Amazon Relational Database Service

User Guide

API Version 2014-10-31

Amazon Relational Database Service: User Guide

Table of Contents

What Is Amazon Relational Database Service (Amazon RDS)?

Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the cloud. It provides cost-efficient, resizable capacity for an industry-standard relational database and manages common database administration tasks.

Overview of Amazon RDS

Why do you want a managed relational database service? Because Amazon RDS takes over many of the difficult or tedious management tasks of a relational database:

- When you buy a server, you get CPU, memory, storage, and IOPS, all bundled together. With Amazon RDS, these are split apart so that you can scale them independently. If you need more CPU, less IOPS, or more storage, you can easily allocate them.
- Amazon RDS manages backups, software patching, automatic failure detection, and recovery.
- To deliver a managed service experience, Amazon RDS doesn't provide shell access to DB instances, and it restricts access to certain system procedures and tables that require advanced privileges.
- You can have automated backups performed when you need them, or manually create your own backup snapshot. You can use these backups to restore a database. The Amazon RDS restore process works reliably and efficiently.
- You can get high availability with a primary instance and a synchronous secondary instance that you can fail over to when problems occur. You can also use MySQL, MariaDB, or PostgreSQL Read Replicas to increase read scaling.
- You can use the database products you are already familiar with: MySQL, MariaDB, PostgreSQL, Oracle, Microsoft SQL Server, and the new, MySQL-compatible Amazon Aurora DB engine (for information, see Amazon Aurora on Amazon RDS (p. 428)).
- In addition to the security in your database package, you can help control who can access your RDS databases by using AWS Identity and Access Management (IAM) to define users and permissions. You can also help protect your databases by putting them in a virtual private cloud.

If you are new to AWS products and services, begin learning more with the following resources:

- For an overview of all AWS products, see What is Cloud Computing?.
- Amazon Web Services provides a number of database services. For guidance on which service is best for your environment, see Running Databases on AWS.

DB Instances

The basic building block of Amazon RDS is the *DB instance*. A DB instance is an isolated database environment in the cloud. A DB instance can contain multiple user-created databases, and you can access it by using the same tools and applications that you use with a stand-alone database instance. You can create and modify a DB instance by using the AWS Command Line Interface, the Amazon RDS API, or the AWS Management Console.

Each DB instance runs a *DB engine*. Amazon RDS currently supports the MySQL, MariaDB, PostgreSQL, Oracle, and Microsoft SQL Server DB engines. Each DB engine has its own supported features, and each version of a DB engine may include specific features. Additionally, each DB engine has a set of parameters in a DB parameter group that control the behavior of the databases that it manages.

The computation and memory capacity of a DB instance is determined by its *DB instance class*. You can select the DB instance that best meets your needs. If your needs change over time, you can change DB instances. For information, see DB Instance Class (p. 92).

Note
For pricing information on DB instance classes, go to the Pricing section of the Amazon Relational Database Service (Amazon RDS) product page.

DB instance storage comes in three types: Magnetic, General Purpose (SSD), and Provisioned IOPS (PIOPS). They differ in performance characteristics and price, allowing you to tailor your storage performance and cost to the needs of your database. Each DB instance has minimum and maximum storage requirements depending on the storage type and the database engine it supports. It's important to have sufficient storage so that your databases have room to grow and that features for the DB engine have room to write content or log entries. For more information, see Storage for Amazon RDS (p. 410).

You can run a DB instance on a virtual private cloud using the Amazon Virtual Private Cloud (VPC) service. When you use a virtual private cloud, you have control over your virtual networking environment: you can select your own IP address range, create subnets, and configure routing and access control lists. The basic functionality of Amazon RDS is the same whether it is running in a VPC or not; Amazon RDS manages backups, software patching, automatic failure detection, and recovery. There is no additional cost to run your DB instance in a VPC. For more information on VPC and RDS, see Amazon Virtual Private Cloud (VPCs) and Amazon RDS (p. 390).

Amazon RDS uses Network Time Protocol (NTP) to synchronize the time on DB Instances.

Regions and Availability Zones

Amazon cloud computing resources are housed in highly available data center facilities in different areas of the world (for example, North America, Europe, or Asia). Each data center location is called a region.

Each region contains multiple distinct locations called Availability Zones, or AZs. Each Availability Zone is engineered to be isolated from failures in other Availability Zones, and to provide inexpensive, low-latency network connectivity to other Availability Zones in the same region. By launching instances in separate Availability Zones, you can protect your applications from the failure of a single location. For more information, see Regions and Availability Zones (p. 97).

You can run your DB instance in several Availability Zones, an option called a Multi-AZ deployment. When you select this option, Amazon automatically provisions and maintains a synchronous standby replica of your DB instance in a different Availability Zone. The primary DB instance is synchronously replicated across Availability Zones to the standby replica to provide data redundancy, failover support, eliminate I/O freezes, and minimize latency spikes during system backups.

Security

A security group controls the access to a DB instance. It does so by allowing access to IP address ranges or Amazon EC2 instances that you specify.

Amazon RDS uses DB security groups, VPC security groups, and EC2 security groups. In simple terms, a DB security group controls access to a DB instance that is not in a VPC, a VPC security group controls access to a DB instance inside a VPC, and an Amazon EC2 security group controls access to an EC2

instance and can be used with a DB instance. For more information about security groups, see Security in Amazon RDS (p. 326).

Monitoring an Amazon RDS DB Instance

There are several ways that you can track the performance and health of a DB instance. You can use the free Amazon CloudWatch service to monitor the performance and health of a DB instance; performance charts are shown in the Amazon RDS console. You can subscribe to Amazon RDS events to be notified when changes occur with a DB instance, DB Snapshot, DB parameter group, or DB security group. For more information, see Monitoring Amazon RDS (p. 245).

Amazon RDS Interfaces

There are several ways that you can interact with Amazon RDS.

AWS Management Console

The AWS Management Console is a simple web-based user interface. You can manage your DB instances from the console with no programming required. To access the Amazon RDS console, sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.

Command Line Interface

You can use the AWS Command Line Interface (AWS CLI) to access the Amazon RDS API interactively. To install the AWS CLI, see Installing the AWS Command Line Interface. To begin using the AWS CLI for RDS, see AWS Command Line Interface Reference for Amazon RDS.

Programming with Amazon RDS

If you are a developer, you can access the Amazon RDS programmatically. For more information, see Amazon RDS Application Programming Interface (API) (p. 1240).

For application development, we recommend that you use one of the AWS Software Development Kits (SDKs). The AWS SDKs handle low-level details such as authentication, retry logic, and error handling, so that you can focus on your application logic. AWS SDKs are available for a wide variety of languages. For more information, see Tools for Amazon Web Services .

AWS also provides libraries, sample code, tutorials, and other resources to help you get started more easily. For more information, see Sample Code & Libraries.

How You Are Charged for Amazon RDS

When you use Amazon RDS, you pay only for what you use, and there are no minimum or setup fees. You are billed according to the following criteria.

- Instance class – Pricing is based on the class (for example, micro, small, large, xlarge) of the DB instance consumed.
- Running time – You are billed by the instance-hour, which is equivalent to a single instance running for an hour. For example, both a single instance running for two hours and two instances running for one

hour consume two instance-hours. If a DB instance runs for only part of an hour, you are charged for a full instance-hour.

- Storage – The storage capacity that you have provisioned to your DB instance is billed per GB per month. If you scale your provisioned storage capacity within the month, your bill is pro-rated.
- I/O requests per month – Total number of storage I/O requests that you have made in a billing cycle.
- Backup storage – Backup storage is the storage that is associated with automated database backups and any active database snapshots that you have taken. Increasing your backup retention period or taking additional database snapshots increases the backup storage consumed by your database. Amazon RDS provides backup storage up to 100% of your provisioned database storage at no additional charge. For example, if you have 10 GB-months of provisioned database storage, we provide up to 10 GB-months of backup storage at no additional charge. Most databases require less raw storage for a backup than for the primary dataset, so if you don't keep multiple backups, you never pay for backup storage. Backup storage is free only for active DB instances.
- Data transfer –Internet data transfer in and out of your DB instance.

In addition to regular RDS pricing, you can purchase reserved DB instances. Reserved DB instances let you make a one-time up-front payment for a DB instance and reserve the DB instance for a one- or three-year term at significantly lower rates. For more information on reserved DB instances, see Working with Reserved DB Instances (p. 189)

For Amazon RDS pricing information, see the Amazon RDS product page.

What's Next?

The preceding section introduced you to the basic infrastructure components that RDS offers. What should you do next?

Getting Started

Create a DB instance using instructions in the Getting Started with Amazon RDS (p. 10) section.

Database Engine–Specific Topics

You can review information specific to a particular DB engine in the following sections:

- Amazon Aurora on Amazon RDS (p. 428)
- MariaDB on Amazon RDS (p. 666)
- Microsoft SQL Server on Amazon RDS (p. 720)
- MySQL on Amazon RDS (p. 820)
- Oracle on Amazon RDS (p. 931)
- PostgreSQL on Amazon RDS (p. 1144)

Setting Up for Amazon RDS

Before you use Amazon RDS for the first time, complete the following tasks:

1. Sign Up for AWS (p. 5)
2. Create an IAM User (p. 5)
3. Determine Requirements (p. 6)
4. Provide Access to the DB Instance in the VPC by Creating a Security Group (p. 8)

Sign Up for AWS

When you sign up for Amazon Web Services (AWS), your AWS account is automatically signed up for all services in AWS, including Amazon RDS. You are charged only for the services that you use.

With Amazon RDS, you pay only for the resources you use. The Amazon RDS DB instance that you create is live (not running in a sandbox). You incur the standard Amazon RDS usage fees for the instance until you terminate it. For more information about Amazon RDS usage rates, see the Amazon RDS product page. If you are a new AWS customer, you can get started with Amazon RDS for free; for more information, see AWS Free Usage Tier.

If you have an AWS account already, skip to the next task. If you don't have an AWS account, use the following procedure to create one.

To create an AWS account

1. Open https://aws.amazon.com/, and then choose **Create an AWS Account**.

   Note
   This might be unavailable in your browser if you previously signed into the AWS Management Console. In that case, choose **Sign In to the Console**, and then choose **Create a new AWS account**.
2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a PIN using the phone keypad.

Note your AWS account number, because you'll need it for the next task.

Create an IAM User

Services in AWS, such as Amazon RDS, require that you provide credentials when you access them, so that the service can determine whether you have permission to access its resources. The console requires your password. You can create access keys for your AWS account to access the command line interface or API. However, we don't recommend that you access AWS using the credentials for your AWS account; we recommend that you use AWS Identity and Access Management (IAM) instead. Create an IAM user, and then add the user to an IAM group with administrative permissions or grant this user administrative permissions. You can then access AWS using a special URL and the credentials for the IAM user.

If you signed up for AWS but have not created an IAM user for yourself, you can create one using the IAM console.

To create an IAM user for yourself and add the user to an Administrators group

1. Use your AWS account email address and password to sign in to the AWS Management Console as the *AWS account root user*.
2. In the navigation pane of the console, choose **Users**, and then choose **Add user**.
3. For **User name**, type **`Administrator`**.
4. Select the check box next to **AWS Management Console access**, select **Custom password**, and then type the new user's password in the text box. You can optionally select **Require password reset** to force the user to select a new password the next time the user signs in.
5. Choose **Next: Permissions**.
6. On the **Set permissions for user** page, choose **Add user to group**.
7. Choose **Create group**.
8. In the **Create group** dialog box, type **`Administrators`**.
9. For **Filter**, choose **Job function**.
10. In the policy list, select the check box for **AdministratorAccess**. Then choose **Create group**.
11. Back in the list of groups, select the check box for your new group. Choose **Refresh** if necessary to see the group in the list.
12. Choose **Next: Review** to see the list of group memberships to be added to the new user. When you are ready to proceed, choose **Create user**.

You can use this same process to create more groups and users, and to give your users access to your AWS account resources. To learn about using policies to restrict users' permissions to specific AWS resources, go to Access Management and Example Policies.

To sign in as this new IAM user, sign out of the AWS console, then use the following URL, where *your_aws_account_id* is your AWS account number without the hyphens (for example, if your AWS account number is `1234-5678-9012`, your AWS account ID is `123456789012`):

```
https://your_aws_account_id.signin.aws.amazon.com/console/
```

Enter the IAM user name and password that you just created. When you're signed in, the navigation bar displays "*your_user_name* @ *your_aws_account_id*".

If you don't want the URL for your sign-in page to contain your AWS account ID, you can create an account alias. From the IAM dashboard, click **Customize** and enter an alias, such as your company name. To sign in after you create an account alias, use the following URL:

```
https://your_account_alias.signin.aws.amazon.com/console/
```

To verify the sign-in link for IAM users for your account, open the IAM console and check under **AWS Account Alias** on the dashboard.

Determine Requirements

The basic building block of Amazon RDS is the DB instance. The DB instance is where you create your databases. A DB instance provides a network address called the **Endpoint**. Your applications connect to the endpoint exposed by the DB instance whenever they need to access the databases created in that DB

instance. The information you specify when you create the DB instance controls configuration elements such as storage, memory, database engine and version, network configuration, security, and maintenance periods.

You must know your DB instance and network needs before you create a security group and before you create a DB instance. For example, you must know the following:

- What are the memory and processor requirements for your application or service? You will use these settings when you determine what DB instance class you will use when you create your DB instance. For specifications about DB instance classes, see DB Instance Class (p. 92).
- Your DB instance is most likely in a virtual private cloud (VPC); some legacy instances are not in a VPC, but if you are a new RDS user (two years or less) or accessing a new region, you are most likely creating an DB instance inside a VPC. The security group rules you need to connect to a DB instance depend on whether your DB instance is in a default VPC, in a user-defined VPC, or outside of a VPC. For information on determining if your account has a default VPC in a region, see Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform (p. 391). The follow list describes the rules for each VPC option:
  - **Default VPC** — If your AWS account has a default VPC in the region, that VPC is configured to support DB instances. If you specify the default VPC when you create the DB instance:
    - You must create a **VPC security group** that authorizes connections from the application or service to the Amazon RDS DB instance with the database. Note that you must use the Amazon EC2 API or the **Security Group** option on the VPC Console to create VPC security groups. For information, see Step 4: Create a VPC Security Group (p. 403).
    - You must specify the default DB subnet group. If this is the first DB instance you have created in the region, Amazon RDS will create the default DB subnet group when it creates the DB instance.
  - **User-defined VPC** — If you want to specify a user-defined VPC when you create a DB instance:
    - You must create a **VPC security group** that authorizes connections from the application or service to the Amazon RDS DB instance with the database. Note that you must use the Amazon EC2 API or the **Security Group** option on the VPC Console to create VPC security groups. For information, see Step 4: Create a VPC Security Group (p. 403)..
    - The VPC must meet certain requirements in order to host DB instances, such as having at least two subnets, each in a separate availability zone. For information, see Amazon Virtual Private Cloud (VPCs) and Amazon RDS (p. 390).
    - You must specify a DB subnet group that defines which subnets in that VPC can be used by the DB instance. For information, see the DB Subnet Group section in Working with a DB Instance in a VPC (p. 400).
  - **No VPC** — if your AWS account does not have a default VPC, and you do not specify a user-defined VPC:
    - You must create a **DB security group** that authorizes connections from the devices and Amazon RDS instances running the applications or utilities that will access the databases in the DB instance. For more information, see Working with DB Security Groups (EC2-Classic Platform) (p. 380).
- Do you need failover support? On Amazon RDS, a standby replica of your DB instance that can be used in the event of a failover is called a Multi-AZ deployment. If you have production workloads, you should use a Multi-AZ deployment. For test purposes, you can usually get by with a single instance, non-Multi-AZ deployment.
- Does your AWS account have policies that grant the permissions needed to perform Amazon RDS operations? If you are connecting to AWS using IAM credentials, your IAM account must have IAM policies that grant the permissions required to perform Amazon RDS operations. For more information, see Authentication and Access Control for Amazon RDS (p. 327).
- What TCP/IP port will your database be listening on? The firewall at some companies may block connections to the default port for your database engine. If your company firewall blocks the default port, choose another port for the new DB instance. Note that once you create a DB instance that listens on a port you specify, you can change the port by modifying the DB instance.

- What region do you want your database in? Having the database close in proximity to the application or web service could reduce network latency.
- What are your storage requirements? Do you need to use Provisioned IOPS? Amazon RDS provides three storage types: magnetic, General Purpose (SSD), and Provisioned IOPS (input/output operations per second) . Magnetic storage, also called standard storage, offers cost-effective storage that is ideal for applications with light or burst I/O requirements. General purpose, SSD-backed storage, also called *gp2*, can provide faster access than disk-based storage. Provisioned IOPS storage is designed to meet the needs of I/O-intensive workloads, particularly database workloads, that are sensitive to storage performance and consistency in random access I/O throughput. For more information on Amazon RDS storage, see Storage for Amazon RDS (p. 410).

Once you have the information you need to create the security group and the DB instance, continue to the next step.

Provide Access to the DB Instance in the VPC by Creating a Security Group

Your DB instance will most likely be created in a VPC. Security groups provide access to the DB instance in the VPC. They act as a firewall for the associated DB instance, controlling both inbound and outbound traffic at the instance level. DB instances are created by default with a firewall and a default security group that prevents access to the DB instance. You must therefore add rules to a security group that enable you to connect to your DB instance. Use the network and configuration information you determined in the previous step to create rules to allow access to your DB instance.

The security group you need to create is a *VPC security group*, unless you have a legacy DB instance not in a VPC that requires a *DB security group*. If you created your AWS account after March 2013, chances are very good that you have a default VPC, and your DB instance will be created in that VPC. DB instances in a VPC require that you add rules to a VPC security group to allow access to the instance.

For example, if you have an application that will access a database on your DB instance in a VPC, you must add a Custom TCP rule that specifies the port range and IP addresses that application will use to access the database. If you have an application on an Amazon EC2 instance, you can use the VPC or EC2 security group you set up for the Amazon EC2 instance.

To create a VPC security group

1. Sign in to the AWS Management Console and open the Amazon VPC console at https://console.aws.amazon.com/vpc.
2. In the top right corner of the AWS Management Console, select the region in which you want to create the VPC security group and the DB instance. In the list of Amazon VPC resources for that region, it should show that you have at least one VPC and several Subnets. If it does not, you do not have a default VPC in that region.
3. In the navigation pane, click **Security Groups**.
4. Click **Create Security Group**.
5. In the **Create Security Group** window, type the **Name tag**, **Group name**, and **Description** of your security group. Select the **VPC** that you want to create your DB instance in. Click **Yes, Create**.
6. The VPC security group you created should still be selected. The details pane at the bottom of the console window displays the details for the security group, and tabs for working with inbound and outbound rules. Click the **Inbound Rules** tab.
7. On the **Inbound Rules** tab, click **Edit**. Select **Custom TCP Rule** from the **Type** list. Type the port value you will use for your DB instance in the **PortRange** text box, and then type the IP address

range (CIDR value) from where you will access the instance, or select a security group name in the **Source** text box.

8. If you need to add more IP addresses or different port ranges, click **Add another rule**.
9. If you need to, you can use the **Outbound Rules** tab to add rules for outbound traffic.
10. When you have finished, click **Save**.

You will use the VPC security group you just created as the security group for your DB instance when you create it. If your DB instance is not going to be in a VPC, then see the topic Working with DB Security Groups (EC2-Classic Platform) (p. 380) to create a DB security group that you will use when you create your DB instance.

Finally, a quick note about VPC subnets: If you use a default VPC, a default subnet group spanning all of the VPC's subnets has already been created for you. When you use the **Launch a DB Instance** wizard to create a DB instance, you can select the default VPC and use **default** for the **DB Subnet Group**.

Once you have completed the setup requirements, you can use your requirements and the security group you created to launch a DB instance. For information on creating a DB instance, see the relevant documentation in the following table:

Database Engine	Relevant Documentation
Amazon Aurora	Creating a DB Cluster and Connecting to a Database on an Amazon Aurora DB Instance (p. 10)
MariaDB	Creating a MariaDB DB Instance and Connecting to a Database on a MariaDB DB Instance (p. 17)
Microsoft SQL Server	Creating a Microsoft SQL Server DB Instance and Connecting to a DB Instance (p. 25)
MySQL	Creating a MySQL DB Instance and Connecting to a Database on a MySQL DB Instance (p. 35)
Oracle	Creating an Oracle DB Instance and Connecting to a Database on an Oracle DB Instance (p. 44)
PostgreSQL	Creating a PostgreSQL DB Instance and Connecting to a Database on a PostgreSQL DB Instance (p. 51)

Getting Started with Amazon RDS

This section shows you how to create and connect to a DB instance using Amazon RDS. You can create, or launch, a DB instance that uses MySQL, Oracle, PostgreSQL, Microsoft SQL Server, Amazon Aurora, or MariaDB.

> **Important**
> You must complete the tasks in the Setting Up for Amazon RDS (p. 5) section before you can create or connect to a DB instance.

Creating a DB instance and connecting to a database on a DB instance is slightly different for each of the DB engines. Choose the DB engine following that you want to use for detailed information on creating and connecting to the DB instance. After you have created and connected to your DB instance, there are instructions to help you delete the DB instance.

Topics

Creating a DB Cluster and Connecting to a Database on an Amazon Aurora DB Instance

The easiest way to create an Amazon Aurora DB cluster is to use the Amazon RDS console. Once you have created the DB cluster, you can use standard MySQL utilities, such as MySQL Workbench, or PostgreSQL utilities, such as pgAdmin, to connect to a database on the DB cluster.

> **Important**
> You must complete the tasks in the Setting Up for Amazon RDS (p. 5) section before you can create or connect to a DB cluster.

Topics

Create a DB Cluster

Before you create a DB cluster, you must first have an Amazon Virtual Private Cloud (VPC) and an Amazon RDS DB subnet group. Your VPC must have at least one subnet in each of at least two

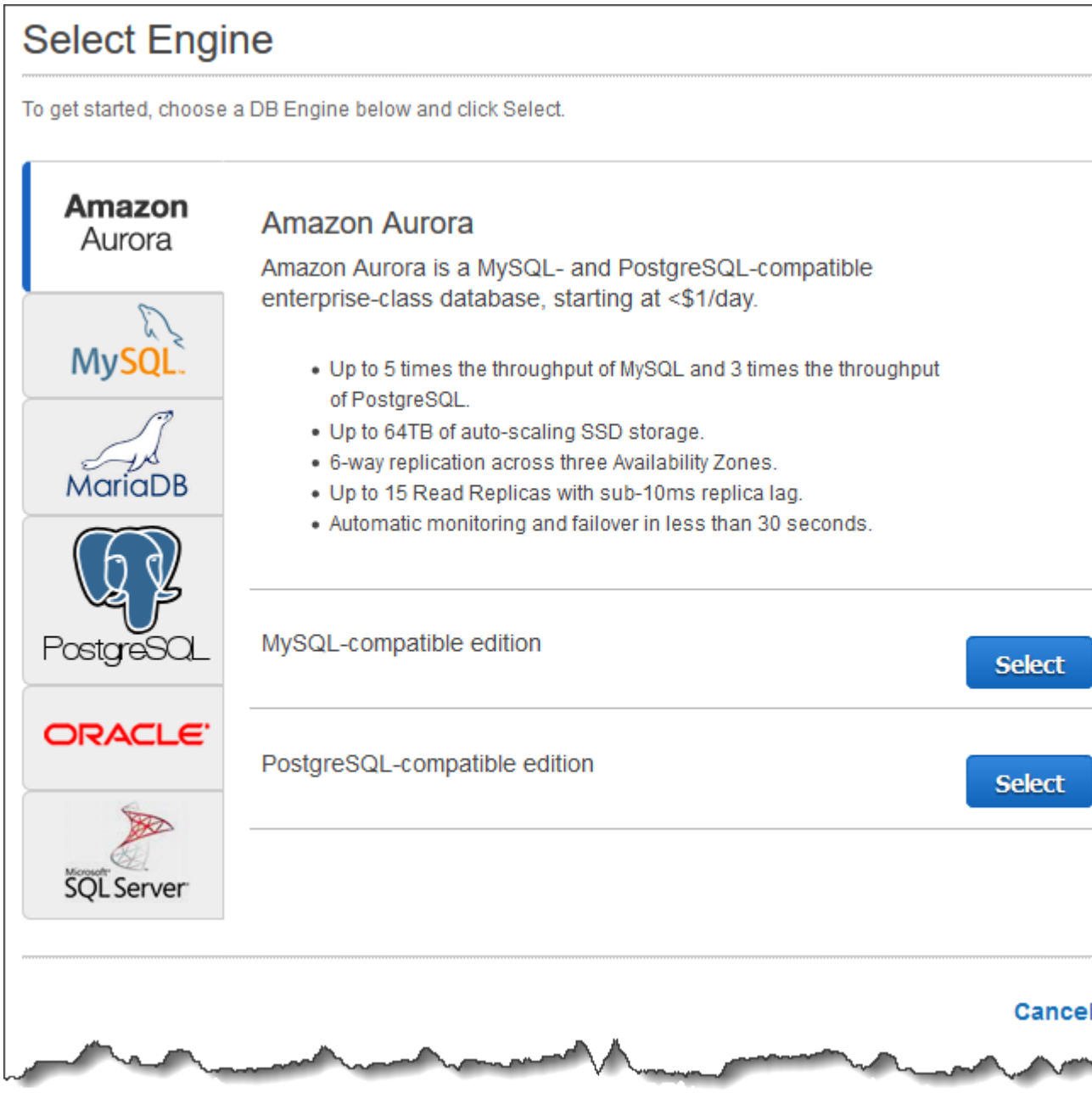
Availability Zones. You can use the default VPC for your AWS account, or you can create your own VPC. The Amazon RDS console makes it easy for you to create your own VPC for use with Amazon Aurora or use an existing VPC with your Aurora DB cluster.

If you want to create a VPC and DB subnet group for use with your Amazon Aurora DB cluster yourself, rather than having Amazon RDS create the VPC and DB subnet group for you, then follow the instructions in How to Create a VPC for Use with Amazon Aurora (p. 452). Otherwise, follow the instructions in this topic to create your DB cluster and have Amazon RDS create a VPC and DB subnet group for you.

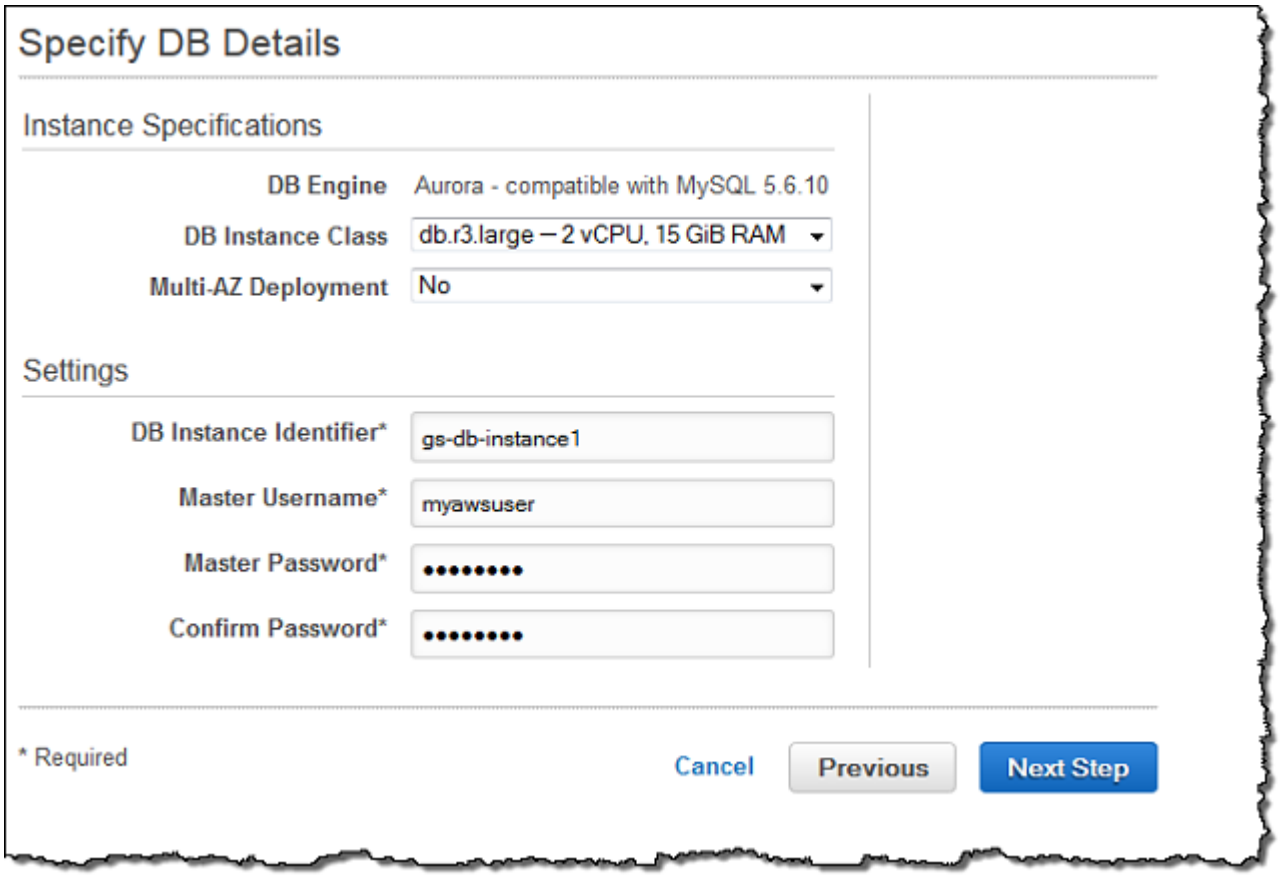
> **Note**
> Aurora is not available in all AWS Regions. For a list of AWS Regions where Aurora is available, see Availability (p. 431).

To launch an Aurora DB cluster

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the top-right corner of the AWS Management Console, choose the AWS Region that you want to create your DB cluster in. For a list of AWS Regions where Aurora is available, see Availability (p. 431).
3. In the left navigation pane, choose **Instances**.
4. Choose **Launch DB Instance** to start the Launch DB Instance Wizard. The wizard opens on the **Select Engine** page.
5. On the **Select Engine** page, choose the **Select** button for the MySQL-compatible edition of Aurora.



6. Set the following values on the **Specify DB Details** page:
   - **DB Instance Class:** `db.r3.large`
   - **DB Instance Identifier:** `gs-db-instance1`
   - **Master Username:** Using alphanumeric characters, type a master user name, used to log on to your DB instances in the DB cluster.
   - **Master Password** and **Confirm Password:** Type a password in the **Master Password** box that contains from 8 to 41 printable ASCII characters (excluding /,", and @) for your master user password, used to log on to your database. Then type the password again in the **Confirm Password** box.



7. Choose **Next** and set the following values on the **Configure Advanced Settings** page:

   - **VPC ID:** If you have an existing VPC, then you can use that VPC with your Amazon Aurora DB cluster by choosing your VPC identifier, for example `vpc-a464d1c1`. For information on using an existing VPC, see How to Create a VPC for Use with Amazon Aurora (p. 452).

     Otherwise, you can choose to have Amazon RDS create a VPC for you by choosing **Create a new VPC**. This example uses the **Create a new VPC** option.

   - **Subnet Group:** If you have an existing subnet group, then you can use that subnet group with your Amazon Aurora DB cluster by choosing your subnet group identifier, for example, `gs-subnet-group1`.

     Otherwise, you can choose to have Amazon RDS create a subnet group for you by choosing **Create a new subnet group**. This example uses the **Create a new subnet group** option.

   - **Publicly Accessible:** `Yes`

     Note
     Your production DB cluster might not need to be in a public subnet, because only your application servers require access to your DB cluster. If your DB cluster doesn't need to be in a public subnet, set **Publicly Accessible** to `No`.

   - **Availability Zone:** `No Preference`
   - **VPC Security Group(s):** If you have one or more existing VPC security groups, then you can use one or more of those VPC security groups with your Amazon Aurora DB cluster by choosing your VPC security group identifiers, for example, `gs-security-group1`.

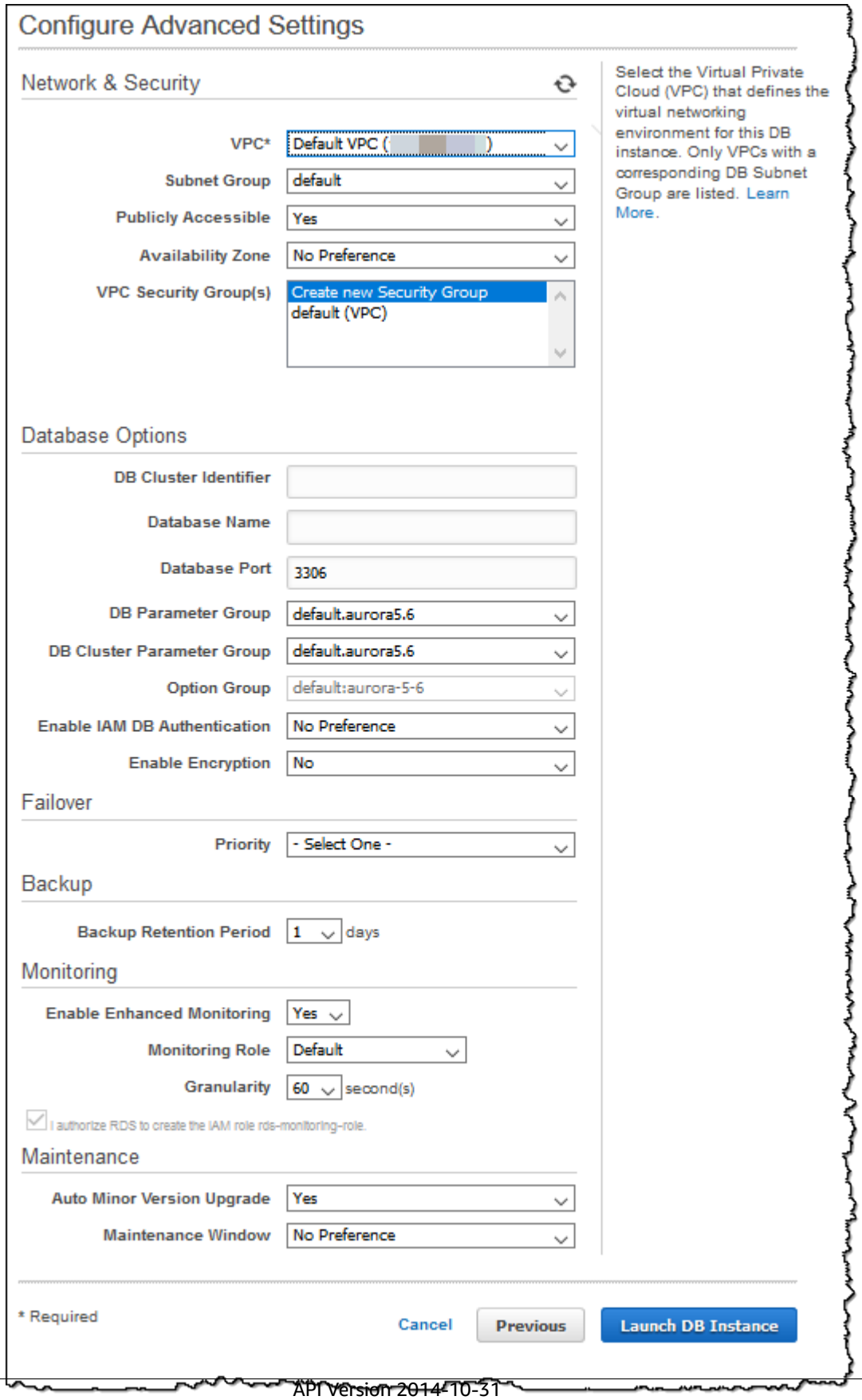
     Otherwise, you can choose to have Amazon RDS create a VPC security group for you by choosing **Create a new Security group**. This example uses the **Create a new Security group** option.

- **DB Cluster Identifier:** `gs-db-cluster1`
- **Database Name:** `sampledb`

  Note
  This creates the default database. To create additional databases, connect to the DB cluster and use the SQL command CREATE DATABASE. For more information about connecting to the DB cluster, see Connecting to an Amazon Aurora DB Cluster (p. 457).

- **Database Port:** `3306`

  Note
  You might be behind a corporate firewall that does not allow access to default ports such as the MySQL default port, 3306. In this case, provide a port value that your corporate firewall allows. Remember that port value later when you connect to the Aurora DB cluster.


Configure Advanced Settings
Network & Security
VPC*
Default VPC (
Subnet Group
default
Publicly Accessible
Yes
Availability Zone
No Preference
VPC Security Group(s)
Create new Security Group
default (VPC)
Select the Virtual Private Cloud (VPC) that defines the virtual networking environment for this DB instance. Only VPCs with a corresponding DB Subnet Group are listed. Learn More.
Database Options
DB Cluster Identifier
Database Name
Database Port
3306
DB Parameter Group
default.aurora5.6
DB Cluster Parameter Group
default.aurora5.6
Option Group
default:aurora-5-6
Enable IAM DB Authentication
No Preference
Enable Encryption
No
Failover
Priority
- Select One -
Backup
Backup Retention Period
1
days
Monitoring
Enable Enhanced Monitoring
Yes
Monitoring Role
Default
Granularity
60
second(s)
I authorize RDS to create the IAM role rds-monitoring-role.
Maintenance
Auto Minor Version Upgrade
Yes
Maintenance Window
No Preference
* Required
Cancel
Previous
Launch DB Instance

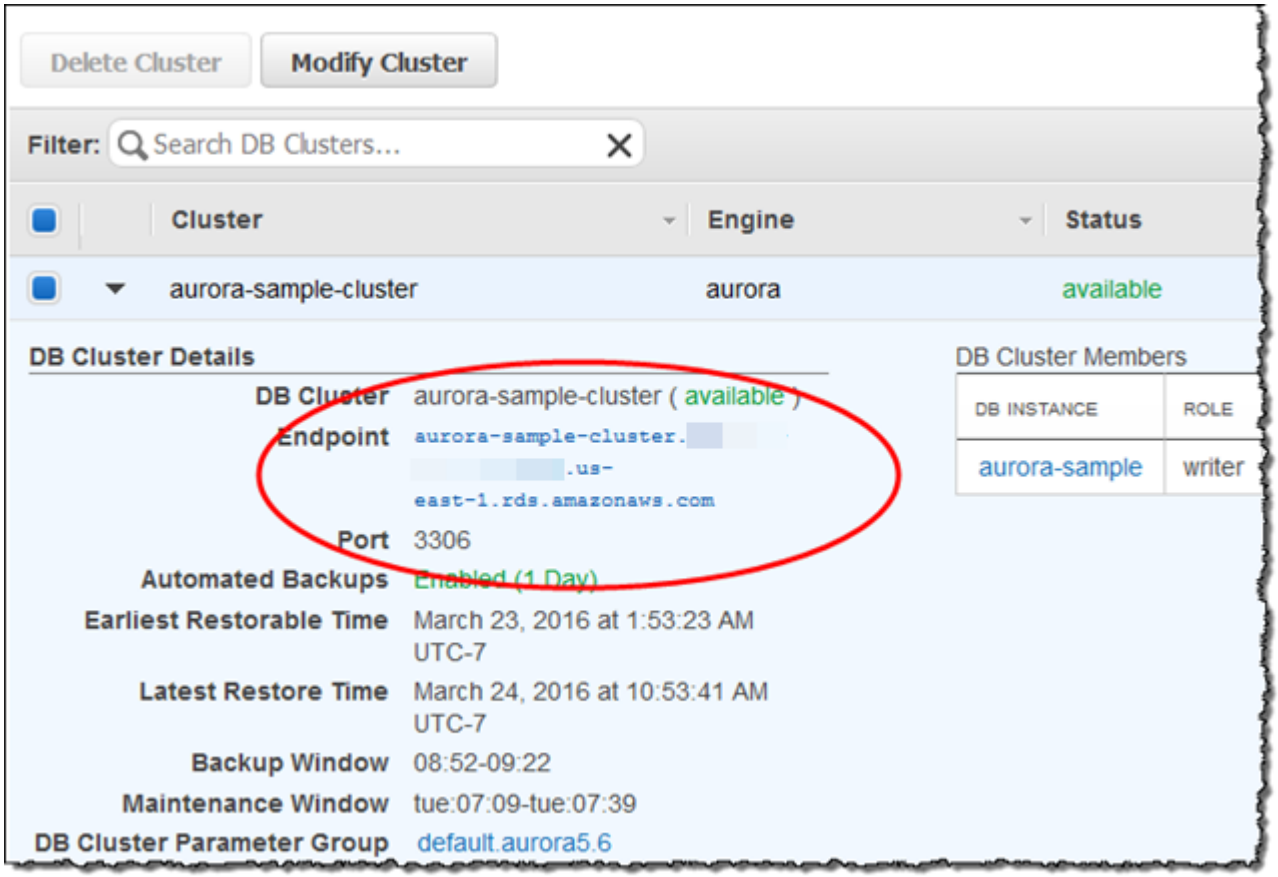
8. Leave the rest of the values as their defaults, and choose **Launch DB Instance** to create the DB cluster and primary instance.

Connect to an Instance in a DB Cluster

Once Amazon RDS provisions your DB cluster and creates the primary instance, you can use any standard SQL client application to connect to a database on the DB cluster. In this example, you connect to a database on the Aurora MySQL DB cluster using MySQL monitor commands. One GUI-based application that you can use to connect is MySQL Workbench. For more information, go to the Download MySQL Workbench page.

To connect to a database on an Aurora MySQL DB cluster using the MySQL monitor

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. Choose **Clusters** and choose the DB cluster from the list to show the DB cluster details. On the details page, copy the value for the endpoint. This endpoint is the cluster endpoint.



3. Type the following command at a command prompt on a client computer to connect to a database on an Aurora MySQL DB cluster using the MySQL monitor. Use the cluster endpoint to connect to the primary instance, and the master user name that you created previously. (You are prompted for a password.) If you supplied a port value other than 3306, use that for the `-P` parameter instead.

```
PROMPT> mysql -h <endpoint> -P 3306 -u <mymasteruser> -p
```

You should see output similar to the following.

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 350
Server version: 5.6.10-log MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Delete the Sample DB Cluster, DB Subnet Group, and VPC

Once you have connected to the sample DB cluster that you created, you can delete the DB cluster, DB subnet group, and VPC (if you created a VPC).

To delete a DB cluster

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. Choose **Instances** and then choose the `gs-db-instance1` DB instance.
3. Choose **Instance Actions**, and then choose **Delete**.
4. Choose **Yes, Delete**.

To delete a DB subnet group

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. Choose **Subnet Groups** and then choose the `gs-subnet-group1` DB subnet group.
3. Choose **Delete**.
4. Choose **Yes, Delete**.

To delete a VPC

1. Sign in to the AWS Management Console and open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. Choose **Your VPCs** and then choose the VPC that was created for this procedure.
3. Choose **Delete**.
4. Choose **Yes, Delete**.

Creating a MariaDB DB Instance and Connecting to a Database on a MariaDB DB Instance

The easiest way to create a MariaDB DB instance is to use the Amazon RDS console. Once you have created the DB instance, you can use command line tools such as `mysql` or standard graphical tools such as HeidiSQL to connect to a database on the DB instance.

> **Important**
> You must complete the tasks in the Setting Up for Amazon RDS (p. 5) section before you can create or connect to a DB instance.

Topics

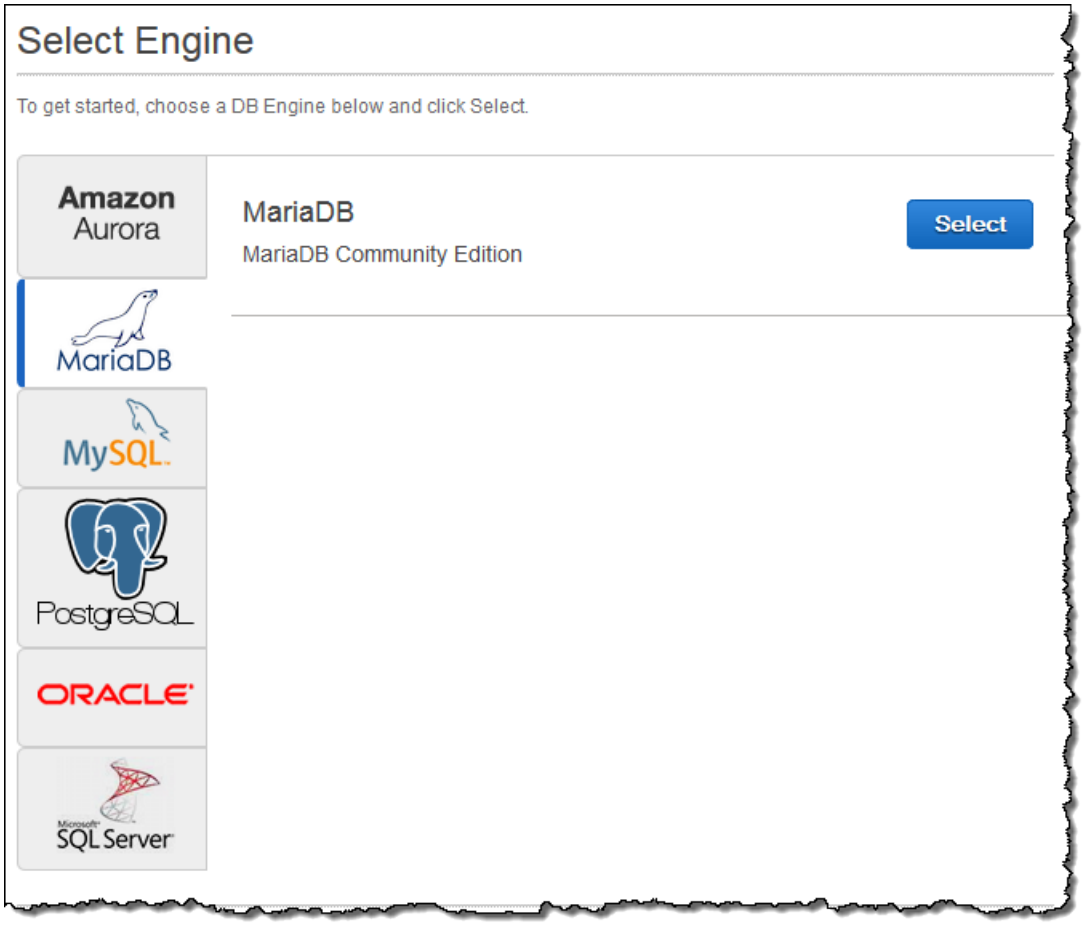
Creating a MariaDB Instance

The basic building block of Amazon RDS is the DB instance. This environment is where you run your MariaDB databases.

In this example, you create a DB instance running the MariaDB database engine called *east1-mariadb-instance1*, with a *db.t2.small* DB instance class, 5 GB of storage, and automated backups enabled with a retention period of one day.

To create a MariaDB DB instance

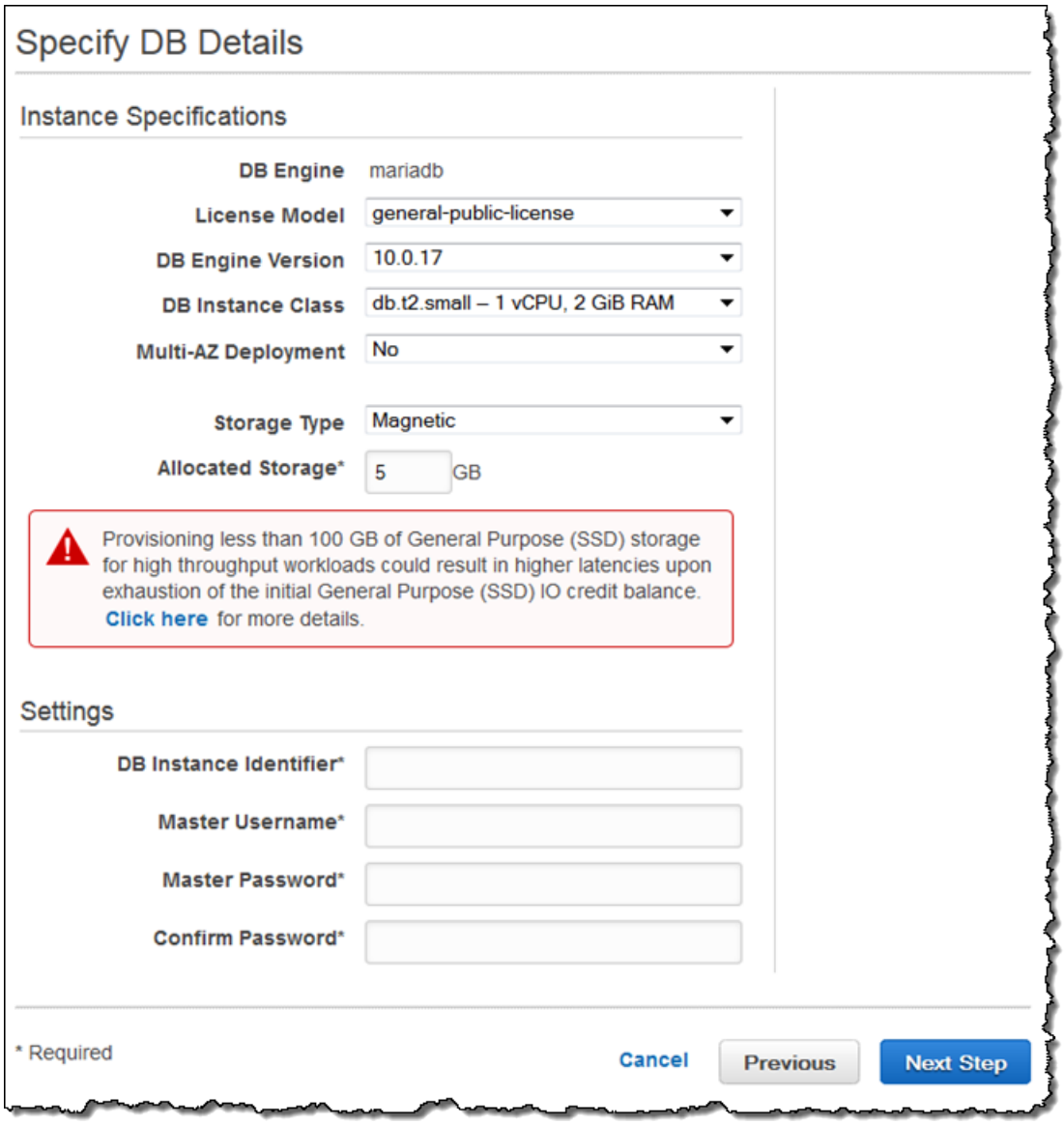
1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the top right corner of the Amazon RDS console, choose the region in which you want to create the DB instance.
3. In the navigation pane, choose **Instances**.
4. Choose **Launch DB Instance**. The **Launch DB Instance Wizard** opens on the **Select Engine** page.



5. On the **Select Engine** page, choose the MariaDB icon, and then choose **Select** for the MariaDB engine.
6. Next, the **Production?** page asks if you plan to use the DB instance you are creating for production. Because this is an example instance, choose **No**. When you are finished, choose **Next**.

   Note
   If you create a production instance, you typically choose **Yes** on this page to enable the failover option Multi-AZ and the Provisioned IOPS storage option.
7. On the **Specify DB Details** page, specify your DB instance information. The following table shows settings for an example DB instance. When the settings are as you want them, choose **Next**.

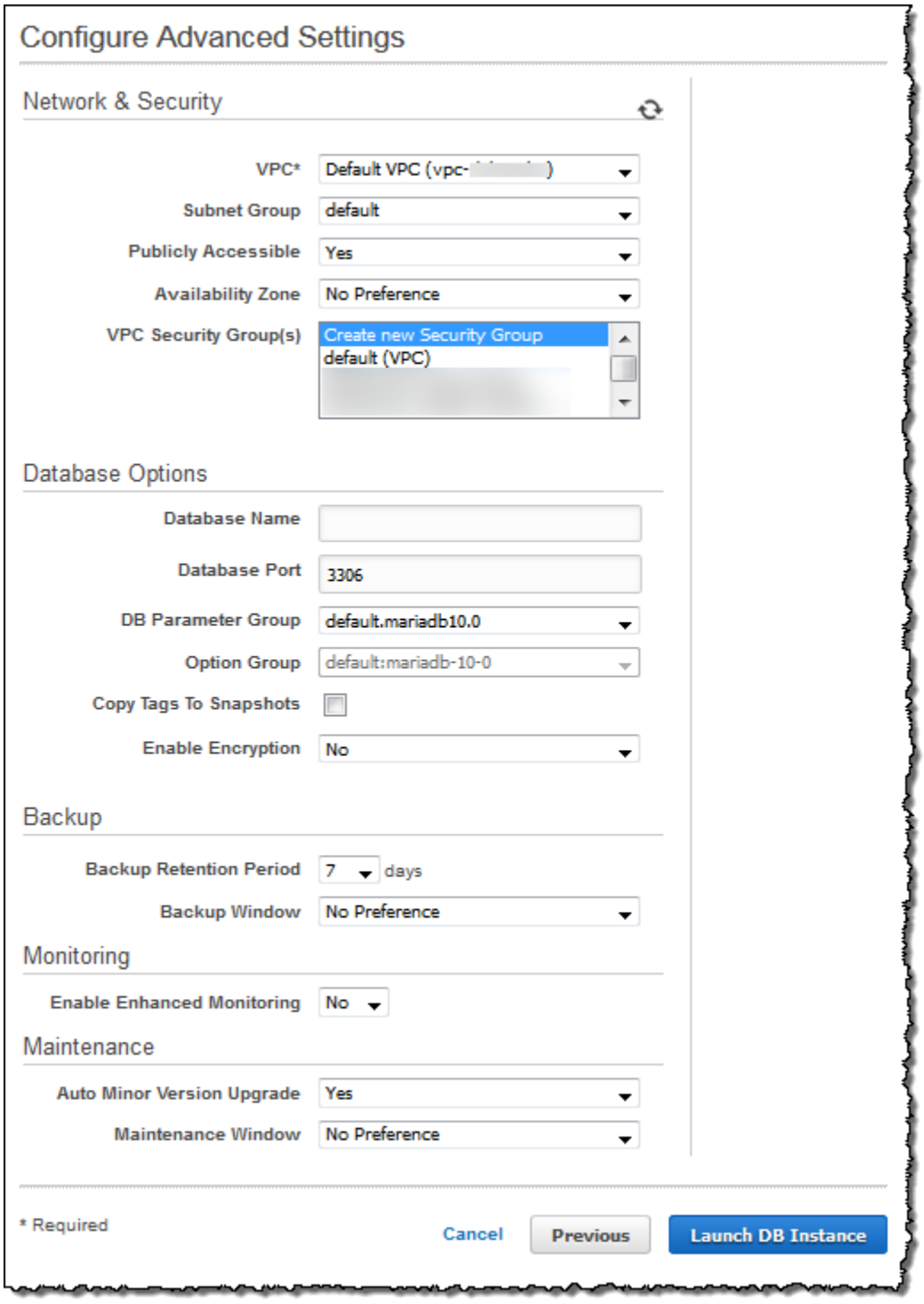
For This Parameter	Do This
License Model	Choose the default, **general-public-license**, to use the GNU General Public License, version 2 for MariaDB. MariaDB has only one license model.
DB Engine Version	Choose the version of MariaDB that you want to use.
DB Instance Class	Choose **db.t2.small** for a configuration that equates to 2 GB memory, 1 ECU (1 virtual core with 1 ECU), 64-bit platform, and moderate I/O capacity.
Multi-AZ Deployment	Choose **`Yes`** to have a standby replica of your DB instance created in another Availability Zone for failover support. We recommend Multi-AZ for production workloads to maintain high availability. For development and testing, you can choose **`No`**. For more information, see High Availability (Multi-AZ) (p. 99).
Storage Type	Choose the storage type **Magnetic**. For more information about storage, see Storage for Amazon RDS (p. 410).
Allocated Storage	Type **`5`** to allocate 5 GB of storage for your database. In some cases, allocating a higher amount of storage for your DB instance than the size of your database can improve I/O performance. For more information about storage allocation, see Amazon Relational Database Service Features.
DB Instance Identifier	Type a name for the DB instance that is unique for your account in the region you chose. You can add some intelligence to the name, such as including the region and DB engine you chose, for example **`east1-mariadb-instance1`**.
Master Username	Type a name using 1-16 alphanumeric characters to use as the master user name to log on to your DB instance. You use this user name to log on to your database on the DB instance for the first time.
Master Password and **Confirm Password**	Type a password that contains from 8 to 41 printable ASCII characters (excluding /,", and @) for your master user password. You use this password with the user name when you log on to your database. Type the password again in the **Confirm Password** box.



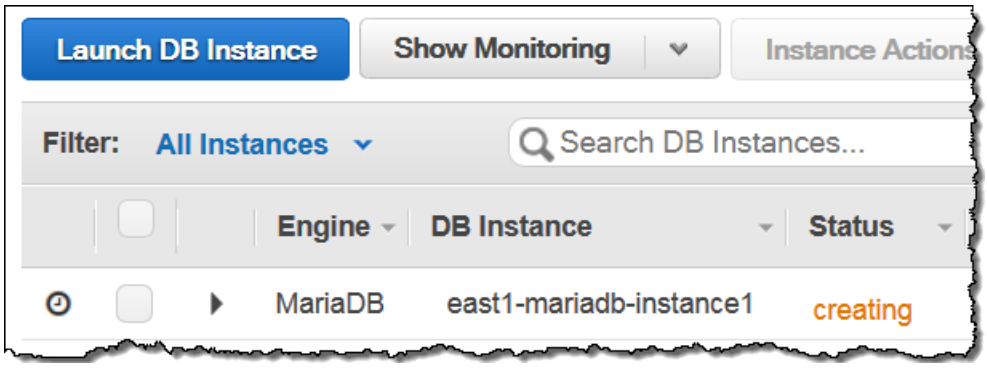
8. On the **Configure Advanced Settings** page, provide additional information that RDS needs to launch the MariaDB DB instance. The table shows settings for an example DB instance. Specify your DB instance information, then choose **Launch DB Instance**.

For This Parameter	Do This
VPC	Choose the name of the Amazon Virtual Private Cloud (Amazon VPC) to host your MariaDB DB instance. For more information about using VPC, see Amazon Virtual Private Cloud (VPCs) and Amazon RDS (p. 390).
Availability Zone	Determine if you want to specify a particular Availability Zone. For more information about Availability Zones, see Regions and Availability Zones (p. 97).

For This Parameter	Do This
VPC Security Groups	Choose the VPC security group you want to use with this DB instance. For more information about VPC security groups, go to Security Groups for Your VPC in the *Amazon Virtual Private Cloud User Guide*.
Database Name	Type a name for your default database that is 1 to 64 alphanumeric characters. If you don't provide a name, Amazon RDS doesn't automatically create a database on the DB instance you are creating. To create additional databases, connect to the DB instance and use the SQL command CREATE DATABASE. For more information about connecting to the DB instance, see Connecting to a DB Instance Running the MariaDB Database Engine (p. 688).
Database Port	Leave the default value of **3306** unless you have a specific port you want to access the database through. MariaDB installations default to port 3306.
DB Parameter Group	Accept the default value of **default.mariadb10.0** unless you created your own DB parameter group. For more information about parameter groups, see Working with DB Parameter Groups (p. 170).
Option Group	Accept the default value of **default.mariadb-10-0**.
Copy Tags To Snapshots	Choose this option to have any DB instance tags copied to a DB snapshot when you create a snapshot. For more information, see Tagging Amazon RDS Resources (p. 129).
Enable Encryption	Choose **No**. **Note** You usually choose **Yes** for production instances to enable encryption at rest for this DB instance. For more information, see Encrypting Amazon RDS Resources (p. 355).
Backup Retention Period	Set the number of days you want automatic backups of your database to be retained. For testing purposes, you can set this value to **1**.
Backup Window	Unless you have a specific time that you want to have your database back up, use the default of **No Preference**.
Enable Enhanced Monitoring	Unless you want to enable gathering metrics in real time for the operating system that your DB instance runs on, use the default of **No**.
Auto Minor Version Upgrade	Choose **Yes** to enable your DB instance to receive minor DB engine version upgrades automatically when they become available.
Maintenance Window	Choose the 30-minute window in which pending modifications to your DB instance are applied. If the time period doesn't matter, choose **No Preference**.



9. On the RDS console, the new DB instance appears in the list of DB instances. The DB instance has a status of **creating** until the DB instance is created and ready for use. When the state changes to **available**, you can connect to a database on the DB instance. Depending on the DB instance class and store allocated, it can take several minutes for the new DB instance to become available.



Connecting to a Database on a DB Instance Running the MariaDB Database Engine

Once Amazon RDS provisions your DB instance, you can use any standard SQL client application to connect to a database on the DB instance. In this example, you connect to a database on a MariaDB DB instance using the `mysql` command-line tool. One GUI-based application you can use to connect is HeidiSQL; for more information, go to the Download HeidiSQL page. For more information on using MariaDB, go to the MariaDB documentation.

To connect to a database on a DB instance using the `mysql` command-line tool

Type the following command at a command prompt on a client computer to connect to a database on a MariaDB DB instance. Substitute the DNS name for your DB instance for `<endpoint>`, the master user name you used for `<mymasteruser>`, and provide the master password you used when prompted for a password.

```
PROMPT> mysql -h <endpoint> -P 3306 -u <mymasteruser> -p <master password>
```

You should see output similar to the following.

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 272
Server version: 5.5.5-10.0.17-MariaDB-log MariaDB Server

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql >
```

Deleting a DB Instance

Once you have connected to the sample DB instance that you created, you should delete the DB instance so you are no longer charged for it.

To delete a DB instance with no final DB snapshot

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.

2. For **Instances**, choose the DB instance you want to delete.
3. For **Instance Actions**, choose **Delete**.
4. For **Create final Snapshot?**, choose **No**.
5. Choose **Yes, Delete**.

Creating a Microsoft SQL Server DB Instance and Connecting to a DB Instance

The basic building block of Amazon RDS is the DB instance. Your Amazon RDS DB instance is similar to your on-premises Microsoft SQL Server. After you create your SQL Server DB instance, you can add one or more custom databases to it.

> **Important**
> You must have an AWS account before you can create a DB instance. If you don't have an AWS account, open https://aws.amazon.com/, and then choose **Create an AWS Account**.

In this topic you create a sample SQL Server DB instance. You then connect to the DB instance and run a simple query. Finally you delete the sample DB instance.

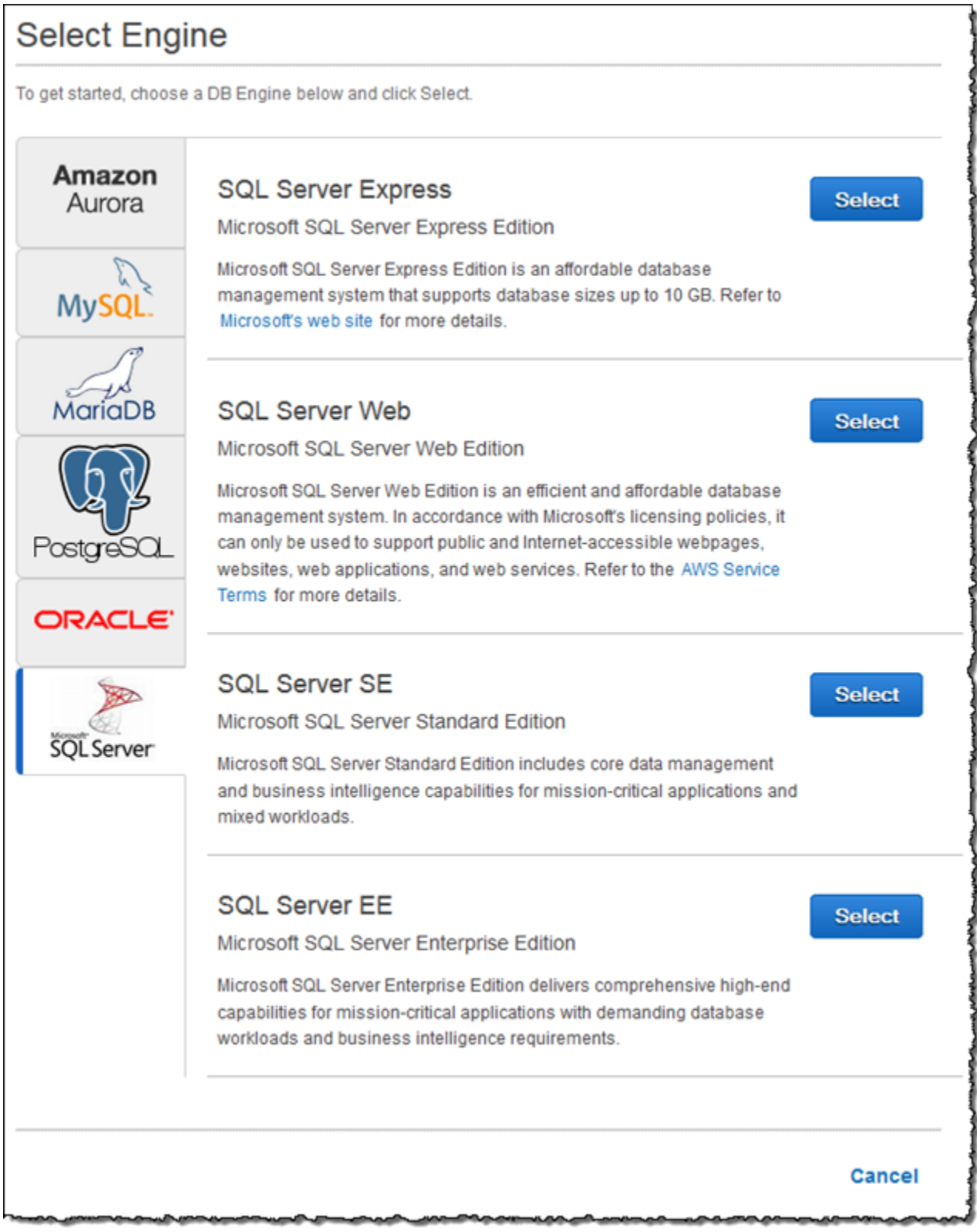
Creating a Sample SQL Server DB Instance

In this procedure you use the AWS Management Console to create a sample DB instance. Since you are only creating a sample DB instance, each setting is not fully explained. For a full explanation of each setting, see Creating a DB Instance Running the Microsoft SQL Server Database Engine (p. 738).

To create a DB instance running the Microsoft SQL Server DB engine

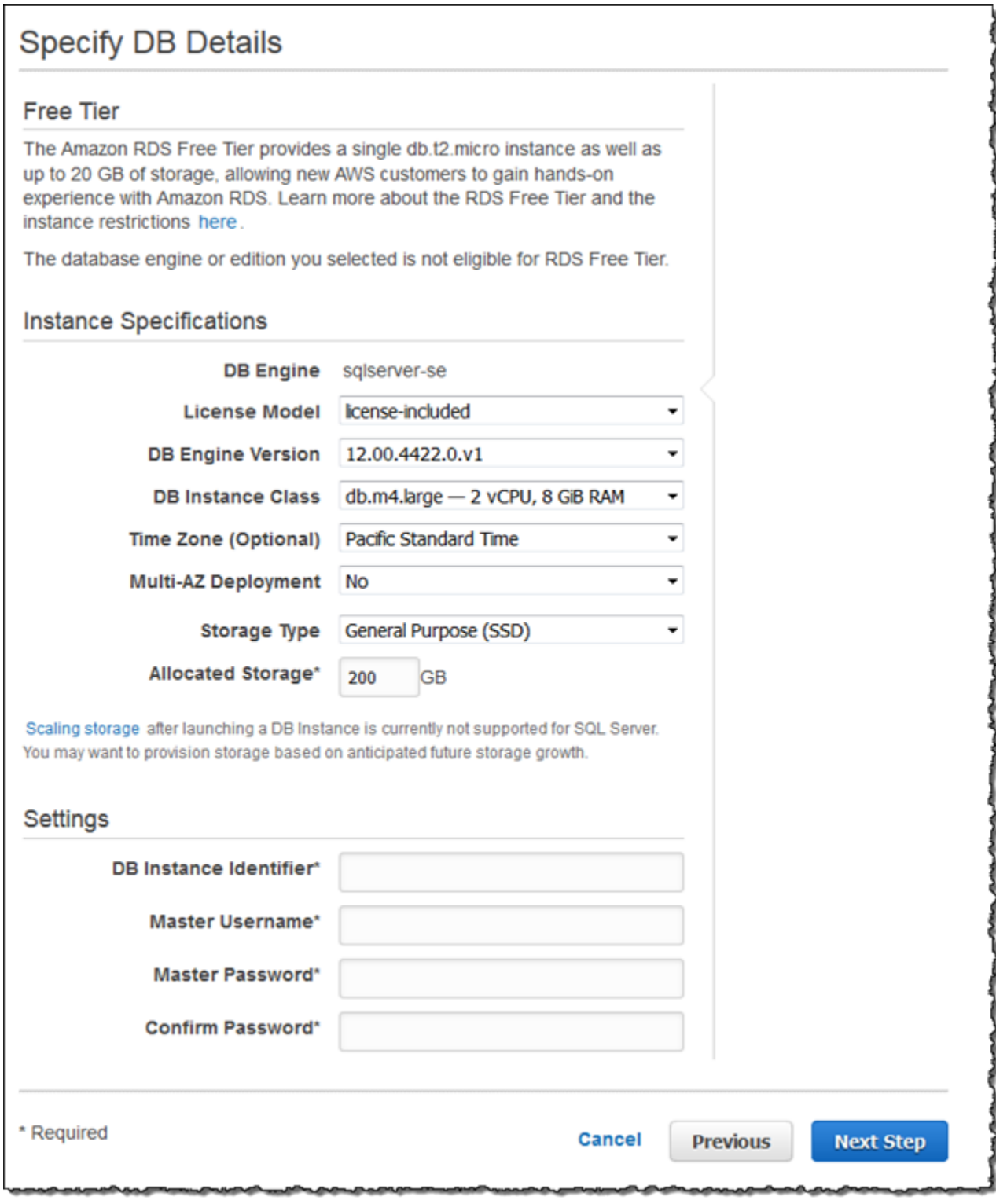
1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the top right corner of the Amazon RDS console, choose the region in which you want to create the DB instance.
3. In the navigation pane, choose **Instances**.
4. Choose **Launch DB Instance**.

   The **Select Engine** page appears.



5. Choose the SQL Server icon, and then choose **Select** for the **SQL Server Express** edition.

The **Specify DB Details** page appears.

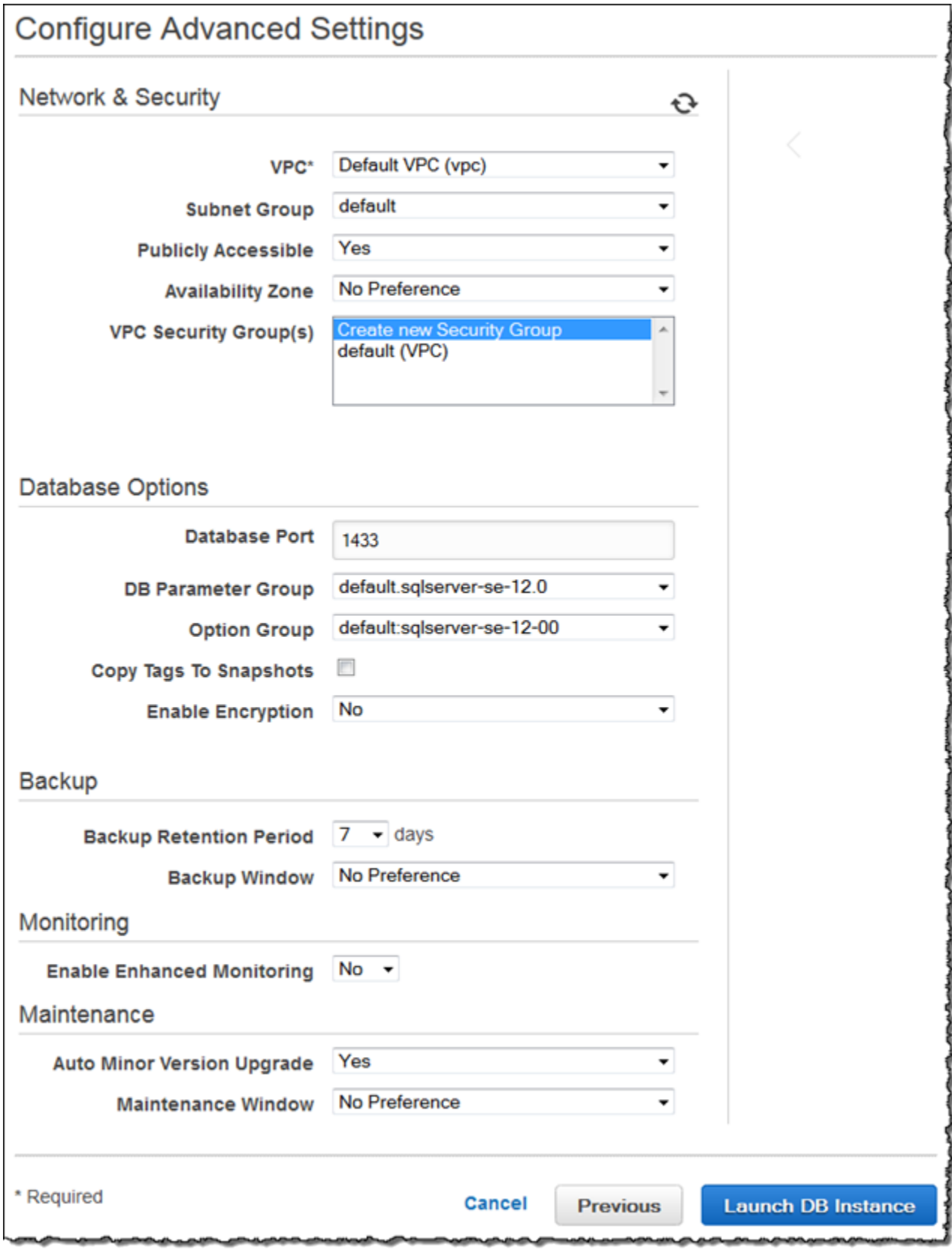


6. On the **Specify DB Details** page, provide the information for your DB instance as shown in the following table:

For This Parameter	Do This
License Model	Choose **license-included** to use the general license agreement for Microsoft SQL Server.
DB Engine Version	Choose the most recent version of SQL Server available in the list.
DB Instance Class	Choose **db.t2.micro**. This instance class is appropriate for testing.
Time Zone	Do not choose a time zone. If you don't choose a time zone, your DB instance uses the default time zone.
Storage Type	Choose the storage type **General Purpose (SSD)**.
Allocated Storage	Type **`20`** to allocate 20 GB of storage for your database. There is a warning that you should consider allocating more storage, but since this is a sample DB instance, 20 GB is sufficient.
DB Instance Identifier	Type **`sample-instance`**.
Master Username	Type a name that you will use as the master user name to log on to your DB Instance with all database privileges. The master user name is a SQL Server Authentication login.
Master Password and **Confirm Password**	Type a password for your master user password. It must contain between 8 and 128 printable ASCII characters (excluding /,", and @).

7. Choose **Next** to continue.

   The **Configure Advanced Settings** page appears.

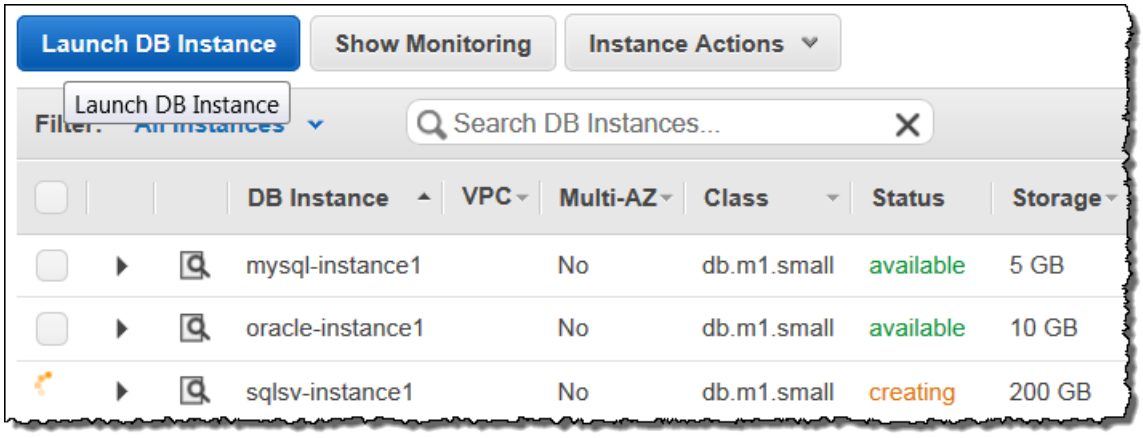

Configure Advanced Settings
Network & Security
VPC*
Default VPC (vpc)
Subnet Group
default
Publicly Accessible
Yes
Availability Zone
No Preference
VPC Security Group(s)
Create new Security Group
default (VPC)
Database Options
Database Port
1433
DB Parameter Group
default.sqlserver-se-12.0
Option Group
default:sqlserver-se-12-00
Copy Tags To Snapshots
Enable Encryption
No
Backup
Backup Retention Period
7
days
Backup Window
No Preference
Monitoring
Enable Enhanced Monitoring
No
Maintenance
Auto Minor Version Upgrade
Yes
Maintenance Window
No Preference
* Required
Cancel
Previous
Launch DB Instance

8. On the **Configure Advanced Settings** page, provide the information for your DB instance as shown in the following table:

For This Parameter	Do This
VPC	Choose **Create new VPC**.
Subnet Group	Choose **Create new DB Subnet Group**.
Publicly Accessible	Choose **Yes**.
Availability Zone	Choose **No Preference**.
VPC Security Group	Choose **Create new Security Group**.
Database Port	Leave the default value of **1433** unless you have a specific port you want to access the database through. SQL Server installations default to port 1433, but in some cases a firewall might block this port. If in doubt, ask your network administrator what port you should use.
DB Parameter Group	Leave the default value.
Option Group	Leave the default value.
Copy Tags To Snapshots	Leave this setting unselected.
Backup Retention Period	Choose **7**.
Backup Window	Choose **No Preference**.
Enable Enhanced Monitoring	Choose **No**.
Auto Minor Version Upgrade	Choose **Yes**.
Maintenance Window	Choose **No Preference**.

9. Choose **Launch DB Instance**.
10. Choose **View Your DB Instances**.

    On the RDS console, the new DB instance appears in the list of DB instances. The DB instance has a status of **creating** until the DB instance is ready to use. When the state changes to **available**, you can connect to the DB instance. Depending on the DB instance class and the amount of storage, it can take up to 20 minutes before the new instance is available.



Connecting to Your Sample SQL Server DB Instance

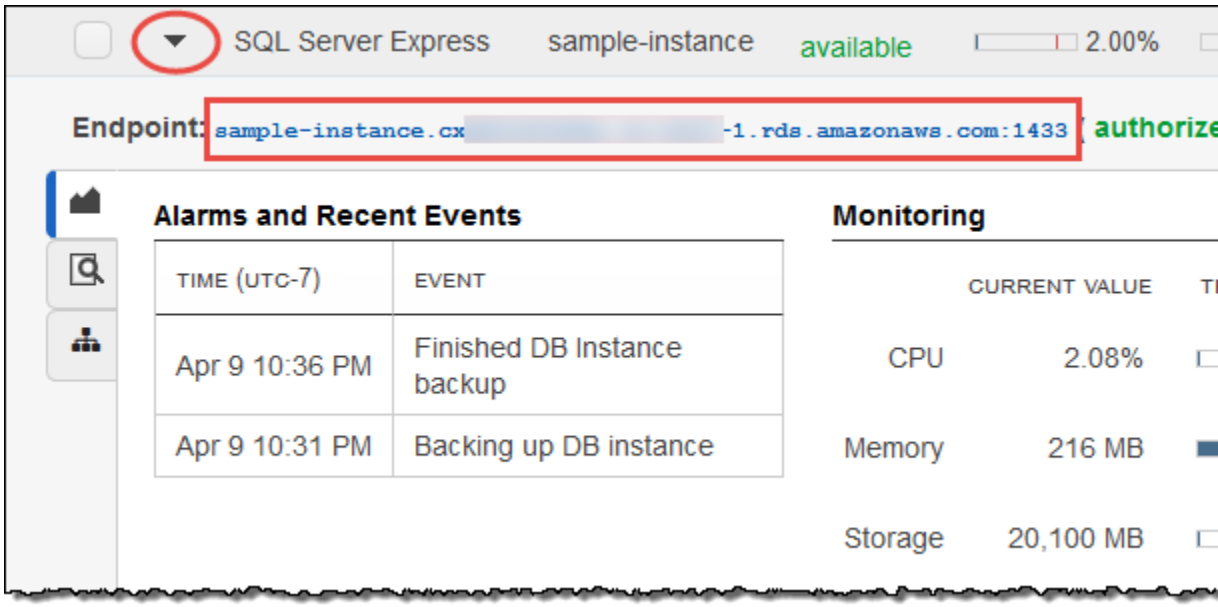
In this procedure you connect to your sample DB instance by using Microsoft SQL Server Management Studio (SSMS). To download a stand-alone version of this utility, see Download SQL Server Management Studio (SSMS) in the Microsoft documentation.

To connect to a DB Instance using SSMS

1. Find the DNS name and port number for your DB Instance.

   a. Open the RDS console and then choose **Instances** to display a list of your DB instances.

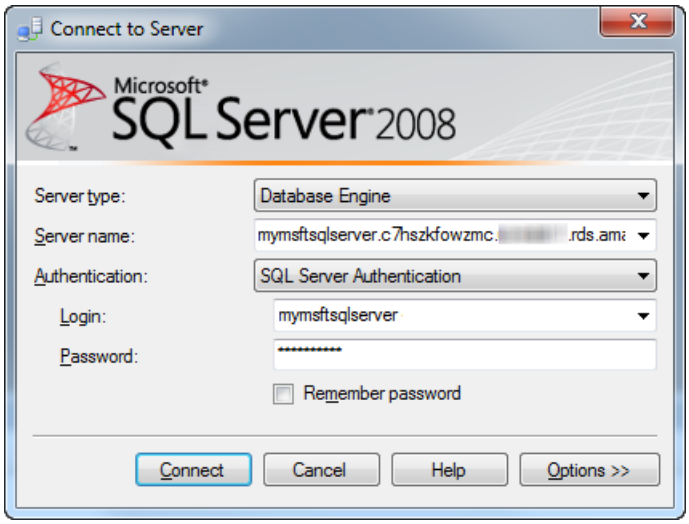
   b. Choose the row for your SQL Server DB instance to display the summary information for the instance.



   c. Copy the endpoint. The **Endpoint** field has two parts separated by a colon (:). The part before the colon is the DNS name for the instance, the part following the colon is the port number. Copy both parts.

2. Start SQL Server Management Studio.

   The **Connect to Server** dialog box appears.



3. Provide the information for your sample DB instance.

   a. For **Server type**, choose **Database Engine**.

   b. For **Server name**, type or paste the DNS name and port number of your sample DB Instance, separated by a comma.

      Important
      Change the colon between the DNS name and port number to a comma.

      For example, your server name should look like the following:

      ```
      sample-instance.cg034hpkmmjt.us-east-1.rds.amazonaws.com,1433
      ```

   c. For **Authentication**, choose **SQL Server Authentication**.

   d. For **Login**, type the master user name you chose earlier for your sample DB instance.

   e. For **Password**, type the password you chose earlier for your sample DB instance.

4. Choose **Connect**.

   After a few moments, SSMS connects to your DB instance. If you can't connect to your DB instance, see Troubleshooting the Connection to Your SQL Server DB Instance (p. 754).

Exploring Your Sample SQL Server DB Instance

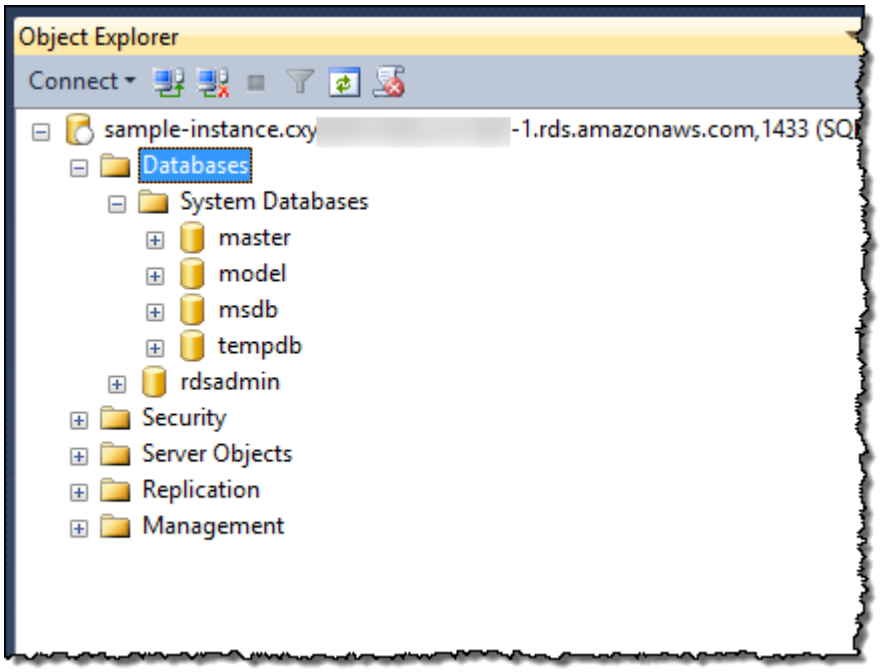
In this procedure you continue the previous procedure and explore your sample DB instance by using Microsoft SQL Server Management Studio (SSMS).

To explore a DB Instance using SSMS

1. Your SQL Server DB instance comes with SQL Server's standard built-in system databases (master, model, msdb, and tempdb). To explore the system databases, do the following:

   a. In SSMS, on the **View** menu, choose **Object Explorer**.

   b. Expand your DB instance, expand **Databases**, and then expand **System Databases** as shown following.



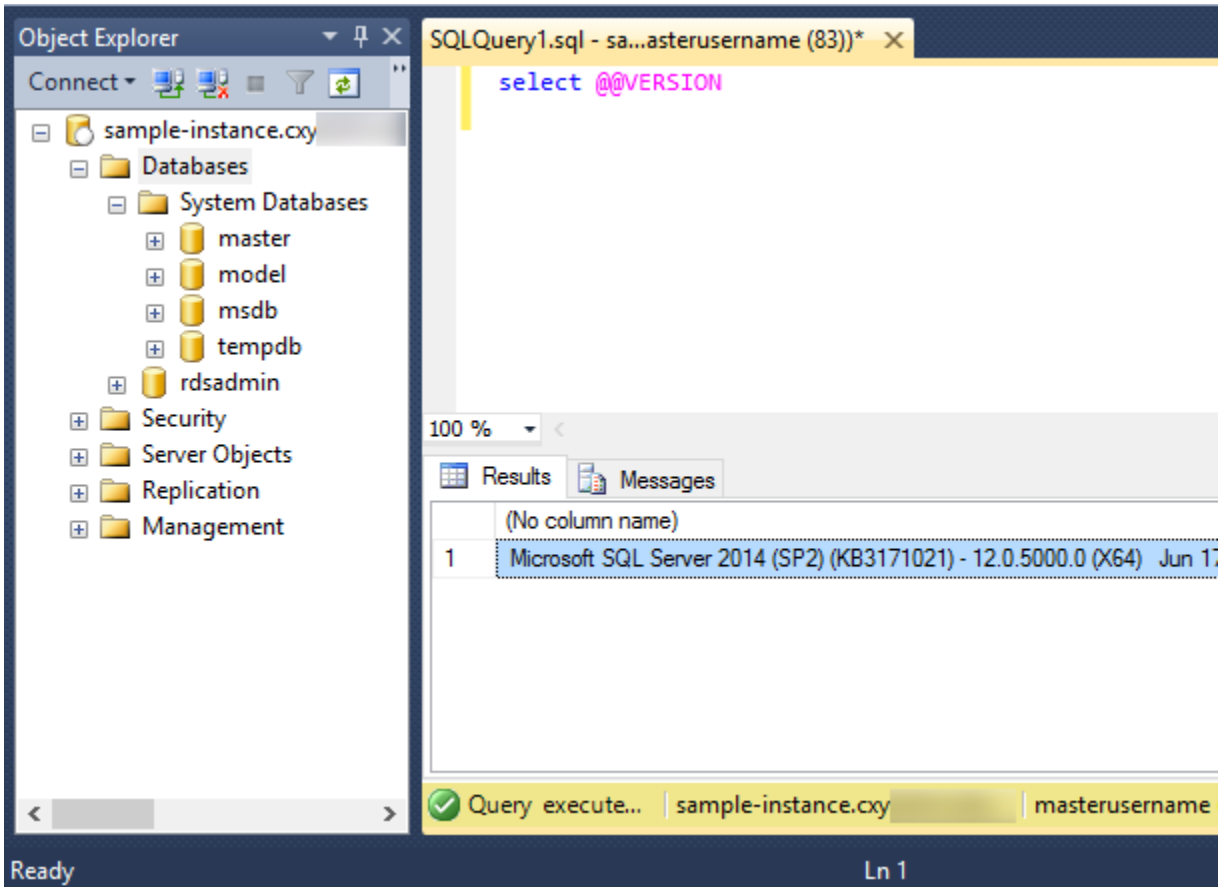
2. Your SQL Server DB instance also comes with a database named `rdsadmin`. Amazon RDS uses this database to store the objects that it uses to manage your database. The `rdsadmin` database also includes stored procedures that you can run to perform advanced tasks.
3. You can now start creating your own databases and running queries against your DB instance and databases as usual. To run a test query against your sample DB instance, do the following:

   a. In SSMS, on the **File** menu point to **New** and then choose **Query with Current Connection**.

   b. Type the following SQL query:

   ```
   select @@VERSION
   ```

   c. Run the query. SSMS returns the SQL Server version of your Amazon RDS DB instance.



Deleting Your Sample DB Instance

Once you are done exploring the sample DB instance that you created, you should delete the DB instance so that you are no longer charged for it.

To delete a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the **Instances** list, choose your sample DB instance.
3. Choose **Instance Actions**, and then choose **Delete**.
4. For **Create final Snapshot**, choose **No**.

   Note
   You should create a final snapshot for any production DB instance that you delete.
5. Choose **Delete**.

Related Topics

- Tutorial: Create an Amazon VPC for Use with an Amazon RDS DB Instance (p. 406)
- Creating a DB Instance Running the Microsoft SQL Server Database Engine (p. 738)
- Connecting to a DB Instance Running the Microsoft SQL Server Database Engine (p. 749)

- Modifying a DB Instance Running the Microsoft SQL Server Database Engine (p. 756)
- Microsoft SQL Server on Amazon RDS (p. 720)

Creating a MySQL DB Instance and Connecting to a Database on a MySQL DB Instance

The easiest way to create a DB instance is to use the AWS Management Console. Once you have created the DB instance, you can use standard MySQL utilities such as MySQL Workbench to connect to a database on the DB instance.

> **Important**
> You must complete the tasks in the Setting Up for Amazon RDS (p. 5) section before you can create or connect to a DB instance.

Topics

- Creating a MySQL DB Instance (p. 35)
- Connecting to a Database on a DB Instance Running the MySQL Database Engine (p. 42)
- Deleting a DB Instance (p. 42)

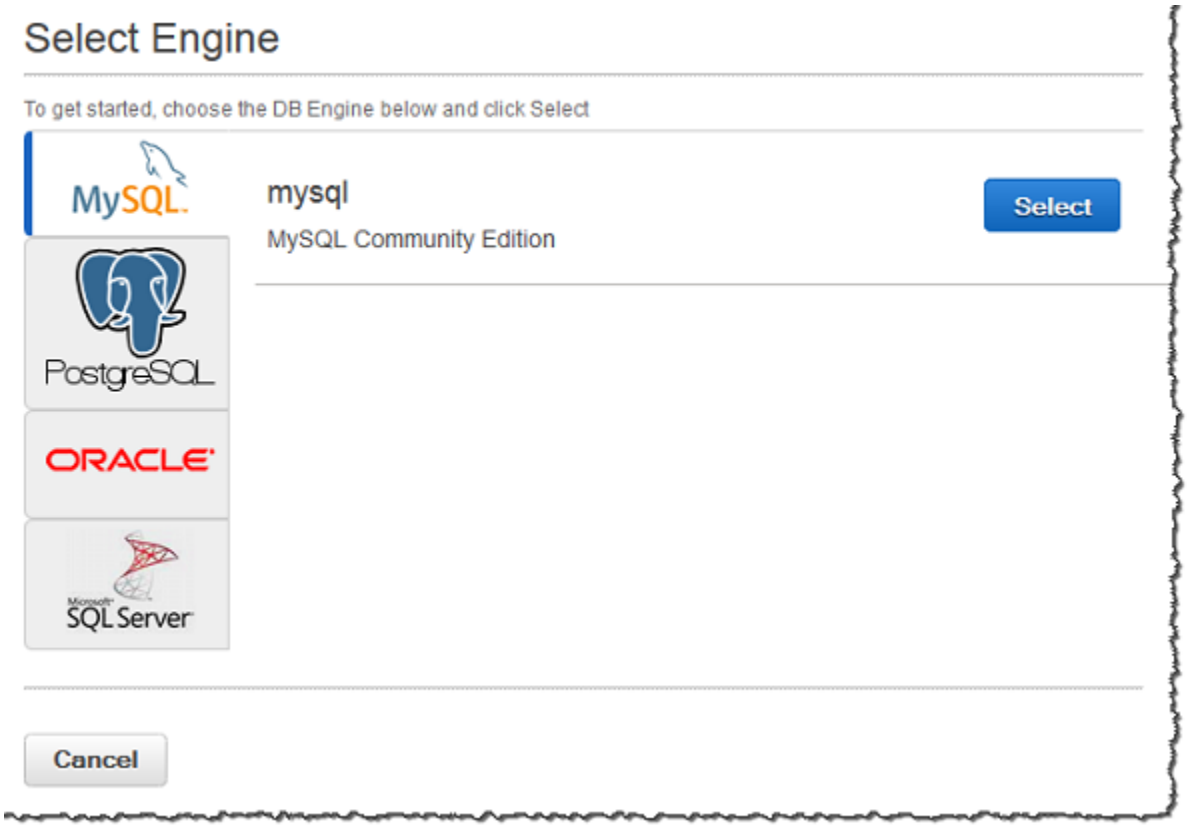
Creating a MySQL DB Instance

The basic building block of Amazon RDS is the DB instance. This is the environment in which you run your MySQL databases.

In this example, you create a DB instance running the MySQL database engine called *west2-mysql-instance1*, with a *db.m1.small* DB instance class, 5 GB of storage, and automated backups enabled with a retention period of one day.

To create a MySQL DB instance

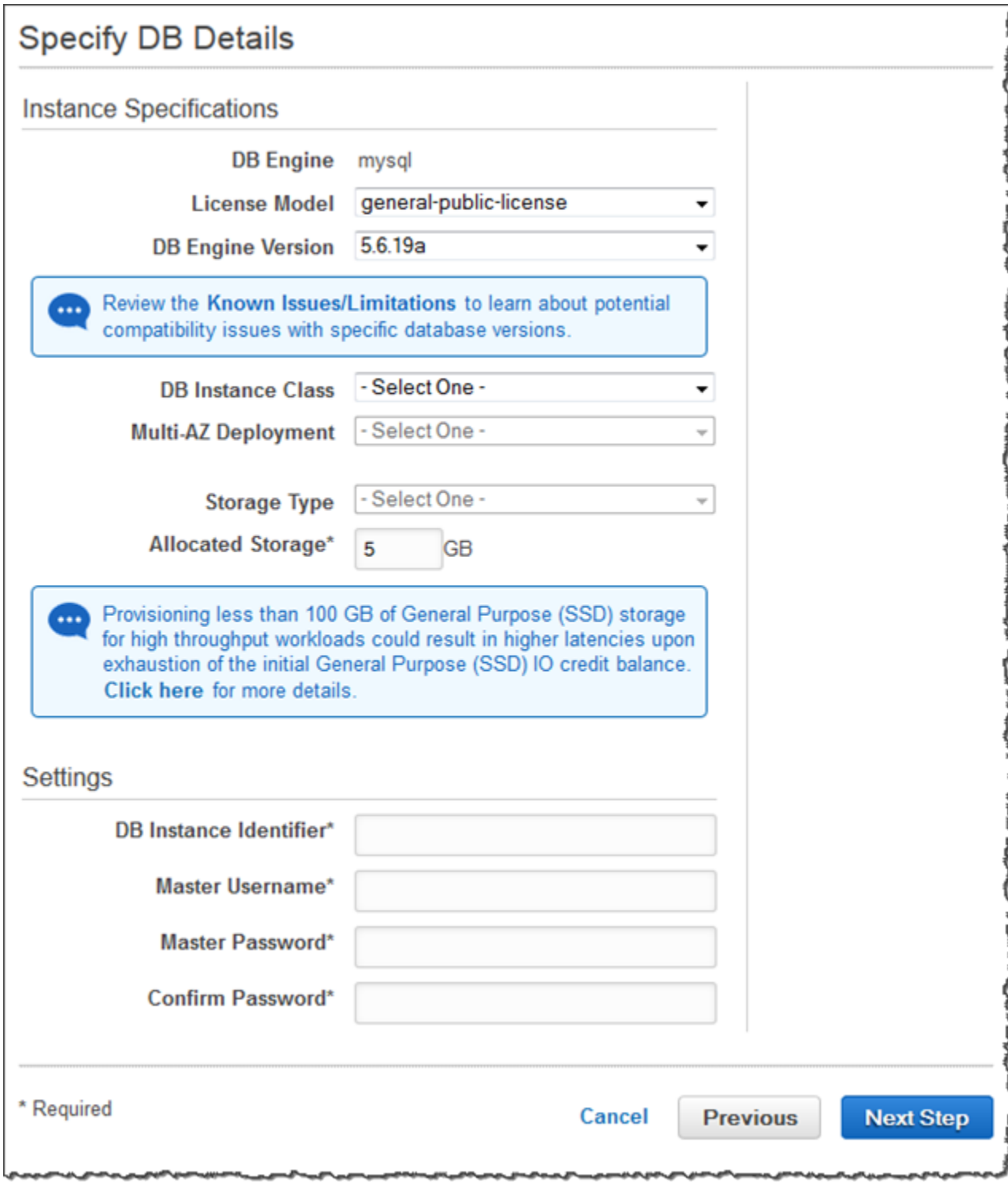
1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the top right corner of the Amazon RDS console, choose the region in which you want to create the DB instance.
3. In the navigation pane, choose **Instances**.
4. Choose **Launch DB Instance**. The **Launch DB Instance Wizard** opens on the **Select Engine** page.



5. On the **Select Engine** page, choose the MySQL icon and then choose **Select** for the MySQL DB engine.
6. On the **Specify DB Details** page, specify your DB instance information. The following table shows settings for an example DB instance. When the settings are as you want them, choose **Next**.

For This Parameter	Do This
License Model	Choose the default, **`general-public-license`**, to use the general license agreement for MySQL. MySQL has only one license model.
DB Engine Version	Choose the default version of MySQL. Amazon RDS supports multiple versions of MySQL in some regions.
DB Instance Class	Choose **`db.m1.small`** for a configuration that equates to 1.7 GB memory, 1 ECU (1 virtual core with 1 ECU), 64-bit platform, and moderate I/O capacity.
Multi-AZ Deployment	Choose **`Yes`** to have a standby replica of your DB instance created in another Availability Zone for failover support. We recommend Multi-AZ for production workloads to maintain high availability. For development and testing, you can choose **`No`**. For more information, see High Availability (Multi-AZ) (p. 99).

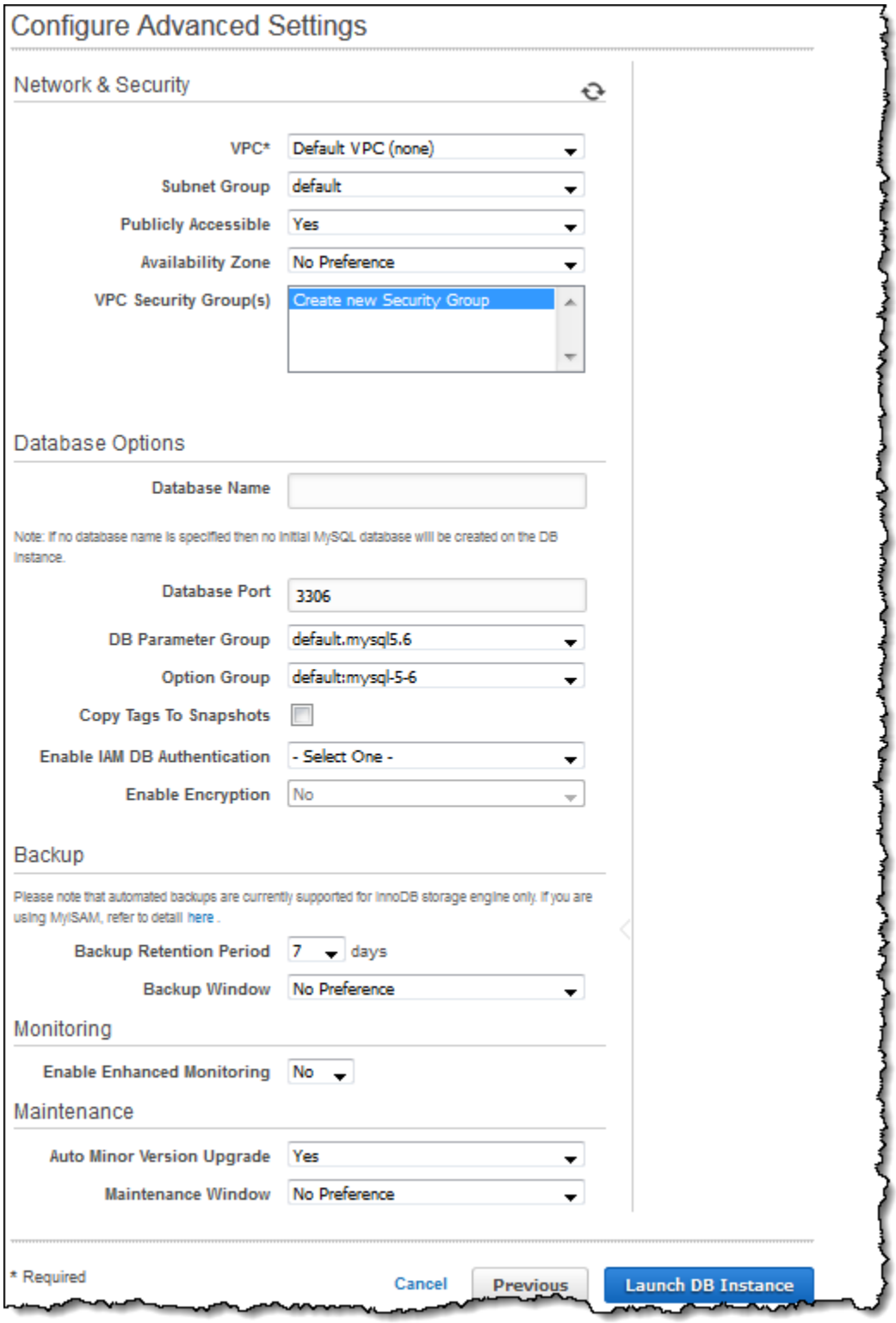
For This Parameter	Do This
Allocated Storage	Type **5** to allocate 5 GB of storage for your database. In some cases, allocating a higher amount of storage for your DB instance than the size of your database can improve I/O performance. For more information about storage allocation, see Amazon Relational Database Service Features.
Storage Type	Choose the storage type **Magnetic**. For more information about storage, see Storage for Amazon RDS (p. 410).
DB Instance Identifier	Type a name for the DB instance that is unique for your account in the region you chose. You can add some intelligence to the name, such as including the region and DB engine you chose, for example **west2-mysql-instance1**.
Master Username	Type a name using alphanumeric characters to use as the master user name to log on to your DB instance. This is the user name you use to log on to your database on the DB instance for the first time.
Master Password and **Confirm Password**	Type a password that contains from 8 to 41 printable ASCII characters (excluding /,", and @) for your master user password. This is the password to use when you use the user name to log on to your database. Then type the password again in the **Confirm Password** box.



7. On the **Configure Advanced Settings** page, provide additional information that RDS needs to launch the MySQL DB instance. The table shows settings for an example DB instance. Specify your DB instance information, then choose **Launch DB Instance**.

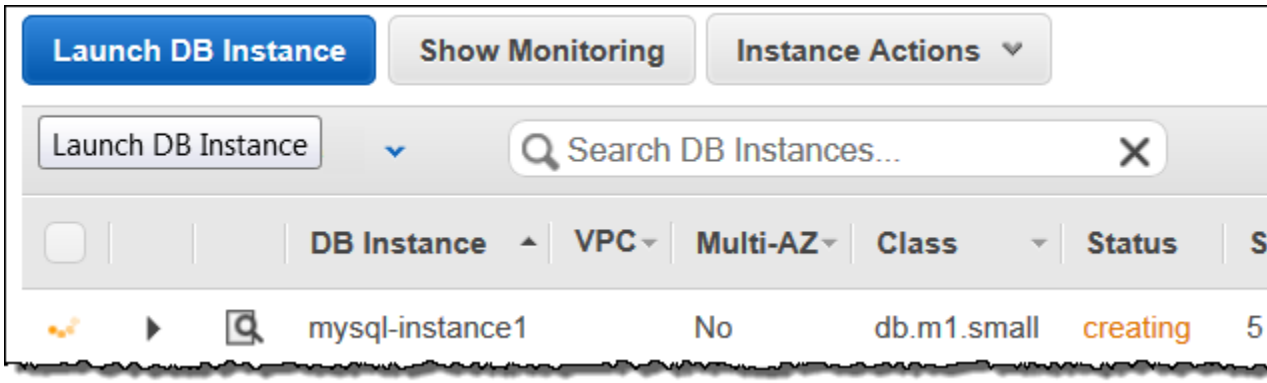
For This Parameter	Do This
VPC	Choose the name of the Amazon Virtual Private Cloud (VPC) to host your MySQL DB instance. If your DB instance isn't hosted in a VPC, choose **Not in VPC**. For more information about VPC, see Amazon Virtual Private Cloud (VPCs) and Amazon RDS (p. 390).
Availability Zone	Determine if you want to specify a particular Availability Zone. If you chose **Yes** for the Multi-AZ Deployment parameter on the previous page, you don't have any options here. For more information about Availability Zones, see Regions and Availability Zones (p. 97).
DB Security Groups	Choose the security group you want to use with this DB instance. For more information about security groups, see Working with DB Security Groups (EC2-Classic Platform) (p. 380).
Database Name	Type a name for your default database that is 1 to 64 alpha-numeric characters. If you don't provide a name, Amazon RDS doesn't automatically create a database on the DB instance you are creating. To create additional databases, connect to the DB instance and use the SQL command CREATE DATABASE. For more information about connecting to the DB instance, see Connecting to a DB Instance Running the MySQL Database Engine (p. 840).
Database Port	Leave the default value of **3306** unless you have a specific port you want to access the database through. MySQL installations default to port 3306.
DB Parameter Group	Leave the default value unless you created your own DB parameter group. For more information about parameter groups, see Working with DB Parameter Groups (p. 170).
Option Group	Choose the default value because this option group is used with the MySQL version you chose on the previous page.
Copy Tags To Snapshots	Choose this option to have any DB instance tags copied to a DB snapshot when you create a snapshot. For more information, see Tagging Amazon RDS Resources (p. 129).
Enable Encryption	Choose **Yes** to enable encryption at rest for this DB instance. For more information, see Encrypting Amazon RDS Resources (p. 355).
Backup Retention Period	Set the number of days you want automatic backups of your database to be retained. For testing purposes, you can set this value to **1**.

For This Parameter	Do This
Backup Window	Unless you have a specific time that you want to have your database backup, use the default of **`No Preference`**.
Enable Enhanced Monitoring	Unless you want to enable gathering metrics in real time for the operating system that your DB instance runs on, use the default of **No**.
Auto Minor Version Upgrade	Choose **`Yes`** to enable your DB instance to receive minor DB engine version upgrades automatically when they become available.
Maintenance Window	Choose the 30-minute window in which pending modifications to your DB instance are applied. If the time period doesn't matter, choose **`No Preference`**.



8. On the RDS console, the new DB instance appears in the list of DB instances. The DB instance has a status of **creating** until the DB instance is created and ready for use. When the state changes to

available, you can connect to a database on the DB instance. Depending on the DB instance class and store allocated, it could take several minutes for the new DB instance to become available.



Connecting to a Database on a DB Instance Running the MySQL Database Engine

Once Amazon RDS provisions your DB instance, you can use any standard SQL client application to connect to a database on the DB instance. In this example, you connect to a database on a MySQL DB instance using MySQL monitor commands. One GUI-based application you can use to connect is MySQL Workbench; for more information, go to the Download MySQL Workbench page. For more information on using MySQL, go to the MySQL documentation.

To connect to a database on a DB instance using MySQL monitor

- Type the following command at a command prompt on a client computer to connect to a database on a MySQL DB instance using the MySQL monitor. Substitute the DNS name for your DB instance for <endpoint>, the master user name you used for <mymasteruser>, and the master password you used for <password>.

```
PROMPT> mysql -h <endpoint> -P 3306 -u <mymasteruser> -p
```

You should see output similar to the following.

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 350
Server version: 5.6.27-log MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Deleting a DB Instance

Once you have connected to the sample DB instance that you created, you should delete the DB instance so you are no longer charged for it.

To delete a DB instance with no final DB snapshot

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.

2. In the **Instances** list, choose the DB instance you wish to delete.
3. Choose **Instance Actions**, and then choose **Delete**.
4. Choose **No** for **Create final Snapshot?**
5. Choose **Yes, Delete**.

Creating an Oracle DB Instance and Connecting to a Database on an Oracle DB Instance

The basic building block of Amazon RDS is the DB instance. Your Amazon RDS DB instance is similar to your on-premises Oracle database.

> **Important**
> You must have an AWS account before you can create a DB instance. If you don't have an AWS account, open https://aws.amazon.com/, and then choose **Create an AWS Account**.

In this topic you create a sample Oracle DB instance. You then connect to the DB instance and run a simple query. Finally you delete the sample DB instance.

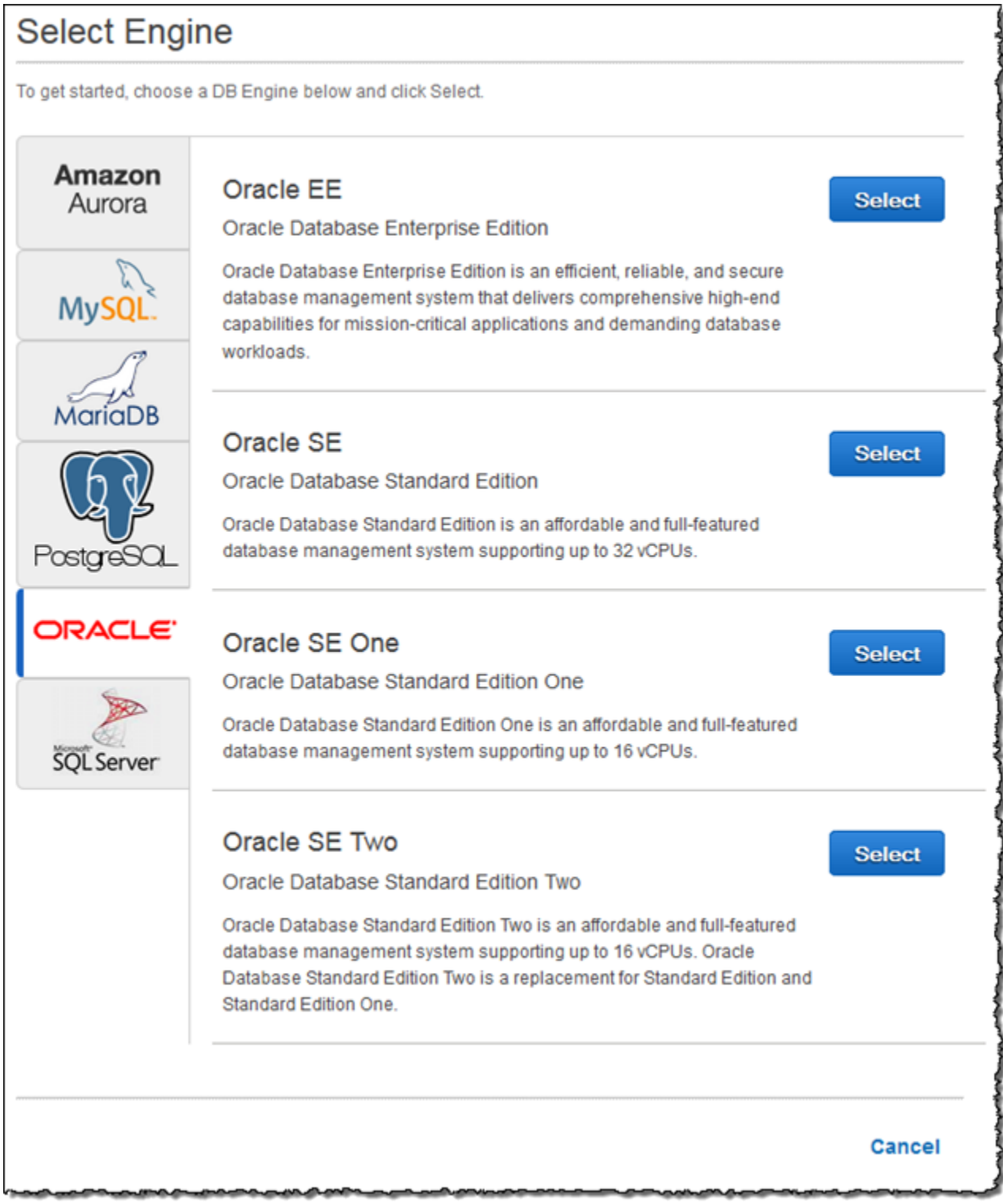
Creating a Sample Oracle DB Instance

In this procedure you use the AWS Management Console to create a sample DB instance. Since you are only creating a sample DB instance, each setting is not fully explained. For a full explanation of each setting, see Creating a DB Instance Running the Oracle Database Engine (p. 949).

To create a DB instance running the Oracle database engine

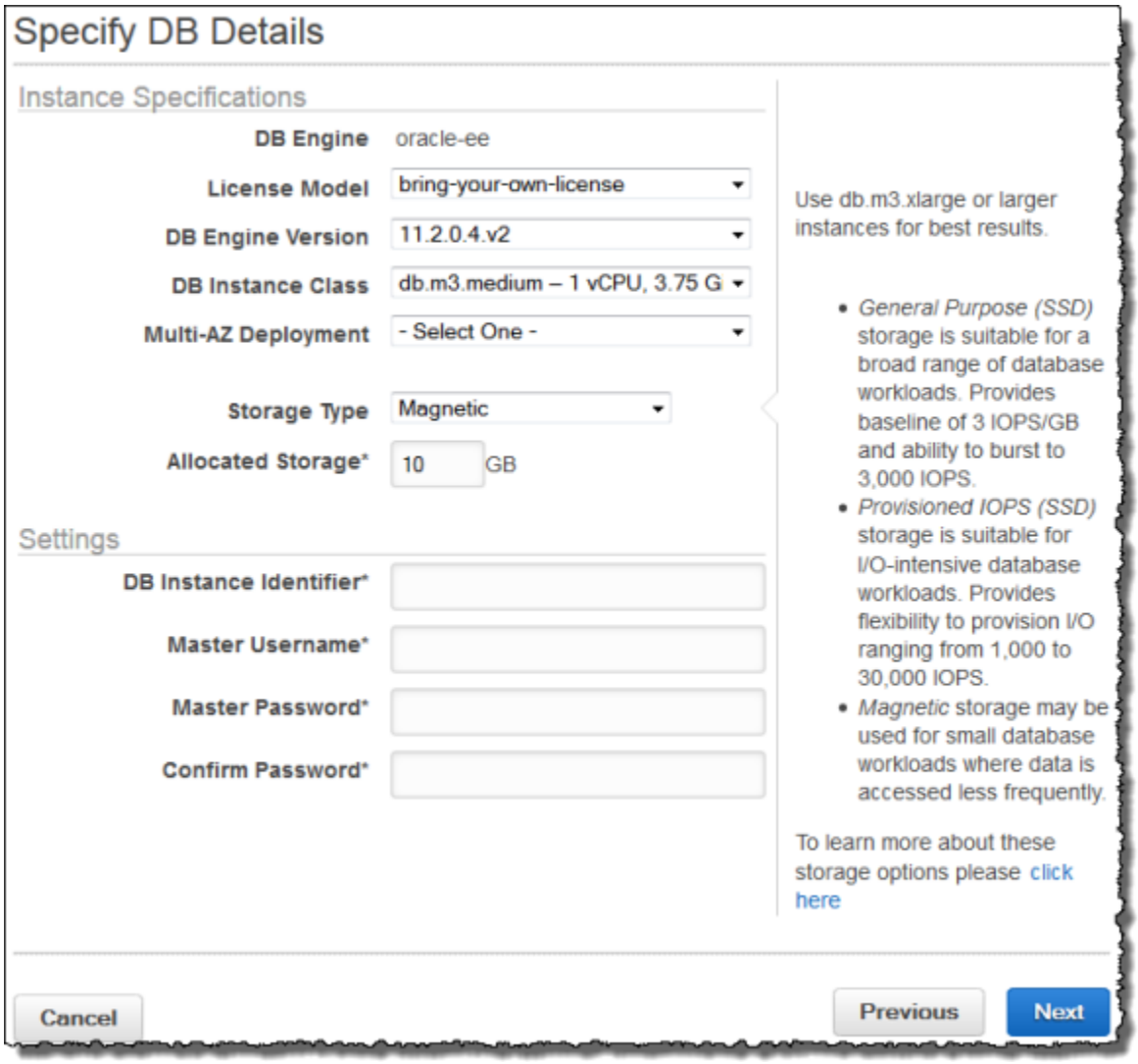
1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the top right corner of the Amazon RDS console, choose the region in which you want to create the DB instance.
3. In the navigation pane, choose **Instances**.
4. Choose **Launch DB Instance**.

   The **Select Engine** page appears.



5. Choose the Oracle icon, and then choose **Select** for the **Oracle SE Two** edition.
6. The **Production?** page asks if you are planning to use the DB instance you are creating for production. Choose **Dev/Test** and then choose **Next Step**.

   The **Specify DB Details** page appears.



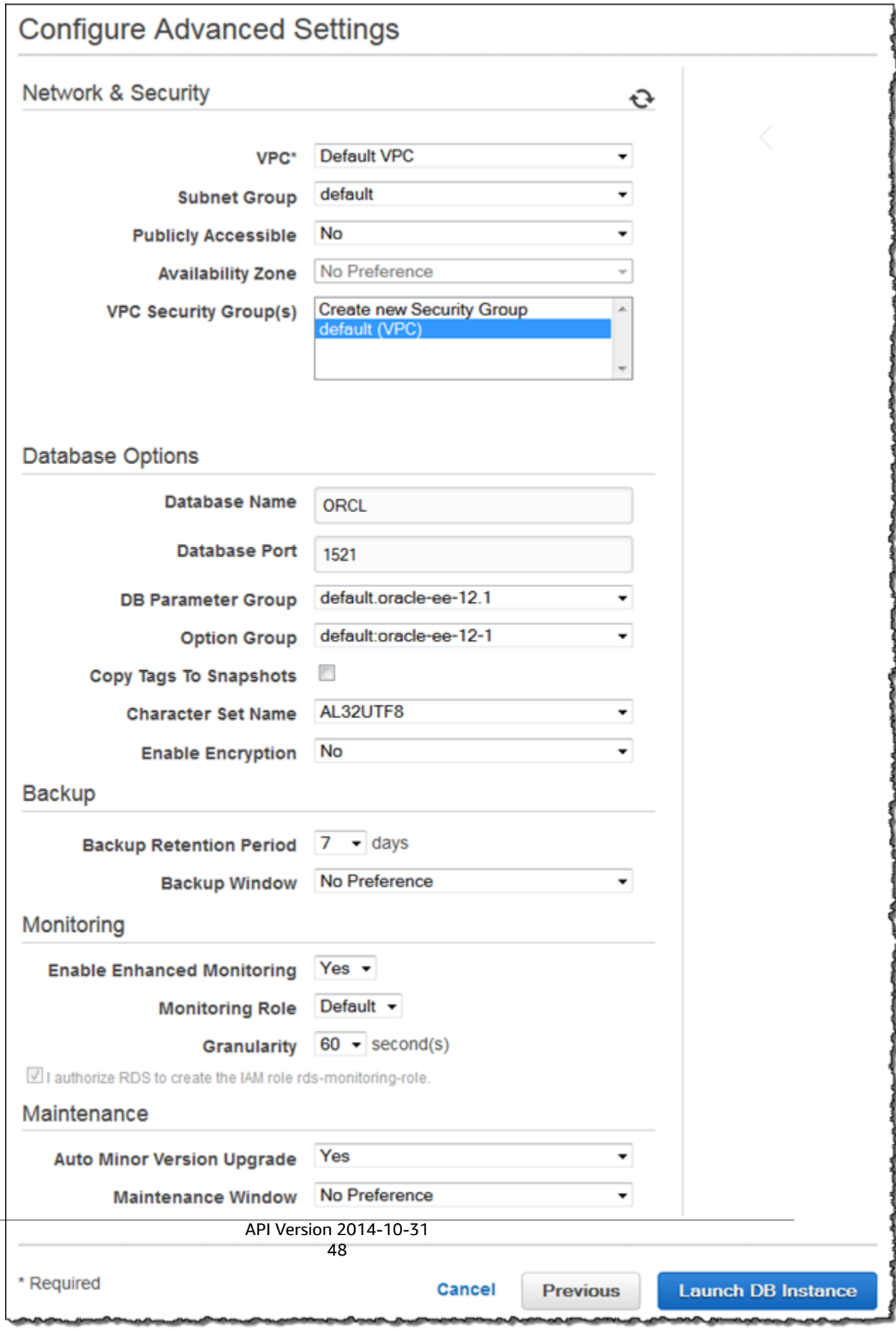
7. On the **Specify DB Details** page, provide the information for your DB instance as shown in the following table:

For This Parameter	Do This
License Model	Choose **license-included** to use the general license agreement for Oracle.
DB Engine Version	Choose the most recent version of Oracle available in the list.
DB Instance Class	Choose **db.t2.small**. This instance class is appropriate for testing.
Multi-AZ Deployment	For development and testing, choose **No**.
Storage Type	Choose the storage type **General Purpose (SSD)**.
Allocated Storage	Type **10** to allocate 10 GB of storage for your database. There is a warning that you should consider allocating

For This Parameter	Do This
	more storage, but since this is a sample DB instance, 10 GB is sufficient.
DB Instance Identifier	Type **`sample-instance`**.
Master Username	Type a name that you will use as the master user name to log on to your DB Instance with all database privileges. The master user name is a SQL Server Authentication login.
Master Password and **Confirm Password**	Type a password for your master user password. It must contain between 8 and 128 printable ASCII characters (excluding /,", and @).

8. Choose **Next** to continue.

   The **Configure Advanced Settings** page appears.

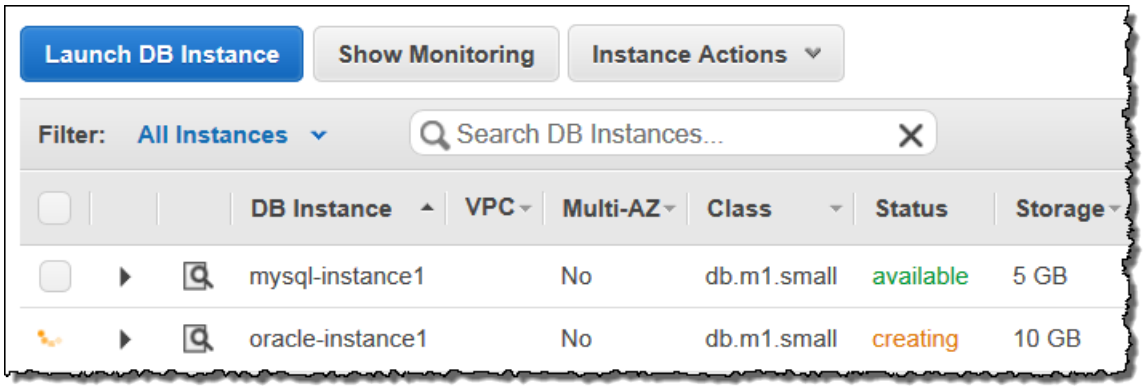

Configure Advanced Settings
Network & Security
VPC*
Default VPC
Subnet Group
default
Publicly Accessible
No
Availability Zone
No Preference
VPC Security Group(s)
Create new Security Group
default (VPC)
Database Options
Database Name
ORCL
Database Port
1521
DB Parameter Group
default.oracle-ee-12.1
Option Group
default:oracle-ee-12-1
Copy Tags To Snapshots
Character Set Name
AL32UTF8
Enable Encryption
No
Backup
Backup Retention Period
7
days
Backup Window
No Preference
Monitoring
Enable Enhanced Monitoring
Yes
Monitoring Role
Default
Granularity
60
second(s)
I authorize RDS to create the IAM role rds-monitoring-role.
Maintenance
Auto Minor Version Upgrade
Yes
Maintenance Window
No Preference
* Required
Cancel
Previous
Launch DB Instance

9. On the **Configure Advanced Settings** page, provide the information for your DB instance as shown in the following table:

For This Parameter	Do This
VPC	Choose **Create new VPC**.
Subnet Group	Choose **Create new DB Subnet Group**.
Publicly Accessible	Choose **Yes**.
Availability Zone	Choose **No Preference**.
VPC Security Group	Choose **Create new Security Group**.
Database Name	Type ORCL
Database Port	Leave the default value of **1521** unless you have a specific port you want to access the database through. Oracle installations default to port 1521, but in some cases a firewall might block this port. If in doubt, ask your network administrator what port you should use.
DB Parameter Group	Leave the default value.
Option Group	Leave the default value.
Copy Tags To Snapshots	Leave this setting unselected.
Character Set Name	Choose the default value of **AL32UTF8** for the Unicode 5.0 UTF-8 Universal character set.
Enable Encryption	Choose **No** to enable encryption at rest for this DB instance.
Backup Retention Period	Choose **7**.
Backup Window	Choose **No Preference**.
Enable Enhanced Monitoring	Choose **No**.
Auto Minor Version Upgrade	Choose **Yes**.
Maintenance Window	Choose **No Preference**.

10. Choose **Launch DB Instance**.
11. Choose **View Your DB Instances**.

    On the RDS console, the new DB instance appears in the list of DB instances. The DB instance has a status of **creating** until the DB instance is ready to use. When the state changes to **available**, you can connect to the DB instance. Depending on the DB instance class and the amount of storage, it can take up to 20 minutes before the new instance is available.

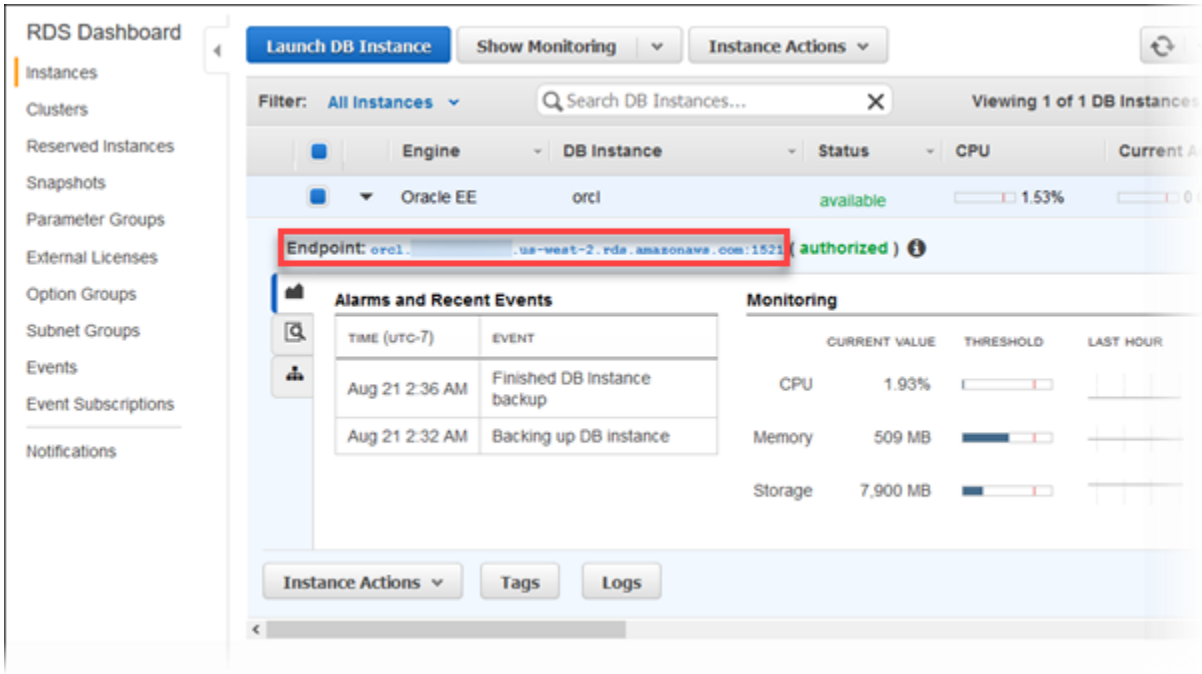


Connecting to Your Sample Oracle DB Instance

After Amazon RDS provisions your DB instance, you can use any standard SQL client application to connect to the instance. In this procedure you connect to your sample DB instance by using the Oracle *sqlplus* command line utility. To download a stand-alone version of this utility, see SQL*Plus User's Guide and Reference.

To connect to a DB Instance using SQL*Plus

1. Find the DNS name and port number for your DB Instance.
   a. Open the RDS console and then choose **Instances** to display a list of your DB instances.
   b. Choose the row for your Oracle DB instance to display the summary information for the instance.



   c. Copy the endpoint. The **Endpoint** field has two parts separated by a colon (:). The part before the colon is the DNS name for the instance, the part following the colon is the port number.
2. Type the following command on one line at a command prompt to connect to your DB instance by using the sqlplus utility. The value for `Host` is the DNS name for your DB instance, the value for

`Port` is the port you assigned the DB instance, and the value for the Oracle `SID` is the name of the DB instance's database that you specified when you created the DB instance, not the name of the DB instance.

```
PROMPT>sqlplus 'mydbusr@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=endpoint DNS name)
 (PORT=1521))(CONNECT_DATA=(SID=ORCL)))'>
```

You should see output similar to the following.

```
SQL*Plus: Release 11.1.0.7.0 - Production on Wed May 25 15:13:59 2011

SQL>
```

Deleting Your Sample DB Instance

Once you are done exploring the sample DB instance that you created, you should delete the DB instance so that you are no longer charged for it.

To delete a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the **Instances** list, choose your sample DB instance.
3. Choose **Instance Actions**, and then choose **Delete**.
4. For **Create final Snapshot**, choose **No**.

   Note
   You should create a final snapshot for any production DB instance that you delete.
5. Choose **Delete**.

Related Topics

- Tutorial: Create an Amazon VPC for Use with an Amazon RDS DB Instance (p. 406)
- Creating a DB Instance Running the Oracle Database Engine (p. 949)
- Connecting to a DB Instance Running the Oracle Database Engine (p. 959)
- Modifying a DB Instance Running the Oracle Database Engine (p. 967)
- Oracle on Amazon RDS (p. 931)

Creating a PostgreSQL DB Instance and Connecting to a Database on a PostgreSQL DB Instance

The easiest way to create a DB instance is to use the RDS console. Once you have created the DB instance, you can use standard SQL client utilities to connect to the DB instance such as the pgAdmin utility. In this example, you create a DB instance running the PostgreSQL database engine called west2-postgres1, with a db.m1.small DB instance class, 10 GB of storage, and automated backups enabled with a retention period of one day.

Important
You must complete the tasks in the Setting Up for Amazon RDS (p. 5) section before you can create or connect to a DB instance.

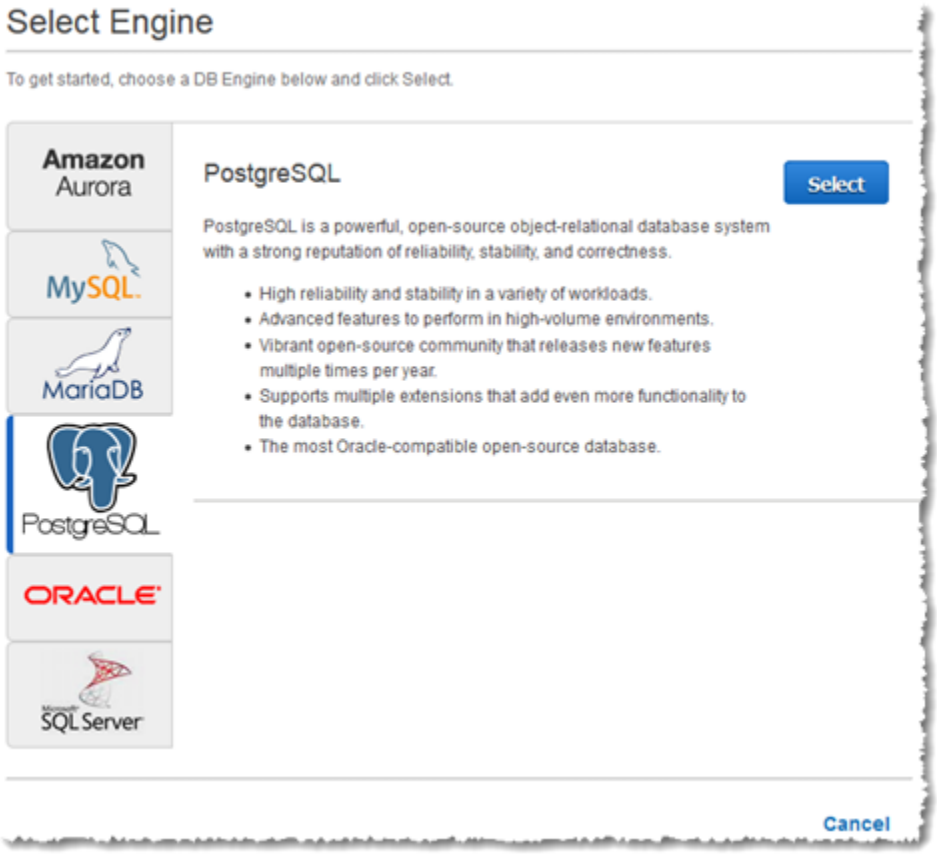
Topics

Creating a PostgreSQL DB Instance

To create a DB Instance Running the PostgreSQL DB Engine

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the top right corner of the AWS Management Console, choose the region in which you want to create the DB instance.
3. In the navigation pane, choose **Instances**.
4. Choose **Launch DB Instance** to start the **Launch DB Instance Wizard**.

   The wizard opens on the **Select Engine** page.

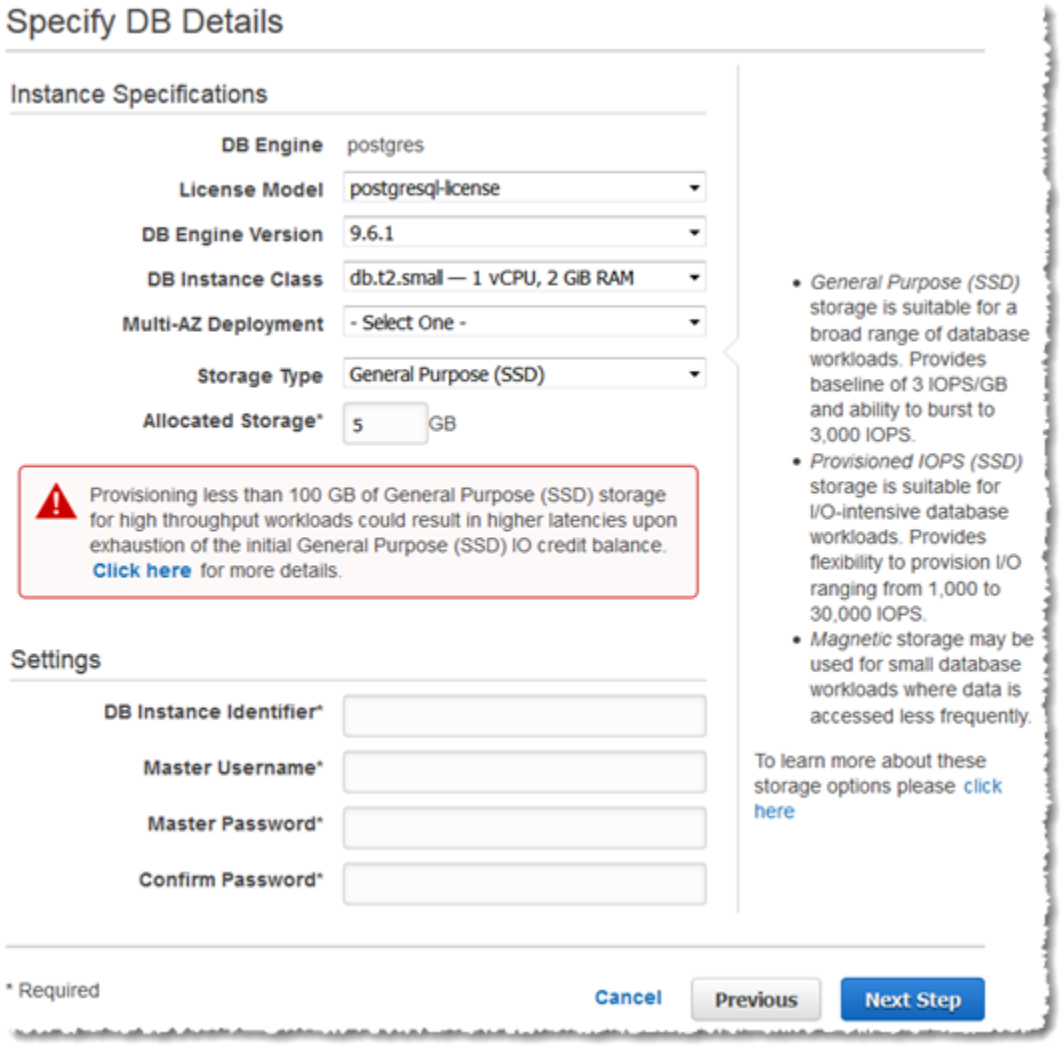


5. On the **Select Engine** page, choose the PostgreSQL icon, and then choose **Select**.
6. Next, the **Production?** page asks if you are planning to use the DB instance you are creating for production. If you are, choose **PostgreSQL** under **Production**. If you choose this option, the failover

option **Multi-AZ** and the **Provisioned IOPS** storage options are preselected in the following step. Choose **Next Step** when you are finished.

7. On the **Specify DB Details** page, specify your DB instance information. Choose **Next Step** when you are finished.

For This Parameter	Do This
License Model	PostgreSQL has only one license model. Choose **`postgresql-license`** to use the general license agreement for PostgreSQL.
DB Engine Version	Choose the version of PostgreSQL you want to use.
DB Instance Class	Choose **`db.t2.small`** for a configuration that equates to 2 GB memory, 1 ECU (1 virtual core with 1 ECU), 64-bit platform, and moderate I/O capacity. For more information about all the DB instance class options, see DB Instance Class (p. 92).
Multi-AZ Deployment	Choose **`Yes`** to have a standby replica of your DB instance created in another Availability Zone for failover support. We recommend Multi-AZ for production workloads to maintain high availability. For development and testing, you can choose **`No`**. For more information, see High Availability (Multi-AZ) (p. 99).
Storage Type	Choose the storage type **`General Purpose (SSD)`**. For more information about storage, see Storage for Amazon RDS (p. 410).
Allocated Storage	Type **`5`** to allocate 5 GB of storage for your database. In some cases, allocating a higher amount of storage for your DB instance than the size of your database can improve I/O performance. For more information about storage allocation, see Amazon Relational Database Service Features.
DB Instance Identifier	Type a name for the DB instance that is unique for your account in the region you chose. You can add some intelligence to the name, such as including the region and DB engine you chose, for example **`postgreSQL-test`**.
Master Username	Type a name using alphanumeric characters to use as the master user name to log on to your DB instance. For information on the default privileges granted to the master user name, see Amazon RDS PostgreSQL Planning Information (p. 1147)
Master Password and **Confirm Password**	Type a password that contains from 8 to 128 printable ASCII characters (excluding /,", and @) for your master password, then type the password again in the **Confirm Password** box.

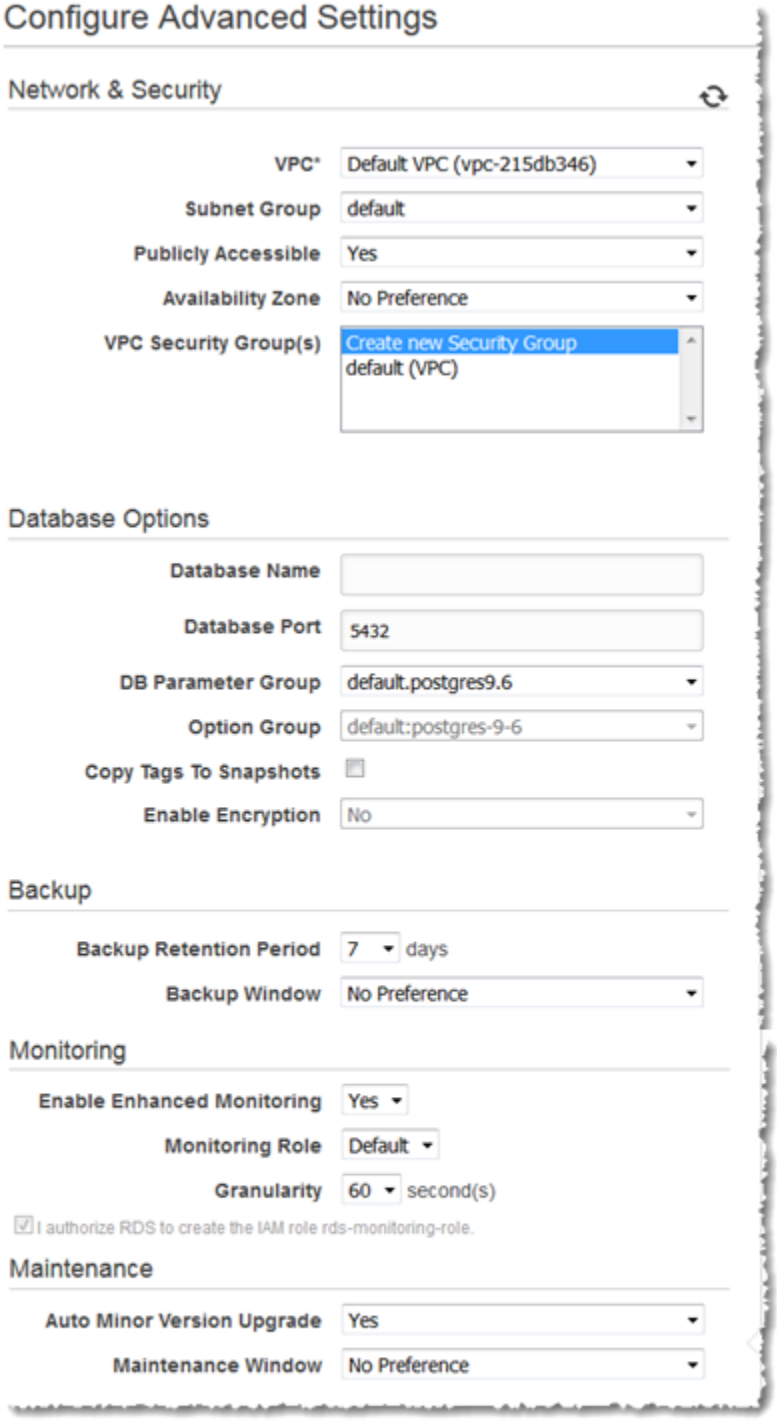


8. On the **Configure Advanced Settings** page, provide additional information that RDS needs to launch the PostgreSQL DB instance. The table shows settings for an example DB instance. Specify your DB instance information, then choose **Launch DB Instance**.

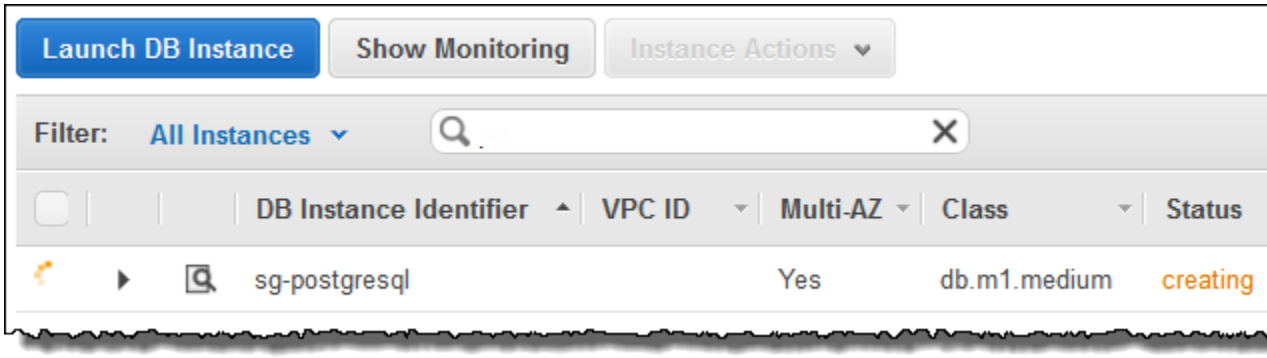
For This Parameter	Do This
VPC	This setting depends on the platform you are on. If you are a new customer to AWS, choose the default VPC shown. If you are creating a DB instance on the previous E2-Classic platform that does not use a VPC, choose **`Not in VPC`**. For more information about VPC, see Amazon Virtual Private Cloud (VPCs) and Amazon RDS (p. 390).
Subnet Group	This setting depends on the platform you are on. If you are a new customer to AWS, choose **`default`**, which is the default DB subnet group that was created for your account. If you are creating a DB instance on the previous E2-Classic platform and you want your DB instance in a

For This Parameter	Do This
	specific VPC, choose the DB subnet group you created for that VPC. For more information about VPC, see Amazon Virtual Private Cloud (VPCs) and Amazon RDS (p. 390).
Publicly Accessible	Choose **`Yes`** to give the DB instance a public IP address, meaning that it is accessible outside the VPC; otherwise, choose **`No`**, so the DB instance is only accessible from inside the VPC. For more information about hiding DB instances from public access, see Hiding a DB Instance in a VPC from the Internet (p. 401).
Availability Zone	Use the default value of **`No Preference`** unless you want to specify an Availability Zone.
VPC Security Group	If you are a new customer to AWS, choose the default VPC. If you created a VPC security group, choose the VPC security group you previously created.
Database Name	Type a name for your database of up to 63 alpha-numeric characters. If you do not provide a name, the default "postgres" database is created. To create additional databases, connect to the DB instance and use the SQL command CREATE DATABASE. For more information about connecting to the DB instance, see Connecting to a DB Instance Running the PostgreSQL Database Engine (p. 1179).
Database Port	Specify a port you want to use to access the database. PostgreSQL installations default to port 5432.
DB Parameter Group	Use the default value unless you have created your own parameter group.
Option Group	Use the default value unless you have created your own option group.
Copy Tags To Snapshots	Choose this option to have any DB instance tags copied to a DB snapshot when you create a snapshot. For more information, see Tagging Amazon RDS Resources (p. 129).
Enable Encryption	Choose **`Yes`** to enable encryption at rest for this DB instance. For more information, see Encrypting Amazon RDS Resources (p. 355).
Backup Retention Period	Set the number of days you want automatic backups of your database to be retained. For testing purposes, you can set this value to **`1`**.
Backup Window	Unless you have a specific time that you want to have your database backup, use the default of **`No Preference`**.

For This Parameter	Do This
Enable Enhanced Monitoring	Choose **`Yes`** to enable real-time OS monitoring. Amazon RDS provides metrics in real time for the operating system (OS) that your DB instance runs on. You are only charged for Enhanced Monitoring that exceeds the free tier provided by Amazon CloudWatch Logs.
Monitoring Role	Choose **`Default`** to use the default IAM role.
Granularity	Choose **`60`** to monitor the instance every minute.
Auto Minor Version Upgrade	Choose **`Yes`** to enable your DB instance to receive minor DB engine version upgrades automatically when they become available.
Maintenance Window	Choose the 30-minute window in which pending modifications to your DB instance are applied. If the time period doesn't matter, choose **`No Preference`**.


Configure Advanced Settings
Network & Security
VPC*
Default VPC (vpc-215db346)
Subnet Group
default
Publicly Accessible
Yes
Availability Zone
No Preference
VPC Security Group(s)
Create new Security Group
default (VPC)
Database Options
Database Name
Database Port
5432
DB Parameter Group
default.postgres9.6
Option Group
default:postgres-9-6
Copy Tags To Snapshots
Enable Encryption
No
Backup
Backup Retention Period
7
days
Backup Window
No Preference
Monitoring
Enable Enhanced Monitoring
Yes
Monitoring Role
Default
Granularity
60
second(s)
I authorize RDS to create the IAM role rds-monitoring-role.
Maintenance
Auto Minor Version Upgrade
Yes
Maintenance Window
No Preference

9. On the final page of the wizard, choose **View Your DB Instances**.
10. On the Amazon RDS console, the new DB instance appears in the list of DB instances. The DB instance has a status of **creating** until the DB instance is created and ready for use. When the state changes to **available**, you can connect to the DB instance. Depending on the DB instance class and store allocated, it could take several minutes for the new instance to be available.



Connecting to a PostgreSQL DB Instance

After Amazon RDS provisions your DB instance, you can use any standard SQL client application to connect to the instance. It is important to note that the security group you assigned to the DB instance when you created it must allow access to the DB instance. If you have difficulty connecting to the DB instance, the problem is most often with the access rules you set up in the security group you assigned to the DB instance.

This section shows two ways to connect to a PostgreSQL DB instance. The first example uses *pgAdmin*, a popular Open Source administration and development tool for PostgreSQL. You can download and use *pgAdmin* without having a local instance of PostgreSQL on your client computer. The second example uses *psql*, a command line utility that is part of a PostgreSQL installation. To use *psql*, you must have a PostgreSQL installed on your client computer or have installed the *psql* client on your machine.

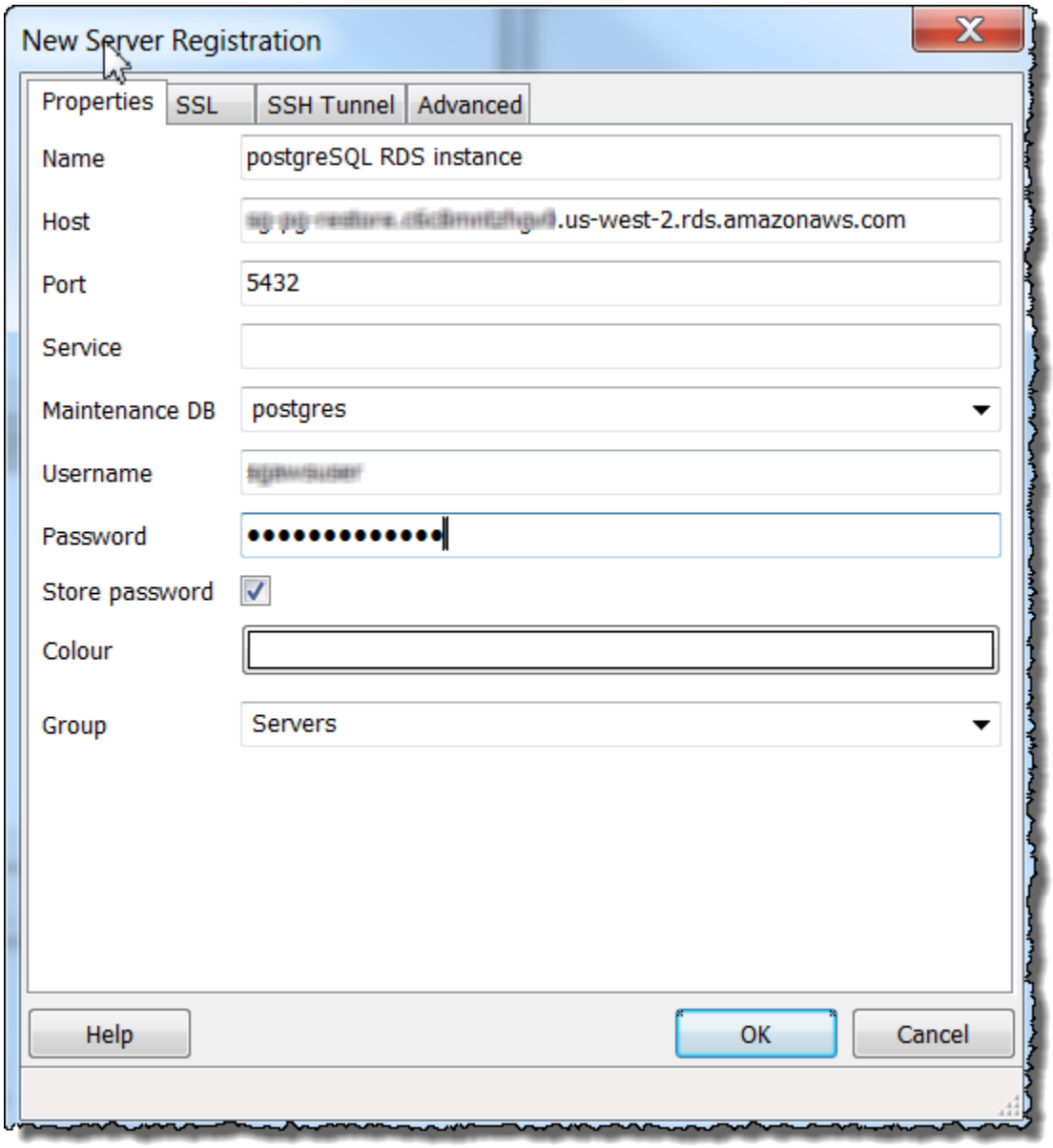
In this example, you connect to a PostgreSQL DB instance using pgAdmin.

Using pgAdmin to Connect to a PostgreSQL DB Instance

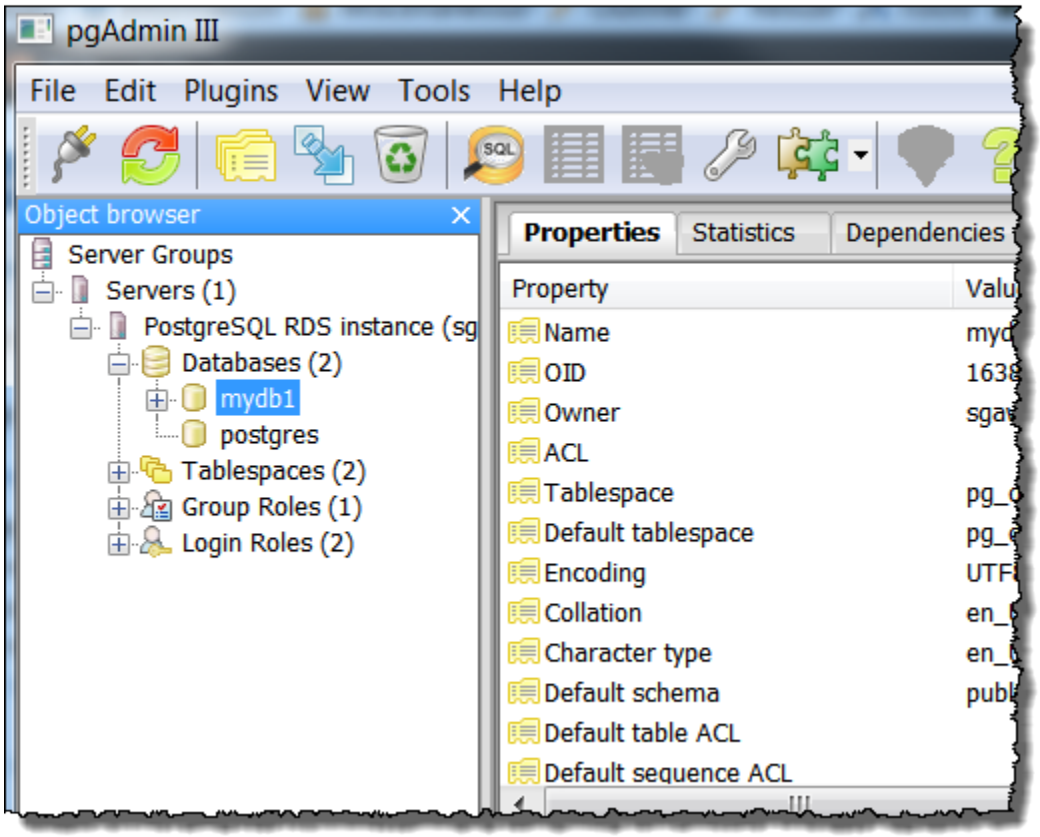
To connect to a PostgreSQL DB instance using pgAdmin

1. Launch the *pgAdmin* application on your client computer. You can install *pgAdmin* from http://www.pgadmin.org/.
2. Choose **Add Server** from the **File** menu.
3. In the **New Server Registration** dialog box, enter the DB instance endpoint (for example, mypostgresql.c6c8dntfzzhgv0.us-west-2.rds.amazonaws.com) in the **Host** box. Do not include the colon or port number as shown on the Amazon RDS console (mypostgresql.c6c8dntfzzhgv0.us-west-2.rds.amazonaws.com:*5432*).

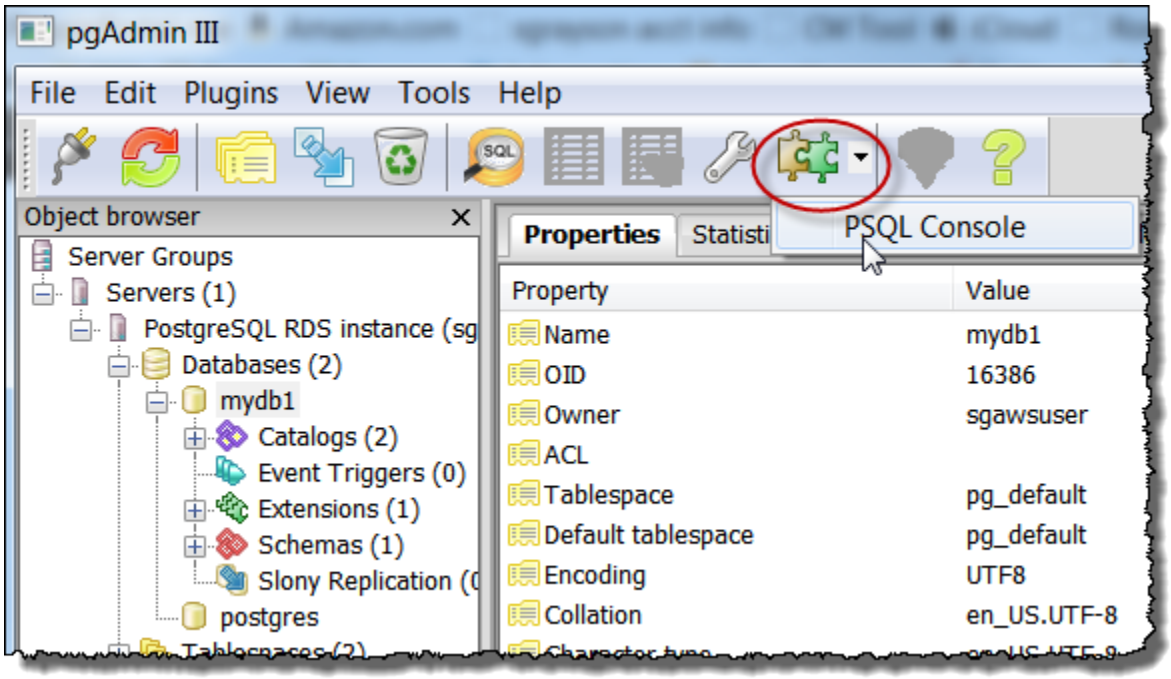
   Enter the port you assigned to the DB instance into the **Port** box. Enter the user name and user password you entered when you created the DB instance into the **Username** and **Password** boxes, respectively.



4. Choose **OK**.
5. In the **Object browser**, expand the **Server Groups**. Choose the Server (the DB instance) you created, and then choose the database name.



6. Choose the plugin icon and choose **PSQL Console**. The *psql* command window opens for the default database you created.



7. Use the command window to enter SQL or *psql* commands. Type `\q` to close the window.

Using *psql* to Connect to a PostgreSQL DB Instance

If your client computer has PostgreSQL installed, you can use a local instance of *psql* to connect to a PostgreSQL DB instance. To connect to your PostgreSQL DB instance using *psql*, you need to provide host information and access credentials.

The following format is used to connect to a PostgreSQL DB instance on Amazon RDS:

```
psql --host=<DB instance endpoint> --port=<port> --username=<master user name> --password
 --dbname=<database name>
```

For example, the following command connects to a database called `mypgdb` on a PostgreSQL DB instance called `mypostgresql` using fictitious credentials:

```
psql --host=mypostgresql.c6c8mwvfdgv0.us-west-2.rds.amazonaws.com --port=5432 --
username=awsuser --password --dbname=mypgdb
```

Troubleshooting Connection Issues

By far the most common problem that occurs when attempting to connect to a database on a DB instance is the access rules in the security group assigned to the DB instance. If you used the default DB security group when you created the DB instance, chances are good that the security group did not have the rules that allow you to access the instance. For more information about Amazon RDS security groups, see Amazon RDS Security Groups (p. 375)

The most common error is *could not connect to server: Connection timed out*. If you receive this error, check that the host name is the DB instance endpoint and that the port number is correct. Check that the security group assigned to the DB instance has the necessary rules to allow access through any firewall your connection may be going through.

Deleting a DB Instance

Once you have connected to the sample DB instance that you created, you should delete the DB instance so you are no longer charged for it.

To delete a DB instance with no final DB snapshot

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the **Instances** list, choose the DB instance you wish to delete.
3. Choose **Instance Actions**, and then choose **Delete**.
4. Choose **No** for **Create final Snapshot?**
5. Choose **Yes, Delete**.

Tutorial: Create a Web Server and an Amazon RDS Database

This tutorial helps you install an Apache web server with PHP, and create a MySQL database. The web server runs on an Amazon EC2 instance using Amazon Linux, and the MySQL database is an Amazon RDS MySQL DB instance. Both the Amazon EC2 instance and the Amazon RDS DB instance run in a VPC based in Amazon Virtual Private Cloud service (Amazon VPC).

> **Note**
> This tutorial works with Amazon Linux and might not work for other versions of Linux such as Ubuntu.

Before you begin this tutorial, you must have a VPC with both public and private subnets, and corresponding security groups. If you don't have these, complete the following tasks in Tutorial: Create an Amazon VPC for Use with an Amazon RDS DB Instance (p. 406):

- Create a VPC with Private and Public Subnets (p. 406)
- Create Additional Subnets (p. 407)
- Create a VPC Security Group for a Public Web Server (p. 408)
- Create a VPC Security Group for a Private Amazon RDS DB Instance (p. 409)

In this tutorial, you perform the following procedures:

- Step 1: Create an RDS DB Instance (p. 62)
- Step 2: Create an EC2 Instance and Install a Web Server (p. 66)

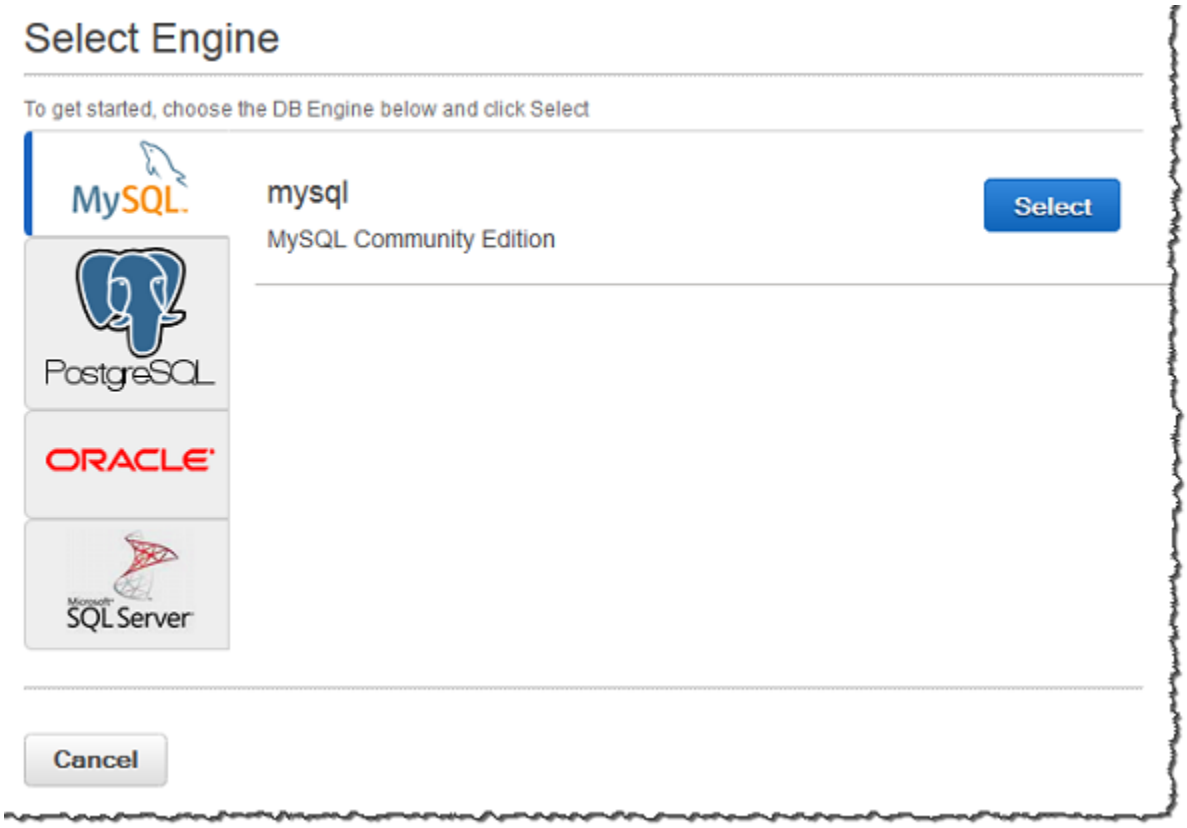
Step 1: Create an RDS DB Instance

In this step you create an Amazon RDS MySQL DB instance that maintains the data used by a web application.

> **Important**
> Before you begin this step, you must have a VPC with both public and private subnets, and corresponding security groups. If you don't have these, see Tutorial: Create an Amazon VPC for Use with an Amazon RDS DB Instance (p. 406). Complete the steps in Create a VPC with Private and Public Subnets (p. 406), Create Additional Subnets (p. 407), Create a VPC Security Group for a Public Web Server (p. 408), and Create a VPC Security Group for a Private Amazon RDS DB Instance (p. 409).

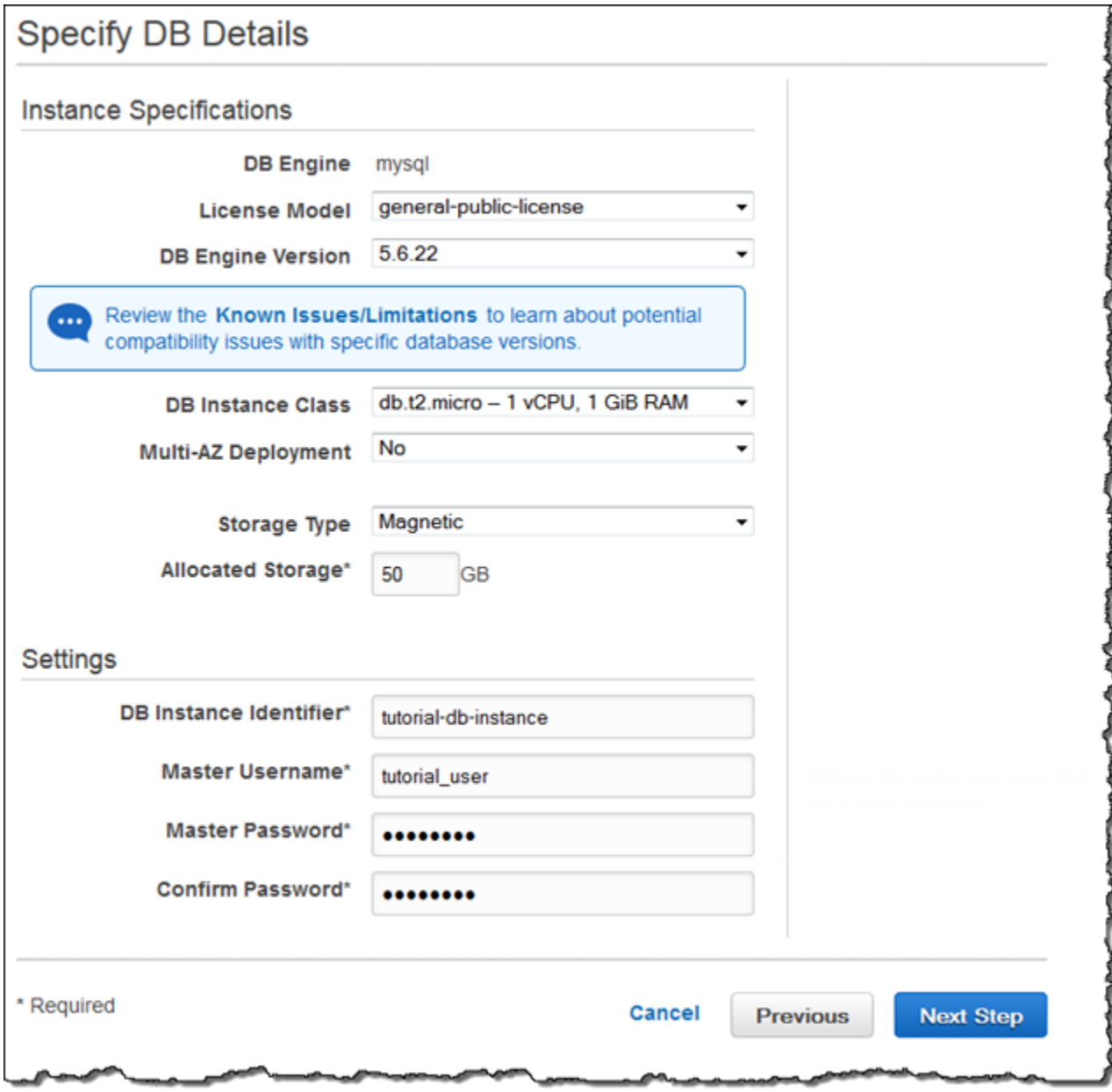
To launch a MySQL DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the top-right corner of the AWS Management Console, choose the region in which you want to create the DB instance. This example uses the US West (Oregon) region.
3. Choose **Instances**.
4. Choose **Launch DB Instance**.
5. On the **Select Engine** page, shown following, choose the MySQL DB engine, and then choose **Select**.



6. On the **Production** page, below **Dev/Test**, choose **MySQL This instance is intended for use outside of production**, and then choose **Next Step**.
7. On the **Specify DB Details** page, shown following, set these values:

   - **DB Engine Version:** Use the default value.
   - **DB Instance Class:** `db.t2.micro`
   - **Multi-AZ Deployment:** `No`
   - **Storage Type:** `General Purpose (SSD)`
   - **Allocated Storage:** `50 GB`
   - **DB Instance Identifier:** `tutorial-db-instance`
   - **Master Username:** `tutorial_user`
   - **Master Password:** Choose a password.
   - **Confirm Password:** Retype the password.



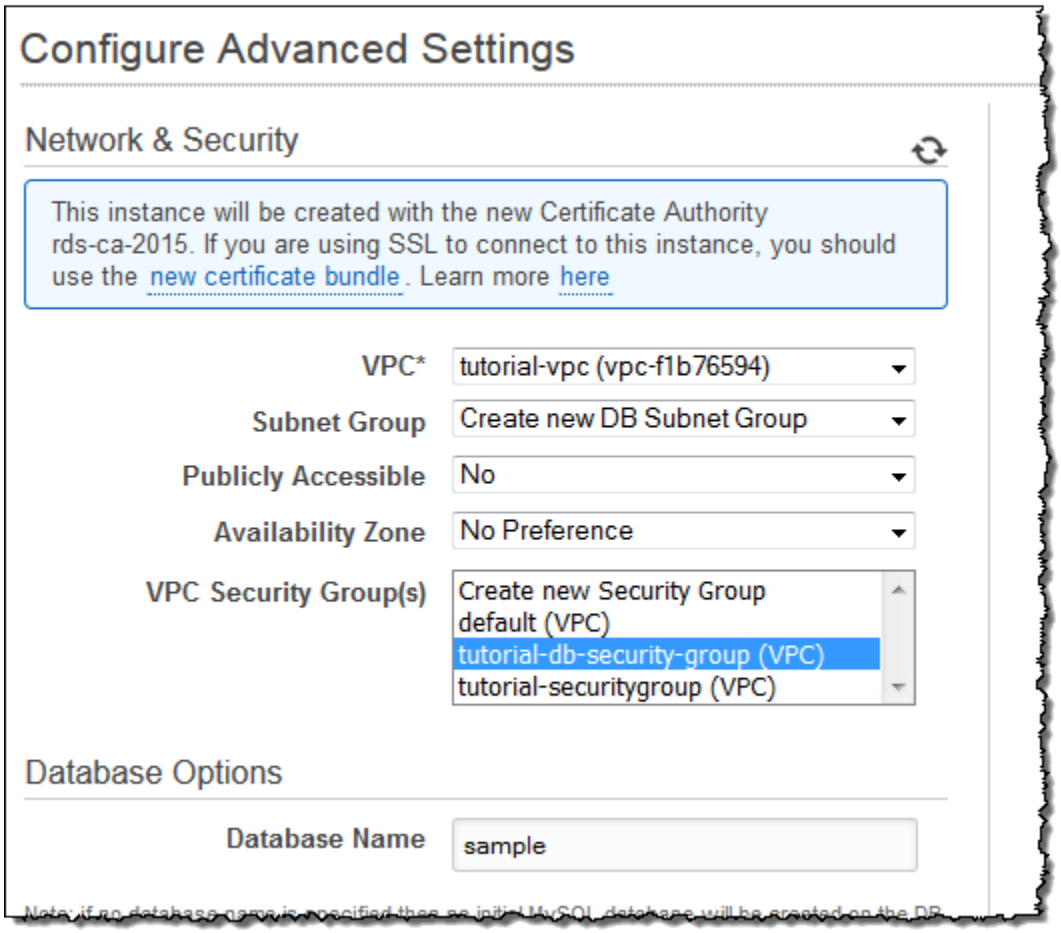
8. Choose **Next Step** and set the following values in the **Configure Advanced Settings** page, shown following:

   - **VPC:** Choose an existing VPC with both public and private subnets, such as the `tutorial-vpc` (vpc-*identifier*) created in Create a VPC with Private and Public Subnets (p. 406)

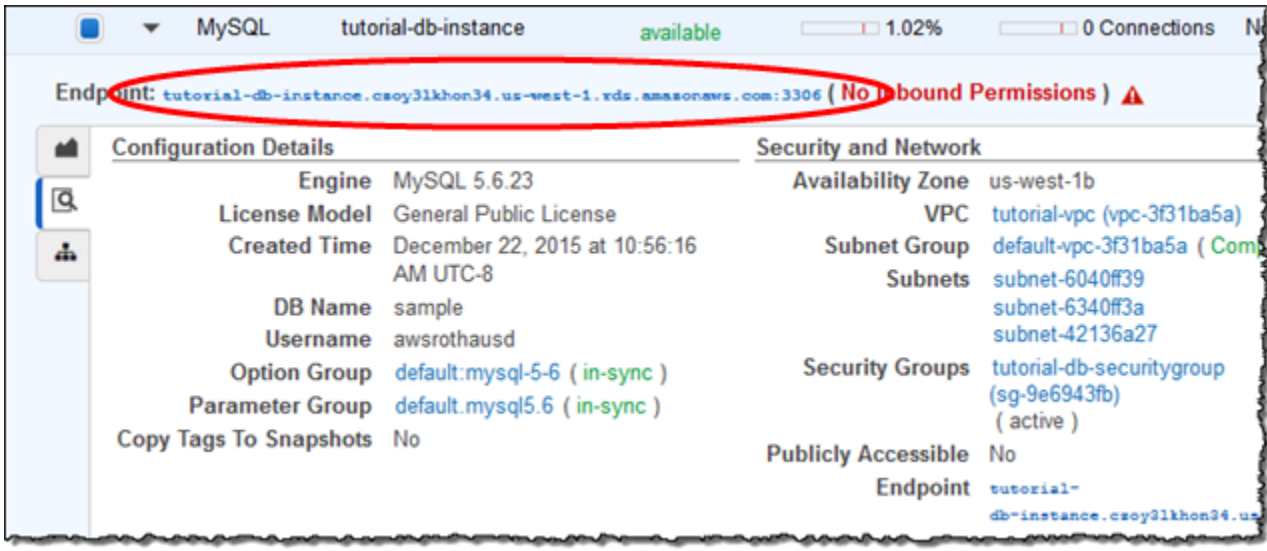
     Note
     The VPC must have subnets in different availability zones.

   - **Subnet group:** `Create a new DB Subnet Group`
   - **Publicly Accessible:** `No`
   - **Availability Zone:** `No Preference`

- **VPC Security Group(s):** Choose an existing security group that is configured for private access, such as the `tutorial-db-securitygroup` created in Create a VPC Security Group for a Private Amazon RDS DB Instance (p. 409)
- **Database Name:** `sample`



9. To create your Amazon RDS MySQL DB instance, choose **Launch DB Instance**.
10. On the next page, choose **View Your DB Instances** to view your RDS MySQL DB instance.
11. Wait for the status of your new DB instance to show as `available`. Then choose the selection box to the left of your DB instance to display the DB instance details, shown following.



Make note of the endpoint for your DB instance. This endpoint shows the server name and port that you use to connect your web server to your RDS DB instance.

To make sure your RDS MySQL DB instance is as secure as possible, verify that sources outside of the VPC cannot connect to your RDS MySQL DB instance.

Next Step

Step 2: Create an EC2 Instance and Install a Web Server (p. 66)

Step 2: Create an EC2 Instance and Install a Web Server

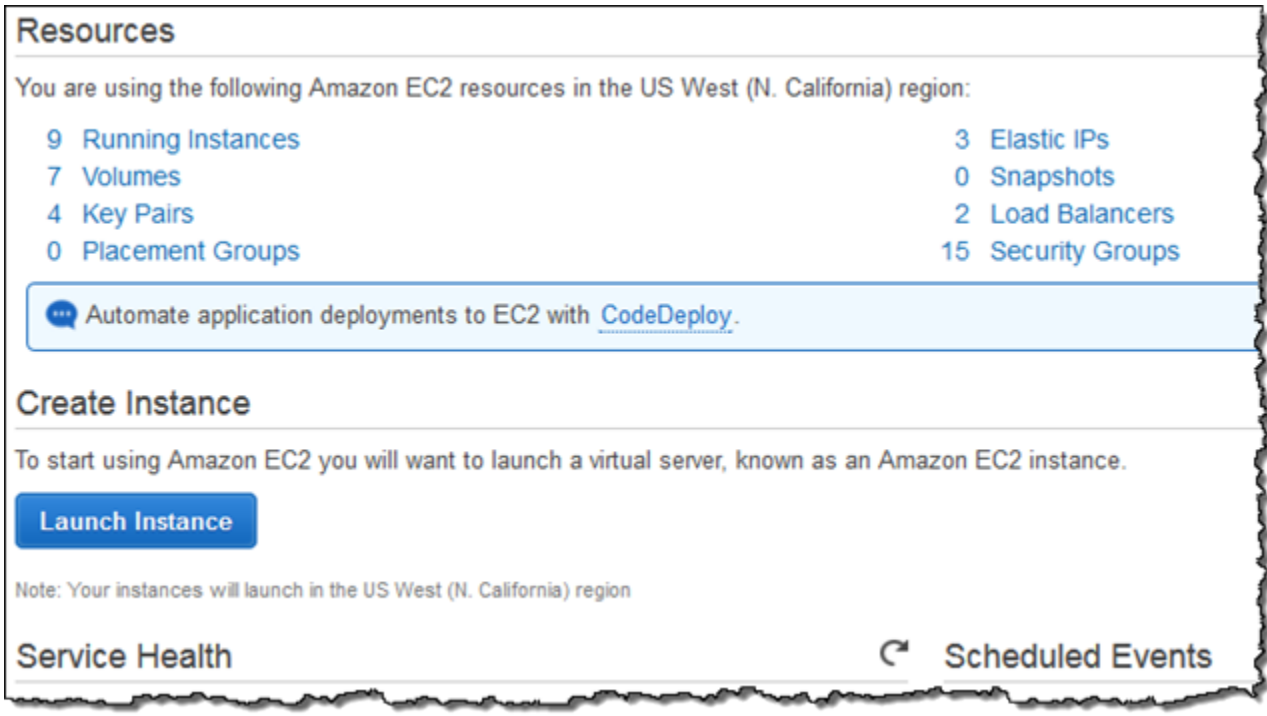
In this step you create a web server to connect to the Amazon RDS DB instance that you created in Step 1: Create an RDS DB Instance (p. 62).

Launch an EC2 Instance

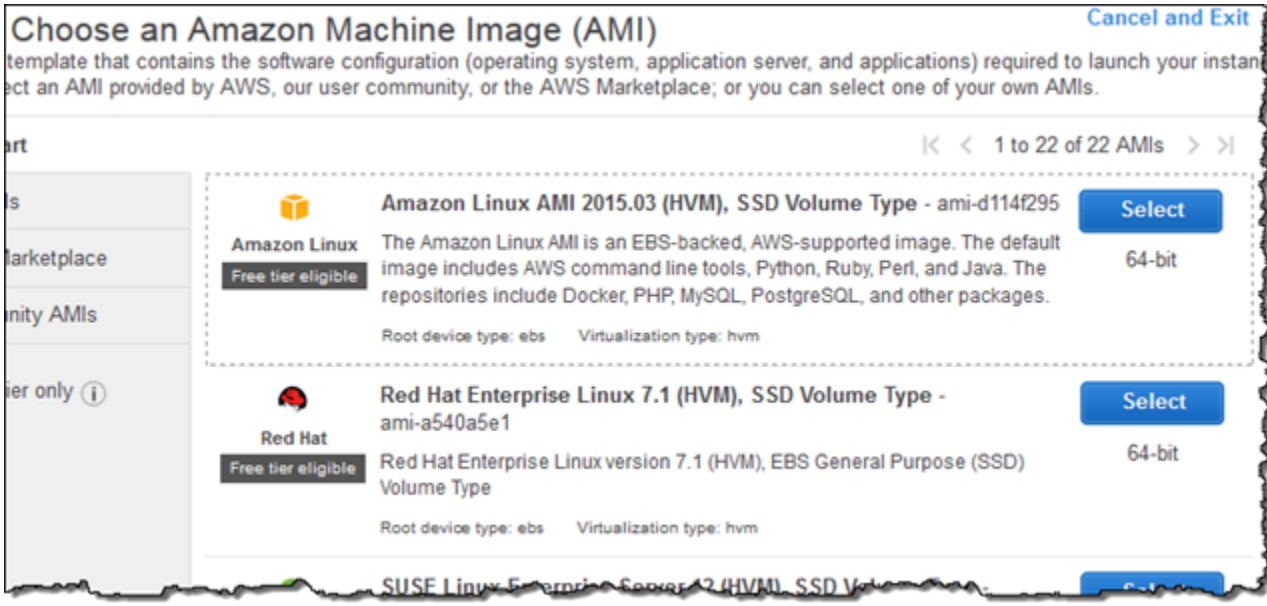
First you create an Amazon EC2 instance in the public subnet of your VPC.

To launch an EC2 instance

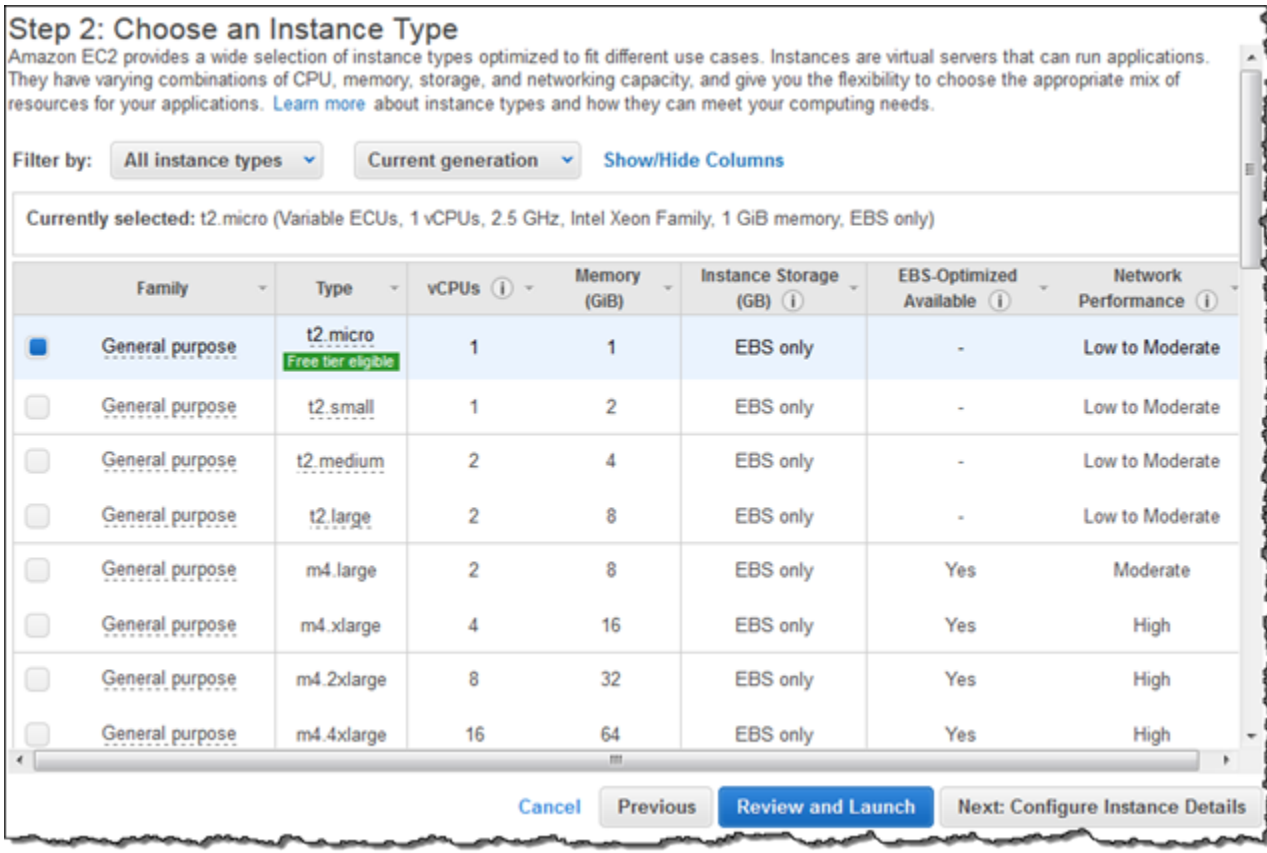
1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Choose **EC2 Dashboard**, and then choose **Launch Instance**, as shown following.



3. Choose the **Amazon Linux** Amazon Machine Image (AMI), as shown following.

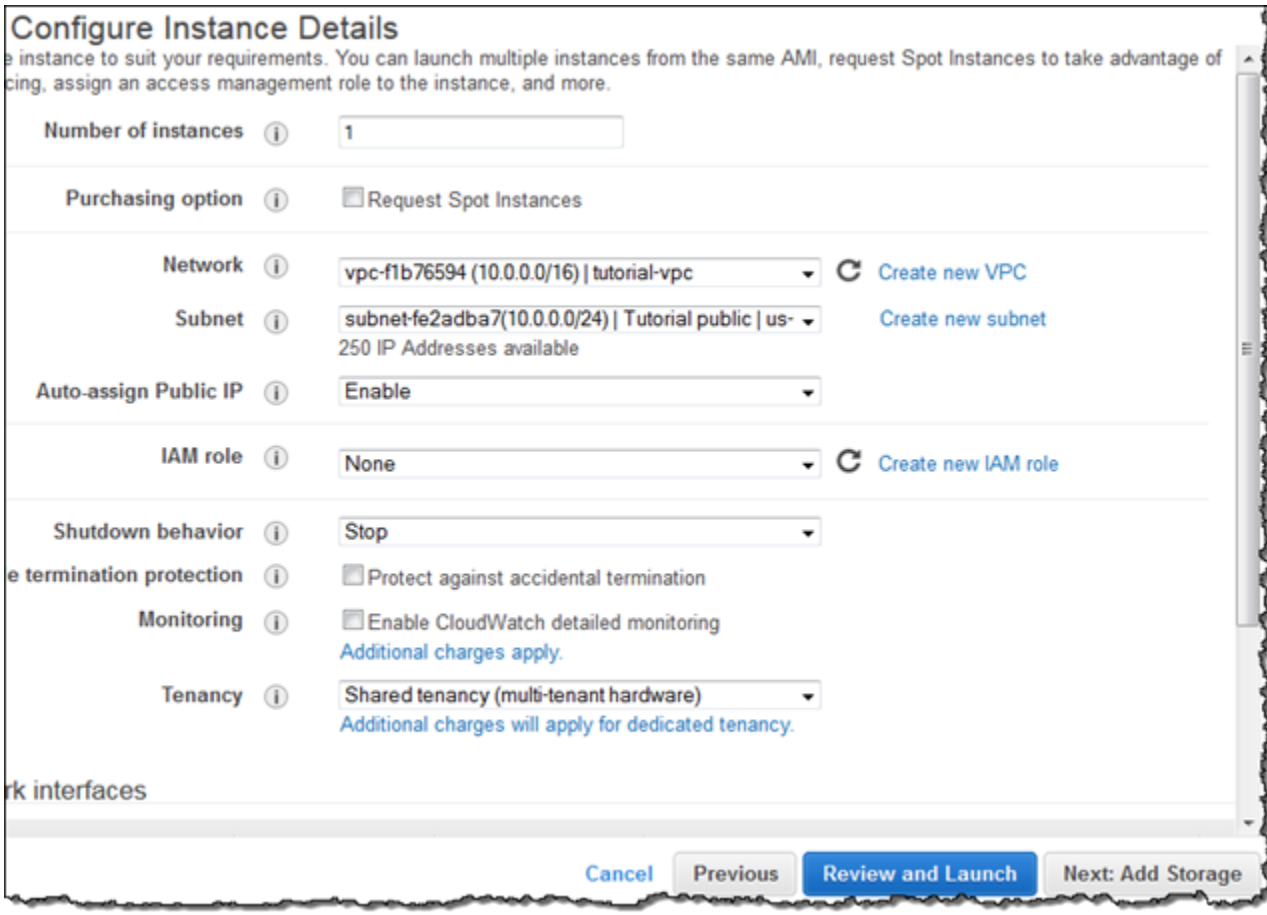


4. Choose the `t2.micro` instance type, as shown following, and then choose **Next: Configure Instance Details**.

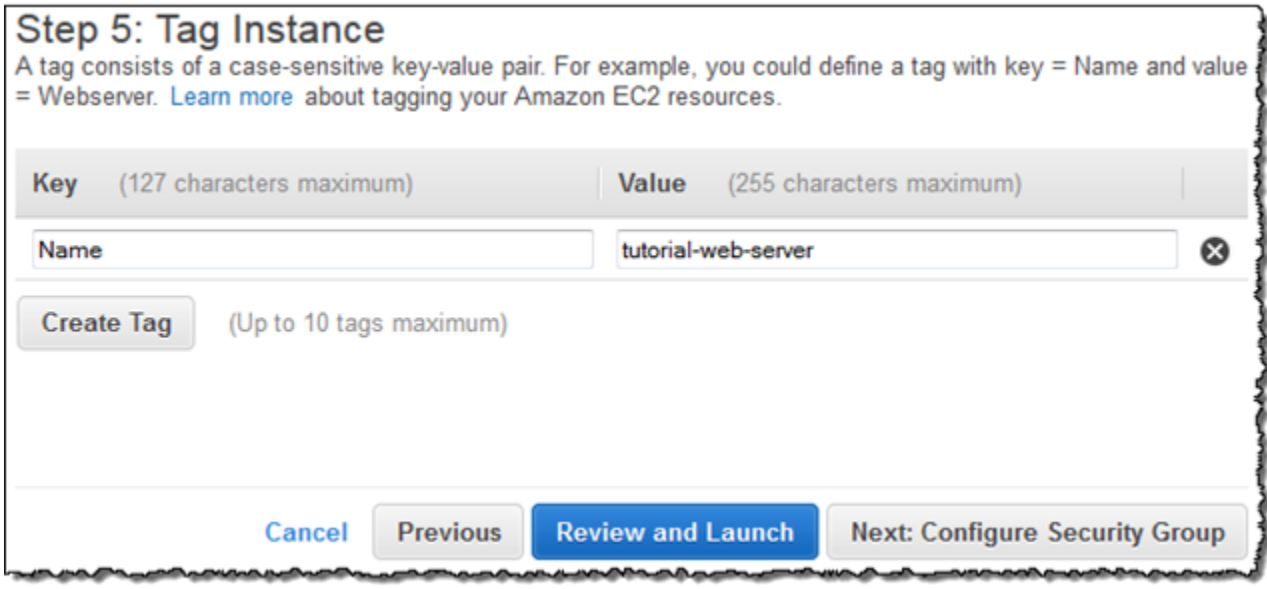


5. On the **Configure Instance Details** page, shown following, set these values and leave the other values as their defaults:

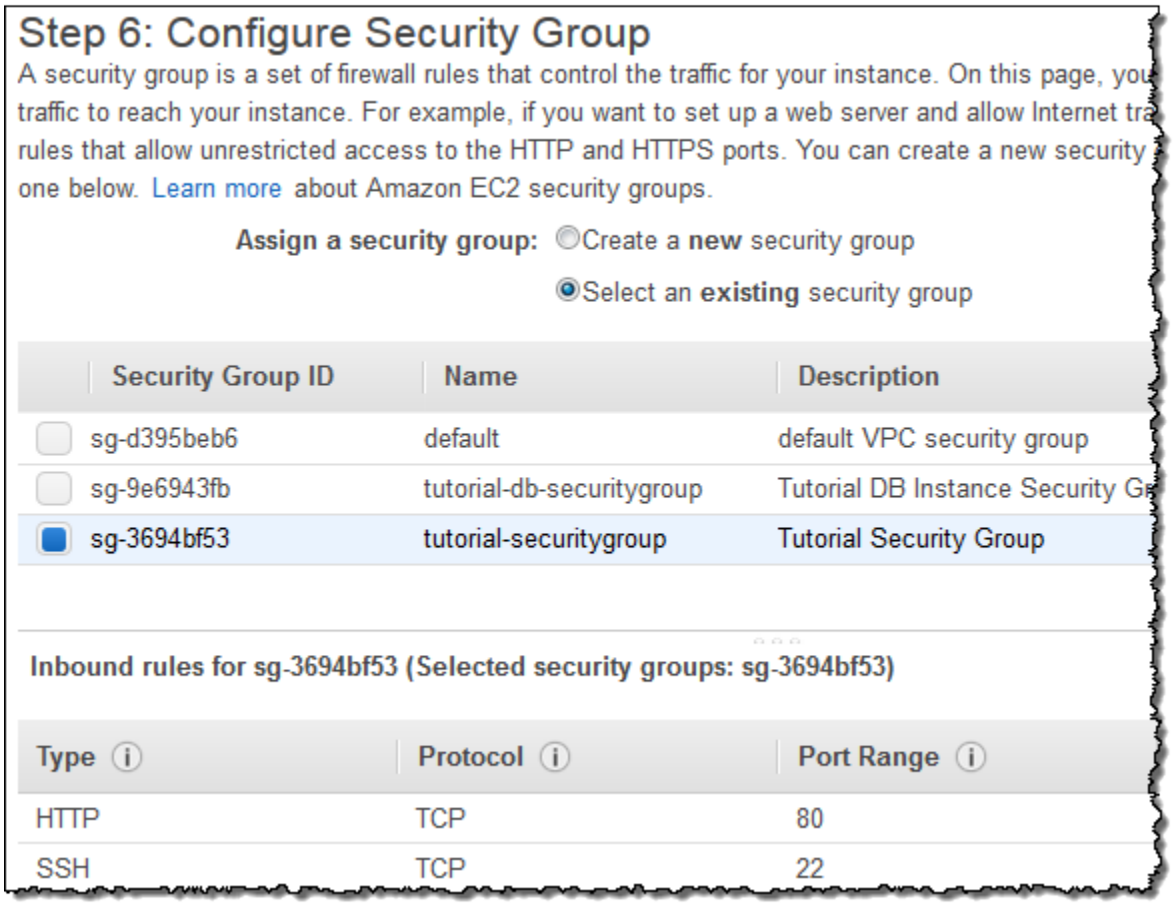
   - **Network:** Choose the VPC with both public and private subnets that you chose for the DB instance, such as the `tutorial-vpc` (vpc-*identifier*) created in Create a VPC with Private and Public Subnets (p. 406)
   - **Subnet:** Choose an existing public subnet, such as `subnet-`*identifier* `| Tutorial public | us-west-2a` created in Create a VPC Security Group for a Public Web Server (p. 408)
   - **Auto-assign Public IP:** `Enable`



6. Choose **Next: Add Storage**.
7. On the **Add Storage** page, leave the default values and choose **Next: Tag Instance**.
8. On the **Tag Instance** page, shown following, choose **Create Tag** and set **Value** for the `Name` tag to `tutorial-web-server`, and then choose **Next: Configure Security Group**.

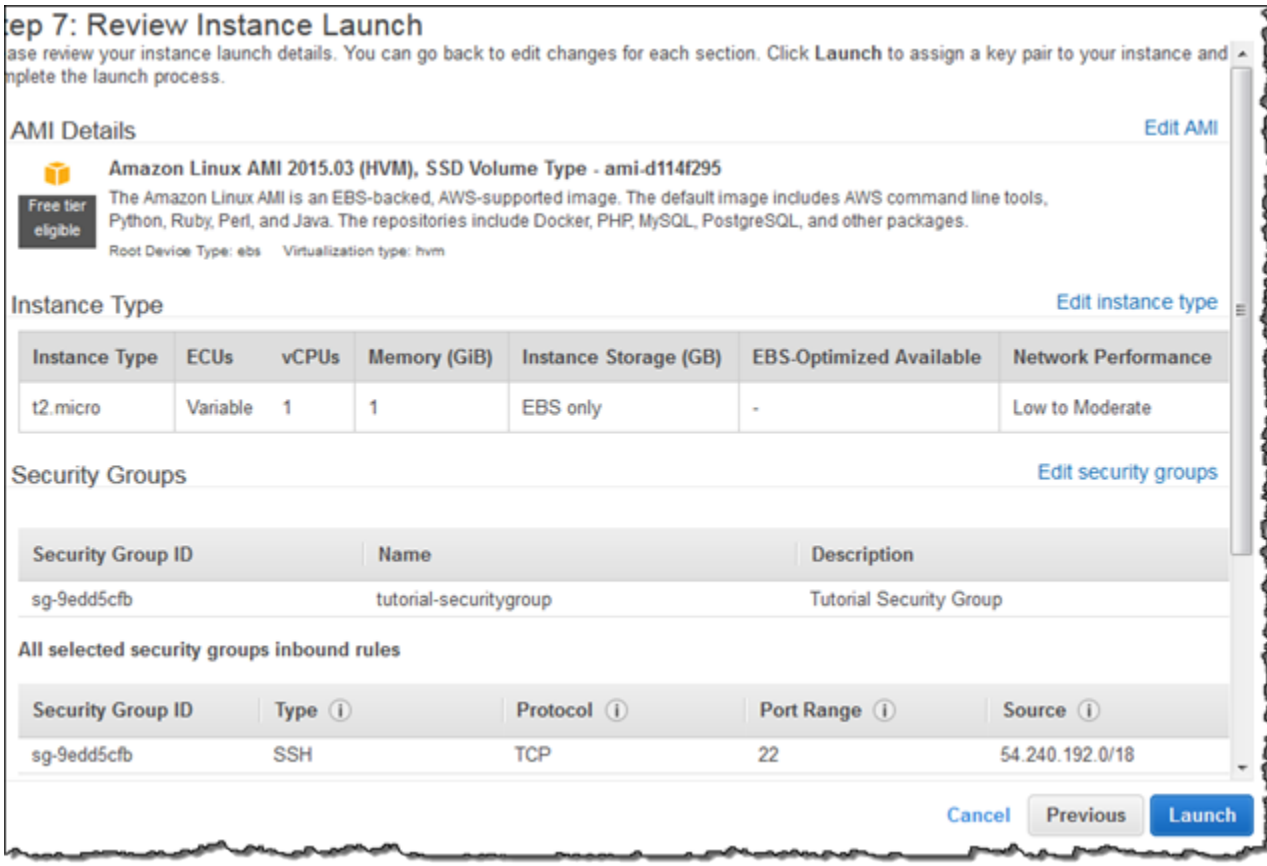


9. On the **Configure Security Group** page, shown following, choose **Select an existing security group**, and then choose an existing security group, such as the `tutorial-securitygroup` created in Create a VPC Security Group for a Public Web Server (p. 408). The security group must include inbound rules for SSH and HTTP access.

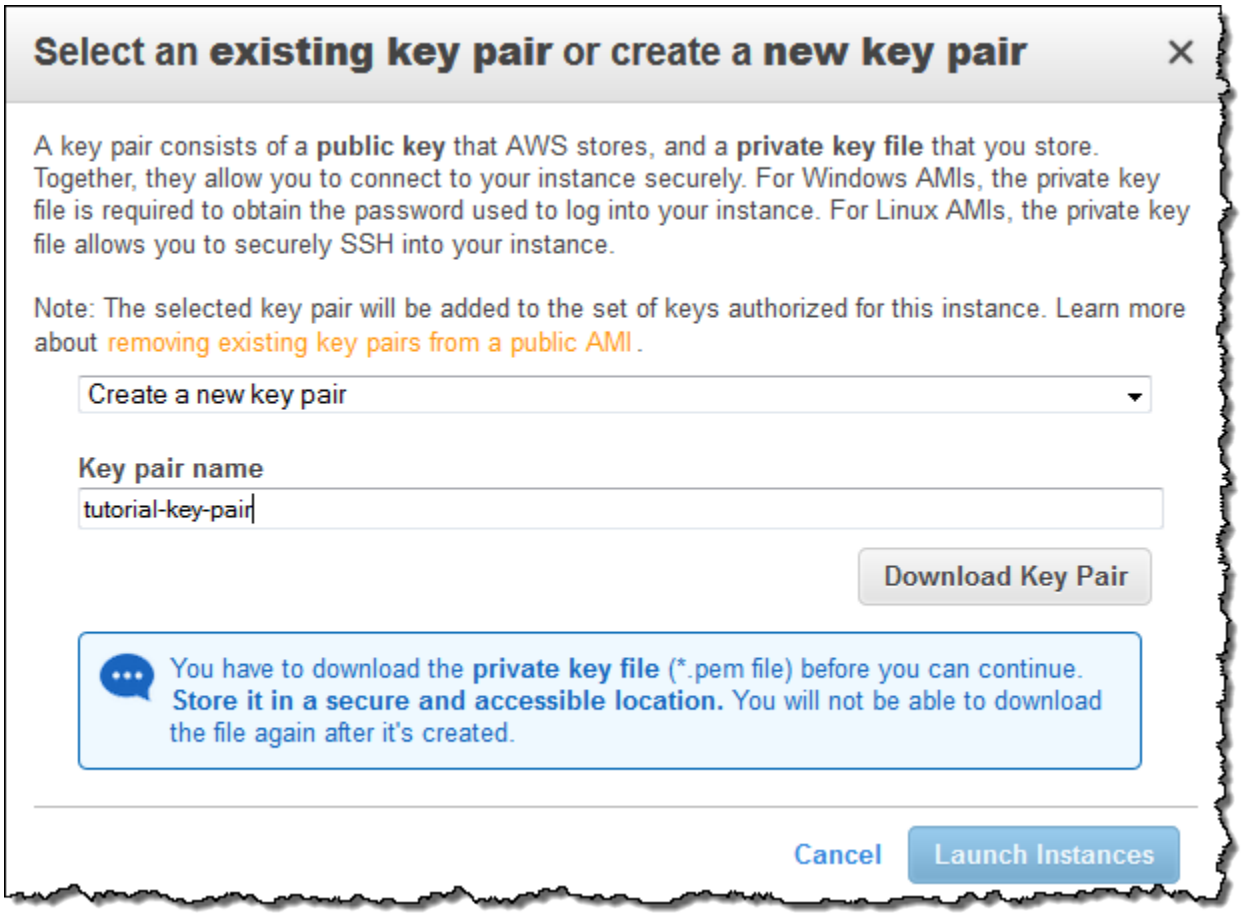


10. Choose **Review and Launch**.

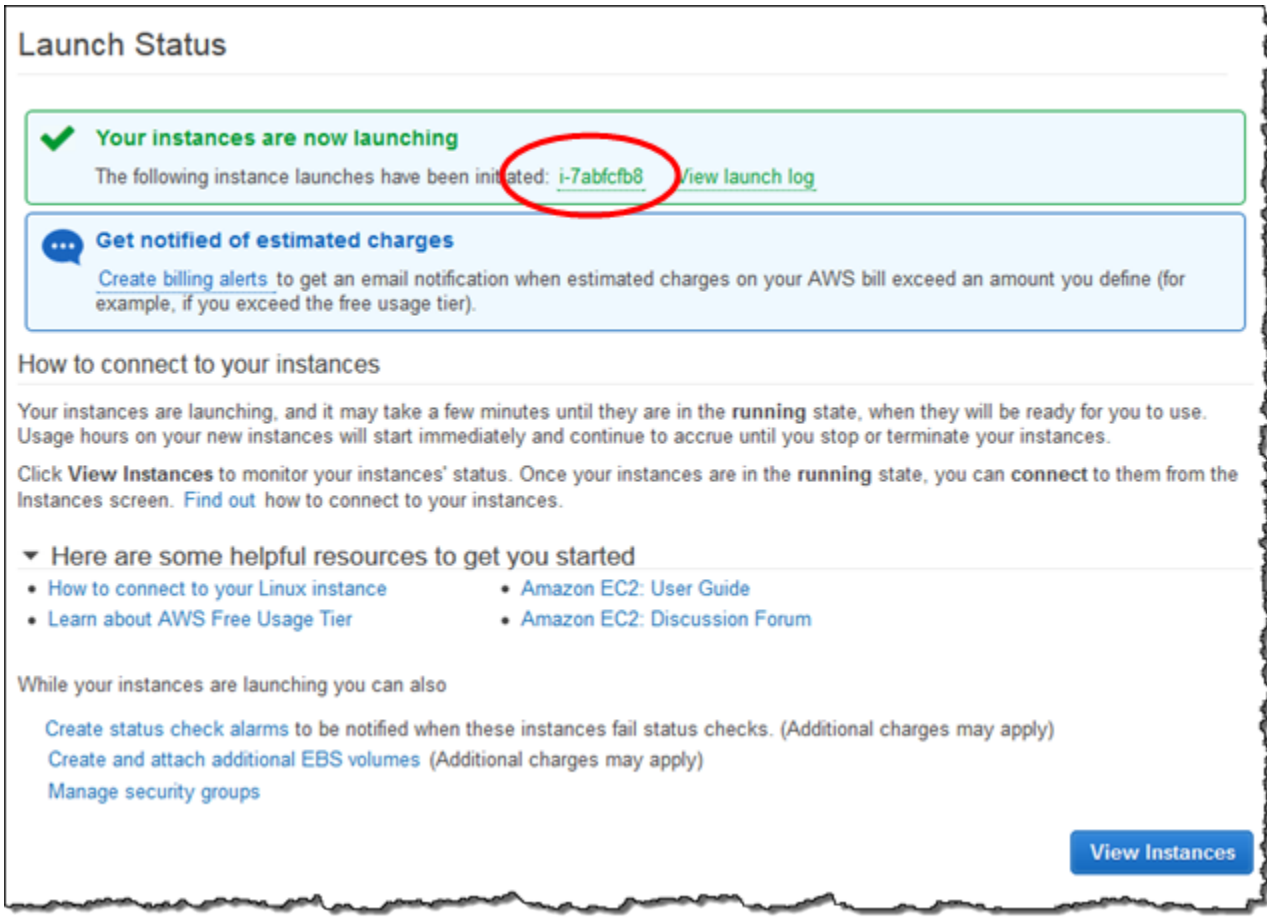
11. On the **Review Instance Launch** page, shown following, verify your settings and then choose **Launch**.



12. On the **Select an existing key pair or create a new key pair** page, shown following, choose `Create a new key pair` and set **Key pair name** to `tutorial-key-pair`. Choose **Download Key Pair**, and then save the key pair file on your local machine. You use this key pair file to connect to your EC2 instance.



13. To launch your EC2 instance, choose **Launch Instances**. On the **Launch Status** page, shown following, note the identifier for your new EC2 instance, for example: `i-7abfcfb8`.



14. To find your instance, choose **View Instances**.
15. Wait until **Instance Status** for your instance reads as `running` before continuing.

Install an Apache web server with PHP

Next you connect to your EC2 instance and install the web server.

To connect to your EC2 instance and install the Apache web server with PHP

1. To connect to the EC2 instance that you created earlier, follow the steps in Connect to Your Instance.
2. To get the latest bug fixes and security updates, update the software on your EC2 instance by using the following command:

   Note
   The `-y` option installs the updates without asking for confirmation. To examine updates before installing, omit this option.

```
[ec2-user ~]$ sudo yum update -y
```

3. After the updates complete, install the Apache web server with the PHP software package using the **yum install** command, which installs multiple software packages and related dependencies at the same time:

```
[ec2-user ~]$ sudo yum install -y httpd24 php56 php56-mysqlnd
```

4. Start the web server with the command shown following:

```
[ec2-user ~]$ sudo service httpd start
```

You can test that your web server is properly installed and started by entering the public DNS name of your EC2 instance in the address bar of a web browser, for example: `http://ec2-42-8-168-21.us-west-1.compute.amazonaws.com`. If your web server is running, then you see the Apache test page. If you don't see the Apache test page, then verify that your inbound rules for the VPC security group that you created in Tutorial: Create an Amazon VPC for Use with an Amazon RDS DB Instance (p. 406) include a rule allowing HTTP (port 80) access for the IP address you use to connect to the web server.

Note
The Apache test page appears only when there is no content in the document root directory, `/var/www/html`. After you add content to the document root directory, your content appears at the public DNS address of your EC2 instance instead of the Apache test page.

5. Configure the web server to start with each system boot using the **chkconfig** command:

```
[ec2-user ~]$ sudo chkconfig httpd on
```

To allow `ec2-user` to manage files in the default root directory for your Apache web server, you need to modify the ownership and permissions of the `/var/www` directory. In this tutorial, you add a group named `www` to your EC2 instance, and then you give that group ownership of the `/var/www` directory and add write permissions for the group. Any members of that group can then add, delete, and modify files for the web server.

To set file permissions for the Apache web server

1. Add the `www` group to your EC2 instance with the following command:

```
[ec2-user ~]$ sudo groupadd www
```

2. Add the `ec2-user` user to the `www` group:

```
[ec2-user ~]$ sudo usermod -a -G www ec2-user
```

3. To refresh your permissions and include the new `www` group, log out:

```
[ec2-user ~]$ exit
```

4. Log back in again and verify that the `www` group exists with the `groups` command:

```
[ec2-user ~]$ groups
ec2-user wheel www
```

5. Change the group ownership of the `/var/www` directory and its contents to the `www` group:

```
[ec2-user ~]$ sudo chown -R root:www /var/www
```

6. Change the directory permissions of `/var/www` and its subdirectories to add group write permissions and set the group ID on subdirectories created in the future:

```
[ec2-user ~]$ sudo chmod 2775 /var/www
[ec2-user ~]$ find /var/www -type d -exec sudo chmod 2775 {} +
```

7. Recursively change the permissions for files in the `/var/www` directory and its subdirectories to add group write permissions:

```
[ec2-user ~]$ find /var/www -type f -exec sudo chmod 0664 {} +
```

Connect your Apache web server to your RDS DB instance

Next, you add content to your Apache web server that connects to your Amazon RDS DB instance.

To add content to the Apache web server that connects to your RDS DB instance

1. While still connected to your EC2 instance, change the directory to `/var/www` and create a new subdirectory named `inc`:

```
[ec2-user ~]$ cd /var/www
[ec2-user ~]$ mkdir inc
[ec2-user ~]$ cd inc
```

2. Create a new file in the `inc` directory named `dbinfo.inc`, and then edit the file by calling nano (or the editor of your choice).

```
[ec2-user ~]$ >dbinfo.inc
[ec2-user ~]$ nano dbinfo.inc
```

3. Add the following contents to the `dbinfo.inc` file, where *`endpoint`* is the endpoint of your RDS MySQL DB instance, without the port, and *`master password`* is the master password for your RDS MySQL DB instance.

   Note
   Placing the user name and password information in a folder that is not part of the document root for your web server reduces the possibility of your security information being exposed.

```
<?php

define('DB_SERVER', 'endpoint');
define('DB_USERNAME', 'tutorial_user');
define('DB_PASSWORD', 'master password');
define('DB_DATABASE', 'sample');

?>
```

4. Save and close the `dbinfo.inc` file.

5. Change the directory to `/var/www/html`:

```
[ec2-user ~]$ cd /var/www/html
```

6. Create a new file in the `html` directory named `SamplePage.php`, and then edit the file by calling nano (or the editor of your choice).

```
[ec2-user ~]$ >SamplePage.php
[ec2-user ~]$ nano SamplePage.php
```

7. Add the following contents to the `SamplePage.php` file:

   Note
   Placing the user name and password information in a folder that is not part of the document root for your web server reduces the possibility of your security information being exposed.

```
<?php include "../inc/dbinfo.inc"; ?>
<html>
<body>
<h1>Sample page</h1>
<?php

  /* Connect to MySQL and select the database. */
  $connection = mysqli_connect(DB_SERVER, DB_USERNAME, DB_PASSWORD);

  if (mysqli_connect_errno()) echo "Failed to connect to MySQL: " .
 mysqli_connect_error();

  $database = mysqli_select_db($connection, DB_DATABASE);

  /* Ensure that the Employees table exists. */
  VerifyEmployeesTable($connection, DB_DATABASE);

  /* If input fields are populated, add a row to the Employees table. */
  $employee_name = htmlentities($_POST['Name']);
  $employee_address = htmlentities($_POST['Address']);

  if (strlen($employee_name) || strlen($employee_address)) {
    AddEmployee($connection, $employee_name, $employee_address);
  }
?>

<!-- Input form -->
<form action="<?PHP echo $_SERVER['SCRIPT_NAME'] ?>" method="POST">
  <table border="0">
    <tr>
      <td>Name</td>
      <td>Address</td>
    </tr>
    <tr>
      <td>
        <input type="text" name="Name" maxlength="45" size="30" />
      </td>
      <td>
        <input type="text" name="Address" maxlength="90" size="60" />
      </td>
      <td>
        <input type="submit" value="Add Data" />
      </td>
    </tr>
```

```
  </table>
</form>

<!-- Display table data. -->
<table border="1" cellpadding="2" cellspacing="2">
  <tr>
    <td>ID</td>
    <td>Name</td>
    <td>Address</td>
  </tr>

<?php

$result = mysqli_query($connection, "SELECT * FROM Employees");

while($query_data = mysqli_fetch_row($result)) {
  echo "<tr>";
  echo "<td>",$query_data[0], "</td>",
       "<td>",$query_data[1], "</td>",
       "<td>",$query_data[2], "</td>";
  echo "</tr>";
}
?>

</table>

<!-- Clean up. -->
<?php

  mysqli_free_result($result);
  mysqli_close($connection);

?>

</body>
</html>


<?php

/* Add an employee to the table. */
function AddEmployee($connection, $name, $address) {
   $n = mysqli_real_escape_string($connection, $name);
   $a = mysqli_real_escape_string($connection, $address);

   $query = "INSERT INTO `Employees` (`Name`, `Address`) VALUES ('$n', '$a');";

   if(!mysqli_query($connection, $query)) echo("<p>Error adding employee data.</p>");
}

/* Check whether the table exists and, if not, create it. */
function VerifyEmployeesTable($connection, $dbName) {
  if(!TableExists("Employees", $connection, $dbName))
  {
     $query = "CREATE TABLE `Employees` (
         `ID` int(11) NOT NULL AUTO_INCREMENT,
         `Name` varchar(45) DEFAULT NULL,
         `Address` varchar(90) DEFAULT NULL,
         PRIMARY KEY (`ID`),
         UNIQUE KEY `ID_UNIQUE` (`ID`)
       ) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=latin1";

     if(!mysqli_query($connection, $query)) echo("<p>Error creating table.</p>");
  }
}
```

```
/* Check for the existence of a table. */
function TableExists($tableName, $connection, $dbName) {
  $t = mysqli_real_escape_string($connection, $tableName);
  $d = mysqli_real_escape_string($connection, $dbName);

  $checktable = mysqli_query($connection,
      "SELECT TABLE_NAME FROM information_schema.TABLES WHERE TABLE_NAME = '$t' AND
 TABLE_SCHEMA = '$d'");

  if(mysqli_num_rows($checktable) > 0) return true;

  return false;
}
?>
```

8. Save and close the `SamplePage.php` file.
9. Verify that your web server successfully connects to your RDS MySQL DB instance by opening a web browser and browsing to `http://`*`EC2 instance endpoint`*`/SamplePage.php`, for example: `http://ec2-55-122-41-31.us-west-2.compute.amazonaws.com/SamplePage.php`.

You can use `SamplePage.php` to add data to your RDS MySQL DB instance. The data that you add is then displayed on the page.

To make sure your RDS MySQL DB instance is as secure as possible, verify that sources outside of the VPC cannot connect to your RDS MySQL DB instance.

Tutorials

The following tutorials show you how to perform common tasks that use Amazon RDS:

- Tutorial: Create an Amazon VPC for Use with an Amazon RDS DB Instance (p. 406)
- Tutorial: Create a Web Server and an Amazon RDS Database (p. 62)
- Tutorial: Restore a DB Instance from a DB Snapshot (p. 239)

For videos, see AWS Instructional Videos and Labs.

Best Practices for Amazon RDS

Learn best practices for working with Amazon RDS. As new best practices are identified, we will keep this section up to date.

Topics

Amazon RDS Basic Operational Guidelines

The following are basic operational guidelines that everyone should follow when working with Amazon RDS. Note that the Amazon RDS Service Level Agreement requires that you follow these guidelines:

- Monitor your memory, CPU, and storage usage. Amazon CloudWatch can be setup to notify you when usage patterns change or when you approach the capacity of your deployment, so that you can maintain system performance and availability.
- Scale up your DB instance when you are approaching storage capacity limits. You should have some buffer in storage and memory to accommodate unforeseen increases in demand from your applications.
- Enable automatic backups and set the backup window to occur during the daily low in write IOPS.
- If your database workload requires more I/O than you have provisioned, recovery after a failover or database failure will be slow. To increase the I/O capacity of a DB instance, do any or all of the following:
  - Migrate to a DB instance class with High I/O capacity.
  - Convert from standard storage to either General Purpose or Provisioned IOPS storage, depending on how much of an increase you need. For information on available storage types, see Amazon RDS Storage Types (p. 410).

    If you convert to Provisioned IOPS storage, make sure you also use a DB instance class that is optimized for Provisioned IOPS. For information on Provisioned IOPS, see Provisioned IOPS Storage (p. 413).
  - If you are already using Provisioned IOPS storage, provision additional throughput capacity.
- If your client application is caching the Domain Name Service (DNS) data of your DB instances, set a time-to-live (TTL) value of less than 30 seconds. Because the underlying IP address of a DB instance

can change after a failover, caching the DNS data for an extended time can lead to connection failures if your application tries to connect to an IP address that no longer is in service.

- Test failover for your DB instance to understand how long the process takes for your use case and to ensure that the application that accesses your DB instance can automatically connect to the new DB instance after failover.

DB Instance RAM Recommendations

An Amazon RDS performance best practice is to allocate enough RAM so that your working set resides almost completely in memory. To tell if your working set is almost all in memory, check the ReadIOPS metric (using Amazon CloudWatch) while the DB instance is under load. The value of ReadIOPS should be small and stable. If scaling up the DB instance class—to a class with more RAM—results in a dramatic drop in ReadIOPS, your working set was not almost completely in memory. Continue to scale up until ReadIOPS no longer drops dramatically after a scaling operation, or ReadIOPS is reduced to a very small amount. For information on monitoring a DB instance's metrics, see Viewing DB Instance Metrics (p. 254).

Amazon RDS Security Best Practices

Use AWS IAM accounts to control access to Amazon RDS API actions, especially actions that create, modify, or delete RDS resources such as DB instances, security groups, option groups, or parameter groups, and actions that perform common administrative actions such as backing up and restoring DB instances, or configuring Provisioned IOPS storage.

- Assign an individual IAM account to each person who manages RDS resources. Do not use AWS root credentials to manage Amazon RDS resources; you should create an IAM user for everyone, including yourself.
- Grant each user the minimum set of permissions required to perform his or her duties.
- Use IAM groups to effectively manage permissions for multiple users.
- Rotate your IAM credentials regularly.

For more information about IAM, go to AWS Identity and Access Management. For information on IAM best practices, go to IAM Best Practices.

Using Enhanced Monitoring to Identify Operating System Issues

Amazon RDS provides metrics in real time for the operating system (OS) that your DB instance runs on. You can view the metrics for your DB instance using the console, or consume the Enhanced Monitoring JSON output from Amazon CloudWatch Logs in a monitoring system of your choice. For more information about Enhanced Monitoring, see Enhanced Monitoring (p. 258)

Enhanced Monitoring is available for the following database engines:

- Amazon Aurora
- MariaDB
- Microsoft SQL Server

- MySQL version 5.5 or later
- Oracle
- PostgreSQL

Enhanced monitoring is available for all DB instance classes except for `db.m1.small`. Enhanced Monitoring is available in all regions except for AWS GovCloud (US).

Using Metrics to Identify Performance Issues

To identify performance issues caused by insufficient resources and other common bottlenecks, you can monitor the metrics available for your Amazon RDS DB instance.

Viewing Performance Metrics

You should monitor performance metrics on a regular basis to see the average, maximum, and minimum values for a variety of time ranges. If you do so, you can identify when performance is degraded. You can also set Amazon CloudWatch alarms for particular metric thresholds so you are alerted if they are reached.

In order to troubleshoot performance issues, it's important to understand the baseline performance of the system. When you set up a new DB instance and get it running with a typical workload, you should capture the average, maximum, and minimum values of all of the performance metrics at a number of different intervals (for example, one hour, 24 hours, one week, two weeks) to get an idea of what is normal. It helps to get comparisons for both peak and off-peak hours of operation. You can then use this information to identify when performance is dropping below standard levels.

To view performance metrics

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the left navigation pane, select **Instances**, and then select a DB instance.
3. Select **Show Monitoring**. The first eight performance metrics display. The metrics default to showing information for the current day.
4. Use the numbered buttons at top right to page through the additional metrics, or select **Show All** to see all metrics.
5. Select a performance metric to adjust the time range in order to see data for other than the current day. You can change the **Statistic**, **Time Range**, and **Period** values to adjust the information displayed. For example, to see the peak values for a metric for each day of the last two weeks, set **Statistic** to **Maximum**, **Time Range** to **Last 2 Weeks**, and **Period** to **Day**.

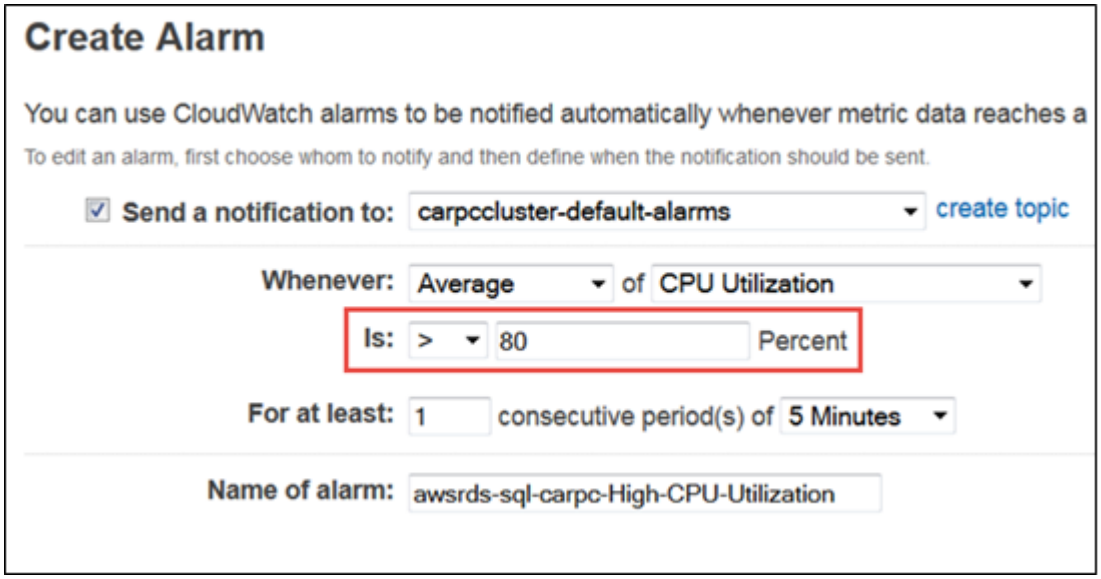
   Note
   Changing the **Statistic**, **Time Range**, and **Period** values changes them for all metrics. The updated values persist for the remainder of your session or until you change them again.

You can also view performance metrics using the CLI or API. For more information, see Viewing DB Instance Metrics (p. 254).

To set a CloudWatch alarm

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.

2. In the left navigation pane, select **Instances**, and then select a DB instance.
3. Select **Show Monitoring**, and then select a performance metric to bring up the expanded view.
4. Select **Create Alarm**.
5. On the **Create Alarm** page, identify what email address should receive the alert by selecting a value in the **Send a notification to** box. Select **create topic** to the right of that box to create a new alarm recipient if necessary.
6. In the **Whenever** list, select the alarm statistic to set.
7. In the **of** box, select the alarm metric.
8. In the **Is** box and the unlabeled box to the right of it, set the alarm threshold, as shown following:



9. In the **For at least** box, enter the number of times that the specified threshold must be reached in order to trigger the alarm.
10. In the **consecutive period(s) of** box, select the period during which the threshold must have been reached in order to trigger the alarm.
11. In the **Name of alarm** box, enter a name for the alarm.
12. Select **Create Alarm**.

The performance metrics page appears, and you can see the new alarm in the **CloudWatch Alarms** status bar. If you don't see the status bar, refresh your page.

Evaluating Performance Metrics

A DB instance has a number of different categories of metrics, and how to determine acceptable values depends on the metric.

CPU

- CPU Utilization – Percentage of computer processing capacity used.

Memory

- Freeable Memory – How much RAM is available on the DB instance, in megabytes.
- Swap Usage – How much swap space is used by the DB instance, in megabytes.

Disk space

- Free Storage Space – How much disk space is not currently being used by the DB instance, in megabytes.

Input/output operations

- Read IOPS, Write IOPS – The average number of disk read or write operations per second.
- Read Latency, Write Latency – The average time for a read or write operation in milliseconds.
- Read Throughput, Write Throughput – The average number of megabytes read from or written to disk per second.
- Queue Depth – The number of I/O operations that are waiting to be written to or read from disk.

Network traffic

- Network Receive Throughput, Network Transmit Throughput – The rate of network traffic to and from the DB instance in megabytes per second.

Database connections

- DB Connections – The number of client sessions that are connected to the DB instance.

For more detailed individual descriptions of the performance metrics available, see Amazon RDS Dimensions and Metrics.

Generally speaking, acceptable values for performance metrics depend on what your baseline looks like and what your application is doing. Investigate consistent or trending variances from your baseline. Advice about specific types of metrics follows:

- **High CPU or RAM consumption –** High values for CPU or RAM consumption might be appropriate, provided that they are in keeping with your goals for your application (like throughput or concurrency) and are expected.
- **Disk space consumption –** Investigate disk space consumption if space used is consistently at or above 85 percent of the total disk space. See if it is possible to delete data from the instance or archive data to a different system to free up space.
- **Network traffic –** For network traffic, talk with your system administrator to understand what expected throughput is for your domain network and Internet connection. Investigate network traffic if throughput is consistently lower than expected.
- **Database connections –** Consider constraining database connections if you see high numbers of user connections in conjunction with decreases in instance performance and response time. The best number of user connections for your DB instance will vary based on your instance class and the complexity of the operations being performed. You can determine the number of database connections by associating your DB instance with a parameter group where the *User Connections* parameter is set to other than 0 (unlimited). You can either use an existing parameter group or create a new one. For more information, see Working with DB Parameter Groups (p. 170).
- **IOPS metrics –** The expected values for IOPS metrics depend on disk specification and server configuration, so use your baseline to know what is typical. Investigate if values are consistently different than your baseline. For best IOPS performance, make sure your typical working set will fit into memory to minimize read and write operations.

For issues with any performance metrics, one of the first things you can do to improve performance is tune the most used and most expensive queries to see if that lowers the pressure on system resources. For more information, see Tuning Queries (p. 85)

If your queries are tuned and an issue persists, consider upgrading your Amazon RDS DB Instance Class (p. 92) to one with more of the resource (CPU, RAM, disk space, network bandwidth, I/O capacity) that is related to the issue you are experiencing.

Tuning Queries

One of the best ways to improve DB instance performance is to tune your most commonly used and most resource-intensive queries to make them less expensive to run.

MySQL Query Tuning

Go to Optimizing SELECT Statements in the MySQL documentation for more information on writing queries for better performance. You can also go to MySQL Performance Tuning and Optimization Resources for additional query tuning resources.

Oracle Query Tuning

Go to the Database SQL Tuning Guide in the Oracle documentation for more information on writing and analyzing queries for better performance.

SQL Server Query Tuning

Go to Analyzing a Query in the SQL Server documentation to improve queries for SQL Server DB instances. You can also use the execution-, index- and I/O-related data management views (DMVs) described in the Dynamic Management Views and Functions documentation to troubleshoot SQL Server query issues.

A common aspect of query tuning is creating effective indexes. You can use the Database Engine Tuning Advisor to get potential index improvements for your DB instance. For more information, see Analyzing Your Database Workload on an Amazon RDS DB Instance with SQL Server Tuning Advisor (p. 803).

PostgreSQL Query Tuning

Go to Using EXPLAIN in the PostgreSQL documentation to learn how to analyze a query plan. You can use this information to modify a query or underlying tables in order to improve query performance. You can also go to Controlling the Planner with Explicit JOIN Clauses to get tips about how to specify joins in your query for the best performance.

MariaDB Query Tuning

Go to Query Optimizations in the MariaDB documentation for more information on writing queries for better performance.

Best Practices for Working with Amazon Aurora

You have several different options for improving performance and stability in Amazon Aurora, depending on the database engine used by your Aurora DB cluster and DB instances. For more information about best practices with Amazon Aurora, see Best Practices with Amazon Aurora (p. 663).

Best Practices for Working with MySQL Storage Engines

On a MySQL DB instance, observe the following table creation limits:

- You're limited to 10,000 tables if you are either using Provisioned IOPS storage, or using General Purpose storage and the instance is 200 GB or larger in size.
- You're limited to 1000 tables if you are either using standard storage, or using General Purpose storage and the instance is less than 200 GB in size.

We recommend these limits because having large numbers of tables significantly increases database recovery time after a failover or database crash. If you need to create more tables than recommended, set the `innodb_file_per_table` parameter to 0. For more information, see Working with InnoDB Tablespaces to Improve Crash Recovery Times (p. 906) and Working with DB Parameter Groups (p. 170).

For MySQL DB instances that use version 5.7 or later, you can exceed these table creation limits due to improvements in InnoDB crash recovery. However, we still recommend that you take caution due to the potential performance impact of creating very large numbers of tables.

On a MySQL DB instance, avoid tables in your database growing too large. Provisioned storage limits restrict the maximum size of a MySQL table file to 16 TB. Instead, partition your large tables so that file sizes are well under the 16 TB limit. This approach can also improve performance and recovery time. For more information, see MySQL File Size Limits (p. 911).

The Point-In-Time Restore and snapshot restore features of Amazon RDS for MySQL require a crash-recoverable storage engine and are supported for the InnoDB storage engine only. Although MySQL supports multiple storage engines with varying capabilities, not all of them are optimized for crash recovery and data durability. For example, the MyISAM storage engine does not support reliable crash recovery and might prevent a Point-In-Time Restore or snapshot restore from working as intended. This might result in lost or corrupt data when MySQL is restarted after a crash.

InnoDB is the recommended and supported storage engine for MySQL DB instances on Amazon RDS. InnoDB instances can also be migrated to Aurora, while MyISAM instances can't be migrated. However, MyISAM performs better than InnoDB if you require intense, full-text search capability. If you still choose to use MyISAM with Amazon RDS, following the steps outlined in Automated Backups with Unsupported MySQL Storage Engines (p. 205) can be helpful in certain scenarios for snapshot restore functionality.

If you want to convert existing MyISAM tables to InnoDB tables, you can use the process outlined in the MySQL documentation. MyISAM and InnoDB have different strengths and weaknesses, so you should fully evaluate the impact of making this switch on your applications before doing so.

In addition, Federated Storage Engine is currently not supported by Amazon RDS for MySQL.

Best Practices for Working with MariaDB Storage Engines

The Point-In-Time Restore and snapshot restore features of Amazon RDS for MariaDB require a crash-recoverable storage engine and are supported for the XtraDB storage engine only. Although MariaDB supports multiple storage engines with varying capabilities, not all of them are optimized for crash recovery and data durability. For example, although Aria is a crash-safe replacement for MyISAM, it might still prevent a Point-In-Time Restore or snapshot restore from working as intended. This might result in lost or corrupt data when MariaDB is restarted after a crash.

XtraDB is the recommended and supported storage engine for MariaDB DB instances on Amazon RDS. If you still choose to use Aria with Amazon RDS, following the steps outlined in Automated Backups with Unsupported MariaDB Storage Engines (p. 206) can be helpful in certain scenarios for snapshot restore functionality.

Best Practices for Working with PostgreSQL

Two important areas where you can improve performance with PostgreSQL on Amazon RDS are when loading data into a DB instance and when using the PostgreSQL autovacuum feature. The following sections cover some of the practices we recommend for these areas.

Loading Data into a PostgreSQL DB Instance

When loading data into an Amazon RDS PostgreSQL DB instance, you should modify your DB instance settings and your DB parameter group values to allow for the most efficient importing of data into your DB instance.

Modify your DB instance settings to the following:

- Disable DB instance backups (set backup_retention to 0)
- Disable Multi-AZ

Modify your DB parameter group to include the following settings. You should test the parameter settings to find the most efficient settings for your DB instance:

- Increase the value of the `maintenance_work_mem` parameter. For more information about PostgreSQL resource consumption parameters, see the PostgreSQL documentation.
- Increase the value of the `checkpoint_segments` and `checkpoint_timeout` parameters to reduce the number of writes to the wal log.
- Disable the `synchronous_commit` parameter (do not turn off FSYNC).
- Disable the PostgreSQL autovacuum parameter.

Use the `pg_dump -Fc` (compressed) or `pg_restore -j` (parallel) commands with these settings.

Working with the fsync and full_page_writes database parameters

In PostgreSQL 9.4.1 on Amazon RDS, the `fsync` and `full_page_writes` database parameters are not modifiable. Disabling the `fsync` and `full_page_writes` database parameters can lead to data corruption, so we have enabled them for you. We recommend that customers with other 9.3 DB engine versions of PostgreSQL not disable the `fsync` and `full_page_writes` parameters.

Working with the PostgreSQL Autovacuum Feature

The autovacuum feature for PostgreSQL databases is a feature that we strongly recommend you use to maintain the health of your PostgreSQL DB instance. Autovacuum automates the execution of the VACUUM and ANALYZE command; using autovacuum is required by PostgreSQL, not imposed by Amazon RDS, and its use is critical to good performance. The feature is enabled by default for all new Amazon RDS PostgreSQL DB instances, and the related configuration parameters are appropriately set by default.

Your database administrator needs to know and understand this maintenance operation. For the PostgreSQL documentation on autovacuum, see http://www.postgresql.org/docs/current/static/routine-vacuuming.html#AUTOVACUUM.

Autovacuum is not a “resource free” operation, but it works in the background and yields to user operations as much as possible. When enabled, autovacuum checks for tables that have had a large

number of updated or deleted tuples. It also protects against loss of very old data due to transaction ID wraparound.

Autovacuum should not be thought of as a high-overhead operation that can be reduced to gain better performance. On the contrary, tables that have a high velocity of updates and deletes will quickly deteriorate over time if autovacuum is not run.

> **Important**
> Not running autovacuum can result in an eventual required outage to perform a much more intrusive vacuum operation. When an Amazon RDS PostgreSQL DB instance becomes unavailable because of an over conservative use of autovacuum, the PostgreSQL database will shut down to protect itself. At that point, Amazon RDS must perform a single-user–mode full vacuum directly on the DB instance , which can result in a multi-hour outage. Thus, we strongly recommend that you do not turn off autovacuum, which is enabled by default.

The autovacuum parameters determine when and how hard autovacuum works. The `autovacuum_vacuum_threshold` and `autovacuum_vacuum_scale_factor` parameters determine when autovacuum is run. The `autovacuum_max_workers`, `autovacuum_nap_time`, `autovacuum_cost_limit`, and `autovacuum_cost_delay` parameters determine how hard autovacuum works. For more information about autovacuum, when it runs, and what parameters are required, see the PostgreSQL documentation.

The following query shows the number of "dead" tuples in a table named table1 :

```
PROMPT> select relname, n_dead_tup, last_vacuum, last_autovacuum from
pg_catalog.pg_stat_all_tables
where n_dead_tup > 0 and relname =  'table1' order by n_dead_tup desc;
```

The results of the query will resemble the following:

```
relname | n_dead_tup | last_vacuum | last_autovacuum
---------+------------+-------------+-----------------
 tasks   |   81430522 |             |
(1 row)
```

Best Practices for Working with SQL Server

Best practices for a Multi-AZ deployment with a SQL Server DB instance include the following:

- Use Amazon RDS DB events to monitor failovers. For example, you can be notified by text message or email when a DB instance fails over. For more information about Amazon RDS events, see Using Amazon RDS Event Notification (p. 279).
- If your application caches DNS values, set time to live (TTL) to less than 30 seconds. Setting TTL as so is a good practice in case there is a failover, where the IP address might change and the cached value might no longer be in service.
- We recommend that you *do not* enable the following modes because they turn off transaction logging, which is required for Multi-AZ:
  - Simple recover mode
  - Offline mode
  - Read-only mode
- Test to determine how long it takes for your DB instance to failover. Failover time can vary due to the type of database, the instance class, and the storage type you use. You should also test your application's ability to continue working if a failover occurs.
- To shorten failover time, you should do the following:

  - Ensure that you have sufficient Provisioned IOPS allocated for your workload. Inadequate I/O can lengthen failover times. Database recovery requires I/O.
  - Use smaller transactions. Database recovery relies on transactions, so if you can break up large transactions into multiple smaller transactions, your failover time should be shorter.
- Take into consideration that during a failover, there will be elevated latencies. As part of the failover process, Amazon RDS automatically replicates your data to a new standby instance. This replication means that new data is being committed to two different DB instances, so there might be some latency until the standby DB instance has caught up to the new primary DB instance.
- Deploy your applications in all Availability Zones. If an Availability Zone does go down, your applications in the other Availability Zones will still be available.

When working with a Multi-AZ deployment of SQL Server, remember that Amazon RDS mirrors all SQL Server databases on your instance. If you don't want particular databases to be mirrored, set up a separate DB instance that doesn't use Multi-AZ for those databases.

Working with DB Parameter Groups

We recommend that you try out DB parameter group changes on a test DB instance before applying parameter group changes to your production DB instances. Improperly setting DB engine parameters in a DB parameter group can have unintended adverse effects, including degraded performance and system instability. Always exercise caution when modifying DB engine parameters and back up your DB instance before modifying a DB parameter group. For information about backing up your DB instance, see Backing Up and Restoring Amazon RDS DB Instances (p. 200).

Amazon RDS Best Practices Presentation Video

The 2016 AWS Summit conference in Chicago included a presentation on best practices for creating and configuring a secure, highly available database instance using Amazon RDS. A video of the presentation is available here.

Amazon RDS DB Instances

A *DB instance* is an isolated database environment running in the cloud. It is the basic building block of Amazon RDS. A DB instance can contain multiple user-created databases, and can be accessed using the same client tools and applications you might use to access a standalone database instance. DB instances are simple to create and modify with the Amazon AWS command line tools, Amazon RDS API actions, or the AWS Management Console.

> **Note**
> Amazon RDS supports access to databases using any standard SQL client application. Amazon RDS does not allow direct host access.

You can have up to 40 Amazon RDS DB instances. Of these 40, up to 10 can be Oracle or SQL Server DB instances under the "License Included" model. All 40 DB instances can be used for MySQL, MariaDB, or PostgreSQL. You can also have 40 DB instances for SQL Server or Oracle under the "BYOL" licensing model. If your application requires more DB instances, you can request additional DB instances using the form at https://console.aws.amazon.com/support/home#/case/create?issueType=service-limit-increase&limitType=service-code-rds-instances.

Each DB instance has a DB instance identifier. This customer-supplied name uniquely identifies the DB instance when interacting with the Amazon RDS API and AWS CLI commands. The DB instance identifier must be unique for that customer in an AWS Region.

Each DB instance supports a database engine. Amazon RDS currently supports MySQL, MariaDB, PostgreSQL, Oracle, Microsoft SQL Server, and Amazon Aurora database engines.

When creating a DB instance, some database engines require that a database name be specified. A DB instance can host multiple databases, or a single Oracle database with multiple schemas. The database name value depends on the database engine:

- For the MySQL and MariaDB database engines, the database name is the name of a database hosted in your DB instance. Databases hosted by the same DB instance must have a unique name within that instance.
- For the Oracle database engine, database name is used to set the value of ORACLE_SID, which must be supplied when connecting to the Oracle RDS instance.
- For the Microsoft SQL Server database engine, database name is not a supported parameter.
- For the PostgreSQL database engine, the database name is the name of a database hosted in your DB instance. A database name is not required when creating a DB instance. Databases hosted by the same DB instance must have a unique name within that instance.

Amazon RDS creates a master user account for your DB instance as part of the creation process. This master user has permissions to create databases and to perform create, delete, select, update, and insert operations on tables the master user creates. You must set the master user password when you create a DB instance, but you can change it at any time using the Amazon AWS command line tools, Amazon RDS API actions, or the AWS Management Console. You can also change the master user password and manage users using standard SQL commands.

Topics

- DB Instance and DB Cluster Maintenance (p. 102)
- Amazon RDS DB Instance Lifecycle (p. 111)
- Tagging Amazon RDS Resources (p. 129)
- Working with PostgreSQL, MySQL, and MariaDB Read Replicas (p. 134)
- Working with Option Groups (p. 153)
- Working with DB Parameter Groups (p. 170)
- Working with Amazon Resource Names (ARNs) in Amazon RDS (p. 184)
- Working with Reserved DB Instances (p. 189)

DB Instance Class

The DB instance class determines the computation and memory capacity of an Amazon RDS DB instance. The DB instance class you need depends on your processing power and memory requirements.

For more information about instance class pricing, see Amazon RDS Pricing.

DB Instance Class Types

Amazon RDS supports three types of instance classes: Standard, Memory Optimized, and Burstable Performance. For more information about Amazon EC2 instance types, see Instance Type in the Amazon EC2 documentation.

The following are the Standard DB instance classes available:

- **db.m4** – Third-generation instance classes that provide more computing capacity than the second-generation db.m3 instance classes at a lower price.
- **db.m3** – Second-generation instance classes that provide a balance of compute, memory, and network resources, and are a good choice for many applications.
- **db.m1** – First-generation general-purpose instance classes.

The following are the Memory Optimized DB instance classes available:

- **db.r4** – Third-generation instance classes optimized for memory-intensive applications and that offer a better price per GiB of RAM than the db.r3 instance classes.
- **db.r3** – Second-generation instance classes that provide memory optimization and more computing capacity than the first-generation db.m2 instance classes, at a lower price. The db.r3 DB instances classes are not available in the South America (São Paulo) region.
- **db.m2** – First-generation memory-optimized instance classes.

The following are the Burstable Performance DB instance classes available:

- **db.t2** – Instance classes that provide a baseline performance level, with the ability to burst to full CPU usage.

Specifications for All Available DB Instance Classes

The following table provides details of the Amazon RDS DB instance classes. The table columns are explained after the table.

Instance Class	vCPU	ECU[2]	Memor (GiB)	VPC Only	EBS Opti	Max. Bandw (Mbps)	Networ Perfor	Auro MyS	Auro Post	Mari	Micr SQL Serv	MyS	Orac	PostgreSQL
db.m4 – Latest Generation Standard Instance Classes														
db.m4.16xlarge	64	188	256	Yes	Yes	10,000	25 Gbps	No	No	Yes	Yes[8]	MySQL 5.7, 5.6	Yes[9]	No
db.m4.10xlarge	40	124.5	160	Yes	Yes	4,000	10 Gbps	No	No	Yes	Yes[8]	Yes	Yes[9]	Yes

Instance Class	vCPU	ECU[2]	Memor (GiB)	VPC Only	EBS Opti	Max. Bandw (Mbps)	Networ Perfor	Auro MyS	Auro Post	Mari	Micr SQL Serv	MyS	Orac	PostgreSQL
db.m4.4xlarge	16	53.5	64	Yes	Yes	2,000	High	No	No	Yes	Yes[8]	Yes	Yes[9]	Yes
db.m4.2xlarge	8	25.5	32	Yes	Yes	1,000	High	No	No	Yes	Yes[8]	Yes	Yes[9]	Yes
db.m4.xlarge	4	13	16	Yes	Yes	750	High	No	No	Yes	Yes[8]	Yes	Yes[9]	Yes
db.m4.large	2	6.5	8	Yes	Yes	450	Moderate	No	No	Yes	Yes[8]	Yes	Yes[9]	Yes
db.m3 – Current Generation Standard Instance Classes														
db.m3.2xlarge	8	26	30	No	Yes	1,000	High	No	No	No	Yes[8]	No	Yes[9]	Yes
db.m3.xlarge	4	13	15	No	Yes	500	High	No	No	No	Yes[8]	No	Yes[9]	Yes
db.m3.large	2	6.5	7.5	No	No	—	Moderate	No	No	No	Yes[8]	No	Yes[9]	Yes
db.m3.medium	1	3	3.75	No	No	—	Moderate	No	No	No	Yes[8]	No	Yes[9]	Yes
db.m1 – Previous Generation Standard Instance Classes														
db.m1.xlarge	4	4	15	No	Yes	450	High	No	No	No	Yes[8]	No	Yes[9]	Yes
db.m1.large	2	2	7.5	No	Yes	450	Moderate	No	No	No	Yes[8]	No	Yes[9]	Yes
db.m1.medium	1	1	3.75	No	No	—	Moderate	No	No	No	Yes[8]	No	Yes[9]	Yes
db.m1.small	1	1	1.7	No	No	—	Very Low	No	No	No	Yes[8]	No	Yes[9]	Yes
db.r4 – Latest Generation Memory Optimized Instance Classes														
db.r4.16xlarge	64	195	488	No	Yes	14,000	25 Gbps	1.15 and later	Yes	Yes	Yes[8]	MySQL 5.7, 5.6	Yes[9]	No
db.r4.8xlarge	32	99	244	No	Yes	7,000	10 Gbps	1.15 and later	Yes	Yes	Yes[8]	MySQL 5.7, 5.6	Yes[9]	No
db.r4.4xlarge	16	53	122	No	Yes	3,500	Up to 10 Gbps	1.15 and later	Yes	Yes	Yes[8]	MySQL 5.7, 5.6	Yes[9]	No
db.r4.2xlarge	8	27	61	No	Yes	1,750	Up to 10 Gbps	1.15 and later	Yes	Yes	Yes[8]	MySQL 5.7, 5.6	Yes[9]	No
db.r4.xlarge	4	13.5	30.5	No	Yes	875	Up to 10 Gbps	1.15 and later	Yes	Yes	Yes[8]	MySQL 5.7, 5.6	Yes[9]	No
db.r4.large	2	7	15.25	No	Yes	437	Up to 10 Gbps	1.15 and later	Yes	Yes	Yes[8]	MySQL 5.7, 5.6	Yes[9]	No
db.r3 – Current Generation Memory Optimized Instance Classes														

Instance Class	vCPU	ECU[2]	Memor (GiB)	VPC Only	EBS Opti	Max. Bandw (Mbps)	Networ Perfor	Auro MyS	Auro Post	Mari	Micr SQL Serv	MyS	Orac	PostgreSQL
db.r3.8xlarge	32	104	244	No	No	—	10 Gbps	Yes	No	Yes	Yes[8]	Yes	Yes[9]	Yes
db.r3.4xlarge	16	52	122	No	Yes	2,000	High	Yes	No	Yes	Yes[8]	Yes	Yes[9]	Yes
db.r3.2xlarge	8	26	61	No	Yes	1,000	High	Yes	No	Yes	Yes[8]	Yes	Yes[9]	Yes
db.r3.xlarge	4	13	30.5	No	Yes	500	Moderate	Yes	No	Yes	Yes[8]	Yes	Yes[9]	Yes
db.r3.large	2	6.5	15.25	No	No	—	Moderate	Yes	No	Yes	Yes[8]	Yes	Yes[9]	Yes
db.m2 – Previous Generation Memory Optimized Instance Classes														
db.m2.4xlarge	8	26	68.4	No	Yes	1,000	High	No	No	No	Yes[8]	No	Yes[9]	Yes
db.m2.2xlarge	4	13	34.2	No	Yes	500	Moderate	No	No	No	Yes[8]	No	Yes[9]	Yes
db.m2.xlarge	2	6.5	17.1	No	No	—	Moderate	No	No	No	Yes[8]	No	Yes[9]	Yes
db.t2 – Current Generation Burstable Performance Instance Classes														
db.t2.2xlarge	8	8	32	Yes	No	—	Moderate	No	No	Yes	No	MySQL 5.7, 5.6	Yes[9]	No
db.t2.xlarge	4	4	16	Yes	No	—	Moderate	No	No	Yes	No	MySQL 5.7, 5.6	Yes[9]	No
db.t2.large	2	2	8	Yes	No	—	Moderate	No	No	Yes	Yes[8]	Yes	Yes[9]	Yes
db.t2.medium	2	2	4	Yes	No	—	Moderate	Yes	No	Yes	Yes[8]	Yes	Yes[9]	Yes
db.t2.small	1	1	2	Yes	No	—	Low	Yes	No	Yes	Yes[8]	Yes	Yes[9]	Yes
db.t2.micro	1	1	1	Yes	No	—	Low	No	No	Yes	Yes[8]	Yes	Yes[9]	Yes

1. **vCPU** – The number of virtual central processing units (CPUs). A virtual CPU is a unit of capacity that you can use to compare DB instance classes. Instead of purchasing or leasing a particular processor to use for several months or years, you are renting capacity by the hour. Our goal is to provide a consistent amount of CPU capacity no matter what the actual underlying hardware.
2. **ECU** – The relative measure of the integer processing power of an Amazon EC2 instance. To make it easy for developers to compare CPU capacity between different instance classes, we have defined an Amazon EC2 Compute Unit. The amount of CPU that is allocated to a particular instance is expressed in terms of these EC2 Compute Units. One ECU currently provides CPU capacity equivalent to a 1.0–1.2 GHz 2007 Opteron or 2007 Xeon processor.
3. **Memory (GiB)** – The RAM memory, in gibibytes, allocated to the DB instance. There is often a consistent ratio between memory and vCPU. For example, the db.m1 instance class has the same memory to vCPU ratio as the db.m3 instance class, but for most use cases the db.m3 instance class provides better, more consistent performance, than the db.m1 instance class.
4. **VPC Only** – The instance class is supported only for DB instances that are in a VPC. If your current DB instance is not in a VPC, and you want to use an instance class that requires a VPC, first move your DB instance into a VPC. For more information, see Moving a DB Instance Not in a VPC into a VPC (p. 405).

5. **EBS-Optimized** – The DB instance uses an optimized configuration stack and provides additional, dedicated capacity for I/O. This optimization provides the best performance by minimizing contention between I/O and other traffic from your instance. For more information about Amazon EBS–optimized instances, see Amazon EBS–Optimized Instances in the Amazon EC2 documentation.
6. **Max. Bandwidth (Mbps)** – The maximum bandwidth in megabits per second. Divide by 8 to get the expected throughput in megabytes per second.

   Important
   For general purpose (gp2) storage, the maximum throughput is 1,280 Mbps (160 MB/s).
7. **Network Performance** – The network speed relative to other DB instance classes.
8. **Microsoft SQL Server** – Instance class support varies according to the version and edition of SQL Server. For instance class support by version and edition, see DB Instance Class Support for Microsoft SQL Server (p. 723).
9. **Oracle** – Instance class support varies according to the version and edition of Oracle. For instance class support by version and edition, see DB Instance Class Support for Oracle (p. 934).

Changing Your DB Instance Class

You can change the CPU and memory available to a DB instance by changing its DB instance class. To change the DB instance class, modify your DB instance by following the instructions for your specific database engine.

- Modifying a DB Instance Running the MariaDB Database Engine (p. 691)
- Modifying a DB Instance Running the Microsoft SQL Server Database Engine (p. 756)
- Modifying a DB Instance Running the MySQL Database Engine (p. 843)
- Modifying a DB Instance Running the Oracle Database Engine (p. 967)
- Modifying a DB Instance Running the PostgreSQL Database Engine (p. 1183)

MySQL DB instances created after April 23, 2014, can change to the db.r3 instance class by modifying the DB instance just as with any other modification. MySQL DB instances running MySQL versions 5.5 and created before April 23, 2014, must first upgrade to MySQL version 5.6. For more information, see Upgrading the MySQL DB Engine (p. 851).

Some instance classes require that your DB instance is in a VPC. If your current DB instance is not in a VPC, and you want to use an instance class that requires a VPC, first move your DB instance into a VPC. For more information, see Moving a DB Instance Not in a VPC into a VPC (p. 405).

Related Topics

- DB Instance RAM Recommendations (p. 81)
- Storage for Amazon RDS (p. 410)

DB Instance Status

The status of a DB instance indicates the health of the instance. You can view the status of a DB instance by using the RDS console, the AWS CLI command describe-db-instances, or the API action DescribeDBInstances.

Note
Amazon RDS also uses another status called *maintenance status*, which is shown in the Maintenance column of the Amazon RDS console. This value indicates the status of any

maintenance patches that need to be applied to a DB instance. Maintenance status is independent of DB instance status. For more information on *maintenance status*, see Updating the Operating System for a DB Instance or DB Cluster.

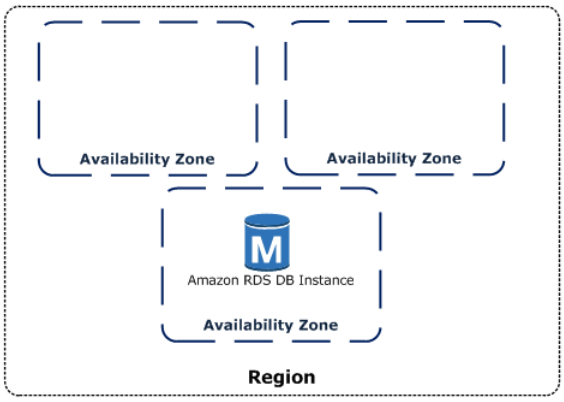
DB Instance Status	Description
available	The instance is healthy and available.
backing-up	The instance is currently being backed up.
configuring-enhanced-monitoring	Enhanced Monitoring is being enabled or disabled for this instance.
creating	The instance is being created. The instance is inaccessible while it is being created.
deleting	The instance is being deleted.
failed	The instance has failed and Amazon RDS was unable to recover it. Perform a point-in-time restore to the latest restorable time of the instance to recover the data.
inaccessible-encryption-credentials	The KMS key used to encrypt or decrypt the DB instance could not be accessed.
incompatible-credentials	The supplied CloudHSM Classic user name or password is incorrect. Please update the CloudHSM Classic credentials for the DB instance.
incompatible-network	Amazon RDS is attempting to perform a recovery action on an instance but is unable to do so because the VPC is in a state that is preventing the action from being completed. This status can occur if, for example, all available IP addresses in a subnet were in use and Amazon RDS was unable to get an IP address for the DB instance.
incompatible-option-group	Amazon RDS attempted to apply an option group change but was unable to do so, and Amazon RDS was unable to roll back to the previous option group state. Consult the Recent Events list for the DB instance for more information. This status can occur if, for example, the option group contains an option such as TDE and the DB instance does not contain encrypted information.
incompatible-parameters	Amazon RDS was unable to start up the DB instance because the parameters specified in the instance's DB parameter group were not compatible. Revert the parameter changes or make them compatible with the instance to regain access to your instance. Consult the Recent Events list for the DB instance for more information about the incompatible parameters.
incompatible-restore	Amazon RDS is unable to do a point-in-time restore. Common causes for this status include using temp tables, using MyISAM tables with MySQL, or using Aria tables with MariaDB.
maintenance	Amazon RDS is applying a maintenance update to the DB instance. This status is used for instance-level maintenance that RDS schedules well in advance. We're evaluating ways to expose additional maintenance actions to customers through this status.
modifying	The instance is being modified because of a customer request to modify the instance.

DB Instance Status	Description
rebooting	The instance is being rebooted because of a customer request or an Amazon RDS process that requires the rebooting of the instance.
renaming	The instance is being renamed because of a customer request to rename it.
resetting-master-credentials	The master credentials for the instance are being reset because of a customer request to reset them.
restore-error	The DB instance encountered an error attempting to restore to a point-in-time or from a snapshot.
starting	The DB instance is starting.
stopping	The DB instance is being stopped.
stopped	The DB instance is stopped.
storage-full	The instance has reached its storage capacity allocation. This is a critical status and should be remedied immediately; you should scale up your storage by modifying the DB instance. Set CloudWatch alarms to warn you when storage space is getting low so you don't run into this situation.
storage-optimization	Your DB instance is being modified to change the storage size or type. The DB instance is fully operational, but while the status of your DB instance is storage-optimization, you can't request any changes to the storage of your DB instance. The storage optimization process is usually short, but can sometimes take up to and even beyond 24 hours.
upgrading	The database engine version is being upgraded.

Regions and Availability Zones

Amazon cloud computing resources are hosted in multiple locations world-wide. These locations are composed of AWS Regions and Availability Zones. Each *AWS Region* is a separate geographic area. Each AWS Region has multiple, isolated locations known as *Availability Zones*. Amazon RDS provides you the ability to place resources, such as instances, and data in multiple locations. Resources aren't replicated across AWS Regions unless you do so specifically.

Amazon operates state-of-the-art, highly-available data centers. Although rare, failures can occur that affect the availability of instances that are in the same location. If you host all your instances in a single location that is affected by such a failure, none of your instances would be available.



It is important to remember that each AWS Region is completely independent. Any Amazon RDS activity you initiate (for example, creating database instances or listing available database instances) runs only in your current default AWS Region. The default AWS Region can be changed in the console, by setting the EC2_REGION environment variable, or it can be overridden by using the `--region` parameter with the AWS Command Line Interface. See Configuring the AWS Command Line Interface, specifically, the sections on Environment Variables and Command Line Options for more information.

Amazon RDS supports a special AWS Region called AWS GovCloud (US) that is designed to allow US government agencies and customers to move more sensitive workloads into the cloud. AWS GovCloud (US) addresses the US government's specific regulatory and compliance requirements. For more information about AWS GovCloud (US), see What Is AWS GovCloud (US)?

To create or work with an Amazon RDS DB instance in a specific AWS Region, use the corresponding regional service endpoint.

Amazon RDS supports the endpoints listed in the following table.

Region	Name	Endpoint
US West (Oregon) Region	us-west-2	https://rds.us-west-2.amazonaws.com
US West (N. California) Region	us-west-1	https://rds.us-west-1.amazonaws.com
US East (Ohio) Region	us-east-2	https://rds.us-east-2.amazonaws.com
US East (N. Virginia) Region	us-east-1	https://rds.us-east-1.amazonaws.com
Asia Pacific (Mumbai) Region	ap-south-1	https://rds.ap-south-1.amazonaws.com
Asia Pacific (Seoul) Region	ap-northeast-2	https://rds.ap-northeast-2.amazonaws.com
Asia Pacific (Singapore) Region	ap-southeast-1	https://rds.ap-southeast-1.amazonaws.com
Asia Pacific (Sydney) Region	ap-southeast-2	https://rds.ap-southeast-2.amazonaws.com
Asia Pacific (Tokyo) Region	ap-northeast-1	https://rds.ap-northeast-1.amazonaws.com
Canada (Central) Region	ca-central-1	https://rds.ca-central-1.amazonaws.com
China (Beijing) Region	cn-north-1	https://rds.cn-north-1.amazonaws.com.cn
EU (Frankfurt) Region	eu-central-1	https://rds.eu-central-1.amazonaws.com
EU (Ireland) Region	eu-west-1	https://rds.eu-west-1.amazonaws.com

Region	Name	Endpoint
EU (London) Region	eu-west-2	https://rds.eu-west-2.amazonaws.com
South America (São Paulo) Region	sa-east-1	https://rds.sa-east-1.amazonaws.com
AWS GovCloud (US)	us-gov-west-1	https://rds.us-gov-west-1.amazonaws.com

If you do not explicitly specify an endpoint, the US West (Oregon) endpoint is the default.

Related Topics

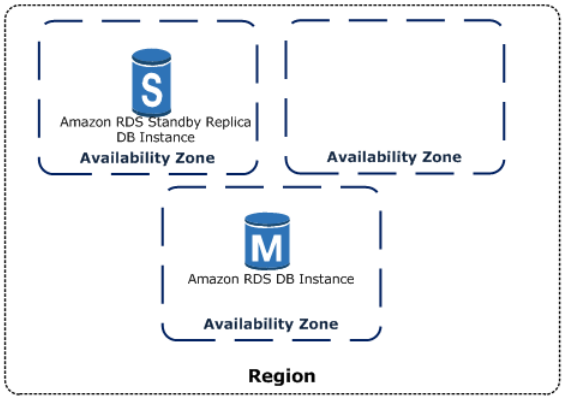
- Regions and Availability Zones in the *Amazon Elastic Compute Cloud User Guide*.
- Amazon RDS DB Instances (p. 90)

High Availability (Multi-AZ)

Amazon RDS provides high availability and failover support for DB instances using Multi-AZ deployments. Amazon RDS uses several different technologies to provide failover support. Multi-AZ deployments for Oracle, PostgreSQL, MySQL, and MariaDB DB instances use Amazon's failover technology. SQL Server DB instances use SQL Server Mirroring. Amazon Aurora instances stores copies of the data in a DB cluster across multiple Availability Zones in a single AWS Region, regardless of whether the instances in the DB cluster span multiple Availability Zones. For more information on Amazon Aurora, see Amazon Aurora on Amazon RDS (p. 428).

In a Multi-AZ deployment, Amazon RDS automatically provisions and maintains a synchronous standby replica in a different Availability Zone. The primary DB instance is synchronously replicated across Availability Zones to a standby replica to provide data redundancy, eliminate I/O freezes, and minimize latency spikes during system backups. Running a DB instance with high availability can enhance availability during planned system maintenance, and help protect your databases against DB instance failure and Availability Zone disruption. For more information on Availability Zones, see Regions and Availability Zones (p. 97).

Note
The high-availability feature is not a scaling solution for read-only scenarios; you cannot use a standby replica to serve read traffic. To service read-only traffic, you should use a Read Replica. For more information, see Working with PostgreSQL, MySQL, and MariaDB Read Replicas (p. 134).



Using the RDS console, you can create a Multi-AZ deployment by simply specifying Multi-AZ when creating a DB instance. You can also use the console to convert existing DB instances to Multi-AZ

deployments by modifying the DB instance and specifying the Multi-AZ option. The RDS console shows the Availability Zone of the standby replica, called the secondary AZ.

You can specify a Multi-AZ deployment using the CLI as well. Use the AWS CLI describe-db-instances command, or the Amazon RDS API DescribeDBInstances action to show the Availability Zone of the standby replica (called the secondary AZ).

The RDS console shows the Availability Zone of the standby replica (called the secondary AZ), or you can use the AWS CLI describe-db-instances command, or the Amazon RDS API DescribeDBInstances action to find the secondary AZ.

DB instances using Multi-AZ deployments may have increased write and commit latency compared to a Single-AZ deployment, due to the synchronous data replication that occurs. You may have a change in latency if your deployment fails over to the standby replica, although AWS is engineered with low-latency network connectivity between Availability Zones. For production workloads, we recommend that you use Provisioned IOPS and DB instance classes (m1.large and larger) that are optimized for Provisioned IOPS for fast, consistent performance.

Modifying a DB Instance to Be a Multi-AZ Deployment

If you have a DB instance in a Single-AZ deployment and you modify it to be a Multi-AZ deployment (for engines other than SQL Server or Amazon Aurora), Amazon RDS takes several steps. First, Amazon RDS takes a snapshot of the primary DB instance from your deployment and then restores the snapshot into another Availability Zone. Amazon RDS then sets up synchronous replication between your primary DB instance and the new instance. This action avoids downtime when you convert from Single-AZ to Multi-AZ, but you can experience a significant performance impact when first converting to Multi-AZ. This impact is more noticeable for large and write-intensive DB instances.

Once the modification is complete, Amazon RDS triggers an event (RDS-EVENT-0025) that indicates the process is complete. You can monitor Amazon RDS events; for more information about events, see Using Amazon RDS Event Notification (p. 279).

Failover Process for Amazon RDS

In the event of a planned or unplanned outage of your DB instance, Amazon RDS automatically switches to a standby replica in another Availability Zone if you have enabled Multi-AZ. The time it takes for the failover to complete depends on the database activity and other conditions at the time the primary DB instance became unavailable. Failover times are typically 60-120 seconds. However, large transactions or a lengthy recovery process can increase failover time. When the failover is complete, it can take additional time for the RDS console UI to reflect the new Availability Zone.

The failover mechanism automatically changes the DNS record of the DB instance to point to the standby DB instance. As a result, you need to re-establish any existing connections to your DB instance. Due to how the Java DNS caching mechanism works, you may need to reconfigure your JVM environment. For more information on how to manage a Java application that caches DNS values in the case of a failover, see the AWS SDK for Java.

Amazon RDS handles failovers automatically so you can resume database operations as quickly as possible without administrative intervention. The primary DB instance switches over automatically to the standby replica if any of the following conditions occur:

- An Availability Zone outage
- The primary DB instance fails
- The DB instance's server type is changed
- The operating system of the DB instance is undergoing software patching

- A manual failover of the DB instance was initiated using **Reboot with failover**

There are several ways to determine if your Multi-AZ DB instance has failed over:

- DB event subscriptions can be setup to notify you via email or SMS that a failover has been initiated. For more information about events, see Using Amazon RDS Event Notification (p. 279)
- You can view your DB events by using the Amazon RDS console or API actions.
- You can view the current state of your Multi-AZ deployment by using the Amazon RDS console and API actions.

For information on how you can respond to failovers, reduce recovery time, and other best practices for Amazon RDS, see Best Practices for Amazon RDS (p. 80).

Related Topics

- Multi-AZ Deployments for Microsoft SQL Server with Database Mirroring (p. 787)
- Licensing Microsoft SQL Server Multi-AZ Deployments (p. 735)
- Licensing Oracle Multi-AZ Deployments (p. 934)

DB Instance and DB Cluster Maintenance

Changes to a DB instance or DB cluster can occur when you manually modify a DB instance or DB cluster, such as when you upgrade the engine version, or when Amazon RDS performs maintenance on a DB instance or DB cluster. Following, you can find information on how to upgrade an engine version and also information on how Amazon RDS performs required maintenance.

Topics

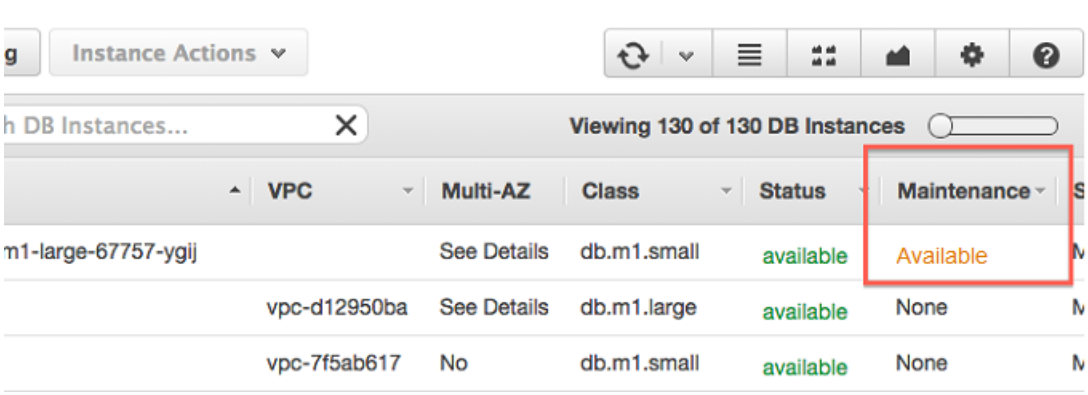
Amazon RDS Maintenance

Periodically, Amazon RDS performs maintenance on Amazon RDS resources. Maintenance most often involves updates to the DB instance's or DB cluster's underlying operating system (OS). Updates to the operating system most often occur for security issues and should be done as soon as possible.

Maintenance items require that Amazon RDS take your DB instance or DB cluster offline for a short time. Maintenance that require a resource to be offline include scale compute operations, which generally take only a few minutes from start to finish, and required operating system or database patching. Required patching is automatically scheduled only for patches that are related to security and instance reliability. Such patching occurs infrequently (typically once every few months) and seldom requires more than a fraction of your maintenance window.

DB instances are not automatically backed up when an OS update is applied, so you should back up your DB instances before you apply an update.

You can view whether a maintenance update is available for your DB instance or DB cluster by using the RDS console, the AWS CLI, or the Amazon RDS API. If an update is available, it is indicated by the word **Available** or **Required** in the **Maintenance** column for the DB instance or DB cluster on the Amazon RDS console, as shown following:



If an update is available, you can take one of the actions in the following table.

Action	Notes
Defer the maintenance items.	Certain OS updates are marked as **Required**. If you defer a required update, you receive a notice from Amazon RDS indicating when the update will be performed on your DB instance or DB cluster. Other updates are **Available**. You can defer these updates indefinitely.

Action	Notes
Apply the maintenance items immediately.	For instructions, see Updating the Operating System for a DB Instance or DB Cluster (p. 108).
Schedule the maintenance items to start during your next maintenance window.	For instructions, see Updating the Operating System for a DB Instance or DB Cluster (p. 108).
Take no action.	The updates are applied during your next maintenance window.

The maintenance window determines when pending operations start, but does not limit the total execution time of these operations. Maintenance operations are not guaranteed to finish before the maintenance window ends, and can continue beyond the specified end time. For more information, see The Amazon RDS Maintenance Window (p. 103).

Multi-AZ Deployments for RDS DB Instances

Running a DB instance as a Multi-AZ deployment can further reduce the impact of a maintenance event, because Amazon RDS will conduct maintenance by following these steps:

1. Perform maintenance on the standby.
2. Promote the standby to primary.
3. Perform maintenance on the old primary, which becomes the new standby.

When you modify the database engine for your DB instance in a Multi-AZ deployment, then Amazon RDS upgrades both the primary and secondary DB instances at the same time. In this case, the database engine for the entire Multi-AZ deployment is shut down during the upgrade.

For more information on Multi-AZ deployments, see High Availability (Multi-AZ) (p. 99).

An Amazon Aurora DB cluster spans multiple Availability Zones (AZs) by default and maintenance is performed on all instances in an Aurora DB cluster during the cluster maintenance window.

The Amazon RDS Maintenance Window

Every DB instance and DB cluster has a weekly maintenance window during which any system changes are applied. You can think of the maintenance window as an opportunity to control when modifications and software patching occur, in the event either are requested or required. If a maintenance event is scheduled for a given week, it is initiated during the 30-minute maintenance window you identify. Most maintenance events also complete during the 30-minute maintenance window, although larger maintenance events may take more than 30 minutes to complete.

The 30-minute maintenance window is selected at random from an 8-hour block of time per region. If you don't specify a preferred maintenance window when you create the DB instance or DB cluster, then Amazon RDS assigns a 30-minute maintenance window on a randomly selected day of the week.

RDS will consume some of the resources on your DB instance or DB cluster while maintenance is being applied. You might observe a minimal effect on performance. For a DB instance, on rare occasions, a Multi-AZ failover might be required for a maintenance update to complete.

Following, you can find the time blocks for each region from which default maintenance windows are assigned.

Region	Time Block
US West (Oregon) Region	06:00–14:00 UTC
US West (N. California) Region	06:00–14:00 UTC
US East (Ohio) Region	03:00–11:00 UTC
US East (N. Virginia) Region	03:00–11:00 UTC
Asia Pacific (Mumbai) Region	17:30–01:30 UTC
Asia Pacific (Seoul) Region	13:00–21:00 UTC
Asia Pacific (Singapore) Region	14:00–22:00 UTC
Asia Pacific (Sydney) Region	12:00–20:00 UTC
Asia Pacific (Tokyo) Region	13:00–21:00 UTC
Canada (Central) Region	06:29–14:29 UTC
EU (Frankfurt) Region	23:00–07:00 UTC
EU (Ireland) Region	22:00–06:00 UTC
EU (London) Region	06:00–14:00 UTC
South America (São Paulo) Region	00:00–08:00 UTC
AWS GovCloud (US)	06:00–14:00 UTC

Adjusting the Preferred DB Instance Maintenance Window

The maintenance window should fall at the time of lowest usage and thus might need modification from time to time. Your DB instance will only be unavailable during this time if the system changes, such as a scale storage operation or a change in DB instance class, are being applied and require an outage, and only for the minimum amount of time required to make the necessary changes.

> **Note**
> For upgrades to the database engine, Amazon Aurora manages the preferred maintenance window for a DB cluster and not individual instances. For information on adjusting the maintenance window for Aurora, see Adjusting the Preferred DB Cluster Maintenance Window (p. 106).

In the following example, you adjust the preferred maintenance window for a DB instance.

For the purpose of this example, we assume that the DB instance named *mydbinstance* exists and has a preferred maintenance window of "Sun:05:00-Sun:06:00" UTC.

AWS Management Console

To adjust the preferred maintenance window

1. Launch the AWS Management Console.

a. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
b. Click on the **DB Instances** link in the Navigation panel on the left side of the console display.

   The **My Instances** list appears.
c. Right-click on the **DB Instance** in the **My DB Instances** list and select **Modify** from the drop-down menu.

   The **Modify DB Instance** window appears.

2. Type the maintenance window into the Maintenance Window text box using the format "day:hour:minute-day:hour:minute".

   Note
   The maintenance window and the backup window for the DB instance cannot overlap. If you enter a value for the maintenance window that overlaps the backup window, an error message appears.
3. Click the **OK** button.

   Changes to the maintenance window take effect immediately.

CLI

To adjust the preferred maintenance window, use the AWS CLI `modify-db-instance` command with the following parameters:

- `--db-instance-identifier`
- `--preferred-maintenance-window`

Example

The following code example sets the maintenance window to Tuesdays from 4:00-4:30AM UTC.

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
--db-instance-identifier mydbinstance \
--preferred-maintenance-window Tue:04:00-Tue:04:30
```

For Windows:

```
aws rds modify-db-instance ^
--db-instance-identifier mydbinstance ^
--preferred-maintenance-window Tue:04:00-Tue:04:30
```

API

To adjust the preferred maintenance window, use the Amazon RDS API `ModifyDBInstance` action with the following parameters:

- `DBInstanceIdentifier = mydbinstance`
- `PreferredMaintenanceWindow = Tue:04:00-Tue:04:30`

Example

The following code example sets the maintenance window to Tuesdays from 4:00-4:30AM UTC.

```
https://rds.us-west-2.amazonaws.com/
  ?Action=ModifyDBInstance
  &DBInstanceIdentifier=mydbinstance
  &PreferredMaintenanceWindow=Tue:04:00-Tue:04:30
  &SignatureMethod=HmacSHA256
  &SignatureVersion=4
  &Version=2014-09-01
  &X-Amz-Algorithm=AWS4-HMAC-SHA256
  &X-Amz-Credential=AKIADQKE4SARGYLE/20140425/us-east-1/rds/aws4_request
  &X-Amz-Date=20140425T192732Z
  &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
  &X-Amz-Signature=1dc9dd716f4855e9bdf188c70f1cf9f6251b070b68b81103b59ec70c3e7854b3
```

Adjusting the Preferred DB Cluster Maintenance Window

The Aurora DB cluster maintenance window should fall at the time of lowest usage and thus might need modification from time to time. Your DB cluster is unavailable during this time only if the updates that are being applied require an outage. The outage is for the minimum amount of time required to make the necessary updates.

AWS Management Console

To adjust the preferred DB cluster maintenance window

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. Choose **Clusters** on the left of the console.
3. Choose the DB cluster that you want to adjust the preferred maintenance window for.
4. Choose **Modify Cluster**.
5. In the **Maintenance** section of the console, set the **Start Day**, **Start Time**, and **Duration** to the values for your new, preferred maintenance window.
6. Choose **Apply Immediately**, and then choose **Continue**.
7. Verify your updated values, and then choose **Modify Cluster**.

CLI

To adjust the preferred DB cluster maintenance window, use the AWS CLI `modify-db-cluster` command with the following parameters:

- `--db-cluster-identifier`
- `--preferred-maintenance-window`

Example

The following code example sets the maintenance window to Tuesdays from 4:00-4:30AM UTC.

For Linux, OS X, or Unix:

```
aws rds modify-db-cluster \
--db-cluster-identifier my-cluster \
--preferred-maintenance-window Tue:04:00-Tue:04:30
```

For Windows:

```
aws rds modify-db-cluster ^
```

```
--db-cluster-identifier my-cluster ^
--preferred-maintenance-window Tue:04:00-Tue:04:30
```

API

To adjust the preferred DB cluster maintenance window, use the Amazon RDS API `ModifyDBCluster` action with the following parameters:

- `DBClusterIdentifier = my-cluster`
- `PreferredMaintenanceWindow = Tue:04:00-Tue:04:30`

Example

The following code example sets the maintenance window to Tuesdays from 4:00-4:30AM UTC.

```
https://rds.us-west-2.amazonaws.com/
    ?Action=ModifyDBCluster
    &DBClusterIdentifier=my-cluster
    &PreferredMaintenanceWindow=Tue:04:00-Tue:04:30
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-10-31
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20140725/us-east-1/rds/aws4_request
    &X-Amz-Date=20161017T161457Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=d6d1c65c2e94f5800ab411a3f7336625820b103713b6c063430900514e21d784
```

Related Topics

- Updating the Operating System for a DB Instance or DB Cluster (p. 108)
- Upgrading a DB Instance Engine Version (p. 115)

Updating the Operating System for a DB Instance or DB Cluster

With Amazon RDS, you can choose when to update the underlying operating system. You can decide when Amazon RDS applies OS updates by using the RDS console, AWS Command Line Interface (AWS CLI), or RDS API.

Use the procedures in this topic to immediately upgrade or schedule an upgrade for your DB instance. For more information, see Amazon RDS Maintenance (p. 102).

AWS Management Console

To manage an OS update for a DB instance or DB cluster

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Instances** to manage updates for a DB instance, or **Clusters** to manage updates for an Aurora DB cluster.
3. Select the check box for the DB instance or DB cluster that has a required operating system update.
4. Choose **Instance Actions** for a DB instance, or **Cluster Actions** for a DB cluster, and then choose one of the following:
   - **Upgrade Now**
   - **Upgrade at Next Window**

     Note
     If you choose **Upgrade at Next Window** and later want to delay the OS update, you can select **Defer Upgrade**.

CLI

To apply a pending OS update to a DB instance or DB cluster, use the apply-pending-maintenance-action AWS CLI command.

Example

For Linux, OS X, or Unix:

```
aws rds apply-pending-maintenance-action \
    --resource-identifier arn:aws:rds:us-west-2:001234567890:db:mysql-db \
    --apply-action system-update \
    --opt-in-type immediate
```

For Windows:

```
aws rds apply-pending-maintenance-action ^
    --resource-identifier arn:aws:rds:us-west-2:001234567890:db:mysql-db ^
    --apply-action system-update ^
    --opt-in-type immediate
```

To return a list of resources that have at least one pending OS update, use the describe-pending-maintenance-actions AWS CLI command.

Example

For Linux, OS X, or Unix:

```
aws rds describe-pending-maintenance-actions \
    --resource-identifier arn:aws:rds:us-west-2:001234567890:db:mysql-db
```

For Windows:

```
aws rds describe-pending-maintenance-actions ^
    --resource-identifier arn:aws:rds:us-west-2:001234567890:db:mysql-db
```

You can also return a list of resources for a DB instance or DB cluster by specifying the `--filters` parameter of the `describe-pending-maintenance-actions` AWS CLI command. The format for the `--filters` command is `Name=`*`filter-name`*`,Value=`*`resource-id`*`,...`.

The following are the accepted values for the `Name` parameter of a filter:

- `db-instance-id` – Accepts a list of DB instance identifiers or Amazon Resource Names (ARNs). The returned list only includes pending maintenance actions for the DB instances identified by these identifiers or ARNs.
- `db-cluster-id` – Accepts a list of DB cluster identifiers or ARNs. The returned list only includes pending maintenance actions for the DB clusters identified by these identifiers or ARNs.

For example, the following example returns the pending maintenance actions for the `sample-cluster1` and `sample-cluster2` DB clusters.

Example

For Linux, OS X, or Unix:

```
aws rds describe-pending-maintenance-actions \
 --filters Name=db-cluster-id,Values=sample-cluster1,sample-cluster2
```

For Windows:

```
aws rds describe-pending-maintenance-actions ^
 --filters Name=db-cluster-id,Values=sample-cluster1,sample-cluster2
```

API

To apply an OS update to a DB instance or DB cluster, call the Amazon RDS API `ApplyPendingMaintenanceAction` action.

Example

```
https://rds.us-west-2.amazonaws.com/
   ?Action=ApplyPendingMaintenanceAction
   &ResourceIdentifier=arn:aws:rds:us-east-1:123456781234:db:my-instance
   &ApplyAction=system-update
   &OptInType=immediate
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &Version=2014-10-31
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20141216/us-west-2/rds/aws4_request
```

```
   &X-Amz-Date=20140421T194732Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
   &X-Amz-Signature=6e25c542bf96fe24b28c12976ec92d2f856ab1d2a158e21c35441a736e4fde2b
```

To return a list of resources that have at least one pending OS update, call the Amazon RDS API DescribePendingMaintenanceActions action.

Example

```
https://rds.us-west-2.amazonaws.com/
   ?Action=DescribePendingMaintenanceActions
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &Version=2014-10-31
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20141216/us-west-2/rds/aws4_request
   &X-Amz-Date=20140421T194732Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
   &X-Amz-Signature=6e25c542bf96fe24b28c12976ec92d2f856ab1d2a158e21c35441a736e4fde2b
```

Related Topics

- Amazon RDS Maintenance (p. 102)
- Upgrading a DB Instance Engine Version (p. 115)

Amazon RDS DB Instance Lifecycle

The lifecycle of an Amazon RDS DB instance includes creating, modifying, maintaining and upgrading, performing backups and restores, rebooting, and deleting the instance. This section provides information on and links to more about these processes.

Topics

Creating an Amazon RDS DB Instance

The basic building block of Amazon RDS is the DB instance. To create an Amazon RDS DB instance, follow the instructions for your specific database engine.

- Creating an Amazon Aurora DB Cluster (p. 437)
- Creating a DB Instance Running the MariaDB Database Engine (p. 678)
- Creating a DB Instance Running the Microsoft SQL Server Database Engine (p. 738)
- Creating a DB Instance Running the MySQL Database Engine (p. 830)
- Creating a DB Instance Running the Oracle Database Engine (p. 949)
- Creating a DB Instance Running the PostgreSQL Database Engine (p. 1172)

Connecting to an Amazon RDS DB Instance

After you create an Amazon RDS DB instance, you can use any standard SQL client application to connect to the DB instance. To connect to an Amazon RDS DB instance, follow the instructions for your specific database engine.

Modifying an Amazon RDS DB Instance and Using the Apply Immediately Parameter

Most modifications to a DB instance can be applied immediately or deferred until the next maintenance window. Some modifications, such as parameter group changes, require that you manually reboot your DB instance for the change to take effect.

> **Important**
> Some modifications result in an outage because Amazon RDS must reboot your DB instance for the change to take effect. Review the impact to your database and applications before modifying your DB instance settings.

To modify an Amazon RDS DB instance, follow the instructions for your specific database engine.

- Modifying a DB Instance Running the MariaDB Database Engine (p. 691)
- Modifying a DB Instance Running the Microsoft SQL Server Database Engine (p. 756)
- Modifying a DB Instance Running the MySQL Database Engine (p. 843)
- Modifying a DB Instance Running the Oracle Database Engine (p. 967)
- Modifying a DB Instance Running the PostgreSQL Database Engine (p. 1183)

The Impact of Apply Immediately

When you modify a DB instance, you can apply the changes immediately. To apply changes immediately, you select the **Apply Immediately** option in the AWS Management Console, you use the `--apply-immediately` parameter when calling the AWS CLI, or you set the `ApplyImmediately` parameter to `true` when using the Amazon RDS API.

If you don't choose to apply changes immediately, the changes are put into the pending modifications queue. During the next maintenance window, any pending changes in the queue are applied.

> **Important**
> If you choose to apply changes immediately, any changes in the pending modifications queue are also applied. If any of the pending modifications require downtime, choosing apply immediately can cause unexpected downtime.

Related Topics

- Renaming a DB Instance (p. 116)
- Rebooting a DB Instance (p. 119)
- Stopping an Amazon RDS DB Instance Temporarily (p. 121)
- modify-db-instance
- ModifyDBInstance

Upgrading a DB Instance Engine Version

When Amazon Relational Database Service (Amazon RDS) supports a new version of a database engine, you can upgrade your DB instances to the new version. There are two kinds of upgrades: major version upgrades and minor version upgrades. For more information about major and minor version upgrades, see the following documentation for your DB engine:

- Amazon Aurora Updates (p. 664)
- Upgrading the MariaDB DB Engine (p. 699)
- Upgrading the Microsoft SQL Server DB Engine (p. 764)
- Upgrading the MySQL DB Engine (p. 851)
- Upgrading the Oracle DB Engine (p. 975)
- Upgrading the PostgreSQL DB Engine (p. 1191)

Related Topics

- Amazon RDS Maintenance (p. 102)
- Updating the Operating System for a DB Instance or DB Cluster (p. 108)

Renaming a DB Instance

You can rename a DB instance by using the AWS Management Console, the AWS CLI `modify-db-instance` command, or the Amazon RDS API `ModifyDBInstance` action. Renaming a DB instance can have far-reaching effects; the following is a list of things you should know before you rename a DB instance.

- When you rename a DB instance, the endpoint for the DB instance changes, because the URL includes the name you assigned to the DB instance. You should always redirect traffic from the old URL to the new one.
- When you rename a DB instance, the old DNS name that was used by the DB instance is immediately deleted, although it could remain cached for a few minutes. The new DNS name for the renamed DB instance becomes effective in about 10 minutes. The renamed DB instance is not available until the new name becomes effective.
- You cannot use an existing DB instance name when renaming an instance.
- All read replicas associated with a DB instance remain associated with that instance after it is renamed. For example, suppose you have a DB instance that serves your production database and the instance has several associated read replicas. If you rename the DB instance and then replace it in the production environment with a DB snapshot, the DB instance that you renamed will still have the read replicas associated with it.
- Metrics and events associated with the name of a DB instance are maintained if you reuse a DB instance name. For example, if you promote a Read Replica and rename it to be the name of the previous master, the events and metrics associated with the master are associated with the renamed instance.
- DB instance tags remain with the DB instance, regardless of renaming.
- DB snapshots are retained for a renamed DB instance.

Renaming to Replace an Existing DB Instance

The most common reasons for renaming a DB instance are that you are promoting a Read Replica or you are restoring data from a DB snapshot or PITR. By renaming the database, you can replace the DB instance without having to change any application code that references the DB instance. In these cases, you would do the following:

1. Stop all traffic going to the master DB instance. This can involve redirecting traffic from accessing the databases on the DB instance or some other way you want to use to prevent traffic from accessing your databases on the DB instance.
2. Rename the master DB instance to a name that indicates it is no longer the master as described later in this topic.
3. Create a new master DB instance by restoring from a DB snapshot or by promoting a read replica, and then give the new instance the name of the previous master DB instance.
4. Associate any read replicas with the new master DB instance.

If you delete the old master DB instance, you are responsible for deleting any unwanted DB snapshots of the old master instance.

For information about promoting a Read Replica, see Promoting a Read Replica to Be a DB Instance (p. 140).

AWS Management Console

To rename a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, select **DB Instances**.
3. Select the check box next to the DB instance you want to rename.
4. From the **Instance Actions** dropdown menu, select **Modify**.
5. Enter a new name in the **DB Instance Identifier** text box. Select the **Apply Immediately** check box, and then click **Continue**.
6. Click **Modify DB Instance** to complete the change.

CLI

To rename a DB instance, use the AWS CLI command `modify-db-instance`. Provide the current `--db-instance-identifier` value and `--new-db-instance-identifier` parameter with the new name of the DB instance.

Example

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier DBInstanceIdentifier \
    --new-db-instance-identifier NewDBInstanceIdentifier
```

For Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier DBInstanceIdentifier ^
    --new-db-instance-identifier NewDBInstanceIdentifier
```

API

To rename a DB instance, call Amazon RDS API function `ModifyDBInstance` with the following parameters:

- `DBInstanceIdentifier` = existing name for the instance
- `NewDBInstanceIdentifier` = new name for the instance

```
https://rds.amazonaws.com/
 ?Action=ModifyDBInstance
 &DBInstanceIdentifier=mydbinstance
 &NewDBInstanceIdentifier=mynewdbinstanceidentifier
 &Version=2012-01-15
 &SignatureVersion=2
 &SignatureMethod=HmacSHA256
 &Timestamp=2012-01-20T22%3A06%3A23.624Z
 &AWSAccessKeyId=<AWS Access Key ID>
 &Signature=<Signature>
```

Related Topics

- Modifying a DB Instance Running the MariaDB Database Engine (p. 691)
- Modifying a DB Instance Running the Microsoft SQL Server Database Engine (p. 756)
- Modifying a DB Instance Running the MySQL Database Engine (p. 843)
- Modifying a DB Instance Running the Oracle Database Engine (p. 967)
- Modifying a DB Instance Running the PostgreSQL Database Engine (p. 1183)

Rebooting a DB Instance

You might need to reboot your DB instance, usually for maintenance reasons. For example, if you make certain modifications, or if you change the DB parameter group associated with the DB instance, you must reboot the instance for the changes to take effect.

Rebooting a DB instance restarts the database engine service. Rebooting a DB instance results in a momentary outage, during which the DB instance status is set to *rebooting*. If the Amazon RDS instance is configured for Multi-AZ, the reboot can be conducted with a failover. An Amazon RDS event is created when the reboot is completed.

If your DB instance is a Multi-AZ deployment, you can force a failover from one availability zone to another when you reboot. When you force a failover of your DB instance, Amazon RDS automatically switches to a standby replica in another Availability Zone, and updates the DNS record for the DB instance to point to the standby DB instance. As a result, you need to clean up and re-establish any existing connections to your DB instance. Rebooting with failover is beneficial when you want to simulate a failure of a DB instance for testing, or restore operations to the original AZ after a failover occurs. For more information, see High Availability (Multi-AZ) (p. 99).

When you reboot the primary instance of an Amazon Aurora DB cluster, RDS also automatically reboots all of the Aurora Replicas in that DB cluster. When you reboot the primary instance of an Aurora DB cluster, no failover occurs. When you reboot an Aurora Replica, no failover occurs. To failover an Aurora DB cluster, call the AWS CLI command `failover-db-cluster`, or the API action `FailoverDBCluster`.

You can't reboot your DB instance if it is not in the "Available" state. Your database can be unavailable for several reasons, such as an in-progress backup, a previously requested modification, or a maintenance-window action.

The time required to reboot your DB instance depends on the crash recovery process of your specific database engine. To improve the reboot time, we recommend that you reduce database activities as much as possible during the reboot process Reduce database activity reduces rollback activity for in-transit transactions.

AWS Management Console

To reboot a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Instances**, and then select the DB instance that you want to reboot.
3. Choose **Instance Actions** and then choose **Reboot**.

   The **Reboot DB Instance** page appears.
4. (Optional) Select **Reboot with failover?** to force a failover from one AZ to another.
5. Choose **Reboot** to reboot your DB instance.

   Alternatively, choose **Cancel**.

CLI

To reboot a DB instance by using the AWS CLI, call the `reboot-db-instance` command.

Example Simple Reboot

For Linux, OS X, or Unix:

```
aws rds reboot-db-instance \
    --db-instance-identifier mydbinstance
```

For Windows:

```
aws rds reboot-db-instance ^
    --db-instance-identifier mydbinstance
```

Example Reboot with Failover

To force a failover from one AZ to the other, use the `--force-failover` parameter.

For Linux, OS X, or Unix:

```
aws rds reboot-db-instance \
    --db-instance-identifier mydbinstance \
    --force-failover
```

For Windows:

```
aws rds reboot-db-instance ^
    --db-instance-identifier mydbinstance ^
    --force-failover
```

API

To reboot a DB instance by using the Amazon RDS API, call the `RebootDBInstance` action.

Example Simple Reboot

```
https://rds.amazonaws.com/
 ?Action=RebootDBInstance
    &DBInstanceIdentifier=mydbinstance
    &Version=2014-10-31
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20131016/us-west-1/rds/aws4_request
    &X-Amz-Date=20131016T233051Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=087a8eb41cb1ab5f99e81575f23e73757ffc6a1e42d7d2b30b9cc0be988cff97
```

Example Reboot with Failover

To force a failover from one AZ to the other, set the `ForceFailover` parameter to `true`.

```
https://rds.amazonaws.com/
    ?Action=RebootDBInstance
    &DBInstanceIdentifier=mydbinstance
    &ForceFailover=true
    &Version=2014-10-31
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20131016/us-west-1/rds/aws4_request
    &X-Amz-Date=20131016T233051Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=087a8eb41cb1ab5f99e81575f23e73757ffc6a1e42d7d2b30b9cc0be988cff97
```

Stopping an Amazon RDS DB Instance Temporarily

If you use a DB instance intermittently, for temporary testing, or for a daily development activity, you can stop your Amazon RDS DB instance temporarily to save money. While your DB instance is stopped, you are charged for provisioned storage (including Provisioned IOPS) and backup storage (including manual snapshots and automated backups within your specified retention window), but not for DB instance hours. For more information, see Billing FAQs.

You can stop and start DB instances that are running the following engines: MariaDB, Microsoft SQL Server, MySQL, Oracle, and PostgreSQL. Stopping and starting a DB instance is supported for all DB instance classes, and in all AWS Regions.

When you stop a DB instance, the DB instance performs a normal shutdown and stops running. The status of the DB instance changes to `stopping` and then `stopped`. Any storage volumes remain attached to the DB instance, and their data is kept. Any data stored in the RAM of the DB instance is deleted. Amazon RDS automatically backs up a stopped DB instance.

You can stop a DB instance for up to seven days. If you do not manually start your DB instance after seven days, your DB instance is automatically started.

Benefits

Stopping and starting a DB instance is faster than creating a DB snapshot, and then restoring the snapshot.

When you stop a DB instance it retains its ID, Domain Name Server (DNS) endpoint, parameter group, security group, and option group. When you start a DB instance, it has the same configuration as when you stopped it. In addition, if you stop a DB instance, Amazon RDS retains the Amazon Simple Storage Service (Amazon S3) transaction logs so you can do a point-in-time restore if necessary.

Limitations

The following are some limitations to stopping and starting a DB instance:

- You can't stop a DB instance that has a Read Replica, or that is a Read Replica.
- You can't stop a DB instance that is in a Multi-AZ deployment.
- You can't stop a DB instance that uses Microsoft SQL Server Mirroring.
- You can't modify a stopped DB instance.
- You can't delete an option group that is associated with a stopped DB instance.
- You can't delete a DB parameter group that is associated with a stopped DB instance.

Option and Parameter Group Considerations

You can't remove persistent options (including permanent options) from an option group if there are DB instances associated with that option group. This functionality is also true of any DB instance with a state of `stopping`, `stopped`, or `starting`.

You can change the option group or DB parameter group that is associated with a stopped DB instance, but the change does not occur until the next time you start the DB instance. If you chose to apply changes immediately, the change occurs when you start the DB instance. Otherwise the changes occurs during the next maintenance window after you start the DB instance.

VPC Considerations

When you stop a DB instance it retains its DNS endpoint. If you stop a DB instance that is not in an Amazon Virtual Private Cloud (Amazon VPC), Amazon RDS releases the IP addresses of the DB instance. If you stop a DB instance that is in a VPC, the DB instance retains its IP addresses.

Note
You should always connect to a DB instance using the DNS endpoint, not the IP address.

AWS Management Console

To stop a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **DB Instances**, and then select the DB instance that you want to modify.
3. Choose **Instance Actions**, and then choose **Stop**.
4. Choose **Continue**.

CLI

To stop a DB instance by using the AWS CLI, call the stop-db-instance command with the following parameters:

- `--db-instance-identifier` – the name of the db instance.

Example

```
stop-db-instance --db-instance-identifier mydbinstance
```

API

To stop a DB instance by using the Amazon RDS API, call the StopDBInstance action with the following parameters:

- `DBInstanceIdentifier` – the name of the db instance.

Example

```
https://rds.amazonaws.com/
    ?Action=StopDBInstance
    &DBInstanceIdentifier=mydbinstance
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-10-31
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20131016/us-west-1/rds/aws4_request
    &X-Amz-Date=20131016T233051Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=087a8eb41cb1ab5f99e81575f23e73757ffc6a1e42d7d2b30b9cc0be988cff97
```

Related Topics

- Starting an Amazon RDS DB Instance That Was Previously Stopped (p. 124)
- Deleting a DB Instance (p. 126)
- Rebooting a DB Instance (p. 119)

Starting an Amazon RDS DB Instance That Was Previously Stopped

You can stop your Amazon RDS DB instance temporarily to save money. After you stop your DB instance, you can restart it to begin using it again. For more details about stopping and starting DB instances, see Stopping an Amazon RDS DB Instance Temporarily (p. 121).

When you start a DB instance that you previously stopped, the DB instance retains the ID, Domain Name Server (DNS) endpoint, parameter group, security group, and option group. When you start a stopped instance, you are charged a full instance hour.

AWS Management Console

To start a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **DB Instances**, and then select the DB instance that you want to modify.
3. Choose **Instance Actions**, and then choose **Start**.
4. Choose **Continue**.

CLI

To start a DB instance by using the AWS CLI, call the start-db-instance command with the following parameters:

- `--db-instance-identifier` – the name of the db instance.

Example

```
start-db-instance --db-instance-identifier mydbinstance
```

API

To start a DB instance by using the Amazon RDS API, call the StartDBInstance action with the following parameters:

- `DBInstanceIdentifier` – the name of the db instance.

Example

```
https://rds.amazonaws.com/
    ?Action=StartDBInstance
    &DBInstanceIdentifier=mydbinstance
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-10-31
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20131016/us-west-1/rds/aws4_request
    &X-Amz-Date=20131016T233051Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
```

```
&X-Amz-Signature=087a8eb41cb1ab5f99e81575f23e73757ffc6a1e42d7d2b30b9cc0be988cff97
```

Related Topics

- Deleting a DB Instance (p. 126)
- Rebooting a DB Instance (p. 119)

Deleting a DB Instance

You can delete a DB instance in any state and at any time. To delete a DB instance, you must specify the name of the instance, and specify whether to take a final DB snapshot taken of the instance.

If the DB instance you want to delete has a Read Replica, you should either promote the Read Replica or delete it. For more information, see Promoting a Read Replica to Be a DB Instance (p. 140).

Final Snapshot

When you delete a DB instance, you can choose whether to create a final snapshot of the DB instance. If you want to be able to restore the DB instance at a later time, you should create a final snapshot.

	With Final Snapshot	Without Final Snapshot
How to Choose	You should create a final DB snapshot if you want to be able to restore your deleted DB instance at a later time.	You can skip creating a final DB snapshot if you want to delete a DB instance quickly. **Important** You will not be able to restore the DB instance later. If you have an earlier manual snapshot of the DB instance, you can restore the DB instance to the point-in-time of the earlier manual snapshot.
Automated Backups	All automated backups are deleted and can't be recovered.	All automated backups are deleted and can't be recovered.
Manual Snapshots	Earlier manual snapshots are not deleted.	Earlier manual snapshots are not deleted.

You can't create a final snapshot of your DB instance if it has one of the following statuses: `creating`, `failed`, `incompatible-restore`, or `incompatible-network`. For more information about DB instance statuses, see DB Instance Status (p. 95).

AWS Management Console

To delete a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the **DB Instances** list, select the DB instance that you want to delete.
3. Choose **Instance Actions**, and then choose **Delete**.
4. For **Create final Snapshot?**, choose **Yes** or **No**.
5. If you chose yes in the previous step, for **Final Snapshot name** type the name of your final DB snapshot.
6. Choose **Yes, Delete**.

CLI

To delete a DB instance by using the AWS CLI, call the delete-db-instance command with the following parameters:

- `--db-instance-identifier`
- `--final-db-snapshot-identifier` or `--skip-final-snapshot`

Example With a Final Snapshot

For Linux, OS X, or Unix:

```
aws rds delete-db-instance \
    --db-instance-identifier mydbinstance  \
    --final-db-snapshot-identifier mydbinstancefinalsnapshot
```

For Windows:

```
aws rds delete-db-instance ^
    --db-instance-identifier mydbinstance  ^
    --final-db-snapshot-identifier mydbinstancefinalsnapshot
```

Example Without a Final Snapshot

For Linux, OS X, or Unix:

```
aws rds delete-db-instance \
    --db-instance-identifier mydbinstance \
    --skip-final-snapshot
```

For Windows:

```
aws rds delete-db-instance ^
    --db-instance-identifier mydbinstance ^
    --skip-final-snapshot
```

API

To delete a DB instance by using the Amazon RDS API, call the `DeleteDBInstance` action with the following parameters:

- `DBInstanceIdentifier`
- `FinalDBSnapshotIdentifier` or `SkipFinalSnapshot`

Example With a Final Snapshot

```
https://rds.amazonaws.com/
    ?Action=DeleteDBInstance
    &DBInstanceIdentifier=mydbinstance
    &FinalDBSnapshotIdentifier=mydbinstancefinalsnapshot
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-10-31
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
```

```
    &X-Amz-Credential=AKIADQKE4SARGYLE/20140305/us-west-1/rds/aws4_request
    &X-Amz-Date=20140305T185838Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=b441901545441d3c7a48f63b5b1522c5b2b37c137500c93c45e209d4b3a064a3
```

Example Without a Final Snapshot

```
https://rds.amazonaws.com/
    ?Action=DeleteDBInstance
    &DBInstanceIdentifier=mydbinstance
    &SkipFinalSnapshot=true
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-10-31
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20140305/us-west-1/rds/aws4_request
    &X-Amz-Date=20140305T185838Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=b441901545441d3c7a48f63b5b1522c5b2b37c137500c93c45e209d4b3a064a3
```

Related Topics

- Stopping an Amazon RDS DB Instance Temporarily (p. 121)

Tagging Amazon RDS Resources

You can use Amazon RDS tags to add metadata to your Amazon RDS resources. In addition, these tags can be used with IAM policies to manage access to Amazon RDS resources and to control what actions can be applied to the Amazon RDS resources. Finally, these tags can be used to track costs by grouping expenses for similarly tagged resources.

All Amazon RDS resources can be tagged

- DB instances
- DB clusters
- Read Replicas
- DB snapshots
- DB cluster snapshots
- Reserved DB instances
- Event subscriptions
- DB option groups
- DB parameter groups
- DB cluster parameter groups
- DB security groups
- DB subnet groups

For information on managing access to tagged resources with IAM policies, see Authentication and Access Control for Amazon RDS (p. 327).

Overview of Amazon RDS Resource Tags

An Amazon RDS tag is a name-value pair that you define and associate with an Amazon RDS resource. The name is referred to as the key. Supplying a value for the key is optional. You can use tags to assign arbitrary information to an Amazon RDS resource. You can use a tag key, for example, to define a category, and the tag value might be an item in that category. For example, you might define a tag key of “project” and a tag value of “Salix,” indicating that the Amazon RDS resource is assigned to the Salix project. You can also use tags to designate Amazon RDS resources as being used for test or production by using a key such as environment=test or environment =production. We recommend that you use a consistent set of tag keys to make it easier to track metadata associated with Amazon RDS resources.

Use tags to organize your AWS bill to reflect your own cost structure. To do this, sign up to get your AWS account bill with tag key values included. Then, to see the cost of combined resources, organize your billing information according to resources with the same tag key values. For example, you can tag several resources with a specific application name, and then organize your billing information to see the total cost of that application across several services. For more information, see Cost Allocation and Tagging in *About AWS Billing and Cost Management*.

Each Amazon RDS resource has a tag set, which contains all the tags that are assigned to that Amazon RDS resource. A tag set can contain as many as 10 tags, or it can be empty. If you add a tag to an Amazon RDS resource that has the same key as an existing tag on resource, the new value overwrites the old value.

AWS does not apply any semantic meaning to your tags; tags are interpreted strictly as character strings. Amazon RDS can set tags on a DB instance or other Amazon RDS resources, depending on the settings that you use when you create the resource. For example, Amazon RDS might add a tag indicating that a DB instance is for production or for testing.

- The tag key is the required name of the tag. The string value can be from 1 to 128 Unicode characters in length and cannot be prefixed with "aws:" or "rds:". The string can contain only the set of Unicode letters, digits, white-space, '_', '.', '/', '=', '+', '-' (Java regex: "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$").

- The tag value is an optional string value of the tag. The string value can be from 1 to 256 Unicode characters in length and cannot be prefixed with "aws:". The string can contain only the set of Unicode letters, digits, white-space, '_', '.', '/', '=', '+', '-' (Java regex: "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$").

  Values do not have to be unique in a tag set and can be null. For example, you can have a key-value pair in a tag set of project/Trinity and cost-center/Trinity.

You can use the AWS Management Console, the command line interface, or the Amazon RDS API to add, list, and delete tags on Amazon RDS resources. When using the command line interface or the Amazon RDS API, you must provide the Amazon Resource Name (ARN) for the Amazon RDS resource you want to work with. For more information about constructing an ARN, see Constructing an ARN for Amazon RDS (p. 184).

Tags are cached for authorization purposes. Because of this, additions and updates to tags on Amazon RDS resources can take several minutes before they are available.

Copying Tags

When you create or restore a DB instance, you can specify that the tags from the DB instance are copied to snapshots of the DB instance. Copying tags ensures that the metadata for the DB snapshots matches that of the source DB instance and any access policies for the DB snapshot also match those of the source DB instance. Tags are not copied by default.

You can specify that tags are copied to DB snapshots for the following actions:

- Creating a DB instance.
- Restoring a DB instance.
- Creating a Read Replica.
- Copying a DB snapshot.

Note
If you include a value for the `--tag-key` parameter of the create-db-snapshot AWS CLI command (or supply at least one tag to the CreateDBSnapshot API action) then RDS doesn't copy tags from the source DB instance to the new DB snapshot. This functionality applies even if the source DB instance has the `--copy-tags-to-snapshot` (`CopyTagsToSnapshot`) option enabled. If you take this approach, you can create a copy of a DB instance from a DB snapshot and avoid adding tags that don't apply to the new DB instance. Once you have created your DB snapshot using the AWS CLI `create-db-snapshot` command (or the `CreateDBSnapshot` Amazon RDS API action) you can then add tags as described later in this topic.

AWS Management Console

The process to tag an Amazon RDS resource is similar for all resources. The following procedure shows how to tag an Amazon RDS DB instance.

To add a tag to a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Instances**.

Note
To filter the list of DB instances in the **DB Instances** pane, in the box beside the **Viewing** box, type a text string. Only DB instances that contain the string appear.

3. Select the DB instance that you want to tag. The inline summary appears.
4. In the inline summary, choose the details icon to open the details section.

5. In the details section, scroll down and choose **Tags** to open the tags section.
6. Choose **Add/Edit Tags**. The Tag DB Instance pane appears.

7. Choose **Add another Tag**.
8. Type a key and value for the tag, and then choose **Save Tags**.

To delete a tag from a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, click **Instances**.

Note
To filter the list of DB instances in the **DB Instances** pane, in the box beside the **Viewing** box, type a text string. Only DB instances that contain the string appear.

3. Select the DB instance from which you want to remove a tag. The inline summary appears.
4. In the inline summary, choose the details icon to open the details section.

5. In the details section, scroll down and choose **Tags** to open the tags section.
6. Choose **Add/Edit Tags**. The Tag DB Instance pane appears.

7. Choose the red "X" in the **Remove** column next to the tag you want to delete, and then choose **Save Tags**.

CLI

You can add, list, or remove tags for a DB instance using the AWS CLI.

- To add one or more tags to an Amazon RDS resource, use the AWS CLI command `add-tags-to-resource`.
- To list the tags on an Amazon RDS resource, use the AWS CLI command `list-tags-for-resource`.
- To remove one or more tags from an Amazon RDS resource, use the AWS CLI command `remove-tags-from-resource`.

To learn more about how to construct the required ARN, see Constructing an ARN for Amazon RDS (p. 184).

API

You can add, list, or remove tags for a DB instance using the Amazon RDS API.

- To add a tag to an Amazon RDS resource, use the `AddTagsToResource` operation.
- To list tags that are assigned to an Amazon RDS resource, use the `ListTagsForResource`.
- To remove tags from an Amazon RDS resource, use the `RemoveTagsFromResource` operation.

To learn more about how to construct the required ARN, see Constructing an ARN for Amazon RDS (p. 184).

When working with XML using the Amazon RDS API, tags use the following schema:

```
<Tagging>
    <TagSet>
        <Tag>
            <Key>Project</Key>
            <Value>Trinity</Value>
        </Tag>
        <Tag>
            <Key>User</Key>
            <Value>Jones</Value>
        </Tag>
    </TagSet>
</Tagging>
```

The following table provides a list of the allowed XML tags and their characteristics. Values for Key and Value are case-dependent. For example, project=Trinity and PROJECT=Trinity are two distinct tags.

Tagging Element	Description
TagSet	A tag set is a container for all tags assigned to an Amazon RDS resource. There can be only one tag set per resource. You work with a TagSet only through the Amazon RDS API.
Tag	A tag is a user-defined key-value pair. There can be from 1 to 50 tags in a tag set.
Key	A key is the required name of the tag. The string value can be from 1 to 128 Unicode characters in length and cannot be prefixed with "rds:" or "aws:". The string can only contain only the set of Unicode letters, digits, white-space, '_', '.', '/', '=', '+', '-' (Java regex: "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$").

Tagging Element	Description
	Keys must be unique to a tag set. For example, you cannot have a key-pair in a tag set with the key the same but with different values, such as project/Trinity and project/Xanadu.
Value	A value is the optional value of the tag. The string value can be from 1 to 256 Unicode characters in length and cannot be prefixed with "rds:" or "aws:". The string can only contain only the set of Unicode letters, digits, white-space, '_', '.', '/', '=', '+', '-' (Java regex: "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$"). Values do not have to be unique in a tag set and can be null. For example, you can have a key-value pair in a tag set of project/Trinity and cost-center/Trinity.

Related Topics

- Authentication and Access Control for Amazon RDS (p. 327)

Working with PostgreSQL, MySQL, and MariaDB Read Replicas

Amazon RDS uses the MySQL, MariaDB, and PostgreSQL (version 9.3.5 and later) DB engines' built-in replication functionality to create a special type of DB instance called a Read Replica from a source DB instance. Updates made to the source DB instance are asynchronously copied to the Read Replica. You can reduce the load on your source DB instance by routing read queries from your applications to the Read Replica. Using Read Replicas, you can elastically scale out beyond the capacity constraints of a single DB instance for read-heavy database workloads.

> **Note**
> The information following applies to creating Amazon RDS Read Replicas either in the same AWS Region as the source DB instance, or in a separate AWS Region. The information following doesn't apply to setting up replication with an instance that is running on an Amazon EC2 instance or that is on-premises.

When you create a Read Replica, you first specify an existing DB instance as the source. Then, Amazon RDS takes a snapshot of the source instance and creates a read-only instance from the snapshot. Amazon RDS then uses the asynchronous replication method for the DB engine to update the Read Replica whenever there is a change to the source DB instance. The Read Replica operates as a DB instance that allows only read-only connections. Applications connect to a Read Replica the same way they do to any DB instance. Amazon RDS replicates all databases in the source DB instance.

Amazon RDS sets up a secure communications channel between the source DB instance and a Read Replica if that Read Replica is in a different AWS Region from the DB instance. Amazon RDS establishes any AWS security configurations needed to enable the secure channel, such as adding security group entries. PostgreSQL DB instances use a secure connection that you can encrypt by setting the *ssl* parameter to **1** for both the source and the replica instances.

Topics

Amazon RDS Read Replica Overview

Deploying one or more Read Replicas for a given source DB instance might make sense in a variety of scenarios, including the following:

- Scaling beyond the compute or I/O capacity of a single DB instance for read-heavy database workloads. You can direct this excess read traffic to one or more Read Replicas.
- Serving read traffic while the source DB instance is unavailable. If your source DB instance cannot take I/O requests (for example, due to I/O suspension for backups or scheduled maintenance), you can direct read traffic to your Read Replicas. For this use case, keep in mind that the data on the Read Replica might be "stale" because the source DB instance is unavailable.

- Business reporting or data warehousing scenarios where you might want business reporting queries to run against a Read Replica, rather than your primary, production DB instance.

By default, a Read Replica is created with the same storage type as the source DB instance. However, you can create a Read Replica that has a different storage type from the source DB instance based on the options listed in the following table.

Source DB Instance Storage Type	Source DB Instance Storage Allocation	Read Replica Storage Type Options
PIOPS	100 GB - 3 TB	PIOPS | GP2 | Standard
GP2	100 GB - 3 TB	PIOPS | GP2 | Standard
GP2	Less than 100 GB	GP2 | Standard
Standard	100 GB - 3 TB	PIOPS | GP2 | Standard
Standard	Less than 100 GB	GP2 | Standard

Amazon RDS doesn't support circular replication. You cannot configure a DB instance to serve as a replication source for an existing DB instance; you can only create a new Read Replica from an existing DB instance. For example, if MyDBInstance replicates to ReadReplica1, you cannot configure ReadReplica1 to replicate back to MyDBInstance. From ReadReplica1, you can only create a new Read Replica, such as ReadReplica2.

For MySQL, MariaDB, and PostgreSQL Read Replicas, you can monitor replication lag in Amazon CloudWatch by viewing the Amazon RDS `ReplicaLag` metric. For MySQL and MariaDB, the `ReplicaLag` metric reports the value of the `Seconds_Behind_Master` field of the `SHOW SLAVE STATUS` command. For PostgreSQL, the `ReplicaLag` metric reports the value of `SELECT extract(epoch from now() - pg_last_xact_replay_timestamp()) AS slave_lag`.

Common causes for replication lag for MySQL and MariaDB are the following:

- A network outage.
- Writing to tables with indexes on a Read Replica. If the `read_only` parameter is not set to 0 on the Read Replica, it can break replication.
- Using a non-transactional storage engine such as MyISAM. Replication is only supported for the InnoDB storage engine on MySQL and the XtraDB storage engine on MariaDB.

When the `ReplicaLag` metric reaches 0, the replica has caught up to the source DB instance. If the `ReplicaLag` metric returns -1, then replication is currently not active. `ReplicaLag` = -1 is equivalent to `Seconds_Behind_Master` = `NULL`.

Differences Between PostgreSQL and MySQL or MariaDB Read Replicas

Because the PostgreSQL DB engine implements replication differently than the MySQL and MariaDB DB engines, there are several significant differences you should know about, as shown in the following table.

Feature/Behavior	PostgreSQL	MySQL and MariaDB
What is the replication method?	Physical replication.	Logical replication.

Feature/Behavior	PostgreSQL	MySQL and MariaDB
How are transaction logs purged?	PostgreSQL has a parameter, `wal_keep_segments`, that dictates how many write ahead log (WAL) files are kept to provide data to the Read Replicas. The parameter value specifies the number of logs to keep.	Amazon RDS won't delete any binary logs that have not been applied.
Can a replica be made writable?	No. A PostgreSQL Read Replica is a physical copy and PostgreSQL doesn't allow for a Read Replica to be made writeable.	Yes. You can enable the MySQL or MariaDB Read Replica to be writable.
Can backups be performed on the replica?	Yes, you can create a snapshot of a PostgreSQL Read Replica, but you cannot enable automatic backups.	Yes. You can enable automatic backups on a MySQL or MariaDB Read Replica.
Can you use parallel replication?	No. PostgreSQL has a single process handling replication.	Yes. MySQL version 5.6 and later and all supported MariaDB versions allow for parallel replication threads.

PostgreSQL Read Replicas (Version 9.3.5 and Later)

Amazon RDS PostgreSQL 9.3.5 and later uses PostgreSQL native streaming replication to create a read-only copy of a source (a "master" in PostgreSQL terms) DB instance. This Read Replica (a "standby" in PostgreSQL terms) DB instance is an asynchronously created physical replication of the master DB instance. It is created by a special connection that transmits write ahead log (WAL) data between the source DB instance and the Read Replica where PostgreSQL asynchronously streams database changes as they are made.

PostgreSQL uses a "replication" role to perform streaming replication. The role is privileged, but cannot be used to modify any data. PostgreSQL uses a single process for handling replication.

Creating a PostgreSQL Read Replica doesn't require an outage for the master DB instance. Amazon RDS sets the necessary parameters and permissions for the source DB instance and the Read Replica without any service interruption. A snapshot is taken of the source DB instance, and this snapshot becomes the Read Replica. No outage occurs when you delete a Read Replica.

You can create up to five Read Replicas from one source DB instance. For replication to operate effectively, each Read Replica should have the same amount of compute and storage resources as the source DB instance. If you scale the source DB instance, you should also scale the Read Replicas.

Amazon RDS overrides any incompatible parameters on a Read Replica if it prevents the Read Replica from starting. For example, if the `max_connections` parameter value is higher on the source DB instance than on the Read Replica, Amazon RDS updates the parameter on the Read Replica to be the same value as that on the source DB instance.

Here are some important facts about PostgreSQL Read Replicas:

- PostgreSQL Read Replicas are read-only and cannot be made writeable.
- You cannot create a Read Replica from another Read Replica (that is, you cannot create cascading Read Replicas).
- You can promote a PostgreSQL Read Replica to be a new source DB instance. However, the Read Replica doesn't become the new source DB instance automatically. The Read Replica, when promoted, stops receiving WAL communications and is no longer a read-only instance. You must set up any

replication you intend to have going forward because the promoted Read Replica is now a new source DB instance.

- A PostgreSQL Read Replica reports a replication lag of up to 5 minutes if there are no user transactions occurring on the source DB instance.
- Before a DB instance can serve as a source DB instance, you must enable automatic backups on the source DB instance by setting the backup retention period to a value other than 0.

Situations That Break PostgreSQL Replication

In several situations, a PostgreSQL source DB instance can unintentionally break replication with a Read Replica. These situations include the following:

- The `max_wal_senders` parameter is set too low to provide enough data to the number of Read Replicas. This situation causes replication to stop.
- The PostgreSQL parameter, `wal_keep_segments`, dictates how many WAL files are kept to provide data to the Read Replicas. The parameter value specifies the number of logs to keep. If you set the parameter value too low, you can cause a Read Replica to fall so far behind that streaming replication stops. In this case, Amazon RDS reports a replication error and begins recovery on the Read Replica by replaying the source DB instance's archived WAL logs. This recovery process continues until the Read Replica has caught up enough to continue streaming replication. For more information on this process and how to determine the appropriate parameter setting, see Troubleshooting a PostgreSQL Read Replica Problem (p. 151).
- A PostgreSQL Read Replica requires a reboot if the source DB instance endpoint changes.

When the WAL stream that provides data to a Read Replica is broken, PostgreSQL switches into recovery mode to restore the Read Replica by using archived WAL files. When this process is compete, PostgreSQL will attempt to re-establish streaming replication.

MySQL and MariaDB Read Replicas

Before a MySQL or MariaDB DB instance can serve as a replication source, you must enable automatic backups on the source DB instance by setting the backup retention period to a value other than 0. This requirement also applies to a Read Replica that is the source DB instance for another Read Replica. Automatic backups are supported only for Read Replicas running any version of MariaDB or MySQL 5.6 and later.

You can configure replication based on binary log coordinates for both MySQL and MariaDB instance. For MariaDB instances, you can also configure replication based on global transaction IDs (GTIDs), which provides better crash safety. For more information about configuring replication using GTIDs on a MariaDB DB instance, see Configuring GTID-Based Replication into an Amazon RDS MariaDB DB instance (p. 707).

You can create up to five Read Replicas from one DB instance. In order for replication to operate effectively, each Read Replica should have as much compute and storage resources as the source DB instance. If you scale the source DB instance, you should also scale the Read Replicas.

If a Read Replica is running any version of MariaDB or MySQL 5.6 and later, you can specify it as the source DB instance for another Read Replica. For example, you can create ReadReplica1 from MyDBInstance, and then create ReadReplica2 from ReadReplica1. Updates made to MyDBInstance are replicated to ReadReplica1 and then replicated from ReadReplica1 to ReadReplica2. You cannot have more than four instances involved in a replication chain. For example, you can create ReadReplica1 from MySourceDBInstance, and then create ReadReplica2 from ReadReplica1, and then create ReadReplica3 from ReadReplica2, but you cannot create a ReadReplica4 from ReadReplica3.

To enable automatic backups on a Read Replica for Amazon RDS MariaDB or MySQL version 5.6 and later, first create the Read Replica, then modify the Read Replica to enable automatic backups.

Read Replicas are designed to support read queries, but you might need occasional updates. For example, you might need to add an index to speed the specific types of queries accessing the replica. You can enable updates by setting the `read_only` parameter to **0** in the DB parameter group for the Read Replica.

You can run multiple concurrent Read Replica create or delete actions that reference the same source DB instance, as long as you stay within the limit of five Read Replicas for the source instance.

You can create a Read Replica from either single-AZ or Multi-AZ DB instance deployments. You use a Multi-AZ deployment to improve the durability and availability of a critical system, but you cannot use the Multi-AZ secondary to serve read-only queries. You must create Read Replicas from a high-traffic, Multi-AZ DB instance to offload read queries from the source DB instance. If the source instance of a Multi-AZ deployment fails over to the secondary, any associated Read Replicas are switched to use the secondary as their replication source.

For MySQL and MariaDB DB instances, in some cases Read Replicas cannot be switched to the secondary if some binlog events are not flushed during the failure. In these cases, you must manually delete and recreate the Read Replicas. You can reduce the chance of this happening in MySQL 5.5 by setting the **sync_binlog=1** and **innodb_support_xa=1** dynamic variables. These settings might reduce performance, so test their impact before implementing the changes to a production environment. These problems are less likely to occur if you use MySQL 5.6 and later or MariaDB. For instances running MySQL 5.6 and later or MariaDB, the parameters are set by default to **sync_binlog=1** and **innodb_support_xa=1**.

You usually configure replication between Amazon RDS DB instances, but you can configure replication to import databases from instances of MySQL or MariaDB running outside of Amazon RDS, or to export databases to such instances. For more information, see Importing Data to an Amazon RDS MySQL or MariaDB DB Instance with Reduced Downtime (p. 873) and Exporting Data from a MySQL DB Instance by Using Replication (p. 893).

You can stop and restart the replication process on an Amazon RDS DB instance by calling the system stored procedures mysql.rds_stop_replication (p. 918) and mysql.rds_start_replication (p. 917). You can do this when replicating between two Amazon RDS instances for long-running operations such as creating large indexes. You also need to stop and start replication when importing or exporting databases. For more information, see Importing Data to an Amazon RDS MySQL or MariaDB DB Instance with Reduced Downtime (p. 873) and Exporting Data from a MySQL DB Instance by Using Replication (p. 893).

You must explicitly delete Read Replicas, using the same mechanisms for deleting a DB instance. If you delete the source DB instance without deleting the replicas, each replica is promoted to a stand-alone, single-AZ DB instance.

If you promote a MySQL or MariaDB Read Replica that is in turn replicating to other Read Replicas, those Read Replicas remain active. Consider an example where MyDBInstance1 replicates to MyDBInstance2, and MyDBInstance2 replicates to MyDBInstance3. If you promote MyDBInstance2, replication from MyDBInstance1 to MyDBInstance2 no longer occurs, but MyDBInstance2 still replicates to MyDBInstance3.

If replication is stopped for more than 30 consecutive days, either manually or due to a replication error, Amazon RDS terminates replication between the master DB instance and all Read Replicas. It does so to prevent increased storage requirements on the master DB instance and long failover times. The Read Replica DB instance is still available. However, replication cannot be resumed because the binary logs required by the Read Replica are deleted from the master DB instance after replication is terminated. You can create a new Read Replica for the master DB instance to reestablish replication.

Creating a Read Replica

You can create a Read Replica from an existing MySQL, MariaDB, or PostgreSQL DB instance using the AWS Management Console, AWS CLI, or AWS API. You create a Read Replica by specifying the `SourceDBInstanceIdentifier`, which is the DB instance identifier of the source DB instance from which you wish to replicate.

When you initiate the creation of a Read Replica, Amazon RDS takes a DB snapshot of your source DB instance and begins replication. As a result, you experience a brief I/O suspension on your source DB instance as the DB snapshot occurs. The I/O suspension typically lasts about one minute and can be avoided if the source DB instance is a Multi-AZ deployment (in the case of Multi-AZ deployments, DB snapshots are taken from the standby). An active, long-running transaction can slow the process of creating the Read Replica, so wait for long-running transactions to complete before creating a Read Replica. If you create multiple Read Replicas in parallel from the same source DB instance, Amazon RDS takes only one snapshot at the start of the first create action.

When creating a Read Replica, there are a few things to consider. First, you must enable automatic backups on the source DB instance by setting the backup retention period to a value other than 0. This requirement also applies to a Read Replica that is the source DB instance for another Read Replica. For MySQL DB instances, automatic backups are supported only for Read Replicas running MySQL 5.6 and later, but not for MySQL versions 5.5. To enable automatic backups on an Amazon RDS MySQL version 5.6 and later Read Replica, first create the Read Replica, then modify the Read Replica to enable automatic backups.

Preparing MySQL DB Instances That Use MyISAM

If your MySQL DB instance uses a non-transactional engine such as MyISAM, you need to perform the following steps to successfully set up your Read Replica. These steps are required to ensure that the Read Replica has a consistent copy of your data. These steps are not required if all of your tables use a transactional engine such as InnoDB.

1. Stop all data manipulation language (DML) and data definition language (DDL) operations on non-transactional tables in the source DB instance and wait for them to complete. SELECT statements can continue running.
2. Flush and lock the tables in the source DB instance.
3. Create the Read Replica using one of the methods in the following sections.
4. Check the progress of the Read Replica creation using, for example, the `DescribeDBInstances` API operation. Once the Read Replica is available, unlock the tables of the source DB instance and resume normal database operations.

AWS Management Console

To create a Read Replica from a source MySQL, MariaDB, or PostgreSQL DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **DB Instances**.
3. In the **Instances** pane, choose the MySQL, MariaDB, or PostgreSQL DB instance that you want to use as the source for a Read Replica and choose **Create Read Replica** from **Instance Actions**.
4. Choose the instance specifications you want to use. It is a best practice to use the same DB instance class and storage type for the Read Replica.
5. Choose the settings you want to use. For **DB Instance Identifier**, type a name for the Read Replica. Adjust other settings as needed.
6. Choose the network, security, database, and maintenance settings you want to use.

7. Choose **Create Read Replica**.

CLI

To create a Read Replica from a source MySQL, MariaDB, or PostgreSQL DB instance, use the AWS CLI command `create-db-instance-read-replica`.

Example

For Linux, OS X, or Unix:

```
aws rds create-db-instance-read-replica \
    --db-instance-identifier myreadreplica \
    --source-db-instance-identifier mydbinstance
```

For Windows:

```
aws rds create-db-instance-read-replica ^
    --db-instance-identifier myreadreplica ^
    --source-db-instance-identifier mydbinstance
```

API

To create a Read Replica from a source MySQL, MariaDB, or PostgreSQL DB instance, call the Amazon RDS API function `CreateDBInstanceReadReplica`.

```
https://rds.amazonaws.com/
 ?Action=CreateDBInstanceReadReplica
 &DBInstanceIdentifier=myreadreplica
 &SourceDBInstanceIdentifier=mydbinstance
 &Version=2012-01-15
 &SignatureVersion=2
 &SignatureMethod=HmacSHA256
 &Timestamp=2012-01-20T22%3A06%3A23.624Z
 &AWSAccessKeyId=<AWS Access Key ID>
 &Signature=<Signature>
```

Promoting a Read Replica to Be a DB Instance

You can promote a MySQL, MariaDB, or PostgreSQL Read Replica into a stand-alone, single-AZ DB instance. When you promote a Read Replica, the DB instance is rebooted before it becomes available.

There are several reasons you might want to convert a Read Replica into a single-AZ DB instance:

- **Performing DDL operations (MySQL and MariaDB only)** – DDL operations, such as creating or rebuilding indexes, can take time and impose a significant performance penalty on your DB instance. You can perform these operations on a MySQL or MariaDB Read Replica once the Read Replica is in sync with its source DB instance. Then you can promote the Read Replica and direct your applications to use the promoted instance.
- **Sharding** – Sharding embodies the "share-nothing" architecture and essentially involves breaking a large database into several smaller databases. One common way to split a database is splitting tables that are not joined in the same query onto different hosts. Another method is duplicating a table across multiple hosts and then using a hashing algorithm to determine which host receives a given update. You can create Read Replicas corresponding to each of your shards (smaller databases) and promote them when you decide to convert them into standalone shards. You can then carve out the

key space (if you are splitting rows) or distribution of tables for each of the shards depending on your requirements.

- **Implementing failure recovery** – You can use Read Replica promotion as a data recovery scheme if the source DB instance fails. However, if your use case requires synchronous replication, automatic failure detection, and failover, we recommend that you run your DB instance as a Multi-AZ deployment instead. If you are aware of the ramifications and limitations of asynchronous replication and you still want to use Read Replica promotion for data recovery, you can do so. To do this, first create a Read Replica and then monitor the source DB instance for failures. In the event of a failure, do the following:

  1. Promote the Read Replica.
  2. Direct database traffic to the promoted DB instance.
  3. Create a replacement Read Replica with the promoted DB instance as its source.

  You can perform all of these operations using the Amazon Relational Database Service API Reference, and you can automate the process by using the Amazon Simple Workflow Service Developer Guide.

The new DB instance that is created when you promote a Read Replica retains the backup retention period, backup window period, and parameter group of the former Read Replica source. The promotion process can take several minutes or longer to complete, depending on the size of the Read Replica. Once you promote the Read Replica into a single-AZ DB instance, it is just like any other single-AZ DB instance. For example, you can convert the new DB instance into a Multi-AZ DB instance, and you can create Read Replicas from it. You can also take DB snapshots and perform Point-In-Time Restore operations. Because the promoted DB instance is no longer a Read Replica, you cannot use it as a replication target. If a source DB instance has several Read Replicas, promoting one of the Read Replicas to a DB instance has no effect on the other replicas.

We recommend that you disable automated backups on your Read Replica before promoting the Read Replica. This approach ensures that no backup is performed during the promotion process. Once the instance is promoted to a primary instance, backups are performed based on your backup settings.

The following steps show the general process for promoting a Read Replica to a single-AZ DB instance:

1. Stop any transactions from being written to the Read Replica source DB instance, and then wait for all updates to be made to the Read Replica. Database updates occur on the Read Replica after they have occurred on the source DB instance, and this replication lag can vary significantly. Use the Replica Lag metric to determine when all updates have been made to the Read Replica.
2. For MySQL and MariaDB only: If you need to make changes to the MySQL or MariaDB Read Replica, you must the set the `read_only` parameter to **0** in the DB parameter group for the Read Replica. You can then perform all needed DDL operations, such as creating indexes, on the Read Replica. Actions taken on the Read Replica don't affect the performance of the source DB instance.
3. Promote the Read Replica by using the **Promote Read Replica** option on the Amazon RDS console, the AWS CLI command `promote-read-replica`, or the `PromoteReadReplica` Amazon RDS API operation.

   Note
   The promotion process takes a few minutes to complete. When you promote a Read Replica, replication is stopped and the Read Replica is rebooted. When the reboot is complete, the Read Replica is available as a single-AZ DB instance.

AWS Management Console

To promote a Read Replica to a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the Amazon RDS console, choose **Read Replicas**.

3. In the **Read Replicas** pane, select the check box beside the Read Replica that you want to promote.
4. Choose **Promote Read Replica**.
5. In the **Promote Read Replica** dialog box, enter the backup retention period and the backup window for the new promoted DB instance.
6. When the settings are as you want them, choose **Continue**.
7. On the acknowledgment page, choose **Yes, Promote**.

CLI

To promote a Read Replica to a DB instance, use the AWS CLI `promote-read-replica` command.

Example

For Linux, OS X, or Unix:

```
aws rds promote-read-replica \
    --db-instance-identifier myreadreplica
```

For Windows:

```
aws rds promote-read-replica ^
    --db-instance-identifier myreadreplica
```

API

To promote a Read Replica to a DB instance, call `PromoteReadReplica`.

```
https://rds.amazonaws.com/
 ?Action=PromoteReadReplica
 &DBInstanceIdentifier=myreadreplica
 &Version=2012-01-15
 &SignatureVersion=2
 &SignatureMethod=HmacSHA256
 &Timestamp=2012-01-20T22%3A06%3A23.624Z
 &AWSAccessKeyId=<AWS Access Key ID>
 &Signature=<Signature>
```

Replicating a Read Replica Across AWS Regions

With Amazon Relational Database Service (Amazon RDS), you can create a MySQL, PostgreSQL, or MariaDB Read Replica in a different AWS Region than the source DB instance. You create a Read Replica to do the following:

- Improve your disaster recovery capabilities.
- Scale read operations into an AWS Region closer to your users.
- Make it easier to migrate from a data center in one AWS Region to a data center in another AWS Region.

Creating a MySQL, PostgreSQL, or MariaDB Read Replica in a different AWS Region than the source instance is very similar to creating a replica in the same AWS Region. To create a Read Replica across regions, you can use the AWS Management Console, run the `create-db-instance-read-replica` command, or call the `CreateDBInstanceReadReplica` API action.

To create an encrypted Read Replica in a different AWS Region than the source DB instance, the source DB instance must be encrypted.

> **Note**
> You can also create a replica of an Amazon Aurora MySQL DB cluster in a different AWS Region. For more information, see Replicating Amazon Aurora MySQL DB Clusters Across AWS Regions (p. 528).

Following, you can find information on how to create a Read Replica from a source MySQL, MariaDB, or PostgreSQL DB instance in a different AWS Region.

AWS Management Console

You can create a Read Replica across regions using the AWS Management Console.

To create a Read Replica across regions with the console

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **DB Instances**.
3. In the **Instances** window, choose the MySQL, MariaDB, or PostgreSQL DB instance that you want to use as the source for a Read Replica, and then choose **Create Read Replica** from **Instance Actions**. To create an encrypted Read Replica, the source DB instance must be encrypted. To learn more about encrypting the source DB instance, see Encrypting Amazon RDS Resources (p. 355).
4. Choose the instance specifications you want to use. We recommend that you use the same DB instance class and storage type for the Read Replica.
5. Choose the other settings you want to use:
   - For **DB Instance Identifier**, type a name for the Read Replica.
   - In the **Network & Security** section, choose a value for **Designation Region** and **Designation DB Subnet Group**.
   - To create an encrypted Read Replica in another AWS Region, choose **Enable Encryption**, and then choose **Master Key**. For **Master Key**, choose the KMS key identifier of the destination AWS Region.
   - Choose the remaining network, security, database, and maintenance settings you want to use.
6. Choose **Create Read Replica**.

AWS CLI

To create a Read Replica from a source MySQL, MariaDB, or PostgreSQL DB instance in a different AWS Region, you can use the `create-db-instance-read-replica` command. In this case, you use `create-db-instance-read-replica` from the AWS Region where you want the Read Replica and specify the Amazon Resource Name (ARN) for the source DB instance. An ARN uniquely identifies a resource created in Amazon Web Services.

For example, if your source DB instance is in the US East (N. Virginia) region, the ARN looks similar to the following.

```
arn:aws:rds:us-east-1:123456789012:db:my-mysql-instance
```

For information about ARNs, see Working with Amazon Resource Names (ARNs) in Amazon RDS (p. 184).

To create an encrypted Read Replica in a different AWS Region than the source DB instance, you can use the AWS CLI `create-db-instance-read-replica` command from the destination AWS Region. The following parameters are used to create an encrypted Read Replica in another AWS Region:

- `--source-region` — The AWS Region that the encrypted Read Replica is created in. If `source-region` is not specified, you must specify a `pre-signed-url`. A `pre-signed-url` is a URL that contains a Signature Version 4 signed request for the `CreateDBInstanceReadReplica` action that is called in the source AWS Region where the Read Replica is created from. To learn more about the `pre-signed-url`, see `CreateDBInstanceReadReplica`.
- `--source-db-instance-identifier` — The DB instance identifier for the encrypted Read Replica that is created. This identifier must be in the ARN format for the source AWS Region. The AWS Region specified in `source-db-instance-identifier` must match the AWS Region specified as the `source-region`.
- `--db-instance-identifier` — The identifier for the encrypted Read Replica in the destination AWS Region.
- `--kms-key-id` — The AWS KMS key identifier for the key to use to encrypt the Read Replica in the destination AWS Region.

The following code creates a Read Replica in the `us-west-2` region.

Example

For Linux, OS X, or Unix:

```
aws rds create-db-instance-read-replica \
    --db-instance-identifier DBInstanceIdentifier \
    --region us-west-2 \
    --source-db-instance-identifier arn:aws:rds:us-east-1:123456789012:db:my-mysql-instance
```

For Windows:

```
aws rds create-db-instance-read-replica ^
    --db-instance-identifier DBInstanceIdentifier ^
    --region us-west-2 ^
    --source-db-instance-identifier arn:aws:rds:us-east-1:123456789012:db:my-mysql-instance
```

The following code creates a Read Replica in a different AWS Region than the source DB instance. The AWS Region where you call the `create-db-instance-read-replica` command is the destination AWS Region for the encrypted Read Replica.

Example

For Linux, OS X, or Unix:

```
aws rds create-db-instance-read-replica \
    --db-instance-identifier DBInstanceIdentifier \
    --region us-west-2 \
    --source-db-instance-identifier arn:aws:rds:us-east-1:123456789012:db:my-mysql-instance
 \
    --source-region us-east-1 \
    --kms-key-id my-us-east-1-key
```

For Windows:

```
aws rds create-db-instance-read-replica ^
    --db-instance-identifier DBInstanceIdentifier ^
    --region us-west-2 ^
```

```
    --source-db-instance-identifier arn:aws:rds:us-east-1:123456789012:db:my-mysql-instance
 ^
    --source-region us-east-1 ^
    --kms-key-id my-us-east-1-key
```

API

To create a Read Replica from a source MySQL, MariaDB, or PostgreSQL DB instance in a different AWS Region, you can call the Amazon RDS API function CreateDBInstanceReadReplica. In this case, you call CreateDBInstanceReadReplica from the AWS Region where you want the Read Replica and specify the Amazon Resource Name (ARN) for the source DB instance. An ARN uniquely identifies a resource created in Amazon Web Services.

To create an encrypted Read Replica in a different AWS Region than the source DB instance, you can use the Amazon RDS API `CreateDBInstanceReadReplica` action from the destination AWS Region. To create an encrypted Read Replica in another AWS Region, you must specify a value for `PreSignedURL`. `PreSignedURL` should contain a request for the `CreateDBInstanceReadReplica` action to call in the source AWS Region where the Read Replica is created in. To learn more about `PreSignedUrl`, see `CreateDBInstanceReadReplica`.

For example, if your source DB instance is in the US East (N. Virginia) region, the ARN looks similar to the following.

```
arn:aws:rds:us-east-1:123456789012:db:my-mysql-instance
```

For information about ARNs, see Working with Amazon Resource Names (ARNs) in Amazon RDS (p. 184).

Example

```
https://us-west-2.rds.amazonaws.com/
    ?Action=CreateDBInstanceReadReplica
    &KmsKeyId=my-us-east-1-key
    &PreSignedUrl=https%253A%252F%252Frds.us-west-2.amazonaws.com%252F
         %253FAction%253D CreateDBInstanceReadReplica
         %2526DestinationRegion%253Dus-east-1
         %2526KmsKeyId%253Dmy-us-east-1-key
         %2526SourceDBInstanceIdentifier%253Darn%25253Aaws%25253Ards%25253Aus-
west-2%1234567890
12%25253Adb%25253Amy-mysql-instance
         %2526SignatureMethod%253DHmacSHA256
         %2526SignatureVersion%253D4%2526SourceDBInstanceIdentifier%253Darn%25253Aaws
%25253Ards%25253Aus-west-2%25253A123456789012%25253Ainstance%25253Amysql-instance1-
instance-20161115
         %2526Version%253D2014-10-31
         %2526X-Amz-Algorithm%253DAWS4-HMAC-SHA256
         %2526X-Amz-Credential%253DAKIADQKE4SARGYLE%252F20161117%252Fus-west-2%252Frds
%252Faws4_request
         %2526X-Amz-Date%253D20161117T215409Z
         %2526X-Amz-Expires%253D3600
         %2526X-Amz-SignedHeaders%253Dcontent-type%253Bhost%253Buser-agent%253Bx-amz-
content-sha256%253Bx-amz-date
         %2526X-Amz-Signature
%253D255a0f17b4e717d3b67fad163c3ec26573b882c03a65523522cf890a67fca613
    &DBInstanceIdentifier=myreadreplica
    &SourceDBInstanceIdentifier=arn:aws:rds:us-east-1:123456789012:db:my-mysql-instance
    &Version=2012-01-15
    &SignatureVersion=2
    &SignatureMethod=HmacSHA256
    &Timestamp=2012-01-20T22%3A06%3A23.624Z
```

```
    &AWSAccessKeyId=<AWS Access Key ID>
    &Signature=<Signature>
```

Cross-Region Replication Considerations

All of the considerations for performing replication within an AWS Region apply to cross-region replication. The following extra considerations apply when replicating between regions:

- You can only replicate between regions when using Amazon RDS DB instances of MariaDB, PostgreSQL (versions 9.4.7 and 9.5.2 and later), or MySQL 5.6 and later.
- A source DB instance can have cross-region Read Replicas in multiple regions.
- You can only create a cross-region Amazon RDS Read Replica from a source Amazon RDS DB instance that is not a Read Replica of another Amazon RDS DB instance.
- You cannot set up a replication channel into or out of the AWS GovCloud (US) region.
- You can expect to see a higher level of lag time for any Read Replica that is in a different AWS Region than the source instance, due to the longer network channels between regional data centers.
- Within an AWS Region, all cross-region Read Replicas created from the same source DB instance must either be in the same Amazon VPC or be outside of a VPC. For cross-region Read Replicas, any of the create Read Replica commands that specify the `--db-subnet-group-name` parameter must specify a DB subnet group from the same VPC.
- You can create a cross-region Read Replica in a VPC from a source DB instance that is not in a VPC. You can also create a cross-region Read Replica that is not in a VPC from a source DB instance that is in a VPC.
- Due to the limit on the number of access control list (ACL) entries for a VPC, we cannot guarantee more than five cross-region Read Replica instances.

Cross-Region Replication Costs

The data transferred for cross-region replication incurs Amazon RDS data transfer charges. These cross-region replication actions generate charges for the data transferred out of the source AWS Region:

- When you create a Read Replica, Amazon RDS takes a snapshot of the source instance and transfers the snapshot to the Read Replica region.
- For each data modification made in the source databases, Amazon RDS transfers data from the source AWS Region to the Read Replica region.

For more information about data transfer pricing, see Amazon RDS Pricing.

For MySQL and MariaDB instances, you can reduce your data transfer costs by reducing the number of cross-region Read Replicas that you create. For example, suppose that you have a source DB instance in one AWS Region and want to have three Read Replicas in another AWS Region. In this case, you create only one of the Read Replicas from the source DB instance. You create the other two replicas from the first Read Replica instead of the source DB instance.

For example, if you have `source-instance-1` in one AWS Region, you can do the following:

- Create `read-replica-1` in the new AWS Region, specifying `source-instance-1` as the source.
- Create `read-replica-2` from `read-replica-1`.
- Create `read-replica-3` from `read-replica-1`.

In this example, you are only charged for the data transferred from `source-instance-1` to `read-replica-1`. You are not charged for the data transferred from `read-replica-1` to the other two

replicas because they are all in the same AWS Region. If you create all three replicas directly from `source-instance-1`, you are charged for the data transfers to all three replicas.

How Amazon RDS Does Cross-Region Replication

Amazon RDS uses the following process to create a cross-region Read Replica. Depending on the regions involved and the amount of data in the databases, this process can take hours to complete. You can use this information to determine how far the process has proceeded when you create a cross-region Read Replica:

1. Amazon RDS begins configuring the source DB instance as a replication source and sets the status to *modifying*.
2. Amazon RDS begins setting up the specified Read Replica in the destination AWS Region and sets the status to *creating*.
3. Amazon RDS creates an automated DB snapshot of the source DB instance in the source AWS Region. The format of the DB snapshot name is `rds:<InstanceID>-<timestamp>`, where `<InstanceID>` is the identifier of the source instance, and `<timestamp>` is the date and time the copy started. For example, `rds:mysourceinstance-2013-11-14-09-24` was created from the instance `mysourceinstance` at `2013-11-14-09-24`. During the creation of an automated DB snapshot, the source DB instance status remains *modifying*, the Read Replica status remains *creating*, and the DB snapshot status is *creating*. The progress column of the DB snapshot page in the console reports how far the DB snapshot creation has progressed. When the DB snapshot is complete, the status of both the DB snapshot and source DB instance are set to *available*.
4. Amazon RDS begins a cross-region snapshot copy for the initial data transfer. The snapshot copy is listed as an automated snapshot in the destination AWS Region with a status of *creating*. It has the same name as the source DB snapshot. The progress column of the DB snapshot display indicates how far the copy has progressed. When the copy is complete, the status of the DB snapshot copy is set to *available*.
5. Amazon RDS then uses the copied DB snapshot for the initial data load on the Read Replica. During this phase, the Read Replica is in the list of DB instances in the destination, with a status of *creating*. When the load is complete, the Read Replica status is set to *available*, and the DB snapshot copy is deleted.
6. When the Read Replica reaches the available status, Amazon RDS starts by replicating the changes made to the source instance since the start of the create Read Replica operation. During this phase, the replication lag time for the Read Replica will be greater than 0.

   For MySQL, MariaDB, and PostgreSQL Read Replicas, you can monitor replication lag in Amazon CloudWatch by viewing the Amazon RDS `ReplicaLag` metric. For MySQL and MariaDB, the `ReplicaLag` metric reports the value of the `Seconds_Behind_Master` field of the `SHOW SLAVE STATUS` command. For PostgreSQL, the `ReplicaLag` metric reports the value of `SELECT extract(epoch from now() - pg_last_xact_replay_timestamp()) AS slave_lag`.

   Common causes for replication lag for MySQL and MariaDB are the following:

   - A network outage.
   - Writing to tables with indexes on a Read Replica. If the `read_only` parameter is not set to 0 on the Read Replica, it can break replication.
   - Using a non-transactional storage engine such as MyISAM. Replication is only supported for the InnoDB storage engine on MySQL and the XtraDB storage engine on MariaDB.

   When the `ReplicaLag` metric reaches 0, the replica has caught up to the source DB instance. If the `ReplicaLag` metric returns -1, then replication is currently not active. `ReplicaLag` = -1 is equivalent to `Seconds_Behind_Master` = `NULL`.

   PostgreSQL (versions 9.4.7 and 9.5.2 exclusively) uses physical replication slots to manage Write Ahead Log (WAL) retention on the source instance. For each cross-region Read Replica instance,

Amazon RDS creates a physical replication slot and associates it with the instance. Two Amazon CloudWatch metrics, `Oldest Replication Slot Lag` and `Transaction Logs Disk Usage`, show how far behind the most lagging replica is in terms of WAL data received and how much storage is being used for WAL data. The `Transaction Logs Disk Usage` value can substantially increase when a cross-region Read Replica is lagging significantly.

Cross-Region Replication Examples

Example Create a Cross-Region Read Replica Outside of Any VPC

The following example creates a Read Replica in us-west-2 from a source DB instance in us-east-1. The Read Replica is created outside of a VPC:

For Linux, OS X, or Unix:

```
aws rds create-db-instance-read-replica \
    --db-instance-identifier SimCoProd01Replica01 \
    --region us-west-2
    --source-db-instance-identifier arn:aws:rds:us-east-1:123456789012:db:SimcoProd01
```

For Windows:

```
aws rds create-db-instance-read-replica ^
    --db-instance-identifier SimCoProd01Replica01 ^
    --region us-west-2
    --source-db-instance-identifier arn:aws:rds:us-east-1:123456789012:db:SimcoProd01
```

Example Create Cross-Region Read Replica in a VPC

This example creates a Read Replica in us-west-2 from a source DB instance in us-east-1. The Read Replica is created in the VPC associated with the specified DB subnet group:

For Linux, OS X, or Unix:

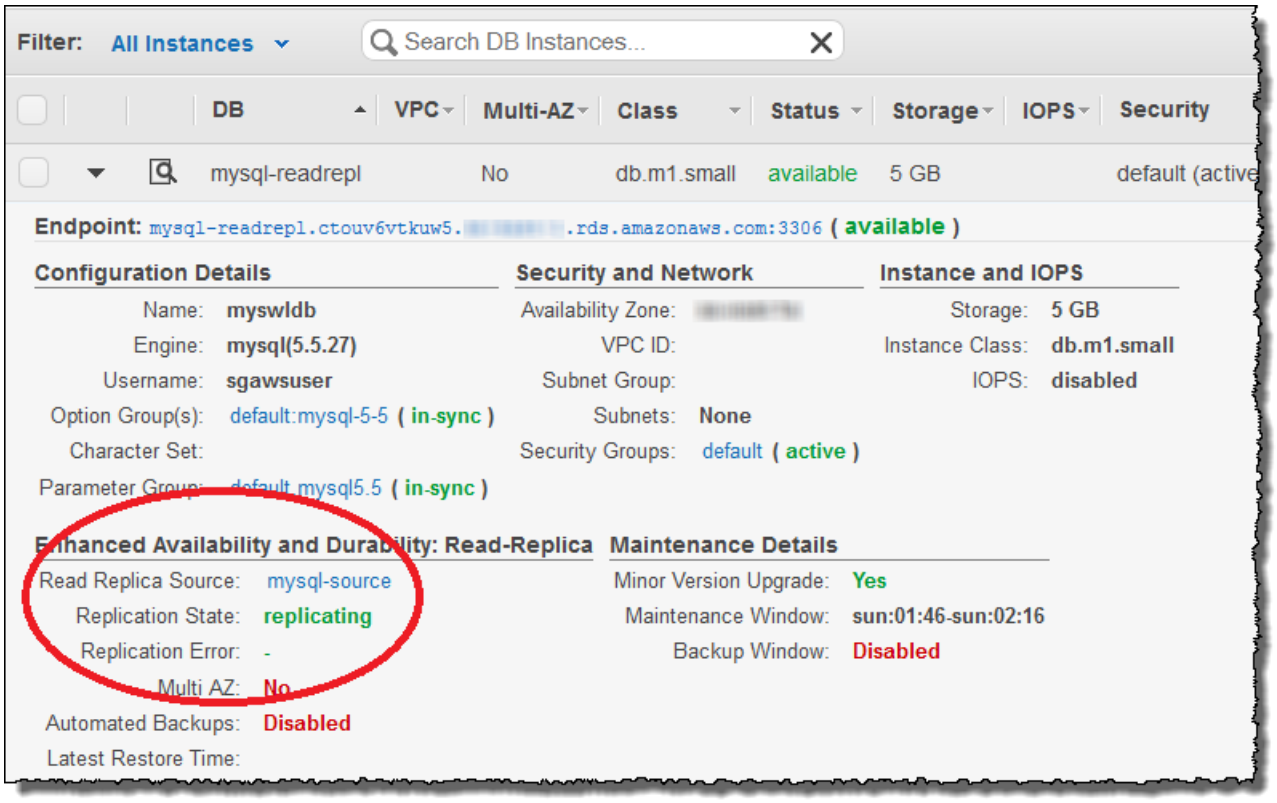
```
aws rds create-db-instance-read-replica \
    --db-instance-identifier SimCoProd01Replica01 \
    --region us-west-2
    --db-subnet-group-name my-us-west-2-subnet
    --source-db-instance-identifier arn:aws:rds:us-east-1:123456789012:db:SimcoProd01
```

For Windows:

```
aws rds create-db-instance-read-replica ^
    --db-instance-identifier SimCoProd01Replica01 ^
    --region us-west-2
    --db-subnet-group-name my-us-west-2-subnet
    --source-db-instance-identifier arn:aws:rds:us-east-1:123456789012:db:SimcoProd01
```

Monitoring Read Replication

You can monitor the status of a Read Replica in several ways. The Amazon RDS console shows the status of a Read Replica; you can also see the status of a Read Replica using the AWS CLI `describe-db-instances` command or the Amazon RDS API `DescribeDBInstances` action.



The status of a Read Replica can be one of the following:

- **Replicating—**The Read Replica is replicating successfully.
- **Error—**An error has occurred with the replication. Check the **Replication Error** field in the Amazon RDS console or the event log to determine the exact error. For more information about troubleshooting a replication error, see Troubleshooting a MySQL or MariaDB Read Replica Problem (p. 150).
- **Stopped—**(MySQL or MariaDB only) Replication has stopped because of a customer initiated request.
- **Terminated—**Replication is terminated. This occurs if replication is stopped for more than thirty consecutive days, either manually or due to a replication error. In this case, Amazon RDS terminates replication between the master DB instance and all Read Replicas in order to prevent increased storage requirements on the master DB instance and long failover times.

  Broken replication can affect storage because the logs can grow in size and number due to the high volume of errors messages being written to the log. Broken replication can also affect failure recovery due to the time Amazon RDS requires to maintain and process the large number of logs during recovery.

You can monitor how far a MySQL or MariaDB Read Replica is lagging the source DB instance by viewing the **Seconds_Behind_Master** data returned by the MySQL or MariaDB `Show Slave Status` command, or the CloudWatch **Replica Lag** statistic. If a replica lags too far behind for your environment, consider deleting and recreating the Read Replica. Also consider increasing the scale of the Read Replica to speed replication.

Troubleshooting a MySQL or MariaDB Read Replica Problem

MySQL and MariaDB's replication technologies are asynchronous. Because they are asynchronous, occasional `BinLogDiskUsage` increases on the source DB instance and `ReplicaLag` on the Read Replica are to be expected. For example, a high volume of write operations to the source DB instance can occur in parallel. In contrast, write operations to the Read Replica are serialized using a single I/O thread, which can lead to a lag between the source instance and Read Replica. For more information about read-only replicas in the MySQL documentation, see Replication Implementation Details. For more information about read-only replicas in the MariaDB documentation, go to Replication Overview.

You can do several things to reduce the lag between updates to a source DB instance and the subsequent updates to the Read Replica, such as the following:

- Sizing a Read Replica to have a storage size and DB instance class comparable to the source DB instance.
- Ensuring that parameter settings in the DB parameter groups used by the source DB instance and the Read Replica are compatible. For more information and an example, see the discussion of the `max_allowed_packet` parameter later in this section.

Amazon RDS monitors the replication status of your Read Replicas and updates the `Replication State` field of the Read Replica instance to `Error` if replication stops for any reason. An example might be if DML queries run on your Read Replica conflict with the updates made on the source DB instance.

You can review the details of the associated error thrown by the MySQL or MariaDB engines by viewing the `Replication Error` field. Events that indicate the status of the Read Replica are also generated, including RDS-EVENT-0045 (p. 283), RDS-EVENT-0046 (p. 283), and RDS-EVENT-0047 (p. 283). For more information about events and subscribing to events, see Using Amazon RDS Event Notification (p. 279). If a MySQL error message is returned, review the error number in the MySQL error message documentation. If a MariaDB error message is returned, review the error in the MariaDB error message documentation.

One common issue that can cause replication errors is when the value for the `max_allowed_packet` parameter for a Read Replica is less than the `max_allowed_packet` parameter for the source DB instance. The `max_allowed_packet` parameter is a custom parameter that you can set in a DB parameter group that is used to specify the maximum size of DML code that can be executed on the database. In some cases, the `max_allowed_packet` parameter value in the DB parameter group associated with a source DB instance is smaller than the `max_allowed_packet` parameter value in the DB parameter group associated with the source's Read Replica. In these cases, the replication process can throw an error (Packet bigger than 'max_allowed_packet' bytes) and stop replication. You can fix the error by having the source and Read Replica use DB parameter groups with the same `max_allowed_packet` parameter values.

Other common situations that can cause replication errors include the following:

- Writing to tables on a Read Replica. If you are creating indexes on a Read Replica, you need to have the `read_only` parameter set to **0** to create the indexes. If you are writing to tables on the Read Replica, it might break replication.
- Using a non-transactional storage engine such as MyISAM. Read replicas require a transactional storage engine. Replication is only supported for the InnoDB storage engine on MySQL and the XtraDB storage engine on MariaDB.
- Using unsafe nondeterministic queries such as `SYSDATE()`. For more information, see Determination of Safe and Unsafe Statements in Binary Logging.

If you decide that you can safely skip an error, you can follow the steps described in the section Skipping the Current Replication Error (p. 905). Otherwise, you can delete the Read Replica and create an instance using the same DB instance identifier so that the endpoint remains the same as that of your old Read Replica. If a replication error is fixed, the `Replication State` changes to *replicating*.

Troubleshooting a PostgreSQL Read Replica Problem

PostgreSQL uses replication slots for cross-region replication, so the process for troubleshooting same-region replication problems and cross-region replication problems is different.

Troubleshooting PostgreSQL Read Replica Problems Within an AWS Region

The PostgreSQL parameter, `wal_keep_segments`, dictates how many Write Ahead Log (WAL) files are kept to provide data to the Read Replicas. The parameter value specifies the number of logs to keep. If you set the parameter value too low, you can cause a Read Replica to fall so far behind that streaming replication stops. In this case, Amazon RDS reports a replication error and begins recovery on the Read Replica by replaying the source DB instance's archived WAL logs. This recovery process continues until the Read Replica has caught up enough to continue streaming replication.

The PostgreSQL log on the Read Replica shows when Amazon RDS is recovering a Read Replica that is this state by replaying archived WAL files.

```
 2014-11-07 19:01:10 UTC::@:[23180]:DEBUG:  switched WAL source from archive to stream
after
     failure 2014-11-07 19:01:10 UTC::@:[11575]:LOG:  started streaming WAL from primary
at
     1A/D3000000 on timeline 1 2014-11-07 19:01:10 UTC::@:[11575]:FATAL:  could not
receive
     data from WAL stream: ERROR:  requested WAL segment 000000010000001A000000D3 has
already been
     removed 2014-11-07 19:01:10 UTC::@:[23180]:DEBUG:  could not restore file
     "00000002.history" from archive: return code 0 2014-11-07 19:01:15
     UTC::@:[23180]:DEBUG:  switched WAL source from stream to archive after failure
     recovering 000000010000001A000000D3 2014-11-07 19:01:16 UTC::@:[23180]:LOG:  restored
log file "000000010000001A000000D3"
       from archive
```

After a certain amount of time, Amazon RDS replays enough archived WAL files on the replica to catch up and allow the Read Replica to begin streaming again. At this point, PostgreSQL resumes streaming and writes a similar line to the following to the log file.

```
 2014-11-07 19:41:36 UTC::@:[24714]:LOG:  started streaming WAL from primary at 1B/
B6000000
     on timeline 1
```

You can determine how many WAL files you should keep by looking at the checkpoint information in the log. The PostgreSQL log shows the following information at each checkpoint. By looking at the "# recycled" transaction log files of these log statements, a user can understand how many transaction files will be recycled during a time range and use this information to tune the `wal_keep_segments` parameter.

```
 2014-11-07 19:59:35 UTC::@:[26820]:LOG:  checkpoint complete: wrote 376 buffers (0.2%); 0
     transaction log file(s) added, 0 removed, 1 recycled; write=35.681 s, sync=0.013 s,
     total=35.703 s; sync files=10, longest=0.013 s, average=0.001 s
```

For example, if the PostgreSQL log shows that 35 files are recycled from the "checkpoint completed" log statements within a 5-minute time frame, we know that with this usage pattern a Read Replica relies on 35 transaction files in five minutes and can't survive 5 minutes in a nonstreaming state if the source DB instance is set to the default `wal_keep_segments` parameter value of 32.

Troubleshooting PostgreSQL Read Replica Problems Across AWS Regions

PostgreSQL (versions 9.4.7 and 9.5.2 exclusively) uses physical replication slots to manage Write Ahead Log (WAL) retention on the source DB instance. For each cross-region Read Replica instance, Amazon RDS creates and associates a physical replication slot. You can use two Amazon CloudWatch metrics, `Oldest Replication Slot Lag` and `Transaction Logs Disk Usage`, to see how far behind the most lagging replica is in terms of WAL data received and to see how much storage is being used for WAL data. The `Transaction Logs Disk Usage` value can substantially increase when a cross-region Read Replica is lagging significantly.

If the workload on your DB instance generates a large amount of WAL data, you might need to change the DB instance class of your source DB instance and Read Replica to one with High / 10Gb network performance for the replica to keep up. The Amazon CloudWatch metric `Transaction Logs Generation` can help you understand the rate at which your workload is generating WAL data.

To determine the status of a cross-region Read Replica, you can query `pg_replication_slots` on the source instance, as in the following example:

```
postgres=# select * from pg_replication_slots;

                  slot_name                  | plugin | slot_type | datoid | database |
 active | active_pid | xmin | catalog_xmin | restart_lsn
-------------------------------------------------------------------------------------------
 rds_us_east_1_db_uzwlholddgpblksce6hgw4nkte |        | physical  |        |          | t
      |      12598 |      |              | 4E/95000060
(1 row)
```

Working with Option Groups

Some DB engines offer additional features that make it easier to manage data and databases, and to provide additional security for your database. Amazon RDS uses option groups to enable and configure these features. An *option group* can specify features, called options, that are available for a particular Amazon RDS DB instance. Options can have settings that specify how the option works. When you associate a DB instance with an option group, the specified options and option settings are enabled for that DB instance.

Amazon RDS supports options for the following database engines:

Database Engine	Relevant Documentation
`MariaDB`	Appendix: Options for MariaDB Database Engine (p. 709)
`Microsoft SQL Server`	Options for the Microsoft SQL Server Database Engine (p. 795)
`MySQL`	Options for MySQL DB Instances (p. 897)
`Oracle`	Options for Oracle DB Instances (p. 993)

Option Groups Overview

Amazon RDS provides an empty default option group for each new DB instance. You cannot modify this default option group, but any new option group that you create derives its settings from the default option group. To apply an option to a DB instance, you must do the following:

1. Create a new option group, or copy or modify an existing option group.
2. Add one or more options to the option group.
3. Associate the option group with the DB instance.

Both DB instances and DB snapshots can be associated with an option group. When you restore from a DB snapshot or perform a point-in-time restore for a DB instance, the option group associated with the DB snapshot or DB instance will, by default, be associated with the restored DB instance. You can associate a different option group with a restored DB instance. However, the new option group must contain any persistent or permanent options that were included in the original option group. Persistent and permanent options are described following.

Options require additional memory to run on a DB instance, so you might need to launch a larger instance to use them, depending on your current use of your DB instance. For example, Oracle Enterprise Manager Database Control uses about 300 MB of RAM; if you enable this option for a small DB instance, you might encounter performance problems or out-of-memory errors.

Each DB instance indicates the status of its association with an option group. For example, a status of **active** indicates the DB instance is associated with that option group, and a status of **invalid** indicates that the option group associated with the DB instance does not contain the options the DB instance requires. If you query a DB instance for the status of its associated option group, Amazon RDS can also return a status such as **pending** or **applying** when it is attempting to change the association from one state to another. For example, the status of the association of a DB instance in an option group can be **creating/pending**.

Persistent and Permanent Options

Two types of options, persistent and permanent, require special consideration when you add them to an option group.

Persistent options, such as the TDE option for Microsoft SQL Server transparent data encryption (TDE), cannot be removed from an option group while DB instances are associated with the option group. You must disassociate all DB instances from the option group before a persistent option can be removed from the option group. When you restore or perform a point-in-time restore from a DB snapshot, if the option group associated with that DB snapshot contains a persistent option, you can only associate the restored DB instance with that option group.

Permanent options, such as the TDE option for Oracle Advanced Security TDE, can never be removed from an option group, and the option group cannot be disassociated from the DB instance. When you restore or perform a point-in-time restore from a DB snapshot, if the option group associated with that DB snapshot contains a permanent option, you can only associate the restored DB instance with an option group with that permanent option.

VPC and Platform Considerations

When an option group is assigned to a DB instance, it is linked to the platform that the DB instance is on. That platform can either be a VPC supported by the Amazon Virtual Private Cloud (Amazon VPC) service, or EC2-Classic (non-VPC) supported by the Amazon Elastic Compute Cloud (Amazon EC2) service. For details on these two platforms, see Amazon EC2 and Amazon Virtual Private Cloud.

If a DB instance is in a VPC, the option group associated with the instance is linked to that VPC. This means that you cannot use the option group assigned to a DB instance if you attempt to restore the instance into a different VPC or onto a different platform. If you restore a DB instance into a different VPC or onto a different platform, you must either assign the default option group to the DB instance, assign an option group that is linked to that VPC or platform, or create a new option group and assign it to the DB instance. Note that with persistent or permanent options, such as Oracle TDE, you must create a new option group that includes the persistent or permanent option when restoring a DB instance into a different VPC.

Option settings control the behavior of an option. For example, the Oracle Advanced Security option `NATIVE_NETWORK_ENCRYPTION` has a setting that you can use to specify the encryption algorithm for network traffic to and from the DB instance. Some options settings are optimized for use with Amazon RDS and cannot be changed.

Mutually Exclusive Options

Some options are mutually exclusive. You can use one or the other, but not both at the same time. The following options are mutually exclusive:

- Oracle Enterprise Manager Database Express (p. 1007) and Oracle Management Agent for Enterprise Manager Cloud Control (p. 1010).
- Oracle Native Network Encryption (p. 1003) and Oracle SSL (p. 1021).
- Oracle Transparent Data Encryption (p. 1036) and Using AWS CloudHSM Classic to Store Amazon RDS Oracle TDE Keys (p. 1086).

Creating an Option Group

You can create a new option group that derives its settings from the default option group, and then add one or more options to the new option group. Alternatively, if you already have an existing option group, you can copy that option group with all of its options to a new option group. For more information, see Making a Copy of an Option Group (p. 156).

After you create a new option group, it has no options. To learn how to add options to the option group, see Adding an Option to an Option Group (p. 157). After you have added the options you want, you

can then associate the option group with a DB instance so that the options become available on the DB instance. For information about associating an option group with a DB instance, see the documentation for your specific engine listed at Working with Option Groups (p. 153).

AWS Management Console

One way of creating on option group is by using the AWS Management Console.

To create a new option group by using the console

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Option Groups**.
3. Choose **Create Group**.
4. In the **Create Option Group** dialog box, do the following:
   a. For **Name**, type a name for the option group that is unique within your AWS account. The name can contain only letters, digits, and hyphens.
   b. For **Description**, type a brief description of the option group. The description is used for display purposes.
   c. For **Engine**, choose the DB engine that you want.
   d. For **Major Engine Version**, choose the major version of the DB engine that you want.
5. To continue, choose **Yes, Create**. To cancel the operation instead, choose **Cancel**.

CLI

To create an option group, use the AWS CLI `create-option-group` command with the following required parameters.

- `--option-group-name`
- `--engine-name`
- `--major-engine-version`
- `--option-group-description`

Example

The following example creates an option group named `TestOptionGroup`, which is associated with the Oracle Enterprise Edition DB engine. The description is enclosed in quotation marks.

For Linux, OS X, or Unix:

```
aws rds create-option-group \
    --option-group-name testoptiongroup \
    --engine-name oracle-ee \
    --major-engine-version 11.2 \
    --option-group-description "Test option group"
```

For Windows:

```
aws rds create-option-group ^
    --option-group-name testoptiongroup ^
    --engine-name oracle-ee ^
```

```
    --major-engine-version 11.2 ^
    --option-group-description "Test option group"
```

API

To create an option group, call the Amazon RDS API `CreateOptionGroup` action. Include the following parameters:

- `OptionGroupName = testoptiongroup`
- `EngineName = oracle-ee`
- `MajorEngineVersion = 11.2`
- `OptionGroupDescription = Test%20option%20group`

Example

```
https://rds.us-east-1.amazonaws.com/
   ?Action=CreateOptionGroup
   &EngineName=oracle-ee
   &MajorEngineVersion=11.2
   &OptionGroupDescription=test%20option%20group
   &OptionGroupName=testoptiongroup
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &Version=2014-09-01
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20140425/us-east-1/rds/aws4_request
   &X-Amz-Date=20140425T174519Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
   &X-Amz-Signature=d3a89afa4511d0c4ecab046d6dc760a72bfe6bb15999cce053adeb2617b60384
```

Making a Copy of an Option Group

You can use the AWS CLI or the Amazon RDS API to make a copy of an option group. Copying an option group is a convenient solution when you have already created an option group and you want to include most of the custom parameters and values from that group in a new option group. You can also make a copy of an option group that you use in production and then modify the copy to test other option settings.

CLI

To copy an option group, use the AWS CLI copy-option-group command. Include the following required parameters:

- `--source-option-group-identifier`
- `--target-option-group-identifier`
- `--target-option-group-description`

Example

The following example creates an option group named `new-local-option-group`, which is a local copy of the option group `my-remote-option-group`.

For Linux, OS X, or Unix:

```
aws rds copy-option-group \
    --source-option-group-identifier arn:aws:rds:us-west-2:123456789012:og:my-remote-
option-group \
    --target-option-group-identifier new-local-option-group \
    --target-option-group-description "Option group 2"
```

For Windows:

```
aws rds copy-option-group ^
    --source-option-group-identifier arn:aws:rds:us-west-2:123456789012:og:my-remote-
option-group ^
    --target-option-group-identifier new-local-option-group ^
    --target-option-group-description "Option group 2"
```

API

To copy an option group, call the Amazon RDS API CopyOptionGroup action. Include the following required parameters.

- `SourceOptionGroupIdentifier` = *arn%3Aaws%3Ards%3Aus-west-2%3A123456789012%3og%3Amy-remote-option-group*
- `TargetOptionGroupIdentifier` = *new-local-option-group*
- `TargetOptionGroupDescription` = *Option%20group%202*

Example

The following example creates an option group named `new-local-option-group`, which is a local copy of the option group `my-remote-option-group`.

```
https://rds.us-east-1.amazonaws.com/
   ?Action=CopyOptionGroup
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &SourceOptionGroupIdentifier=arn%3Aaws%3Ards%3Aus-west-2%3A123456789012%3og%3Amy-remote-
option-group
   &TargetOptionGroupDescription=New%20option%20group
   &TargetOptionGroupIdentifier=new-local-option-group
   &Version=2014-09-01
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20140429/us-east-1/rds/aws4_request
   &X-Amz-Date=20140429T175351Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
   &X-Amz-Signature=9164337efa99caf850e874a1cb7ef62f3cea29d0b448b9e0e7c53b288ddffed2
```

Adding an Option to an Option Group

You can add an option to an existing option group. After you have added the options you want, you can then associate the option group with a DB instance so that the options become available on the DB instance. For information about associating an option group with a DB instance, see the documentation for your specific DB engine listed at Working with Option Groups (p. 153).

Option group changes must be applied immediately in two cases:

- When you add an option that adds or updates a port value, such as the `OEM` option.
- When you add or remove an option group with an option that includes a port value.

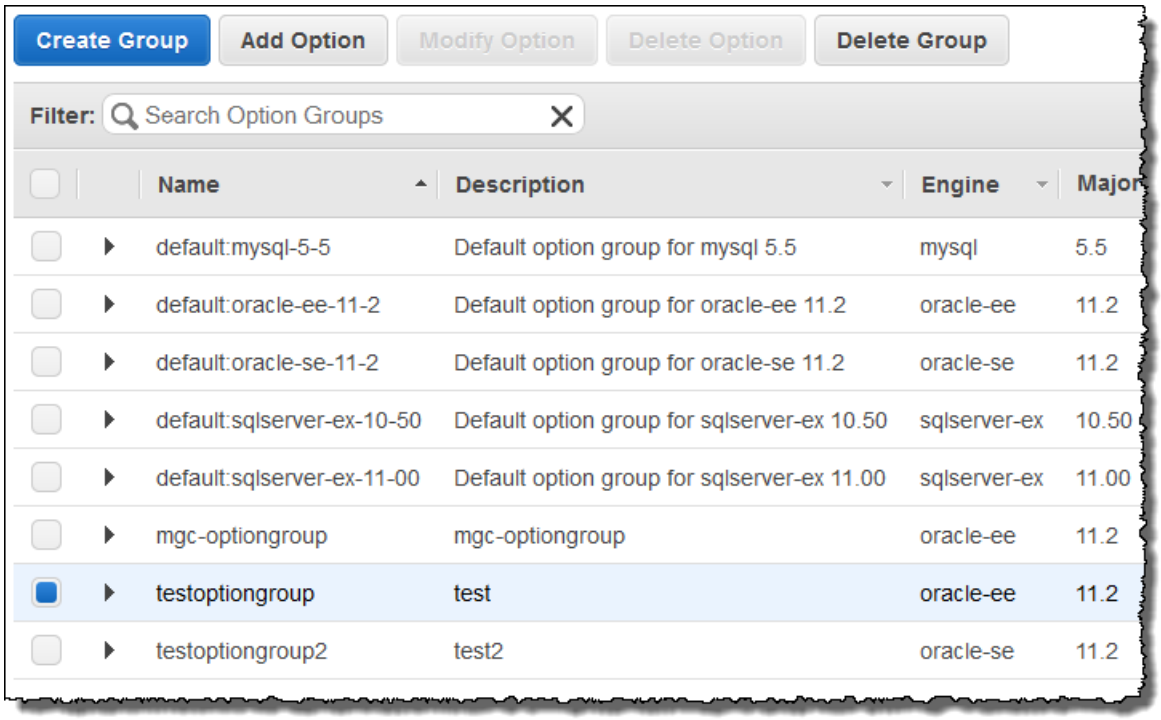
In these cases, you must select the **Apply Immediately** option in the console, or include the `Apply-Immediately` option when using the AWS CLI or set the `Apply-Immediately` parameter to `true` when using the Amazon RDS API. Options that don't include port values can be applied immediately, or can be applied during the next maintenance window for the DB instance.

AWS Management Console

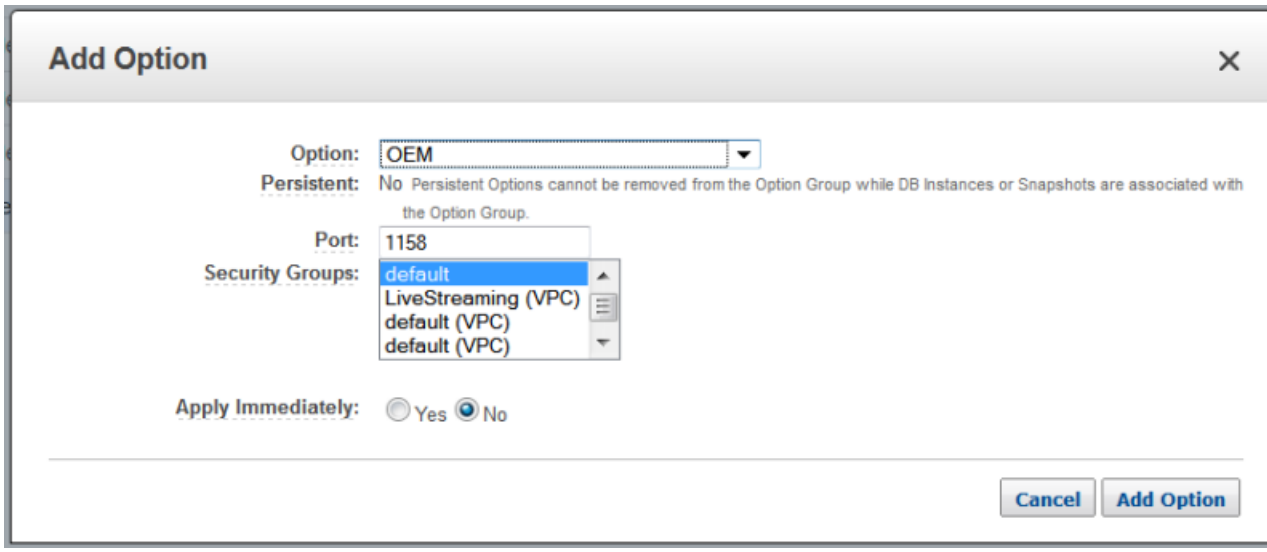
You can use the AWS Management Console to add an option to an option group.

To add an option to an option group by using the console

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Option Groups**.
3. Select the option group that you want to modify, and then choose **Add Option**.



4. In the **Add Option** dialog box, do the following:
   a. Choose the option that you want to add. You might need to provide additional values, depending on the option that you select. For example, when you choose the `OEM` option, you must also type a port value and specify a DB security group.
   b. To enable the option on all associated DB instances as soon as you add it, for **Apply Immediately**, choose **Yes**. If you choose **No** (the default), the option is enabled for each associated DB instance during its next maintenance window.



5. When the settings are as you want them, choose **Add Option**.

CLI

To add an option to an option group, run the AWS CLI add-option-to-option-group command with the option that you want to add. To enable the new option immediately on all associated DB instances, include the `--apply-immediately` parameter. By default, the option is enabled for each associated DB instance during its next maintenance window. Include the following required parameter:

- `--option-group-name`

Example

The following example adds the Oracle Enterprise Manager Database Control (OEM) option to an option group named `TestOptionGroup` and immediately enables it. Note that even if you use the default security group, you must specify that security group.

For Linux, OS X, or Unix:

```
aws rds add-option-to-option-group \
--option-group-name TestOptionGroup \
--option-name OEM \
--security-groups default \
--apply-immediately
```

For Windows:

```
aws rds add-option-to-option-group \
--option-group-name TestOptionGroup \
--option-name OEM \
--security-groups default \
--apply-immediately
```

Command output is similar to the following:

```
OPTIONGROUP testoptiongroup oracle-ee 11.2 Test option group
    OPTION OEM 1158 Oracle Enterprise Manager
        SECGROUP default authorized
```

Example

The following example adds the Oracle OEM option to an option group, specifies a custom port, and specifies a pair of Amazon EC2 VPC security groups to use for that port.

For Linux, OS X, or Unix:

```
aws rds add-option-to-option-group \
    --option-group-name my-option-group \
    --option-name OEM \
    --port 5432 \
    --vpcsg sg-454fa22a,sg-5da54932
```

For Windows:

```
aws rds add-option-to-option-group ^
    --option-group-name my-option-group ^
    --option-name OEM ^
    --port 5432 ^
    --vpcsg sg-454fa22a,sg-5da54932
```

Command output is similar to the following:

```
OPTIONGROUP  my-option-group  oracle-se  11.2  My option group
OPTION  OEM  n  5432  Oracle Enterprise Manager
VPCSECGROUP  sg-454fa22a  active
VPCSECGROUP  sg-5da54932  active
```

Example

The following example adds the Oracle option NATIVE_NETWORK_ENCRYPTION to an option group and specifies the option settings. If no option settings are specified, default values are used.

For Linux, OS X, or Unix:

```
aws rds add-option-to-option-group \
    --option-group-name my-option-group \
    --options NATIVE_NETWORK_ENCRYPTION \
    --settings "SQLNET.ENCRYPTION_SERVER=REQUIRED;
 SQLNET.ENCRYPTION_TYPES_SERVER=AES256,AES192,DES"
```

For Windows:

```
aws rds add-option-to-option-group ^
    --option-group-name my-option-group ^
    --options NATIVE_NETWORK_ENCRYPTION ^
    --settings "SQLNET.ENCRYPTION_SERVER=REQUIRED;
 SQLNET.ENCRYPTION_TYPES_SERVER=AES256,AES192,DES"
```

Command output is similar to the following:

```
OPTIONGROUP  Group Name       Engine     Major Engine Version  Description      VpcSpecific
```

```
OPTIONGROUP  my-option-group  oracle-ee  11.2                  My option group  n
    OPTION  Name                        Persistent Permanent Description
    OPTION  NATIVE_NETWORK_ENCRYPTION   n          n         Oracle Advanced Security -
 Native Network Encryption
      OPTIONSETTING  Name                                  Description
                                  Value              Modifiable
      OPTIONSETTING  SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER  Specifies list of checksumming
 algorithms in order of intended use  SHA1,MD5           true
      OPTIONSETTING  SQLNET.ENCRYPTION_TYPES_SERVER       Specifies list of encryption
 algorithms in order of intended use    AES256,AES192,DES  true
      OPTIONSETTING  SQLNET.ENCRYPTION_SERVER             Specifies the desired encryption
 behavior                          REQUIRED           true
      OPTIONSETTING  SQLNET.CRYPTO_CHECKSUM_SERVER        Specifies the desired data
 integrity behavior                        REQUESTED          true
```

API

To add an option to an option group using the Amazon RDS API, call the ModifyOptionGroup action with the option that you want to add. To enable the new option immediately on all associated DB instances, include the `ApplyImmediately` parameter and set it to `true`. By default, the option is enabled for each associated DB instance during its next maintenance window. Include the following required parameter:

- `OptionGroupName`

Example

```
https://rds.us-east-1.amazonaws.com/
    ?Action=ModifyOptionGroup
    &ApplyImmediately=true
    &OptionGroupName=myawsuser-og02
    &OptionsToInclude.member.1.DBSecurityGroupMemberships.member.1=default
    &OptionsToInclude.member.1.OptionName=MEMCACHED
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-09-01
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20140501/us-east-1/rds/aws4_request
    &X-Amz-Date=20140501T230529Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=4b278baae6294738704a9948e355af0e9bd4fa0913d5b35b0a9a3c916925aced
```

Listing the Options and Option Settings for an Option Group

You can list all the options and option settings for an option group.

AWS Management Console

You can use the AWS Management Console to list all of the options and option settings for an option group.

To list the options and option settings for an option group

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Option Groups**. The **Options** column in the table shows the options and option settings in the option group.

CLI

To list the options and option settings for an option group, use the AWS CLI `describe-option-groups` command. Specify the name of the option group whose options and settings you want to view. If you don't specify an option group name, all option groups are described.

Example

The following example lists the options and option settings for all option groups.

```
aws rds describe-option-groups
```

Example

The following example lists the options and option settings for an option group named `TestOptionGroup`.

```
aws rds describe-option-groups --option-group-name TestOptionGroup
```

API

To list the options and option settings for an option group, use the Amazon RDS API `DescribeOptionGroups` action. Specify the name of the option group whose options and settings you want to view. If you don't specify an option group name, all option groups are described.

Example

The following example lists the options and option settings for all option groups.

```
https://rds.us-west-2.amazonaws.com/
    ?Action=DescribeOptionGroups
    &MaxRecords=100
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-09-01
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20140613/us-west-2/rds/aws4_request
    &X-Amz-Date=20140613T223341Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=5ae331adcd684c27d66e0b794a51933effe32a4c026eba2e994ae483ee47a0ba
```

The output from the preceding action is similar to the following:

```
<DescribeOptionGroupsResponse xmlns="http://rds.amazonaws.com/doc/2014-10-31/">
  <DescribeOptionGroupsResult>
    <OptionGroupsList>
      <OptionGroup>
        <OptionGroupName>default:mysql-5-5</OptionGroupName>
        <AllowsVpcAndNonVpcInstanceMemberships>true</AllowsVpcAndNonVpcInstanceMemberships>
        <MajorEngineVersion>5.5</MajorEngineVersion>
        <EngineName>mysql</EngineName>
        <OptionGroupDescription>Default option group for mysql 5.5</OptionGroupDescription>
        <Options/>
      </OptionGroup>

      <!-- some output omitted for brevity -->

      <OptionGroup>
```

```
        <OptionGroupName>default:postgres-9-3</OptionGroupName>
        <AllowsVpcAndNonVpcInstanceMemberships>true</AllowsVpcAndNonVpcInstanceMemberships>
        <MajorEngineVersion>9.3</MajorEngineVersion>
        <EngineName>postgres</EngineName>
        <OptionGroupDescription>Default option group for postgres 9.3</
OptionGroupDescription>
        <Options/>
      </OptionGroup>
    </OptionGroupsList>
  </DescribeOptionGroupsResult>
  <ResponseMetadata>
    <RequestId>b2ce0772-f55a-11e3-bd0f-bb88ac05a37c</RequestId>
  </ResponseMetadata>
</DescribeOptionGroupsResponse>
```

Example

The following example lists the options and option settings for an option group named `myawsuser-grp1`.

```
https://rds.us-east-1.amazonaws.com/
   ?Action=DescribeOptionGroups
   &MaxRecords=100
   &OptionGroupName=myawsuser-grp1
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &Version=2014-09-01
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20140421/us-east-1/rds/aws4_request
   &X-Amz-Date=20140421T231357Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
   &X-Amz-Signature=fabfbeb85c44e3f151d44211790c5135a9074fdb8d85ec117788ac6cfab6c5bc
```

The output from the preceding action is similar to the following:

```
<DescribeOptionGroupsResponse xmlns="http://rds.amazonaws.com/doc/2014-10-31/">
  <DescribeOptionGroupsResult>
    <OptionGroupsList>
      <OptionGroup>
        <AllowsVpcAndNonVpcInstanceMemberships>true</AllowsVpcAndNonVpcInstanceMemberships>
        <MajorEngineVersion>5.6</MajorEngineVersion>
        <OptionGroupName>myawsuser-grp1</OptionGroupName>
        <EngineName>mysql</EngineName>
        <OptionGroupDescription>my test option group</OptionGroupDescription>
        <Options/>
      </OptionGroup>
    </OptionGroupsList>
  </DescribeOptionGroupsResult>
  <ResponseMetadata>
    <RequestId>8c6201fc-b9ff-11d3-f92b-31fa5e8dbc99</RequestId>
  </ResponseMetadata>
</DescribeOptionGroupsResponse>
```

Modifying an Option Setting

After you have added an option that has modifiable option settings, you can modify the settings at any time. If you change options or option settings in an option group, those changes are applied to all DB instances that are associated with that option group. For more information on what settings are available for the various options, see the documentation for your specific engine listed at Working with Option Groups (p. 153).

Option group changes must be applied immediately in two cases:

- When you add an option that adds or updates a port value, such as the `OEM` option.
- When you add or remove an option group with an option that includes a port value.

In these cases, you must select the **Apply Immediately** option in the console, or include the `Apply-Immediately` option when using the AWS CLI or set the `Apply-Immediately` parameter to `true` when using the Amazon RDS API. Options that don't include port values can be applied immediately, or can be applied during the next maintenance window for the DB instance.

AWS Management Console

You can use the AWS Management Console to modify an option setting.

To modify an option setting by using the console

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Option Groups**.
3. Select the option group whose option that you want to modify, and then choose **Modify Option**.
4. In the **Modify Option** dialog box, from **Installed Options**, choose the option whose setting you want to modify. Make the changes that you want.
5. To enable the option as soon as you add it, for **Apply Immediately**, choose **Yes**. If you choose **No** (the default), the option is enabled for each associated DB instance during its next maintenance window.
6. When the settings are as you want them, choose **Modify Option**.

CLI

To modify an option setting, use the AWS CLI `add-option-to-option-group` command with the option group and option that you want to modify. By default, the option is enabled for each associated DB instance during its next maintenance window. To apply the change immediately to all associated DB instances, include the `--apply-immediately` parameter. To modify an option setting, use the `--settings` argument.

Example

The following example modifies the port that the Oracle Enterprise Manager Database Control (OEM) uses in an option group named `TestOptionGroup` and immediately applies the change.

For Linux, OS X, or Unix:

```
aws rds add-option-to-option-group \
    --option-group-name TestOptionGroup \
    --option-name OEM \
    --port 5432 \
    --apply-immediately
```

For Windows:

```
aws rds add-option-to-option-group ^
    --option-group-name TestOptionGroup ^
    --option-name OEM ^
    --port 5432 ^
```

```
    --apply-immediately
```

Command output is similar to the following:

```
OPTIONGROUP testoptiongroup oracle-ee 11.2 Test Option Group
    OPTION OEM 5432 Oracle Enterprise Manager
        SECGROUP default authorized
```

Example

The following example modifies the Oracle option NATIVE_NETWORK_ENCRYPTION and changes the option settings.

For Linux, OS X, or Unix:

```
aws rds add-option-to-option-group \
    --option-group-name my-option-group \
    --option-name NATIVE_NETWORK_ENCRYPTION \
    --settings "SQLNET.ENCRYPTION_SERVER=REQUIRED;
 SQLNET.ENCRYPTION_TYPES_SERVER=AES256,AES192,DES"
```

For Windows:

```
aws rds add-option-to-option-group ^
    --option-group-name my-option-group ^
    --option-name NATIVE_NETWORK_ENCRYPTION ^
    --settings "SQLNET.ENCRYPTION_SERVER=REQUIRED;
 SQLNET.ENCRYPTION_TYPES_SERVER=AES256,AES192,DES"
```

Command output is similar to the following:

```
OPTIONGROUP  Group Name       Engine     Major Engine Version  Description      VpcSpecific
OPTIONGROUP  my-option-group  oracle-ee  11.2                  My option group  n
    OPTION  Name                         Persistent Permanent Description
    OPTION  NATIVE_NETWORK_ENCRYPTION    n          n         Oracle Advanced Security -
 Native Network Encryption
      OPTIONSETTING  Name                                    Description
                                  Value              Modifiable
      OPTIONSETTING  SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER  Specifies list of checksumming
 algorithms in order of intended use  SHA1,MD5           true
      OPTIONSETTING  SQLNET.ENCRYPTION_TYPES_SERVER       Specifies list of encryption
 algorithms in order of intended use    AES256,AES192,DES  true
      OPTIONSETTING  SQLNET.ENCRYPTION_SERVER             Specifies the desired encryption
 behavior                            REQUIRED           true
      OPTIONSETTING  SQLNET.CRYPTO_CHECKSUM_SERVER        Specifies the desired data
 integrity behavior                       REQUESTED          true
```

API

To modify an option setting, use the Amazon RDS API `ModifyOptionGroup` command with the option group and option that you want to modify. By default, the option is enabled for each associated DB instance during its next maintenance window. To apply the change immediately to all associated DB instances, include the `ApplyImmediately` parameter and set it to `true`.

Example

```
https://rds.us-east-1.amazonaws.com/
```

```
    ?Action=ModifyOptionGroup
    &ApplyImmediately=true
    &OptionGroupName=myawsuser-og02
    &OptionsToInclude.member.1.DBSecurityGroupMemberships.member.1=default
    &OptionsToInclude.member.1.OptionName=MEMCACHED
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-09-01
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20140501/us-east-1/rds/aws4_request
    &X-Amz-Date=20140501T230529Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=4b278baae6294738704a9948e355af0e9bd4fa0913d5b35b0a9a3c916925aced
```

Output from the preceding action should look similar to the following:

```
<ModifyOptionGroupResponse xmlns="http://rds.amazonaws.com/doc/2014-10-31/">
  <ModifyOptionGroupResult>
    <OptionGroup>
      <OptionGroupName>myawsuser-og02</OptionGroupName>
      <MajorEngineVersion>5.6</MajorEngineVersion>
      <AllowsVpcAndNonVpcInstanceMemberships>false</AllowsVpcAndNonVpcInstanceMemberships>
      <EngineName>mysql</EngineName>
      <OptionGroupDescription>my second og</OptionGroupDescription>
      <Options>
        <Option>
          <Port>11211</Port>
          <OptionName>MEMCACHED</OptionName>
          <OptionDescription>Innodb Memcached for MySQL</OptionDescription>
          <Persistent>false</Persistent>
          <OptionSettings>
            <OptionSetting>
              <DataType>BOOLEAN</DataType>
              <IsModifiable>true</IsModifiable>
              <IsCollection>false</IsCollection>
              <Description>If enabled when there is no more memory to store items,
 memcached will return an error rather than evicting items.</Description>
              <Name>ERROR_ON_MEMORY_EXHAUSTED</Name>
              <Value>0</Value>
              <ApplyType>STATIC</ApplyType>
              <AllowedValues>0,1</AllowedValues>
              <DefaultValue>0</DefaultValue>
            </OptionSetting>
            <OptionSetting>
              <DataType>INTEGER</DataType>
              <IsModifiable>true</IsModifiable>
              <IsCollection>false</IsCollection>
              <Description>The backlog queue configures how many network connections can be
 waiting to be processed by memcached</Description>
              <Name>BACKLOG_QUEUE_LIMIT</Name>
              <Value>1024</Value>
              <ApplyType>STATIC</ApplyType>
              <AllowedValues>1-2048</AllowedValues>
              <DefaultValue>1024</DefaultValue>
            </OptionSetting>
          </OptionSettings>
          <VpcSecurityGroupMemberships/>
          <Permanent>false</Permanent>
          <DBSecurityGroupMemberships>
            <DBSecurityGroup>
              <Status>authorized</Status>
              <DBSecurityGroupName>default</DBSecurityGroupName>
            </DBSecurityGroup>
          </DBSecurityGroupMemberships>
        </Option>
```

```
      </Options>
    </OptionGroup>
  </ModifyOptionGroupResult>
  <ResponseMetadata>
    <RequestId>073cfb45-c184-11d3-a537-cef97546330c</RequestId>
  </ResponseMetadata>
</ModifyOptionGroupResponse>
```

Removing an Option from an Option Group

Some options can be removed from an option group, and some cannot. A persistent option cannot be removed from an option group until all DB instances associated with that option group are disassociated. A permanent option can never be removed from an option group. For more information about what options are removable, see the documentation for your specific engine listed at Working with Option Groups (p. 153).

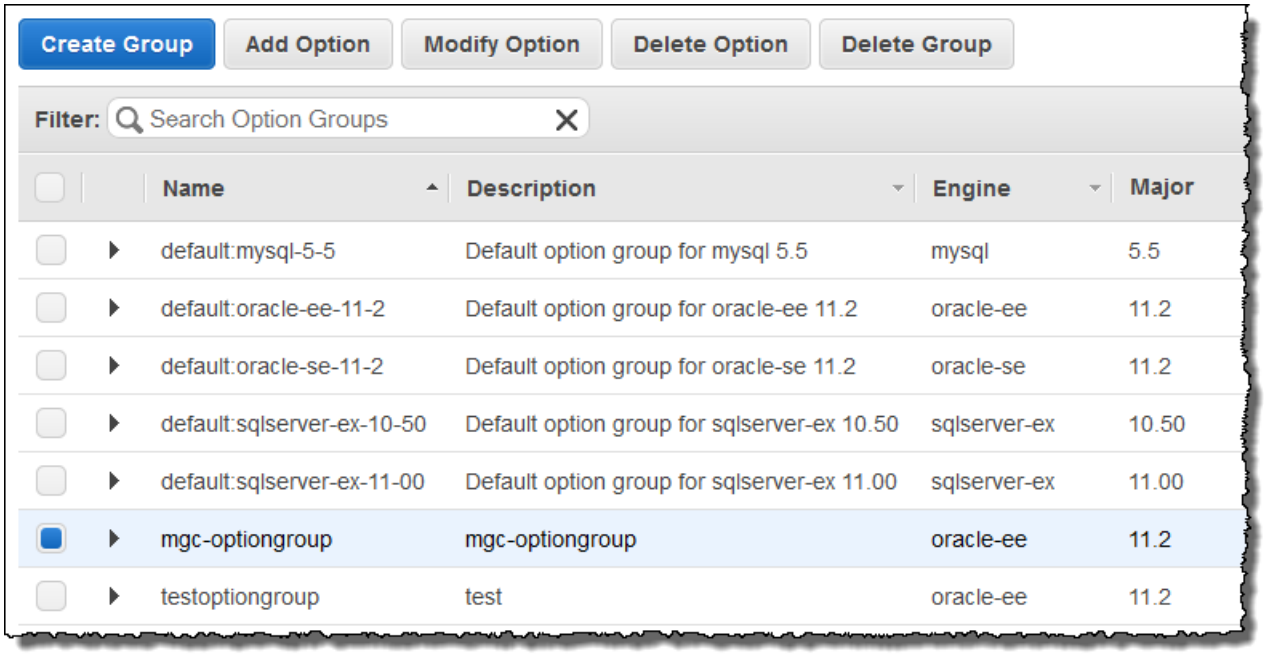
If you remove all options from an option group, Amazon RDS doesn't delete the option group. DB instances that are associated with the empty option group continue to be associated with it; they just won't have any active options. Alternatively, to remove all options from a DB instance, you can associate the DB instance with the default (empty) option group.

AWS Management Console

You can use the AWS Management Console to remove an option from an option group.

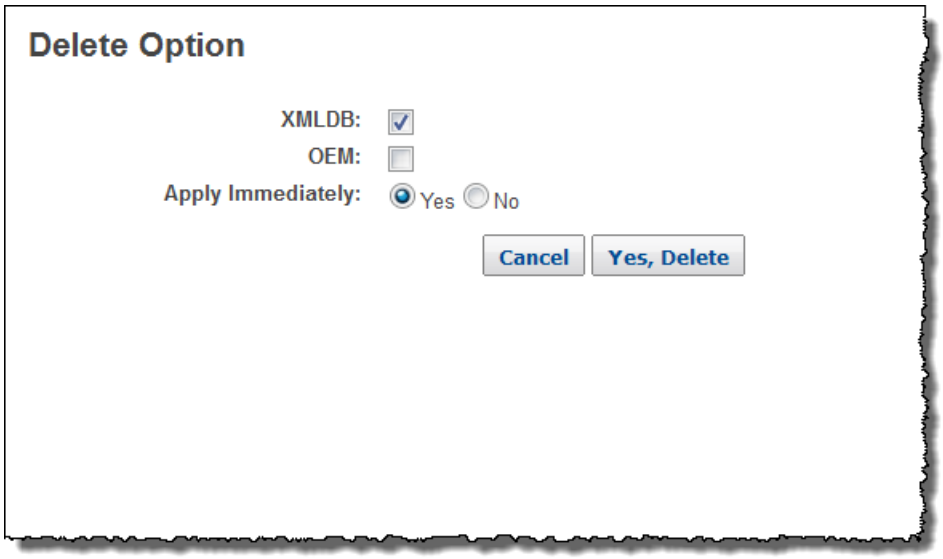
To remove an option from an option group by using the console

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Option Groups**.
3. Select the option group whose option you want to remove, and then choose **Delete Option**.



4. In the **Delete Option** dialog box, do the following:
   - Select the check box for the option that you want to delete.

- For the deletion to take effect as soon as you make it, for **Apply Immediately**, choose **Yes**. If you choose **No** (the default), the option is deleted for each associated DB instance during its next maintenance window.



5. When the settings are as you want them, choose **Yes, Delete**.

CLI

To remove an option from an option group, use the AWS CLI `remove-option-from-option-group` command with the option that you want to delete. By default, the option is removed from each associated DB instance during its next maintenance window. To apply the change immediately, include the `--apply-immediately` parameter.

Example

The following example removes the Oracle Enterprise Manager Database Control (OEM) option from an option group named `TestOptionGroup` and immediately applies the change.

For Linux, OS X, or Unix:

```
aws rds remove-option-from-option-group \
    --option-group-name TestOptionGroup \
    --options OEM \
    --apply-immediately
```

For Windows:

```
aws rds remove-option-from-option-group ^
    --option-group-name TestOptionGroup ^
    --options OEM ^
    --apply-immediately
```

Command output is similar to the following:

```
OPTIONGROUP    testoptiongroup oracle-ee   11.2    Test option group
```

API

To remove an option from an option group, use the Amazon RDS API `ModifyOptionGroup` action. By default, the option is removed from each associated DB instance during its next maintenance window. To apply the change immediately, include the `ApplyImmediately` parameter and set it to `true`.

Include the following parameters:

- `OptionGroupName = myawsuser-og02`
- `OptionsToRemove.OptionName = OEM`

Example

The following example removes the Oracle Enterprise Manager Database Control (OEM) option from an option group named `TestOptionGroup` and immediately applies the change.

```
https://rds.us-east-1.amazonaws.com/
    ?Action=ModifyOptionGroup
    &ApplyImmediately=true
    &OptionGroupName=myawsuser-og02
    &OptionsToRemove.OptionName=OEM
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-09-01
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20140501/us-east-1/rds/aws4_request
    &X-Amz-Date=20140501T231731Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=fd7ee924d39f1014488eb3444a8fdfb028e958b97703f95845a5addc435c1399
```

The output from the preceding command should look something like the following:

```
<ModifyOptionGroupResponse xmlns="http://rds.amazonaws.com/doc/2014-10-31/">
  <ModifyOptionGroupResult>
    <OptionGroup>
      <OptionGroupName>myawsuser-og02</OptionGroupName>
      <AllowsVpcAndNonVpcInstanceMemberships>true</AllowsVpcAndNonVpcInstanceMemberships>
      <MajorEngineVersion>5.6</MajorEngineVersion>
      <EngineName>mysql</EngineName>
      <OptionGroupDescription>my second og</OptionGroupDescription>
      <Options/>
    </OptionGroup>
  </ModifyOptionGroupResult>
  <ResponseMetadata>
    <RequestId>b5f134f3-c185-11d3-f4c6-37db295f7674</RequestId>
  </ResponseMetadata>
</ModifyOptionGroupResponse>
```

Working with DB Parameter Groups

You manage your DB engine configuration through the use of parameters in a DB parameter group. DB parameter groups act as a *container* for engine configuration values that are applied to one or more DB instances.

A default DB parameter group is created if you create a DB instance without specifying a customer-created DB parameter group. This default group contains database engine defaults and Amazon RDS system defaults based on the engine, compute class, and allocated storage of the instance. You cannot modify the parameter settings of a default DB parameter group; you must create your own DB parameter group to change parameter settings from their default value. Note that not all DB engine parameters can be changed in a customer-created DB parameter group.

If you want to use your own DB parameter group, you simply create a new DB parameter group, modify the desired parameters, and modify your DB instance to use the new DB parameter group. All DB instances that are associated with a particular DB parameter group get all parameter updates to that DB parameter group. You can also copy an existing parameter group with the AWS CLI copy-db-parameter-group command. Copying a parameter group is a convenient solution when you have already created a DB parameter group and you want to include most of the custom parameters and values from that group in a new DB parameter group.

Here are some important points you should know about working with parameters in a DB parameter group:

- When you change a dynamic parameter and save the DB parameter group, the change is applied immediately regardless of the **Apply Immediately** setting. When you change a static parameter and save the DB parameter group, the parameter change will take effect after you manually reboot the DB instance. You can reboot a DB instance using the RDS console or explicitly calling the `RebootDbInstance` API action (without failover, if the DB instance is in a Multi-AZ deployment). The requirement to reboot the associated DB instance after a static parameter change helps mitigate the risk of a parameter misconfiguration affecting an API call, such as calling `ModifyDBInstance` to change DB instance class or scale storage.
- When you change the DB parameter group associated with a DB instance, you must manually reboot the instance before the new DB parameter group is used by the DB instance.
- The value for a DB parameter can be specified as an integer; an integer expression built from formulas, variables, functions, and operators; or as a log expression. For more information, see DB Parameter Values (p. 180)
- Set any parameters that relate to the character set or collation of your database in your parameter group prior to creating the DB instance and before you create a database in your DB instance. This ensures that the default database and new databases in your DB instance use the character set and collation values that you specify. If you change character set or collation parameters for your DB instance, the parameter changes are not applied to existing databases.

  You can change character set or collation values for an existing database using the `ALTER DATABASE` command, for example:

```
ALTER DATABASE database_name CHARACTER SET character_set_name COLLATE collation;
```

- Improperly setting parameters in a DB parameter group can have unintended adverse effects, including degraded performance and system instability. Always exercise caution when modifying database parameters and back up your data before modifying a DB parameter group. You should try out parameter group setting changes on a test DB instance before applying those parameter group changes to a production DB instance.
- Amazon Aurora uses both DB parameter groups and DB cluster parameter groups. Parameters in a DB parameter group apply to a single DB instance in an Aurora DB cluster. Parameters in a DB cluster

parameter group apply to every DB instance in a DB cluster. For more information, see Amazon Aurora DB Cluster and DB Instance Parameters (p. 469).

Topics

Creating a DB Parameter Group

The following section shows you how to create a new DB parameter group.

AWS Management Console

To create a DB parameter group

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. Click **Parameter Groups** in the **Navigation** list on the left side of the window.
3. Click the **Create DB Parameter Group** button.

   The **Create DB Parameter Group** window appears.
4. Select a DB parameter group family in the **DB Parameter Group Family** drop-down list box.
5. Type the name of the new DB parameter group in the **DB Parameter Group** text box.
6. Type a description for the new DB parameter group in the **Description** text box.
7. Click the **Yes, Create** button.

CLI

To create a DB parameter group, use the AWS CLI `create-db-parameter-group` command. The following example creates a DB parameter group named *mydbparametergroup* for MySQL version 5.6 with a description of "*My new parameter group*."

Include the following required parameters:

- `--db-parameter-group-name`
- `--db-parameter-group-family`
- `--description`

Example

For Linux, OS X, or Unix:

```
aws rds create-db-parameter-group \
    --db-parameter-group-name mydbparametergroup \
    --db-parameter-group-family MySQL5.6 \
    --description "My new parameter group"
```

For Windows:

```
aws rds create-db-parameter-group ^
    --db-parameter-group-name mydbparametergroup ^
    --db-parameter-group-family MySQL5.6 ^
    --description "My new parameter group"
```

This command produces output similar to the following:

```
DBPARAMETERGROUP  mydbparametergroup  mysql5.6  My new parameter group
```

API

To create a DB parameter group, use the Amazon RDS API `CreateDBParameterGroup` action. The following example creates a DB parameter group named *mydbparametergroup* for MySQL version 5.6 with a description of "*My new parameter group*."

Include the following required parameters:

- `DBParameterGroupName = mydbparametergroup`
- `DBParameterGroupFamily = MySQL5.6`
- `Description = My new parameter group`

Example

```
https://rds.amazonaws.com/
 ?Action=CreateDBParameterGroup
 &DBParameterGroupName=mydbparametergroup
 &Description=My%20new%20parameter%20group
 &DBParameterGroupFamily=MySQL5.6
 &Version=2012-01-15
 &SignatureVersion=2
 &SignatureMethod=HmacSHA256
 &Timestamp=2012-01-15T22%3A06%3A23.624Z
 &AWSAccessKeyId=<AWS Access Key ID>
 &Signature=<Signature>
```

The command returns a response like the following:

```
<CreateDBParameterGroupResponse xmlns="http://rds.amazonaws.com/admin/2012-01-15/">
  <CreateDBParameterGroupResult>
    <DBParameterGroup>
      <DBParameterGroupFamily>mysql5.6</DBParameterGroupFamily>
      <Description>My new parameter group</Description>
      <DBParameterGroupName>mydbparametergroup</DBParameterGroupName>
    </DBParameterGroup>
  </CreateDBParameterGroupResult>
  <ResponseMetadata>
    <RequestId>700a8afe-0b81-11df-85f9-eb5c71b54ddc</RequestId>
  </ResponseMetadata>
</CreateDBParameterGroupResponse>
```

Modifying Parameters in a DB Parameter Group

You can modify parameter values in a customer-created DB parameter group; you cannot change the parameter values in a default DB parameter group. Changes to parameters in a customer-created DB parameter group are applied to all DB instances that are associated with the DB parameter group.

If you change a parameter value, when the change is applied is determined by the type of parameter. Changes to dynamic parameters are applied immediately. Changes to static parameters require that the DB instance associated with DB parameter group be rebooted before the change takes effect. To determine the type of a parameter, list the parameters in a parameter group using one of the procedures shown in the section Listing DB Parameter Groups (p. 176).

The RDS console shows the status of the DB parameter group associated with a DB instance. For example, if the DB instance is not using the latest changes to its associated DB parameter group, the RDS console shows the DB parameter group with a status of **pending-reboot**. You would need to manually reboot the DB instance for the latest parameter changes to take effect for that DB instance.

AWS Management Console

To modify a DB parameter group

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. Click **Parameter Groups** in the navigation pane on the left side of the window.

   The available DB parameter groups appear in a list.
3. In the list, select the parameter group you want to modify.
4. Select **Edit Parameters**.
5. Change the values of the parameters you want to modify. You can scroll through the parameters using the arrow keys at the top right of the dialog box.

   Note that you cannot change values in a default parameter group.
6. Click **Save Changes**.

CLI

To modify a DB parameter group, use the AWS CLI `modify-db-parameter-group` command with the following required parameters:

- `--db-parameter-group-name`
- `--parameters`

The following example modifies the `max_connections` and `max_allowed_packet` values in the DB parameter group named *mydbparametergroup*.

Note
Amazon RDS does not support passing multiple comma-delimited parameter values for a single parameter.

Example

For Linux, OS X, or Unix:

```
aws rds modify-db-parameter-group \
    --db-parameter-group-name mydbparametergroup \
    --parameters "ParameterName=max_connections,ParameterValue=250,ApplyMethod=immediate" \
    --parameters
 "ParameterName=max_allowed_packet,ParameterValue=1024,ApplyMethod=immediate"
```

For Windows:

```
aws rds modify-db-parameter-group ^
    --db-parameter-group-name mydbparametergroup ^
    --parameters "ParameterName=max_connections,ParameterValue=250,ApplyMethod=immediate" ^
    --parameters
 "ParameterName=max_allowed_packet,ParameterValue=1024,ApplyMethod=immediate"
```

The command produces output like the following:

```
DBPARAMETERGROUP  mydbparametergroup
```

API

To modify a DB parameter group, use the Amazon RDS API `ModifyDBParameterGroup` command with the following required parameters:

- `DBParameterGroupName`
- `Parameters`

The following example modifies the `max_connections` and `max_allowed_packet` values in the DB parameter group named *mydbparametergroup*.

Note
Amazon RDS does not support passing multiple comma-delimited parameter values for a single parameter.

Example

```
https://rds.amazonaws.com/
    ?Action=ModifyDBParameterGroup
    &DBParameterGroupName=mydbparametergroup
    &Parameters.member.1.ParameterName=max_connections
    &Parameters.member.1.ParameterValue=250
    &Parameters.member.1.ApplyMethod=immediate
    &Parameters.member.2.ParameterName=max_allowed_packet
    &Parameters.member.2.ParameterValue=1024
    &Parameters.member.2.ApplyMethod=immediate
    &Version=2012-01-15
    &SignatureVersion=2
```

```
    &SignatureMethod=HmacSHA256
    &Timestamp=2012-01-15T22%3A29%3A47.865Z
```

The command returns a response like the following:

```
<ModifyDBParameterGroupResponse xmlns="http://rds.amazonaws.com/admin/2012-01-15/">
  <ModifyDBParameterGroupResult>
    <DBParameterGroupName>mydbparametergroup</DBParameterGroupName>
  </ModifyDBParameterGroupResult>
  <ResponseMetadata>
    <RequestId>3b824e10-0b87-11df-972f-21e99bc6881d</RequestId>
  </ResponseMetadata>
</ModifyDBParameterGroupResponse>
```

Copying a DB Parameter Group

You can copy custom DB parameter groups that you create. Copying a parameter group is a convenient solution when you have already created a DB parameter group and you want to include most of the custom parameters and values from that group in a new DB parameter group. You can copy a DB parameter group by using the AWS CLI copy-db-parameter-group command or the Amazon RDS API CopyDBParameterGroup action.

After you copy a DB parameter group, you should wait at least 5 minutes before creating your first DB instance that uses that DB parameter group as the default parameter group. This allows Amazon RDS to fully complete the copy action before the parameter group is used as the default for a new DB instance. This is especially important for parameters that are critical when creating the default database for a DB instance, such as the character set for the default database defined by the `character_set_database` parameter. You can use the **Parameter Groups** option of the Amazon RDS console or the describe-db-parameters command to verify that your DB parameter group has been created.

CLI

To copy a DB parameter group, use the AWS CLI copy-db-parameter-group command with the following required parameters:

- `--source-db-parameter-group-identifier`
- `--target-db-parameter-group-identifier`
- `--target-db-parameter-group-description`

The following example creates a new DB parameter group named `mygroup2` that is a copy of the DB parameter group `mygroup1`.

Example

For Linux, OS X, or Unix:

```
aws rds copy-db-parameter-group \
    --source-db-parameter-group-identifier mygroup1 \
    --target-db-parameter-group-identifier mygroup2 \
    --target-db-parameter-group-description "DB parameter group 2"
```

For Windows:

```
aws rds copy-db-parameter-group ^
    --source-db-parameter-group-identifier mygroup1 ^
```

```
    --target-db-parameter-group-identifier mygroup2 ^
    --target-db-parameter-group-description "DB parameter group 2"
```

API

To copy a DB parameter group, use the RDS API CopyDBParameterGroup action with the following required parameters:

- `SourceDBParameterGroupIdentifier = arn%3Aaws%3Ards%3Aus-west-2%3A123456789012%3Apg%3Amygroup1`
- `TargetDBParameterGroupIdentifier = mygroup2`
- `TargetDBParameterGroupDescription = DB%20parameter%20group%202`

The following example creates a new DB parameter group named `mygroup2` that is a copy of the DB parameter group `mygroup1`.

Example

```
https://rds.us-east-1.amazonaws.com/
   ?Action=CopyDBParameterGroup
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &SourceDBParameterGroupIdentifier=arn%3Aaws%3Ards%3Aus-west-2%3A123456789012%3Apg
%3Amygroup1
   &TargetDBParameterGroupIdentifier=mygroup2
   &TargetDBParameterGroupDescription=DB%20parameter%20group%202
   &Version=2014-09-01
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20140922/us-east-1/rds/aws4_request
   &X-Amz-Date=20140922T175351Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
   &X-Amz-Signature=5164017efa99caf850e874a1cb7ef62f3ddd29d0b448b9e0e7c53b288ddffed2
```

The command returns a response like the following:

```
<CopyDBParameterGroupResponse xmlns="http://rds.amazonaws.com/doc/2014-10-31/">
  <CopyDBParameterGroupResult>
    <DBParameterGroup>
      <DBParameterGroupFamily>mysql5.6</DBParameterGroupFamily>
      <Description>DB parameter group 2</Description>
      <DBParameterGroupName>mygroup2</DBParameterGroupName>
    </DBParameterGroup>
  </CopyDBParameterGroupResult>
  <ResponseMetadata>
    <RequestId>3328d60e-beb6-11d3-8e5c-3ccda5460d76</RequestId>
  </ResponseMetadata>
</CopyDBParameterGroupResponse>
```

Listing DB Parameter Groups

You can list the DB parameter groups you've created for your AWS account.

> **Note**
> Default parameter groups are automatically created from a default parameter template when you create a DB instance for a particular DB engine and version. These default parameter groups contain preferred parameter settings and cannot be modified. When you create a custom parameter group, you can modify parameter settings.

AWS Management Console

To list all DB parameter groups for an AWS account

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. Click **Parameter Groups** in the navigation pane on the left side of the window.

   The DB parameter groups appear in a list.

CLI

To list all DB parameter groups for an AWS account, use the AWS CLI `describe-db-parameter-groups` command.

Example

The following example lists all available DB parameter groups for an AWS account.

```
aws rds describe-db-parameter-groups
```

The command returns a response like the following:

```
DBPARAMETERGROUP  default.mysql5.5     mysql5.5  Default parameter group for MySQL5.5
DBPARAMETERGROUP  default.mysql5.6     mysql5.6  Default parameter group for MySQL5.6
DBPARAMETERGROUP  mydbparametergroup   mysql5.6  My new parameter group
```

The following example describes the *mydbparamgroup1* parameter group.

For Linux, OS X, or Unix:

```
aws rds describe-db-parameter-groups \
    --db-parameter-group-name mydbparamgroup1
```

For Windows:

```
aws rds describe-db-parameter-groups ^
    --db-parameter-group-name mydbparamgroup1
```

The command returns a response like the following:

```
DBPARAMETERGROUP  mydbparametergroup1  mysql5.5  My new parameter group
```

API

To list all DB parameter groups for an AWS account, use the RDS API `DescribeDBParameterGroups` action.

Example

The following example lists all available DB parameter groups for an AWS account.

```
https://rds.amazonaws.com/
 ?Action=DescribeDBParameterGroups
 &MaxRecords=100
 &Version=2012-01-15
```

```
 &SignatureVersion=2
 &SignatureMethod=HmacSHA256
 &Timestamp=2009-10-22T19%3A31%3A42.262Z
 &AWSAccessKeyId=<AWS Access Key ID>
 &Signature=<Signature>
```

The command returns a response like the following:

```
<DescribeDBParameterGroupsResponse xmlns="http://rds.amazonaws.com/admin/2012-01-15/">
    <DescribeDBParameterGroupsResult>
        <DBParameterGroups>
            <DBParameterGroup>
                <Engine>mysql5.6</Engine>
                <Description>Default parameter group for MySQL5.6</Description>
                <DBParameterGroupName>default.mysql5.6</DBParameterGroupName>
            </DBParameterGroup>
            <DBParameterGroup>
                <Engine>mysql5.6</Engine>
                <Description>My new parameter group</Description>
                <DBParameterGroupName>mydbparametergroup</DBParameterGroupName>
            </DBParameterGroup>
        </DBParameterGroups>
    </DescribeDBParameterGroupsResult>
    <ResponseMetadata>
        <RequestId>41731881-0b82-11df-9a9b-c1bd5894571c</RequestId>
    </ResponseMetadata>
</DescribeDBParameterGroupsResponse>
```

The following example describes the *mydbparamgroup1* parameter group.

```
https://rds.amazonaws.com/
 ?Action=DescribeDBParameterGroups
 &DBParameterGroupName=mydbparamgroup1
 &MaxRecords=100
 &Version=2012-01-15
 &SignatureVersion=2
 &SignatureMethod=HmacSHA256
 &Timestamp=2009-10-22T19%3A31%3A42.262Z
 &AWSAccessKeyId=<AWS Access Key ID>
 &Signature=<Signature>
```

The command returns a response like the following:

```
<DescribeDBParameterGroupsResponse xmlns="http://rds.amazonaws.com/admin/2012-01-15/">
    <DescribeDBParameterGroupsResult>
        <DBParameterGroups>
            <DBParameterGroup>
                <Engine>mysql5.6</Engine>
                <Description>My new parameter group</Description>
                <DBParameterGroupName>mydbparamgroup1</DBParameterGroupName>
            </DBParameterGroup>
        </DBParameterGroups>
    </DescribeDBParameterGroupsResult>
    <ResponseMetadata>
        <RequestId>41731881-0b82-11df-9a9b-c1bd5894571c</RequestId>
    </ResponseMetadata>
</DescribeDBParameterGroupsResponse>
```

Viewing Parameter Values for a DB Parameter Group

You can get a list of all parameters in a DB parameter group and their values.

AWS Management Console

To view the parameter values for a DB parameter group

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. Click **Parameter Groups** in the navigation pane on the left side of the window.

   The DB parameter groups appear in a list.
3. Select a DB parameter group from the list. Click the Details page icon to see the list of parameters for the selected DB parameter group.

CLI

To view the parameter values for a DB parameter group, use the AWS CLI `describe-db-parameters` command with the following required parameter.

- `--db-parameter-group-name`

Example

The following example lists the parameters and parameter values for a DB parameter group named *mydbparametergroup*.

```
aws rds describe-db-parameters --db-parameter-group-name mydbparametergroup
```

The command returns a response like the following:

```
DBPARAMETER  Parameter Name            Parameter Value  Source           Data Type  Apply
 Type  Is Modifiable
DBPARAMETER  allow-suspicious-udfs                      engine-default   boolean    static
     false
DBPARAMETER  auto_increment_increment                   engine-default   integer    dynamic
     true
DBPARAMETER  auto_increment_offset                      engine-default   integer    dynamic
     true
DBPARAMETER  binlog_cache_size         32768            system           integer    dynamic
     true
DBPARAMETER  socket                    /tmp/mysql.sock  system           string     static
     false
```

API

To view the parameter values for a DB parameter group, use the Amazon RDS API `DescribeDBParameters` command with the following required parameter.

- `DBParameterGroupName` = *mydbparametergroup*

Example

The following example lists the parameters and parameter values for a DB parameter group named *mydbparametergroup*.

```
https://rds.amazonaws.com/
    ?Action=DescribeDBParameters
```

```
    &DBParameterGroupName=mydbparametergroup
    &MaxRecords=100
    &Version=2012-01-15
    &SignatureVersion=2
    &SignatureMethod=HmacSHA256
    &Timestamp=2009-10-22T19%3A31%3A42.262Z
    &AWSAccessKeyId=<AWS Access Key ID>
    &Signature=<Signature>
```

The command returns a response like the following:

```
<DescribeDBParametersResponse xmlns="http://rds.amazonaws.com/admin/2012-01-15/">
  <DescribeDBParametersResult>
    <Marker>bWF4X3RtcF90YWJsZXM=</Marker>
    <Parameters>
      <Parameter>
        <DataType>boolean</DataType>
        <Source>engine-default</Source>
        <IsModifiable>false</IsModifiable>
        <Description>Controls whether user-defined functions that have only an xxx symbol
 for the main function can be loaded</Description>
        <ApplyType>static</ApplyType>
        <AllowedValues>0,1</AllowedValues>
        <ParameterName>allow-suspicious-udfs</ParameterName>
      </Parameter>
      <Parameter>
        <DataType>integer</DataType>
        <Source>engine-default</Source>
        <IsModifiable>true</IsModifiable>
        <Description>Intended for use with master-to-master replication, and can be used to
 control the operation of AUTO_INCREMENT columns</Description>
        <ApplyType>dynamic</ApplyType>
        <AllowedValues>1-65535</AllowedValues>
        <ParameterName>auto_increment_increment</ParameterName>
      </Parameter>
      <Parameter>
        <DataType>integer</DataType>
        <Source>engine-default</Source>
        <IsModifiable>true</IsModifiable>
        <Description>Determines the starting point for the AUTO_INCREMENT column value</
Description>
        <ApplyType>dynamic</ApplyType>
        <AllowedValues>1-65535</AllowedValues>
        <ParameterName>auto_increment_offset</ParameterName>
      </Parameter>

      (... sample truncated...)

    </Parameters>
  </DescribeDBParametersResult>
  <ResponseMetadata>
    <RequestId>99c0937a-0b83-11df-85f9-eb5c71b54ddc</RequestId>
  </ResponseMetadata>
</DescribeDBParametersResponse>
```

DB Parameter Values

The value for a DB parameter can be specified as:

- An integer constant
- A DB parameter formula
- A DB parameter function

- A character string constant
- A log expression (the log function represents log base 2), such as value=**{log(DBInstanceClassMemory/8187281418)*1000}**

DB Parameter Formulas

A DB parameter formula is an expression that resolves to an integer value, and is enclosed in braces: {}. Formulas can be specified for either a DB parameter value or as an argument to a DB parameter function.

Syntax

```
{FormulaVariable}
```

```
{FormulaVariable*Integer}
```

```
{FormulaVariable*Integer/Integer}
```

```
{FormulaVariable/Integer}
```

DB Parameter Formula Variables

Formula variables return integers. The names of the variables are case sensitive.

AllocatedStorage

Returns the size, in bytes, of the data volume.

DBInstanceClassMemory

Returns the number of bytes of memory allocated to the DB instance class associated with the current DB instance, less the memory used by the Amazon RDS processes that manage the instance.

EndPointPort

Returns the number of the port used when connecting to the DB instance.

DB Parameter Formula Operators

DB parameter formulas support two operators: division and multiplication.

Division Operator: /

Divides the dividend by the divisor, returning an integer quotient. Decimals in the quotient are truncated, not rounded.

Syntax

```
dividend / divisor
```

The dividend and divisor arguments must be integer expressions.

Multiplication Operator: *

Divides the dividend by the divisor, returning an integer quotient. Decimals in the quotient are truncated, not rounded.

Syntax

```
expression * expression
```

Both expressions must be integers.

DB Parameter Functions

The parameter arguments can be specified as either integers or formulas. Each function must have at least one argument. Multiple arguments can be specified as a comma-separated list. The list cannot have any empty members, such as *argument1*,,*argument3*. Function names are case insensitive.

Note
DB Parameter functions are not currently supported in CLI.

GREATEST()

Returns the largest value from a list of integers or parameter formulas.

Syntax

```
GREATEST(argument1, argument2,...argumentn)
```

Returns an integer.

LEAST()

Returns the smallest value from a list of integers or parameter formulas.

Syntax

```
LEAST(argument1, argument2,...argumentn)
```

Returns an integer.

SUM()

Adds the values of the specified integers or parameter formulas.

Syntax

```
SUM(argument1, argument2,...argumentn)
```

Returns an integer.

DB Parameter Value Examples

These examples show using formulas and functions in the values for DB parameters.

Warning
Improperly setting parameters in a DB parameter group can have unintended adverse effects, including degraded performance and system instability. Always exercise caution when modifying database parameters and back up your data before modifying your DB parameter group. You should try out parameter group changes on a test DB instances, created using point-in-time-restores, before applying those parameter group changes to your production DB instances.

You can specify the GREATEST function in an Oracle processes parameter to set the number of user processes to the larger of either 80 or DBInstanceClassMemory divided by 9868951.

```
GREATEST({DBInstanceClassMemory/9868951},80)
```

You can specify the LEAST() function in a MySQL max_binlog_cache_size parameter value to set the maximum cache size a transaction can use in a MySQL instance to the lesser of 1MB or DBInstanceClass/256:

```
LEAST({DBInstanceClassMemory/256},10485760)
```

Working with Amazon Resource Names (ARNs) in Amazon RDS

Resources created in Amazon Web Services are each uniquely identified with an Amazon Resource Name (ARN). For certain Amazon RDS operations, you must uniquely identify an Amazon RDS resource by specifying its ARN. For example, when you create an RDS DB instance Read Replica, you must supply the ARN for the source DB instance.

Constructing an ARN for Amazon RDS

Resources created in Amazon Web Services are each uniquely identified with an Amazon Resource Name (ARN). You can construct an ARN for an Amazon RDS resource using the following syntax.

`arn:aws:rds:<region>:<account number>:<resourcetype>:<name>`

Region	Name	Endpoint
US West (Oregon) Region	us-west-2	https://rds.us-west-2.amazonaws.com
US West (N. California) Region	us-west-1	https://rds.us-west-1.amazonaws.com
US East (Ohio) Region	us-east-2	https://rds.us-east-2.amazonaws.com
US East (N. Virginia) Region	us-east-1	https://rds.us-east-1.amazonaws.com
Asia Pacific (Mumbai) Region	ap-south-1	https://rds.ap-south-1.amazonaws.com
Asia Pacific (Seoul) Region	ap-northeast-2	https://rds.ap-northeast-2.amazonaws.com
Asia Pacific (Singapore) Region	ap-southeast-1	https://rds.ap-southeast-1.amazonaws.com
Asia Pacific (Sydney) Region	ap-southeast-2	https://rds.ap-southeast-2.amazonaws.com
Asia Pacific (Tokyo) Region	ap-northeast-1	https://rds.ap-northeast-1.amazonaws.com
Canada (Central) Region	ca-central-1	https://rds.ca-central-1.amazonaws.com
China (Beijing) Region	cn-north-1	https://rds.cn-north-1.amazonaws.com.cn
EU (Frankfurt) Region	eu-central-1	https://rds.eu-central-1.amazonaws.com
EU (Ireland) Region	eu-west-1	https://rds.eu-west-1.amazonaws.com
EU (London) Region	eu-west-2	https://rds.eu-west-2.amazonaws.com
South America (São Paulo) Region	sa-east-1	https://rds.sa-east-1.amazonaws.com
AWS GovCloud (US)	us-gov-west-1	https://rds.us-gov-west-1.amazonaws.com

The following table shows the format that you should use when constructing an ARN for a particular Amazon RDS resource type.

Resource Type	ARN Format
DB instance	arn:aws:rds:*<region>*:*<account>*:db:*<name>*

Resource Type	ARN Format
	For example: `arn:aws:rds:us-east-2:123456789012:db:my-mysql-instance-1`
DB cluster	arn:aws:rds:*<region>*:*<account>*:cluster:*<name>* For example: `arn:aws:rds:us-east-2:123456789012:cluster:my-aurora-cluster-1`
Event subscription	arn:aws:rds:*<region>*:*<account>*:es:*<name>* For example: `arn:aws:rds:us-east-2:123456789012:es:my-subscription`
DB option group	arn:aws:rds:*<region>*:*<account>*:og:*<name>* For example: `arn:aws:rds:us-east-2:123456789012:og:my-og-oracle-tde`
DB parameter group	arn:aws:rds:*<region>*:*<account>*:pg:*<name>* For example: `arn:aws:rds:us-east-2:123456789012:pg:my-param-enable-logs`
DB cluster parameter group	arn:aws:rds:*<region>*:*<account>*:cluster-pg:*<name>* For example: `arn:aws:rds:us-east-2:123456789012:cluster-pg:my-cluster-param-timezone`
Reserved DB instance	arn:aws:rds:*<region>*:*<account>*:ri:*<name>* For example: `arn:aws:rds:us-east-2:123456789012:ri:my-reserved-postgresql`
DB security group	arn:aws:rds:*<region>*:*<account>*:secgrp:*<name>* For example: `arn:aws:rds:us-east-2:123456789012:secgrp:my-public`

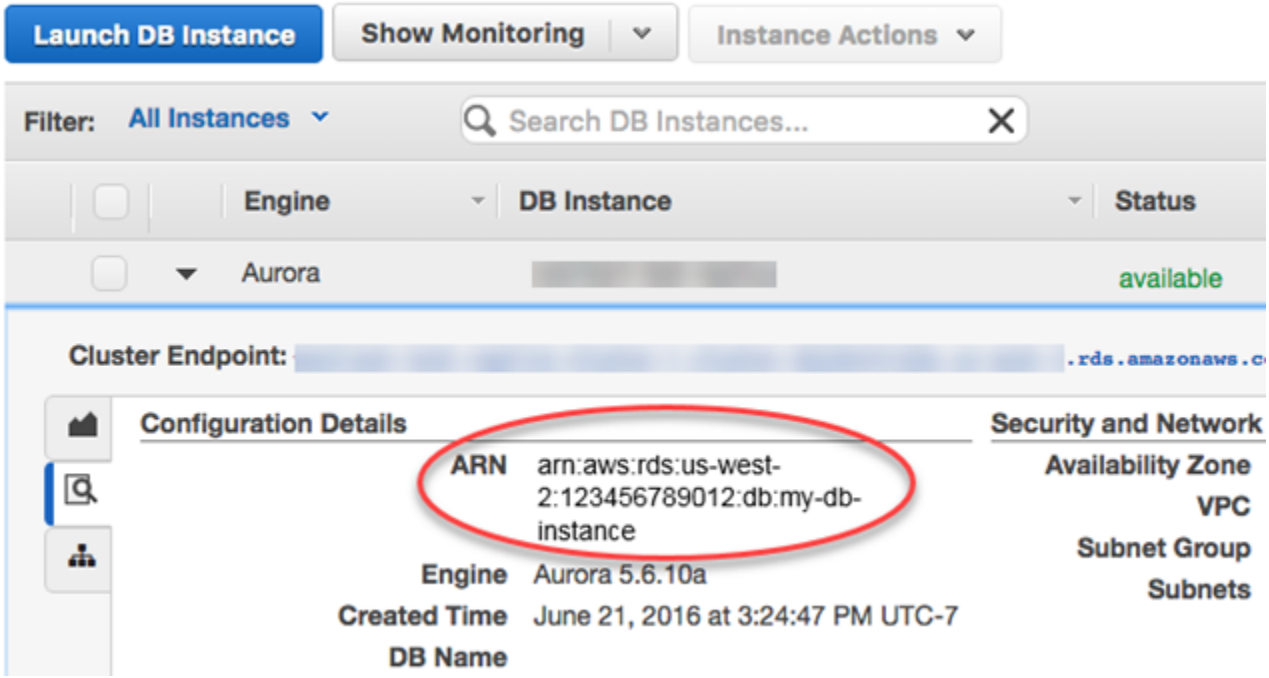
Resource Type	ARN Format
DB snapshot	arn:aws:rds:*<region>*:*<account>*:snapshot:*<name>* For example: `arn:aws:rds:us-east-2:123456789012:snapshot:my-mysql-snap-20130507`
DB cluster snapshot	arn:aws:rds:*<region>*:*<account>*:cluster-snapshot:*<name>* For example: `arn:aws:rds:us-east-2:123456789012:cluster-snapshot:my-aurora-snap-20160809`
DB subnet group	arn:aws:rds:*<region>*:*<account>*:subgrp:*<name>* For example: `arn:aws:rds:us-east-2:123456789012:subgrp:my-subnet-10`

Getting an Existing ARN

You can get the ARN of an RDS resource by using the AWS Management Console, AWS Command Line Interface (AWS CLI), or RDS API.

AWS Management Console

To get an ARN from the AWS Management Console, navigate to the resource you want an ARN for, and choose **See Details** for that resource. For example, you can get the ARN for a DB instance from the **Configuration Details** page as shown following.



AWS CLI

To get an ARN from the AWS CLI for a particular RDS resource, you use the `describe` command for that resource. The following table shows each RDS CLI command, and the ARN property used with the command to get an ARN.

RDS CLI Command	ARN Property
describe-event-subscriptions	EventSubscriptionArn
describe-certificates	CertificateArn
describe-db-parameter-groups	DBParameterGroupArn
describe-db-cluster-parameter-groups	DBClusterParameterGroupArn
describe-db-instances	DBInstanceArn
describe-db-security-groups	DBSecurityGroupArn
describe-db-snapshots	DBSnapshotArn
describe-events	SourceArn
describe-reserved-db-instances	ReservedDBInstanceArn
describe-db-subnet-groups	DBSubnetGroupArn
describe-option-groups	OptionGroupArn
describe-db-clusters	DBClusterArn
describe-db-cluster-snapshots	DBClusterSnapshotArn

For example, the following AWS CLI command gets the ARN for a DB instance.

Example

For Linux, OS X, or Unix:

```
aws rds describe-db-instances \
--db-instance-identifier DBInstanceIdentifier \
--region us-west-2
```

For Windows:

```
aws rds describe-db-instances ^
--db-instance-identifier DBInstanceIdentifier ^
--region us-west-2
```

API

To get an ARN for a particular RDS resource, you can call the following RDS API actions and use the ARN properties shown following.

RDS CLI Command	ARN Property
DescribeEventSubscriptions	EventSubscriptionArn
DescribeCertificates	CertificateArn
DescribeDBParameterGroups	DBParameterGroupArn
DescribeDBClusterParameterGroups	DBClusterParameterGroupArn
DescribeDBInstances	DBInstanceArn
DescribeDBSecurityGroups	DBSecurityGroupArn
DescribeDBSnapshots	DBSnapshotArn
DescribeEvents	SourceArn
DescribeReservedDBInstances	ReservedDBInstanceArn
DescribeDBSubnetGroups	DBSubnetGroupArn
DescribeOptionGroups	OptionGroupArn
DescribeDBClusters	DBClusterArn
DescribeDBClusterSnapshots	DBClusterSnapshotArn

Related Topics

- Tagging Amazon RDS Resources (p. 129)
- Amazon RDS DB Instance Lifecycle (p. 111)

Working with Reserved DB Instances

Reserved DB instances let you reserve a DB instance for a one- or three-year term. Reserved DB instances provide you with a significant discount compared to on-demand DB instance pricing. Reserved DB instances are not physical instances, but rather a billing discount applied to the use of certain on-demand DB instances in your account. Discounts for reserved DB instances are tied to instance type and region.

The general process for working with reserved DB instances is: First get information about available reserved DB instance offerings, then purchase a reserved DB instance offering, and finally get information about your existing reserved DB instances.

Overview of Reserved Instances

When you purchase a reserved instance in Amazon RDS, you purchase a commitment to getting a discounted rate, on a specific DB instance type, for the duration of the reserved instance. To use an Amazon RDS reserved instance, you create a new DB instance just like you do for an on-demand instance. The new DB instance you create must match the specifications of the reserved instance. If the specifications of the new DB instance matches an existing reserved instance for your account, you are billed at the discounted rate offered for the reserved instance; otherwise, the DB instance is billed at an on-demand rate.

> **Note**
> You can move a reserved DB instance from an EC2-Classic (non-VPC) instance into an Amazon Virtual Private Cloud (Amazon VPC) without additional charge.

For more information about reserved DB instances, including pricing, see Amazon RDS Reserved Instances.

Offering Types

Reserved DB instances are available in three varieties—No Upfront, Partial Upfront, and All Upfront—that let you optimize your Amazon RDS costs based on your expected usage.

No Upfront

This option provides access to a reserved DB instance without requiring an upfront payment. Your No Upfront reserved DB instance bills a discounted hourly rate for every hour within the term, regardless of usage, and no upfront payment is required. This option is only available as a one-year reservation.

Partial Upfront

This option requires a part of the reserved DB instance to be paid upfront. The remaining hours in the term are billed at a discounted hourly rate, regardless of usage. This option is the replacement for the previous Heavy Utilization option.

All Upfront

Full payment is made at the start of the term, with no other costs incurred for the remainder of the term regardless of the number of hours used.

Size-Flexible Reserved Instances

When you purchase a reserved instance, one of the things that you specify is the instance class, for example db.m4.large. For more information about instance classes, see DB Instance Class (p. 92).

If you have a DB instance, and you need to scale it to larger capacity, your reserved instance is automatically applied to your scaled DB instance. That is, your reserved instances are automatically applied across all DB instance class sizes. Size-flexible reserved instances are available for DB instances with the same AWS Region, database engine, and instance family. Reserved instance benefits also apply for both Multi-AZ and Single-AZ configurations.

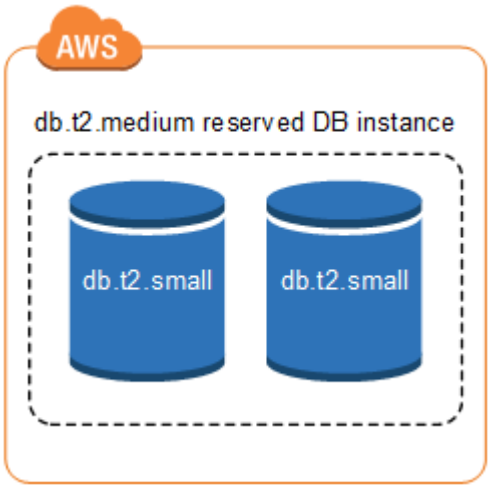
Size-flexible reserved instances are available for the following database engines:

- Amazon Aurora
- MariaDB
- MySQL
- Oracle, Bring Your Own License
- PostgreSQL

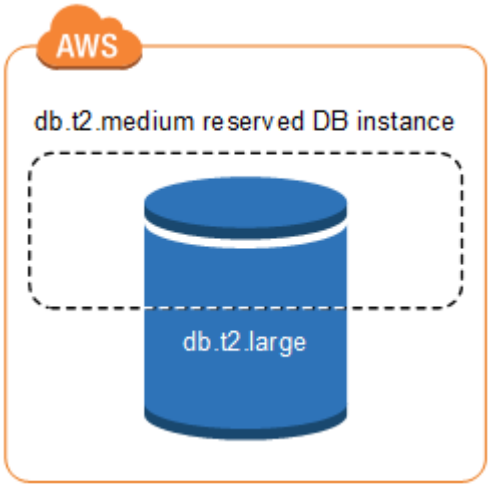
You can compare usage for different reserved instance sizes by using normalized units. For example, one unit of usage on two db.m3.large DB instances is equivalent to 8 normalized units of usage on one db.m3.small. The following table shows the number of normalized units for each DB instance size.

Instance Size	Single-AZ Normalized Units	Multi-AZ Normalized Units
micro	0.5	1
small	1	2
medium	2	4
large	4	8
xlarge	8	16
2xlarge	16	32
4xlarge	32	64
8xlarge	64	128
10xlarge	80	160
16xlarge	132	264

For example, if you purchase a `db.t2.medium` reserved DB instance, and you have two running `db.t2.small` db instances in your account in the same region, the billing benefit is applied in full to both instances.



Alternatively, if you have one `db.t2.large` instance running in your account in the same region, the billing benefit is applied to 50% of the usage of the DB instance.



Deleting a Reserved Instance

The terms for a reserved instance involve a one-year or three-year commitment. You can't cancel a reserved instance. However, you can delete a DB instance that is covered by a reserved instance discount. The process for deleting a DB instance that is covered by a reserved instance discount is the same as for any other DB instance.

Your upfront payment for a reserved DB instance reserves the resources for your use. Because these resources are reserved for you, you are billed for the resources regardless of whether you use them.

If you delete a DB instance that is covered by a reserved instance discount, you can launch another DB instance with compatible specifications and continue to get the discounted rate during the reservation term (one or three years).

AWS Management Console

You can use the AWS Management Console to work with reserved instances as shown in the following procedures.

To get pricing and information about available reserved DB instance offerings

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Reserved Instances**.
3. Choose **Purchase Reserved DB Instance**.
4. For **Product Description**, choose the DB engine and licensing type.
5. For **DB Instance Class**, choose the DB instance class.
6. For **Multi-AZ Deployment**, choose whether or not you want a Multi-AZ deployment.

   Note
   Reserved Amazon Aurora instances always have the **Multi-AZ Deployment** option set to `No`. When you create an Amazon Aurora DB cluster from your reserved instance, the cluster is automatically created as Multi-AZ.
7. For **Term**, choose the length of time you want the DB instance reserved.
8. For **Offering Type**, choose the offering type.

   After you select the offering type, you can see the pricing information.

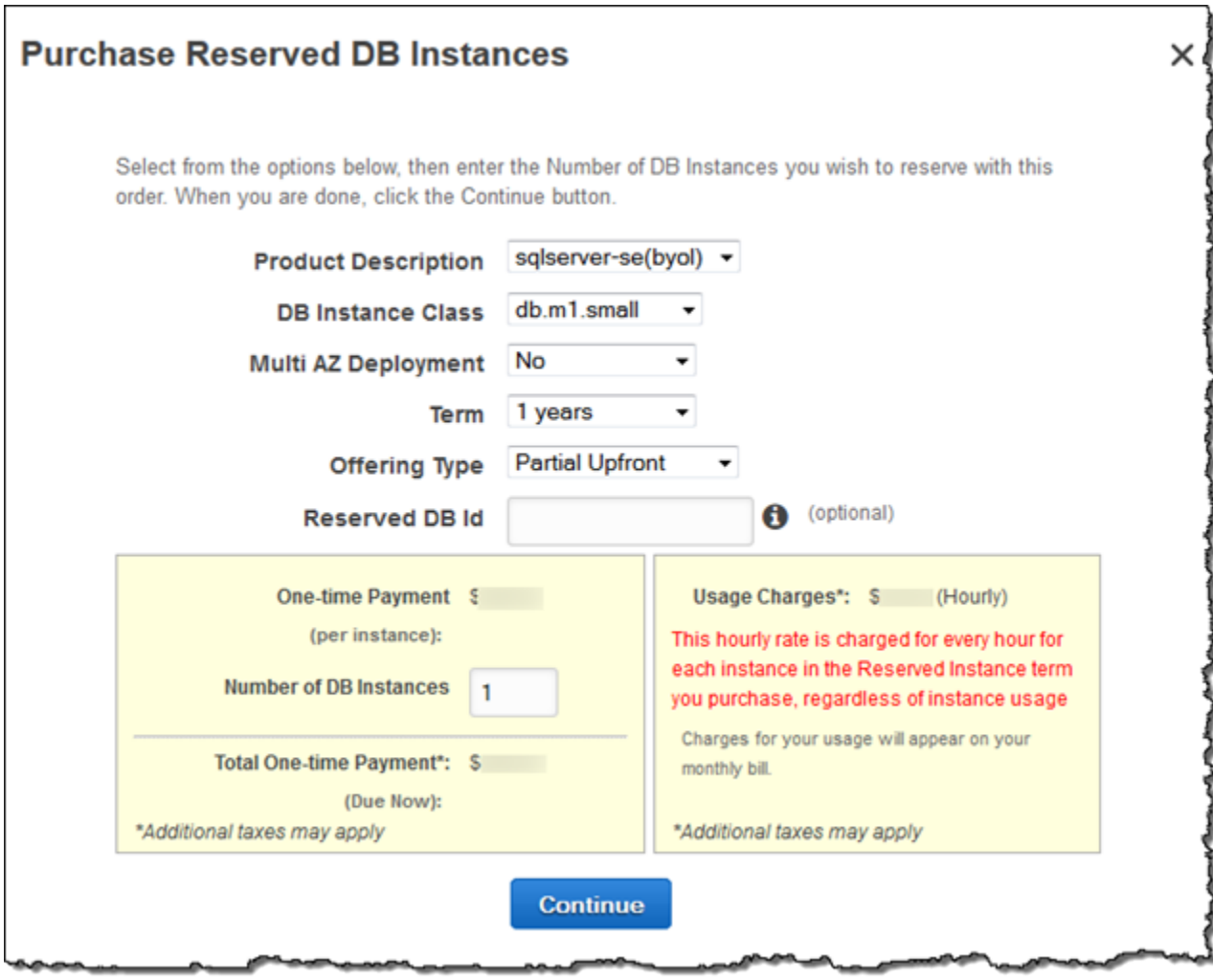
   Important
   Choose **X** in the upper-right corner of the page to avoid purchasing the reserved instance and incurring any charges.

After you have information about the available reserved DB instance offerings, you can use the information to purchase an offering as shown in the following procedure.

To purchase a reserved DB instance

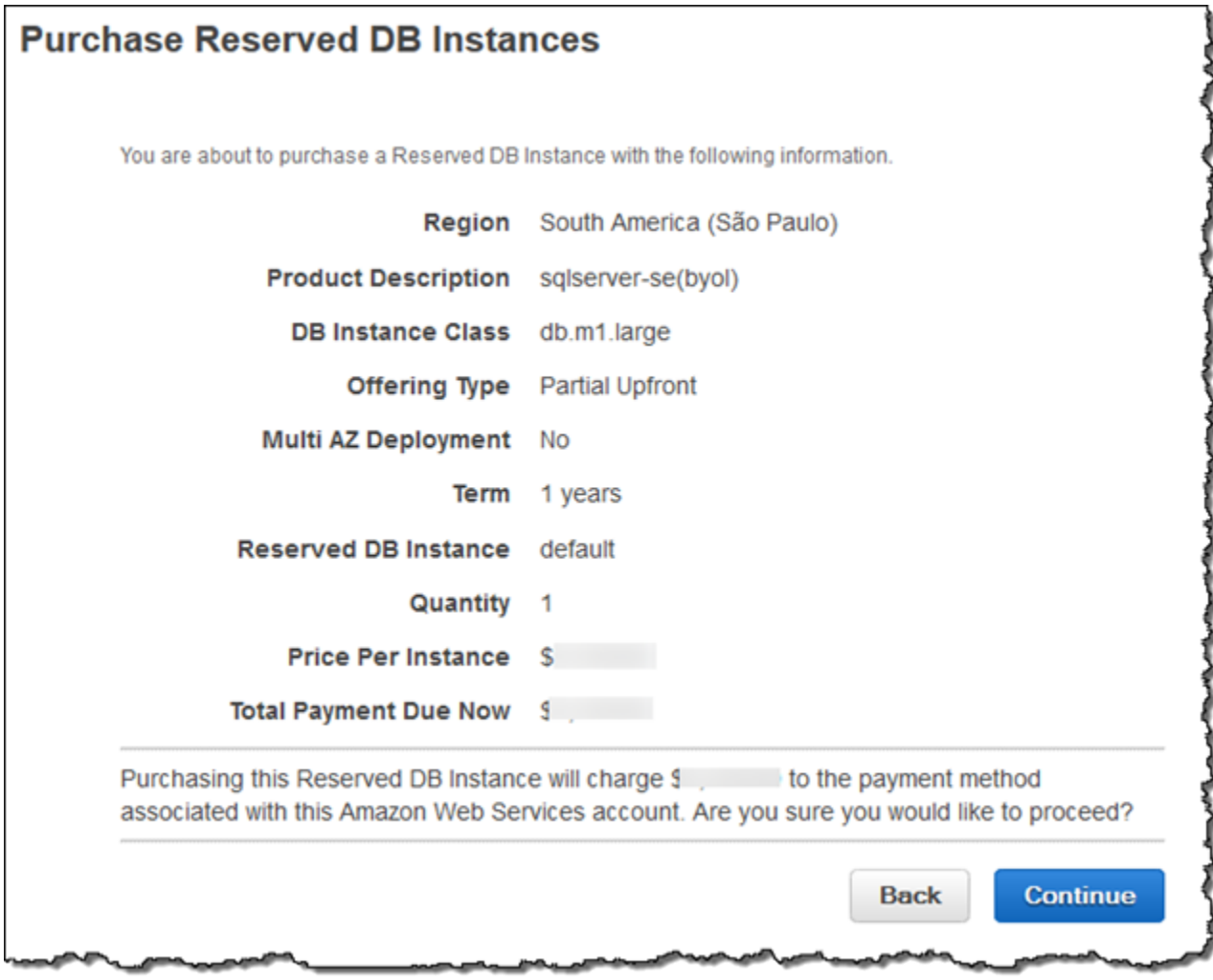
1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the **Navigation** pane, choose **Reserved Instances**.
3. Choose **Purchase Reserved DB Instance**.
4. For **Product Description**, choose the DB engine type.
5. For **DB Instance Class**, choose the DB instance class.
6. For **Multi-AZ Deployment**, choose whether or not you want a Multi-AZ deployment.

   Note
   Reserved Amazon Aurora instances always have the **Multi-AZ Deployment** option set to `No`. When you create an Amazon Aurora DB cluster from your reserved instance, the cluster is automatically created as Multi-AZ.
7. For **Term**, choose the length of time you want the DB instance reserved.
8. For **Offering Type**, choose the offering type.
9. (Optional.) You can assign your own identifier to the reserved instances that you purchase to help you keep track of them. For **Reserved Id**, type an identifier for your reserved DB instance.
10. After you select the offering type, you can see the pricing information, as shown following.



11. Choose **Continue**.

    The **Purchase Reserved DB Instance** dialog box appears, with a summary of the reserved DB instance attributes that you've selected and the payment due, as shown following.



12. On the confirmation page, review your reserved DB instance. If the information is correct, choose **Purchase** to purchase the reserved DB instance.

    Alternatively, choose **Back** to edit your reserved DB instance.

After you have purchased reserved DB instances, you can get information about your reserved DB instances as shown in the following procedure.

To get information about reserved DB instances for your AWS account

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the **Navigation** pane, choose **Reserved Instances**.

   The reserved DB instances for your account appear. You can choose any of the reserved DB instances in the list to see detailed information about that reserved DB instance in the detail pane at the bottom of the console.

CLI

You can use the AWS CLI to work with reserved instances as shown in the following examples.

Example Get Available Reserved Instance Offerings

To get information about available reserved DB instance offerings, call the AWS CLI command `describe-reserved-db-instances-offerings`.

```
aws rds describe-reserved-db-instances-offerings
```

This call returns output similar to the following:

```
OFFERING  OfferingId                            Class         Multi-AZ  Duration  Fixed
 Price  Usage Price  Description  Offering Type
OFFERING  438012d3-4052-4cc7-b2e3-8d3372e0e706  db.m1.large   y         1y        1820.00
 USD  0.368 USD    mysql        Partial  Upfront
OFFERING  649fd0c8-cf6d-47a0-bfa6-060f8e75e95f  db.m1.small   n         1y         227.50
 USD  0.046 USD    mysql        Partial  Upfront
OFFERING  123456cd-ab1c-47a0-bfa6-12345667232f  db.m1.small   n         1y         162.00
 USD   0.00 USD    mysql        All      Upfront
    Recurring Charges:   Amount  Currency  Frequency
    Recurring Charges:   0.123   USD       Hourly
OFFERING  123456cd-ab1c-37a0-bfa6-12345667232d  db.m1.large   y         1y         700.00
 USD   0.00 USD    mysql        All      Upfront
    Recurring Charges:   Amount  Currency  Frequency
    Recurring Charges:   1.25    USD       Hourly
OFFERING  123456cd-ab1c-17d0-bfa6-12345667234e  db.m1.xlarge  n         1y        4242.00
 USD   2.42 USD    mysql        No       Upfront
```

After you have information about the available reserved DB instance offerings, you can use the information to purchase an offering as shown in the following example.

Example Purchase a Reserved Instance

To purchase a reserved DB instance, use the AWS CLI command `purchase-reserved-db-instances-offering` with the following parameters:

- `--reserved-db-instances-offering-id` – the id of the offering that you want to purchase. See the preceding example to get the offering ID.
- `--reserved-db-instance-id` – you can assign your own identifier to the reserved instances that you purchase to help you keep track of them.

The following example purchases the reserved DB instance offering with ID *649fd0c8-cf6d-47a0-bfa6-060f8e75e95f*, and assigns the identifier of *MyReservation*.

For Linux, OS X, or Unix:

```
aws rds purchase-reserved-db-instances-offering \
    --reserved-db-instances-offering-id 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f \
    --reserved-db-instance-id MyReservation
```

For Windows:

```
aws rds purchase-reserved-db-instances-offering ^
    --reserved-db-instances-offering-id 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f ^
```

```
    --reserved-db-instance-id MyReservation
```

The command returns output similar to the following:

```
RESERVATION  ReservationId      Class        Multi-AZ  Start Time                Duration
 Fixed Price  Usage Price  Count  State            Description  Offering Type
RESERVATION  MyReservation      db.m1.small  y         2011-12-19T00:30:23.247Z  1y
 455.00 USD   0.092 USD    1      payment-pending  mysql        Partial  Upfront
```

After you have purchased reserved DB instances, you can get information about your reserved DB instances as shown in the following example.

Example Get Your Reserved Instances

To get information about reserved DB instances for your AWS account, call the AWS CLI command `describe-reserved-db-instances`.

```
aws rds describe-reserved-db-instances
```

The command returns output similar to the following:

```
RESERVATION  ReservationId     Class        Multi-AZ  Start Time                Duration
 Fixed Price  Usage Price  Count  State    Description  Offering Type
RESERVATION  MyReservation     db.m1.small  y         2011-12-09T23:37:44.720Z  1y
 455.00 USD   0.092 USD    1      retired  mysql        Partial  Upfront
```

API

You can use the RDS API to work with reserved instances as shown in the following examples.

Example Get Available Reserved Instance Offerings

To get information about available reserved DB instance offerings, call the Amazon RDS API function `DescribeReservedDBInstancesOfferings`.

```
https://rds.us-east-1.amazonaws.com/
   ?Action=DescribeReservedDBInstancesOfferings
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &Version=2014-09-01
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20140411/us-east-1/rds/aws4_request
   &X-Amz-Date=20140411T203327Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
   &X-Amz-Signature=545f04acffeb4b80d2e778526b1c9da79d0b3097151c24f28e83e851d65422e2
```

This call returns output similar to the following:

```
<DescribeReservedDBInstancesOfferingsResponse xmlns="http://rds.amazonaws.com/
doc/2014-10-31/">
  <DescribeReservedDBInstancesOfferingsResult>
    <ReservedDBInstancesOfferings>
      <ReservedDBInstancesOffering>
        <Duration>31536000</Duration>
        <OfferingType>Partial Upfront</OfferingType>
        <CurrencyCode>USD</CurrencyCode>
        <RecurringCharges/>
        <FixedPrice>1820.0</FixedPrice>
```

```
        <ProductDescription>mysql</ProductDescription>
        <UsagePrice>0.368</UsagePrice>
        <MultiAZ>true</MultiAZ>
        <ReservedDBInstancesOfferingId>438012d3-4052-4cc7-b2e3-8d3372e0e706</
ReservedDBInstancesOfferingId>
        <DBInstanceClass>db.m1.large</DBInstanceClass>
      </ReservedDBInstancesOffering>
      <ReservedDBInstancesOffering>
        <Duration>31536000</Duration>
        <OfferingType>Partial Upfront</OfferingType>
        <CurrencyCode>USD</CurrencyCode>
        <RecurringCharges/>
        <FixedPrice>227.5</FixedPrice>
        <ProductDescription>mysql</ProductDescription>
        <UsagePrice>0.046</UsagePrice>
        <MultiAZ>false</MultiAZ>
        <ReservedDBInstancesOfferingId>649fd0c8-cf6d-47a0-bfa6-060f8e75e95f</
ReservedDBInstancesOfferingId>
        <DBInstanceClass>db.m1.small</DBInstanceClass>
      </ReservedDBInstancesOffering>
    </ReservedDBInstancesOfferings>
  </DescribeReservedDBInstancesOfferingsResult>
  <ResponseMetadata>
    <RequestId>5e4ec40b-2978-11e1-9e6d-771388d6ed6b</RequestId>
  </ResponseMetadata>
</DescribeReservedDBInstancesOfferingsResponse>
```

After you have information about the available reserved DB instance offerings, you can use the information to purchase an offering as shown in the following example.

Example Purchase a Reserved Instance

To purchase a reserved DB instance, call the Amazon RDS API action `PurchaseReservedDBInstancesOffering` with the following parameters:

- `--reserved-db-instances-offering-id` – the id of the offering that you want to purchase. See the preceding example to get the offering ID.
- `--reserved-db-instance-id` – you can assign your own identifier to the reserved instances that you purchase to help you keep track of them.

The following example purchases the reserved DB instance offering with ID *649fd0c8-cf6d-47a0-bfa6-060f8e75e95f*, and assigns the identifier of *MyReservation*.

```
https://rds.us-east-1.amazonaws.com/
   ?Action=PurchaseReservedDBInstancesOffering
   &ReservedDBInstanceId=MyReservation
   &ReservedDBInstancesOfferingId=438012d3-4052-4cc7-b2e3-8d3372e0e706
   &DBInstanceCount=10
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &Version=2014-09-01
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20140415/us-east-1/rds/aws4_request
   &X-Amz-Date=20140415T232655Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
   &X-Amz-Signature=c2ac761e8c8f54a8c0727f5a87ad0a766fbb0024510b9aa34ea6d1f7df52fb11
```

This call returns output similar to the following:

```
<PurchaseReservedDBInstancesOfferingResponse xmlns="http://rds.amazonaws.com/
doc/2014-10-31/">
```

```
  <PurchaseReservedDBInstancesOfferingResult>
    <ReservedDBInstance>
      <OfferingType>Partial Upfront</OfferingType>
      <CurrencyCode>USD</CurrencyCode>
      <RecurringCharges/>
      <ProductDescription>mysql</ProductDescription>
      <ReservedDBInstancesOfferingId>649fd0c8-cf6d-47a0-bfa6-060f8e75e95f</
ReservedDBInstancesOfferingId>
      <MultiAZ>true</MultiAZ>
      <State>payment-pending</State>
      <ReservedDBInstanceId>MyReservation</ReservedDBInstanceId>
      <DBInstanceCount>10</DBInstanceCount>
      <StartTime>2011-12-18T23:24:56.577Z</StartTime>
      <Duration>31536000</Duration>
      <FixedPrice>123.0</FixedPrice>
      <UsagePrice>0.123</UsagePrice>
      <DBInstanceClass>db.m1.small</DBInstanceClass>
    </ReservedDBInstance>
  </PurchaseReservedDBInstancesOfferingResult>
  <ResponseMetadata>
    <RequestId>7f099901-29cf-11e1-bd06-6fe008f046c3</RequestId>
  </ResponseMetadata>
</PurchaseReservedDBInstancesOfferingResponse>
```

After you have purchased reserved DB instances, you can get information about your reserved DB instances as shown in the following example.

Example Get Your Reserved Instances

To get information about reserved DB instances for your AWS account, call the Amazon RDS API action `DescribeReservedDBInstances`.

```
https://rds.us-west-2.amazonaws.com/
   ?Action=DescribeReservedDBInstances
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &Version=2014-09-01
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20140420/us-west-2/rds/aws4_request
   &X-Amz-Date=20140420T162211Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
   &X-Amz-Signature=3312d17a4c43bcd209bc22a0778dd23e73f8434254abbd7ac53b89ade3dae88e
```

The API returns output similar to the following:

```
<DescribeReservedDBInstancesResponse xmlns="http://rds.amazonaws.com/doc/2014-10-31/">
  <DescribeReservedDBInstancesResult>
    <ReservedDBInstances>
      <ReservedDBInstance>
        <OfferingType>Partial Upfront</OfferingType>
        <CurrencyCode>USD</CurrencyCode>
        <RecurringCharges/>
        <ProductDescription>mysql</ProductDescription>
        <ReservedDBInstancesOfferingId>649fd0c8-cf6d-47a0-bfa6-060f8e75e95f</
ReservedDBInstancesOfferingId>
        <MultiAZ>false</MultiAZ>
        <State>payment-failed</State>
        <ReservedDBInstanceId>MyReservation</ReservedDBInstanceId>
        <DBInstanceCount>1</DBInstanceCount>
        <StartTime>2010-12-15T00:25:14.131Z</StartTime>
        <Duration>31536000</Duration>
        <FixedPrice>227.5</FixedPrice>
        <UsagePrice>0.046</UsagePrice>
```

```
        <DBInstanceClass>db.m1.small</DBInstanceClass>
      </ReservedDBInstance>
      <ReservedDBInstance>
        <OfferingType>Partial Upfront</OfferingType>
        <CurrencyCode>USD</CurrencyCode>
        <RecurringCharges/>
        <ProductDescription>mysql</ProductDescription>
        <ReservedDBInstancesOfferingId>649fd0c8-cf6d-47a0-bfa6-060f8e75e95f</
ReservedDBInstancesOfferingId>
        <MultiAZ>false</MultiAZ>
        <State>payment-failed</State>
        <ReservedDBInstanceId>MyReservation</ReservedDBInstanceId>
        <DBInstanceCount>1</DBInstanceCount>
        <StartTime>2010-12-15T01:07:22.275Z</StartTime>
        <Duration>31536000</Duration>
        <FixedPrice>227.5</FixedPrice>
        <UsagePrice>0.046</UsagePrice>
        <DBInstanceClass>db.m1.small</DBInstanceClass>
      </ReservedDBInstance>
    </ReservedDBInstances>
  </DescribeReservedDBInstancesResult>
  <ResponseMetadata>
    <RequestId>23400d50-2978-11e1-9e6d-771388d6ed6b</RequestId>
  </ResponseMetadata>
</DescribeReservedDBInstancesResponse>
```

Related Topics

- How You Are Charged for Amazon RDS (p. 3)

Backing Up and Restoring Amazon RDS DB Instances

This section shows how to back up and restore a DB instance.

Topics

Working With Backups

Amazon RDS creates and saves automated backups of your DB instance. Amazon RDS creates a storage volume snapshot of your DB instance, backing up the entire DB instance and not just individual databases.

Amazon RDS creates automated backups of your DB instance during the backup window of your DB instance. Amazon RDS saves the automated backups of your DB instance according to the backup retention period that you specify. If necessary, you can recover your database to any point in time during the backup retention period.

Your DB instance must be in the `ACTIVE` state for automated backups to occur. If your database is in another state, for example `STORAGE_FULL`, automated backups do not occur.

You can also backup your DB instance manually, by manually creating a DB snapshot. For more information about creating a DB snapshot, see Creating a DB Snapshot (p. 207).

You can copy both automatic and manual DB snapshots, and share manual DB snapshots. For more information about copying a DB snapshot, see Copying a DB Snapshot or DB Cluster Snapshot (p. 213). For more information about sharing a DB snapshot, see Sharing a DB Snapshot or DB Cluster Snapshot (p. 230).

Backup Storage

Your Amazon RDS backup storage for each region is composed of the automated backups and manual DB snapshots for that region. Your backup storage is equivalent to the sum of the database storage for all instances in that region. Moving a DB snapshot to another region increases the backup storage in the destination region.

For more information about backup storage costs, see Amazon RDS Pricing.

All automated backups are deleted when you delete a DB instance. After you delete a DB instance, the automated backups can't be recovered. If you choose to have Amazon RDS create a final DB snapshot before it deletes your DB instance, you can use that to recover your DB instance.

Manual snapshots are not deleted.

The Backup Window

Automated backups occur daily during the preferred backup window. If the backup requires more time than allotted to the backup window, the backup continues after the window ends, until it finishes. The backup window can't overlap with the weekly maintenance window for the DB instance.

During the automatic backup window, storage I/O might be suspended briefly while the backup process initializes (typically under a few seconds). You may experience elevated latencies for a few minutes during backups for Multi-AZ deployments. For MariaDB, MySQL, Oracle, and PostgreSQL, I/O activity is not suspended on your primary during backup for Multi-AZ deployments, because the backup is taken from the standby. For SQL Server, I/O activity is suspended briefly during backup for Multi-AZ deployments.

If you don't specify a preferred backup window when you create the DB instance, Amazon RDS assigns a default 30-minute backup window which is selected at random from an 8-hour block of time per region. The following table lists the time blocks for each region from which the default backups windows are assigned.

Region	Time Block
US West (Oregon) Region	06:00–14:00 UTC

Region	Time Block
US West (N. California) Region	06:00–14:00 UTC
US East (Ohio) Region	03:00–11:00 UTC
US East (N. Virginia) Region	03:00–11:00 UTC
Asia Pacific (Mumbai) Region	16:30–00:30 UTC
Asia Pacific (Seoul) Region	13:00–21:00 UTC
Asia Pacific (Singapore) Region	14:00–22:00 UTC
Asia Pacific (Sydney) Region	12:00–20:00 UTC
Asia Pacific (Tokyo) Region	13:00–21:00 UTC
Canada (Central) Region	06:29–14:29 UTC
EU (Frankfurt) Region	20:00–04:00 UTC
EU (Ireland) Region	22:00–06:00 UTC
EU (London) Region	06:00–14:00 UTC
South America (São Paulo) Region	23:00–07:00 UTC
AWS GovCloud (US)	03:00–11:00 UTC

The Backup Retention Period

You can set the backup retention period when you create a DB instance. If you don't set the backup retention period, the default backup retention period is one day if you create the DB instance using the Amazon RDS API or the AWS CLI, or seven days if you create the DB instance using the AWS Console. For Amazon Aurora DB clusters, the default backup retention period is one day regardless of how the DB cluster is created. After you create a DB instance, you can modify the backup retention period. You can set the backup retention period to between 1 and 35 days. For non–Aurora DB engines, you can also set the backup retention period to 0, which disables automated backups. Manual snapshot limits (100 per region) do not apply to automated backups.

Important
An outage occurs if you change the backup retention period from 0 to a non-zero value or from a non-zero value to 0.

Note
You cannot disable automated backups on Aurora. The backup retention period for Aurora is managed by the DB cluster. For more information, see Backing Up and Restoring an Aurora DB Cluster (p. 468).

Disabling Automated Backups

For non–Aurora DB engines, you may want to temporarily disable automated backups in certain situations; for example, while loading large amounts of data.

Important
We highly discourage disabling automated backups because it disables point-in-time recovery. Disabling automatic backups for a DB instance deletes all existing automated backups for the instance. If you disable and then re-enable automated backups, you are only able to restore starting from the time you re-enabled automated backups.

In this example, you disable automated backups for a DB instance named *mydbinstance* by setting the backup retention parameter to 0.

AWS Management Console

To disable automated backups immediately

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **DB Instances**, and then select the DB instance that you want to modify.
3. Choose **Instance Actions**, and then choose **Modify**. The **Modify DB Instance** window appears.
4. For **Backup Retention Period**, choose **0**.
5. Select **Apply Immediately**.
6. Choose **Continue**.
7. On the confirmation page, choose **Modify DB Instance** to save your changes and disable automated backups.

CLI

To disable automated backups immediately, use the modify-db-instance command and set the backup retention period to 0 with `--apply-immediately`.

Example

The following example immediately disabled automatic backups.

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --backup-retention-period 0 \
    --apply-immediately
```

For Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --backup-retention-period 0 ^
    --apply-immediately
```

To know when the modification is in effect, call `describe-db-instances` for the DB instance until the value for backup retention period is 0 and mydbinstance status is available.

```
aws rds describe-db-instances --db-instance-identifier mydbinstance
```

API

To disable automated backups immediately, call the ModifyDBInstance action with the following parameters:

- `DBInstanceIdentifier = mydbinstance`
- `BackupRetentionPeriod = 0`

Example

```
https://rds.amazonaws.com/
    ?Action=ModifyDBInstance
    &DBInstanceIdentifier=mydbinstance
    &BackupRetentionPeriod=0
    &SignatureVersion=2
    &SignatureMethod=HmacSHA256
    &Timestamp=2009-10-14T17%3A48%3A21.746Z
    &AWSAccessKeyId=<AWS Access Key ID>
    &Signature=<Signature>
```

Enabling Automated Backups

If your DB instance doesn't have automated backups enabled, you can enable them at any time. You enable automated backups by setting the backup retention period to a positive non-zero value. When automated backups are enabled, an outage occurs and a backup is immediately created.

In this example, you enable automated backups for a DB instance named *mydbinstance* by setting the backup retention period to a positive non-zero value (in this case, 3).

AWS Management Console

To enable automated backups immediately

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **DB Instances**, and then select the DB instance that you want to modify.
3. Choose **Instance Actions**, and then choose **Modify**. The **Modify DB Instance** page appears.
4. For **Backup Retention Period**, choose a positive non-zero value, for example 3.
5. Select **Apply Immediately**.
6. Choose **Continue**.
7. On the confirmation page, choose **Modify DB Instance** to save your changes and enable automated backups.

CLI

To enable automated backups immediately, use the AWS CLI `modify-db-instance` command.

In this example, we will enable automated backups by setting the backup retention period to 3 days.

Include the following parameters:

- `--db-instance-identifier`
- `--backup-retention-period`

- `--apply-immediately` or `--no-apply-immediately`

Example

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance  \
    --backup-retention-period 3 \
    --apply-immediately
```

For Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance  ^
    --backup-retention-period 3 ^
    --apply-immediately
```

API

To enable automated backups immediately, use the AWS CLI `ModifyDBInstance` command.

In this example, we will enable automated backups by setting the backup retention period to 3 days.

Include the following parameters:

- `DBInstanceIdentifier`
- `BackupRetentionPeriod`
- `ApplyImmediately = true`

Example

```
https://rds.amazonaws.com/
 ?Action=ModifyDBInstance
 &DBInstanceIdentifier=mydbinstance
 &BackupRetentionPeriod=3
 &ApplyImmediately=true
 &SignatureVersion=2
 &SignatureMethod=HmacSHA256
 &Timestamp=2009-10-14T17%3A48%3A21.746Z
 &AWSAccessKeyId=<AWS Access Key ID>
 &Signature=<Signature>
```

Automated Backups with Unsupported MySQL Storage Engines

For the MySQL DB engine, automated backups are only supported for the InnoDB storage engine; use of these features with other MySQL storage engines, including MyISAM, may lead to unreliable behavior while restoring from backups. Specifically, since storage engines like MyISAM do not support reliable crash recovery, your tables can be corrupted in the event of a crash. For this reason, we encourage you to use the InnoDB storage engine.

- To convert existing MyISAM tables to InnoDB tables, you can use alter table command. For example: `ALTER TABLE table_name ENGINE=innodb, ALGORITHM=COPY;`

- If you choose to use MyISAM, you can attempt to manually repair tables that become damaged after a crash by using the REPAIR command (see: http://dev.mysql.com/doc/refman/5.5/en/repair-table.html). However, as noted in the MySQL documentation, there is a good chance that you will not be able to recover all your data.
- If you want to take a snapshot of your MyISAM tables prior to restoring, follow these steps:
  1. Stop all activity to your MyISAM tables (that is, close all sessions).

     You can close all sessions by calling the mysql.rds_kill command for each process that is returned from the `SHOW FULL PROCESSLIST` command.
  2. Lock and flush each of your MyISAM tables. For example, the following commands lock and flush two tables named `myisam_table1` and `myisam_table2`:

```
mysql> FLUSH TABLES myisam_table, myisam_table2 WITH READ LOCK;
```

  3. Create a snapshot of your DB instance. When the snapshot has completed, release the locks and resume activity on the MyISAM tables. You can release the locks on your tables using the following command:

```
mysql> UNLOCK TABLES;
```

  These steps force MyISAM to flush data stored in memory to disk thereby ensuring a clean start when you restore from a DB snapshot. For more information on creating a DB snapshot, see Creating a DB Snapshot (p. 207).

Automated Backups with Unsupported MariaDB Storage Engines

For the MariaDB DB engine, automated backups are only supported for the XtraDB storage engine; use of these features with other MariaDB storage engines, including Aria, might lead to unreliable behavior while restoring from backups. Even though Aria is a crash-resistant alternative to MyISAM, your tables can still be corrupted in the event of a crash. For this reason, we encourage you to use the XtraDB storage engine.

- To convert existing Aria tables to XtraDB tables, you can use ALTER TABLE command. For example: `ALTER TABLE table_name ENGINE=xtradb, ALGORITHM=COPY;`
- If you choose to use Aria, you can attempt to manually repair tables that become damaged after a crash by using the REPAIR TABLE command. For more information, see http://mariadb.com/kb/en/mariadb/repair-table/.
- If you want to take a snapshot of your Aria tables prior to restoring, follow these steps:
  1. Stop all activity to your Aria tables (that is, close all sessions).
  2. Lock and flush each of your Aria tables.
  3. Create a snapshot of your DB instance. When the snapshot has completed, release the locks and resume activity on the Aria tables. These steps force Aria to flush data stored in memory to disk, thereby ensuring a clean start when you restore from a DB snapshot.

Related Topics

- Restoring a DB Instance to a Specified Time (p. 237)

Creating a DB Snapshot

Amazon RDS creates a storage volume snapshot of your DB instance, backing up the entire DB instance and not just individual databases. Creating this DB snapshot on a Single-AZ DB instance results in a brief I/O suspension that can last from a few seconds to a few minutes, depending on the size and class of your DB instance. Multi-AZ DB instances are not affected by this I/O suspension since the backup is taken on the standby.

When you create a DB snapshot, you need to identify which DB instance you are going to back up, and then give your DB snapshot a name so you can restore from it later. If you have IAM database authentication enabled, then this setting is inherited from the source DB instance.

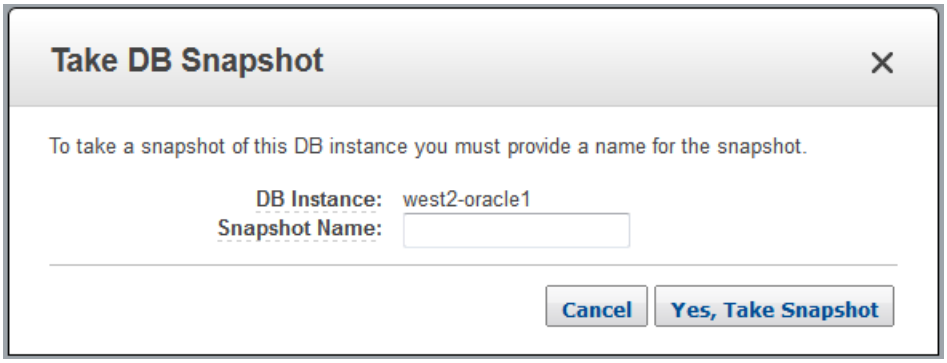
The amount of time it takes to create a snapshot varies with the size your databases. Since the snapshot includes the entire storage volume, the size of files, such as temporary files, also affects the amount of time it takes to create the snapshot.

AWS Management Console

To create a DB snapshot

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, click **DB Instances**.
3. Click **Instance Actions**, and then click **Take DB Snapshot**.

   The **Take DB Snapshot** window appears.
4. Type the name of the snapshot in the **Snapshot Name** text box.



5. Click **Yes, Take Snapshot**.

CLI

When you create a DB snapshot using the AWS CLI, you need to identify which DB instance you are going to back up, and then give your DB snapshot a name so you can restore from it later. You can do this by using the AWS CLI `create-db-snapshot` command with the following parameters:

- `--db-instance-identifier`
- `--db-snapshot-identifier`

In this example, you create a DB snapshot called *`mydbsnapshot`* for a DB instance called *`mydbinstance`*.

Example

For Linux, OS X, or Unix:

```
aws rds create-db-snapshot /
    --db-instance-identifier mydbinstance /
    --db-snapshot-identifier mydbsnapshot
```

For Windows:

```
aws rds create-db-snapshot ^
    --db-instance-identifier mydbinstance ^
    --db-snapshot-identifier mydbsnapshot
```

The output from this command should look similar to the following:

```
DBSNAPSHOT  mydbsnapshot  mydbinstance  2009-10-21T01:54:49.521Z  MySQL     50
creating  sa  5.6.27 general-public-license
```

API

When you create a DB snapshot using the Amazon RDS API, you need to identify which DB instance you are going to back up, and then give your DB snapshot a name so you can restore from it later. You can do this by using the Amazon RDS API `CreateDBSnapshot` command with the following parameters:

- DBInstanceIdentifier
- DBSnapshotIdentifier

In this example, you create a DB snapshot called *mydbsnapshot* for a DB instance called *mydbinstance*.

Example

```
https://rds.us-east-1.amazonaws.com/
    ?Action=CreateDBSnapshot
    &DBInstanceIdentifier=mydbinstance
    &DBSnapshotIdentifier=mydbsnapshot
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2013-09-09
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20140423/us-east-1/rds/aws4_request
    &X-Amz-Date=20140423T161105Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=e9649af6edcfbab4016f04d72e1b7fc16d8734c37477afcf25b3def625484ed2
```

Related Topics

- Restoring from a DB Snapshot (p. 209)
- Copying a DB Snapshot or DB Cluster Snapshot (p. 213)
- Sharing a DB Snapshot or DB Cluster Snapshot (p. 230)

Restoring from a DB Snapshot

Amazon RDS creates a storage volume snapshot of your DB instance, backing up the entire DB instance and not just individual databases. You can create a DB instance by restoring from this DB snapshot. When you restore the DB instance, you provide the name of the DB snapshot to restore from, and then provide a name for the new DB instance that is created from the restore. You cannot restore from a DB snapshot to an existing DB instance; a new DB instance is created when you restore.

You can restore a DB instance and use a different storage type than the source DB snapshot. In this case, the restoration process is slower because of the additional work required to migrate the data to the new storage type. If you restore to or from **Magnetic (Standard)** storage, the migration process is the slowest. That's because **Magnetic** storage doesn't have the IOPS capability of **Provisioned IOPS** or **General Purpose (SSD)** storage.

Parameter Group Considerations

When you restore a DB instance, the default DB parameter group is associated with the restored instance. As soon as the restore is complete and your new DB instance is available, you must associate any custom DB parameter group used by the instance you restored from. You must apply these changes by using the RDS console's *Modify* command, the `ModifyDBInstance` Amazon RDS API, or the AWS CLI `modify-db-instance` command.

Important
We recommend that you retain the parameter group for any DB snapshots you create, so that you can associate your restored DB instance with the correct parameter group.

Security Group Considerations

When you restore a DB instance, the default security group is associated with the restored instance. As soon as the restore is complete and your new DB instance is available, you must associate any custom security groups used by the instance you restored from. You must apply these changes by using the RDS console's *Modify* command, the `ModifyDBInstance` Amazon RDS API, or the AWS CLI `modify-db-instance` command.

Option Group Considerations

When you restore a DB instance, the option group associated with the DB snapshot is associated with the restored DB instance after it is created. For example, if the DB snapshot you are restoring from uses Oracle Transparent Data Encryption, the restored DB instance will use the same option group.

When you assign an option group to a DB instance, the option group is also linked to the supported platform the DB instance is on, either VPC or EC2-Classic (non-VPC). If a DB instance is in a VPC, the option group associated with the DB instance is linked to that VPC. This means that you cannot use the option group assigned to a DB instance if you attempt to restore the instance into a different VPC or onto a different platform. If you restore a DB instance into a different VPC or onto a different platform, you must either assign the default option group to the instance, assign an option group that is linked to that VPC or platform, or create a new option group and assign it to the DB instance. For persistent or permanent options, when restoring a DB instance into a different VPC you must create a new option group that includes the persistent or permanent option.

Microsoft SQL Server Considerations

When you restore a Microsoft SQL Server DB snapshot to a new instance, you can always restore to the same edition as your snapshot. In some cases, you can also change the edition of the DB instance. The following are the limitations when you change editions:

- The DB snapshot must have enough storage allocated for the new edition.
- Only the following edition changes are supported:
  - From Standard Edition to Enterprise Edition
  - From Web Edition to Standard Edition or Enterprise Edition
  - From Express Edition to Web Edition, Standard Edition or Enterprise Edition

If you want to change from one edition to a new edition that is not supported by restoring a snapshot, you can try using the native backup and restore feature. SQL Server verifies whether or not your database is compatible with the new edition based on what SQL Server features you have enabled on the database. For more information, see Importing and Exporting SQL Server Databases (p. 769).

Oracle Considerations

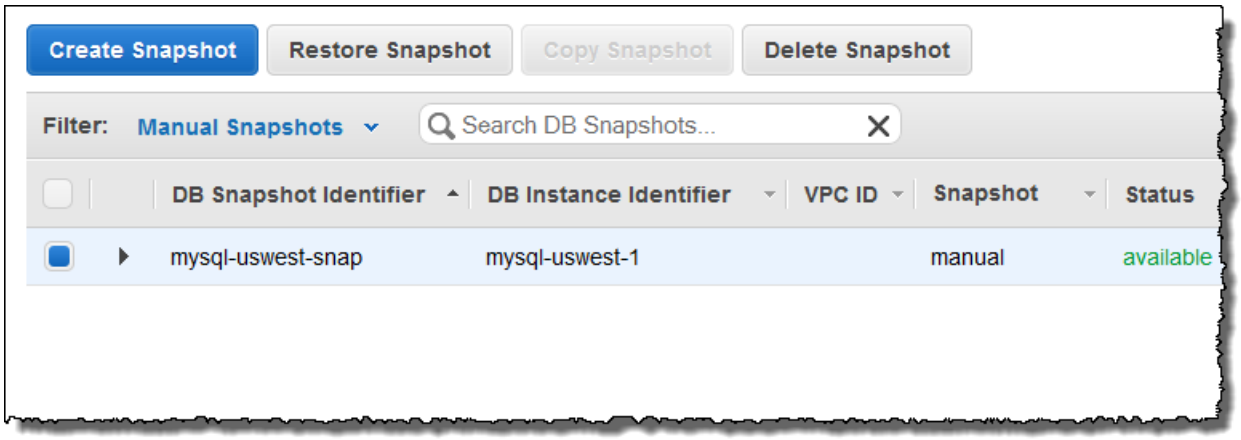
If you use Oracle GoldenGate, always retain the parameter group with the `compatible` parameter. If you restore a DB instance from a DB snapshot, you must modify the restored DB instance to use the parameter group that has a matching or greater `compatible` parameter value. This should be done as soon as possible after the restore action, and you must then reboot your DB instance.

You can upgrade a DB snapshot while it is still a DB snapshot, before you restore it. For more information, see Upgrading an Oracle DB Snapshot (p. 980).

AWS Management Console

To restore a DB instance from a DB snapshot

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Snapshots**.
3. Choose the DB snapshot that you want to restore from.



4. Choose **Restore Snapshot**.

   The **Restore DB Instance** window appears.
5. For **DB Instance Identifier**, type the name for your restored DB instance.
6. Choose **Restore DB Instance**.
7. If you want to restore the functionality of the DB instance to that of the DB instance that the snapshot was created from, you must modify the DB instance to use the security group. The next steps assume that your DB instance is in a VPC. If your DB instance is not in a VPC, use the EC2 Management Console to locate the security group you need for the DB instance.

a. Sign in to the AWS Management Console and open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
b. In the navigation pane, choose **Security Groups**.
c. Select the security group that you want to use for your DB instances. If necessary, add rules to link the security group to a security group for an EC2 instance. For more information, see A DB Instance in a VPC Accessed by an EC2 Instance in the Same VPC (p. 392).

CLI

To restore a DB instance from a DB snapshot, use the AWS CLI command restore-db-instance-from-db-snapshot.

In this example, you restore from a previously created DB snapshot named *mydbsnapshot*. You restore to a new DB instance named *mynewdbinstance*.

Example

For Linux, OS X, or Unix:

```
aws rds restore-db-instance-from-db-snapshot \
    --db-instance-identifier mynewdbinstance \
    --db-snapshot-identifier mydbsnapshot
```

For Windows:

```
aws rds restore-db-instance-from-db-snapshot ^
    --db-instance-identifier mynewdbinstance ^
    --db-snapshot-identifier mydbsnapshot
```

This command returns output similar to the following:

```
DBINSTANCE  mynewdbinstance  db.m3.large  MySQL     50       sa              creating  3  n
  5.6.27  general-public-license
```

After the DB instance has been restored, you must add the DB instance to the security group and parameter group used by the DB instance used to create the DB snapshot if you want the same functionality as that of the previous DB instance.

API

To restore a DB instance from a DB snapshot, call the Amazon RDS API function RestoreDBInstanceFromDBSnapshot with the following parameters:

- `DBSnapshotIdentifier`
- `DBInstanceIdentifier`

In this example, you restore from a previously created DB snapshot named *mydbsnapshot*. You restore to a new DB instance named *mynewdbinstance*.

Example

```
https://rds.us-east-1.amazonaws.com/
```

```
?Action=RestoreDBInstanceFromDBSnapshot
&DBInstanceIdentifier=mynewdbinstance
&DBSnapshotIdentifier=rds%3Amysqldb-2014-04-22-08-15
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&Version=2013-09-09
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20140428/us-east-1/rds/aws4_request
&X-Amz-Date=20140428T232655Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=78ac761e8c8f54a8c0727f4e67ad0a766fbb0024510b9aa34ea6d1f7df52fe92
```

Related Topics

- Tutorial: Restore a DB Instance from a DB Snapshot (p. 239)
- Creating a DB Snapshot (p. 207)
- Copying a DB Snapshot or DB Cluster Snapshot (p. 213)
- Sharing a DB Snapshot or DB Cluster Snapshot (p. 230)

Copying a DB Snapshot or DB Cluster Snapshot

With Amazon Relational Database Service (Amazon RDS), you can copy DB snapshots and DB cluster snapshots. You can copy automated or manual snapshots. After you copy a snapshot, the copy is a manual snapshot.

You can copy a snapshot within the same AWS Region, you can copy a snapshot across AWS Regions, and you can copy a snapshot across AWS accounts.

You can't copy a DB cluster snapshot across regions and accounts in a single step. Perform one step for each of these copy actions. As an alternative to copying, you can also share manual snapshots with other AWS accounts. For more information, see Sharing a DB Snapshot or DB Cluster Snapshot (p. 230).

Limitations

The following are some limitations when you copy snapshots:

- You can't copy a snapshot to or from the following regions: AWS GovCloud (US), China (Beijing).
- If you delete a source snapshot before the target snapshot becomes available, the snapshot copy may fail. Verify that the target snapshot has a status of `AVAILABLE` before you delete a source snapshot.
- You can have up to five snapshot copy requests in progress to a single destination per account.
- You can't copy a DB snapshot across regions if it was created from an Oracle DB instance that is using AWS CloudHSM Classic to store TDE Keys.
- Depending on the regions involved and the amount of data to be copied, a cross-region snapshot copy can take hours to complete. If there is a large number of cross-region snapshot copy requests from a given source AWS Region, Amazon RDS might put new cross-region copy requests from that source AWS Region into a queue until some in-progress copies complete. No progress information is displayed about copy requests while they are in the queue. Progress information is displayed when the copy starts.

Snapshot Retention

Amazon RDS deletes automated snapshots at the end of their retention period, when you disable automated snapshots for a DB instance or DB cluster, or when you delete a DB instance or DB cluster. If you want to keep an automated snapshot for a longer period, copy it to create a manual snapshot, which is retained until you delete it. Amazon RDS storage costs might apply to manual snapshots if they exceed your default storage space.

For more information about backup storage costs, see Amazon RDS Pricing.

Copying Shared Snapshots

You can copy snapshots shared to you by other AWS accounts. If you are copying an encrypted snapshot that has been shared from another AWS account, you must have access to the KMS encryption key that was used to encrypt the snapshot. You can copy shared unencrypted DB snapshots across regions, but you can only copy shared encrypted DB snapshots in the same AWS Region. You can only copy shared DB cluster snapshots, encrypted or not, in the same AWS Region. For more information, see Sharing an Encrypted Snapshot (p. 231).

Handling Encryption

You can copy a snapshot that has been encrypted using an AWS KMS encryption key. If you copy an encrypted snapshot, the copy of the snapshot must also be encrypted. If you copy an encrypted

snapshot within the same AWS Region, you can encrypt the copy with the same KMS encryption key as the original snapshot, or you can specify a different KMS encryption key. If you copy an encrypted snapshot across regions, you can't use the same KMS encryption key for the copy as used for the source snapshot, because KMS keys are region-specific. Instead, you must specify a KMS key valid in the destination AWS Region.

You can also encrypt a copy of an unencrypted snapshot. This way, you can quickly add encryption to a previously unencrypted DB instance or DB cluster. That is, you can create a snapshot of your DB instance or DB cluster when you are ready to encrypt it, and then create a copy of that snapshot and specify a KMS encryption key to encrypt that snapshot copy. You can then restore an encrypted DB instance or DB cluster from the encrypted snapshot. For Amazon Aurora DB cluster snapshots, you also have the option to leave the DB cluster snapshot unencrypted and instead specify a KMS encryption key when restoring. The restored DB cluster is encrypted using the specified key.

Option Group Considerations

Option groups are specific to the AWS Region that they are created in, and you can't use an option group from one AWS Region in another AWS Region.

When you copy a snapshot across regions, you can specify a new option group for the snapshot. We recommend that you prepare the new option group before you copy the snapshot. In the destination AWS Region, create an option group with the same settings as the original DB instance or DB cluster. If one already exists in the new AWS Region, you can use that one.

If you copy a snapshot and you don't specify a new option group for the snapshot, when you restore it the DB instance or DB cluster gets the default option group. To give the new DB instance or DB cluster the same options as the original, you must do the following:

1. In the destination AWS Region, create an option group with the same settings as the original DB instance or DB cluster. If one already exists in the new AWS Region, you can use that one.
2. After you restore the snapshot in the destination AWS Region, modify the new DB instance or DB cluster and add the new or existing option group from the previous step.

Parameter Group Considerations

When you copy a snapshot across regions, the copy doesn't include the parameter group used by the original DB instance or DB cluster. When you restore a snapshot to create a new DB instance or DB cluster, that DB instance or DB cluster gets the default parameter group for the AWS Region it is created in. To give the new DB instance or DB cluster the same parameters as the original, you must do the following:

1. In the destination AWS Region, create a DB parameter group or DB cluster parameter group with the same settings as the original DB instance or DB cluster. If one already exists in the new AWS Region, you can use that one.
2. After you restore the snapshot in the destination AWS Region, modify the new DB instance or DB cluster and add the new or existing parameter group from the previous step.

Copying a DB Snapshot

If your source database engine is MariaDB, Microsoft SQL Server, MySQL, Oracle, or PostgreSQL, then your snapshot is a DB snapshot. For instructions on how to copy a DB snapshot, see Copying a DB Snapshot (p. 215).

Copying a DB Cluster Snapshot

If your source database engine is Aurora, then your snapshot is a DB cluster snapshot. For instructions on how to copy a db cluster snapshot, see Copying a DB Cluster Snapshot (p. 221).

Copying a DB Snapshot

Use the procedures in this topic to copy a DB snapshot. For an overview of copying a snapshot, see Copying a DB Snapshot or DB Cluster Snapshot (p. 213)

If your source database engine is MariaDB, Microsoft SQL Server, MySQL, Oracle, or PostgreSQL, then your snapshot is a DB snapshot. If your source database engine is Aurora, then your snapshot is a DB cluster snapshot. For instructions on how to copy a db cluster snapshot, see Copying a DB Cluster Snapshot (p. 221).

For each AWS account, you can copy up to five DB snapshots at a time from one AWS Region to another. If you copy a DB snapshot to another AWS Region, you create a manual DB snapshot that is retained in that AWS Region. Copying a DB snapshot out of the source AWS Region incurs Amazon RDS data transfer charges.

For more information about data transfer pricing, see Amazon RDS Pricing.

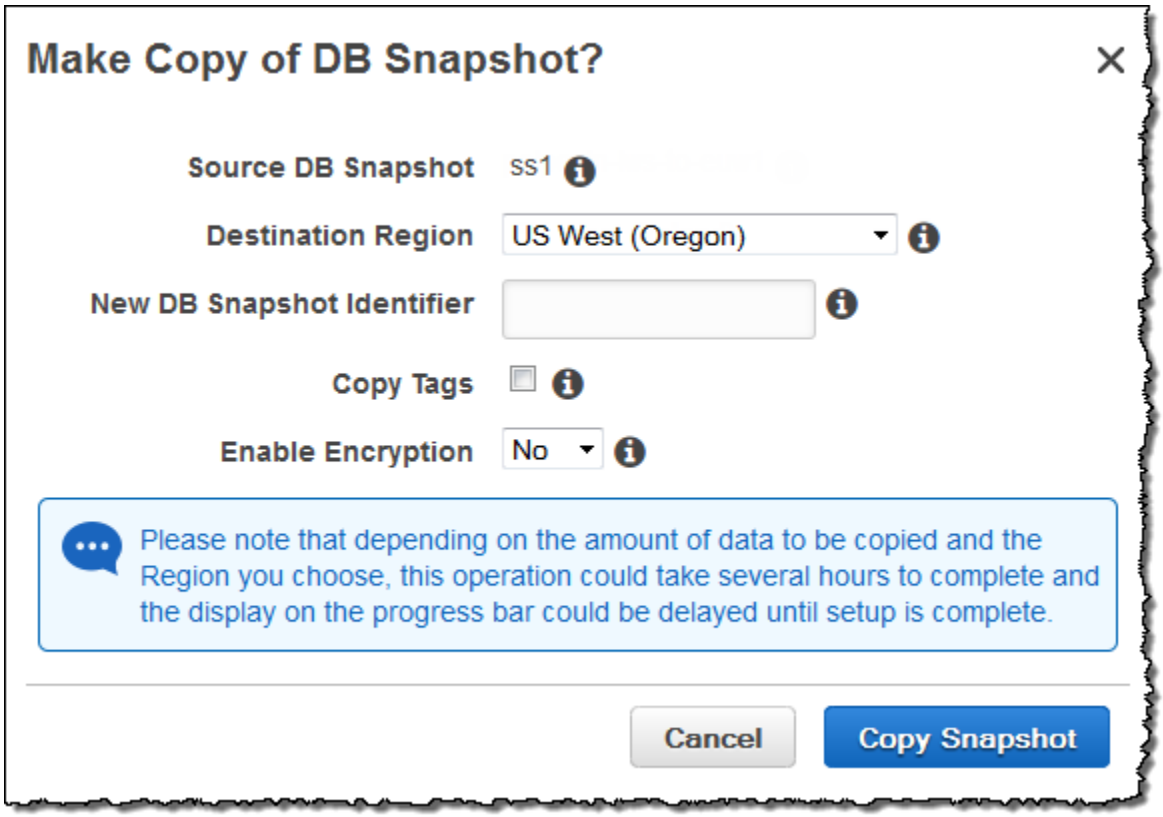
After the DB snapshot copy has been created in the new AWS Region, the DB snapshot copy behaves the same as all other DB snapshots in that AWS Region.

AWS Management Console

This procedure copies an encrypted or unencrypted DB snapshot, in the same AWS Region or across regions, by using the AWS Management Console.

To copy a DB snapshot

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Snapshots**.
3. Select the DB snapshot that you want to copy.
4. Choose **Snapshot Actions**, and then choose **Copy Snapshot**. The **Make Copy of DB Snapshot** page appears.



5. (Optional) To copy the DB snapshot to a different AWS Region, for **Destination Region**, choose the new AWS Region.

   Note
   The destination AWS Region must have the same database engine version available as the source AWS Region.

6. For **New DB Snapshot Identifier**, type the name of the DB snapshot copy.
7. (Optional) For **Target Option Group**, choose a new option group.

   Specify this option if you are copying a snapshot from one AWS Region to another, and your DB instance uses a non-default option group. If your source DB instance uses Transparent Data Encryption for Oracle or Microsoft SQL Server, you must specify this option when copying across regions. For more information, see Option Group Considerations (p. 214).
8. (Optional) Select **Copy Tags** to copy tags and values from the snapshot to the copy of the snapshot.
9. (Optional) For **Enable Encryption**, choose one of the following options:

   - Choose **No** if the DB snapshot isn't encrypted and you don't want to encrypt the copy.
   - Choose **Yes** if the DB snapshot isn't encrypted but you want to encrypt the copy. In this case, for **Master Key**, specify the KMS key identifier to use to encrypt the DB snapshot copy.
   - Choose **Yes** if the DB snapshot is encrypted. In this case, you must encrypt the copy, so **Yes** is already selected. For **Master Key**, specify the KMS key identifier to use to encrypt the DB snapshot copy.

10. Choose **Copy Snapshot**.

CLI

You can copy a DB snapshot by using the AWS CLI command copy-db-snapshot. If you are copying the snapshot to a new AWS Region, run the command in the new AWS Region.

The following options are used to copy a DB snapshot. Not all options are required for all scenarios. Use the descriptions and the examples that follow to determine which options to use.

- `--source-db-snapshot-identifier` – The identifier for the source DB snapshot.
  - If the source snapshot is in the same AWS Region as the copy, specify a valid DB snapshot identifier. For example, `rds:mysql-instance1-snapshot-20130805`.
  - If the source snapshot is in a different AWS Region than the copy, specify a valid DB snapshot ARN. For example, `arn:aws:rds:us-west-2:123456789012:snapshot:mysql-instance1-snapshot-20130805`.
  - If you are copying from a shared manual DB snapshot, this parameter must be the Amazon Resource Name (ARN) of the shared DB snapshot.
  - If you are copying an encrypted snapshot this parameter must be in the ARN format for the source AWS Region, and must match the `SourceDBSnapshotIdentifier` in the `PreSignedUrl` parameter.
- `--target-db-snapshot-identifier` – The identifier for the new copy of the encrypted DB snapshot.
- `--copy-tags` – Include the copy tags option to copy tags and values from the snapshot to the copy of the snapshot.
- `--option-group-name` – The option group to associate with the copy of the snapshot.

  Specify this option if you are copying a snapshot from one AWS Region to another, and your DB instance uses a non-default option group. If your source DB instance uses Transparent Data Encryption for Oracle or Microsoft SQL Server, you must specify this option when copying across regions. For more information, see Option Group Considerations (p. 214).
- `--kms-key-id` – The AWS KMS key ID for an encrypted DB snapshot. The KMS key ID is the Amazon Resource Name (ARN), KMS key identifier, or the KMS key alias for the KMS encryption key.
  - If you copy an encrypted DB snapshot from your AWS account, you can specify a value for this parameter to encrypt the copy with a new KMS encryption key. If you don't specify a value for this parameter, then the copy of the DB snapshot is encrypted with the same KMS key as the source DB snapshot.
  - If you copy an encrypted DB snapshot that is shared from another AWS account, then you must specify a value for this parameter.
  - If you specify this parameter when you copy an unencrypted snapshot, the copy is encrypted.
  - If you copy an encrypted snapshot to a different AWS Region, then you must specify a KMS key for the destination AWS Region. KMS encryption keys are specific to the AWS Region that they are created in, and you cannot use encryption keys from one AWS Region in another AWS Region.
- `--source-region` – The ID of the AWS Region of the source DB snapshot. If you copy an encrypted snapshot to a different AWS Region, then you must specify this option.

Example From Unencrypted, To Same Region

The following code creates a copy of a snapshot, with the new name `mydbsnapshotcopy`, in the same AWS Region as the source snapshot. When the copy is made, all tags on the original snapshot are copied to the snapshot copy.

For Linux, OS X, or Unix:

```
aws rds copy-db-snapshot \
    --source-db-snapshot-identifier mysql-instance1-snapshot-20130805 \
```

```
    --target-db-snapshot-identifier mydbsnapshotcopy \
    --copy-tags
```

For Windows:

```
aws rds copy-db-snapshot ^
    --source-db-snapshot-identifier mysql-instance1-snapshot-20130805 ^
    --target-db-snapshot-identifier mydbsnapshotcopy ^
    --copy-tags
```

Example From Unencrypted, Across Regions

The following code creates a copy of a snapshot, with the new name `mydbsnapshotcopy`, in the AWS Region in which the command is run.

For Linux, OS X, or Unix:

```
aws rds copy-db-snapshot \
    --source-db-snapshot-identifier arn:aws:rds:us-east-1:123456789012:snapshot:mysql-instance1-snapshot-20130805 \
    --target-db-snapshot-identifier mydbsnapshotcopy
```

For Windows:

```
aws rds copy-db-snapshot ^
    --source-db-snapshot-identifier arn:aws:rds:us-east-1:123456789012:snapshot:mysql-instance1-snapshot-20130805 ^
    --target-db-snapshot-identifier mydbsnapshotcopy
```

Example From Encrypted, Across Regions

The following code example copies an encrypted DB snapshot from the us-west-2 region in the us-east-1 region. Run the command in the us-east-1 region.

For Linux, OS X, or Unix:

```
aws rds copy-db-snapshot \
 --source-db-snapshot-identifier arn:aws:rds:us-west-2:123456789012:snapshot:mysql-instance1-snapshot-20161115 \
 --target-db-snapshot-identifier mydbsnapshotcopy \
 --source-region us-west-2 \
 --kms-key-id my-us-east-1-key \
    --option-group-name custom-option-group-name
```

For Windows:

```
aws rds copy-db-snapshot ^
 --source-db-snapshot-identifier arn:aws:rds:us-west-2:123456789012:snapshot:mysql-instance1-snapshot-20161115 ^
 --target-db-snapshot-identifier mydbsnapshotcopy ^
 --source-region us-west-2 ^
 --kms-key-id my-us-east-1-key ^
    --option-group-name custom-option-group-name
```

API

You can copy a DB snapshot by using the Amazon RDS API action CopyDBSnapshot. If you are copying the snapshot to a new AWS Region, perform the action in the new AWS Region.

The following parameters are used to copy a DB snapshot. Not all parameters are required for all scenarios. Use the descriptions and the examples that follow to determine which parameters to use.

- `SourceDBSnapshotIdentifier` – The identifier for the source DB snapshot.
  - If the source snapshot is in the same AWS Region as the copy, specify a valid DB snapshot identifier. For example, `rds:mysql-instance1-snapshot-20130805`.
  - If the source snapshot is in a different AWS Region than the copy, specify a valid DB snapshot ARN. For example, `arn:aws:rds:us-west-2:123456789012:snapshot:mysql-instance1-snapshot-20130805`.
  - If you are copying from a shared manual DB snapshot, this parameter must be the Amazon Resource Name (ARN) of the shared DB snapshot.
  - If you are copying an encrypted snapshot this parameter must be in the ARN format for the source AWS Region, and must match the `SourceDBSnapshotIdentifier` in the `PreSignedUrl` parameter.
- `TargetDBSnapshotIdentifier` – The identifier for the new copy of the encrypted DB snapshot.
- `CopyTags` – Set this parameter to `true` to copy tags and values from the snapshot to the copy of the snapshot. The default is `false`.
- `OptionGroupName` – The option group to associate with the copy of the snapshot.

  Specify this parametr if you are copying a snapshot from one AWS Region to another, and your DB instance uses a non-default option group. If your source DB instance uses Transparent Data Encryption for Oracle or Microsoft SQL Server, you must specify this parameter when copying across regions. For more information, see Option Group Considerations (p. 214).
- `KmsKeyId` – The AWS KMS key ID for an encrypted DB snapshot. The KMS key ID is the Amazon Resource Name (ARN), KMS key identifier, or the KMS key alias for the KMS encryption key.
  - If you copy an encrypted DB snapshot from your AWS account, you can specify a value for this parameter to encrypt the copy with a new KMS encryption key. If you don't specify a value for this parameter, then the copy of the DB snapshot is encrypted with the same KMS key as the source DB snapshot.
  - If you copy an encrypted DB snapshot that is shared from another AWS account, then you must specify a value for this parameter.
  - If you specify this parameter when you copy an unencrypted snapshot, the copy is encrypted.
  - If you copy an encrypted snapshot to a different AWS Region, then you must specify a KMS key for the destination AWS Region. KMS encryption keys are specific to the AWS Region that they are created in, and you cannot use encryption keys from one AWS Region in another AWS Region.
- `PreSignedUrl` – The URL that contains a Signature Version 4 signed request for the `CopyDBSnapshot` API action in the source AWS Region that contains the source DB snapshot to copy.

  You must specify this parameter when you copy an encrypted DB snapshot from another AWS Region by using the Amazon RDS API. You can specify the source region option instead of this parameter when you copy an encrypted DB snapshot from another AWS Region by using the AWS CLI.

  The presigned URL must be a valid request for the `CopyDBSnapshot` API action that can be executed in the source AWS Region that contains the encrypted DB snapshot to be copied. The presigned URL request must contain the following parameter values:

  - `DestinationRegion` - The AWS Region that the encrypted DB snapshot will be copied to. This AWS Region is the same one where the `CopyDBSnapshot` action is called that contains this presigned URL.

    For example, if you copy an encrypted DB snapshot from the us-west-2 region to the us-east-1 region, then you call the `CopyDBSnapshot` action in the us-east-1 region and provide a presigned URL that contains a call to the `CopyDBSnapshot` action in the us-west-2 region. For this example, the `DestinationRegion` in the presigned URL must be set to the us-east-1 region.

- `KmsKeyId` - The KMS key identifier for the key to use to encrypt the copy of the DB snapshot in the destination AWS Region. This is the same identifier for both the `CopyDBSnapshot` action that is called in the destination AWS Region, and the action contained in the presigned URL.
- `SourceDBSnapshotIdentifier` - The DB snapshot identifier for the encrypted snapshot to be copied. This identifier must be in the Amazon Resource Name (ARN) format for the source AWS Region. For example, if you are copying an encrypted DB snapshot from the us-west-2 region, then your `SourceDBSnapshotIdentifier` looks like the following example: `arn:aws:rds:us-west-2:123456789012:snapshot:mysql-instance1-snapshot-20161115`.

For more information on Signature Version 4 signed requests, see the following:

- Authenticating Requests: Using Query Parameters (AWS Signature Version 4) in the Amazon Simple Storage Service API Reference
- Signature Version 4 Signing Process in the AWS General Reference

Example From Unencrypted, To Same Region

The following code creates a copy of a snapshot, with the new name `mydbsnapshotcopy`, in the same AWS Region as the source snapshot. When the copy is made, all tags on the original snapshot are copied to the snapshot copy.

```
https://rds.us-west-1.amazonaws.com/
 ?Action=CopyDBSnapshot
 &CopyTags=true
 &SignatureMethod=HmacSHA256
 &SignatureVersion=4
 &SourceDBSnapshotIdentifier=mysql-instance1-snapshot-20130805
 &TargetDBSnapshotIdentifier=mydbsnapshotcopy
 &Version=2013-09-09
 &X-Amz-Algorithm=AWS4-HMAC-SHA256
 &X-Amz-Credential=AKIADQKE4SARGYLE/20140429/us-west-1/rds/aws4_request
 &X-Amz-Date=20140429T175351Z
 &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
 &X-Amz-Signature=9164337efa99caf850e874a1cb7ef62f3cea29d0b448b9e0e7c53b288ddffed2
```

Example From Unencrypted, Across Regions

The following code creates a copy of a snapshot, with the new name `mydbsnapshotcopy`, in the us-west-1 region.

```
https://rds.us-west-1.amazonaws.com/
 ?Action=CopyDBSnapshot
 &SignatureMethod=HmacSHA256
 &SignatureVersion=4
 &SourceDBSnapshotIdentifier=arn%3Aaws%3Ards%3Aus-east-1%3A123456789012%3Asnapshot%3Amysql-
instance1-snapshot-20130805
 &TargetDBSnapshotIdentifier=mydbsnapshotcopy
 &Version=2013-09-09
 &X-Amz-Algorithm=AWS4-HMAC-SHA256
 &X-Amz-Credential=AKIADQKE4SARGYLE/20140429/us-west-1/rds/aws4_request
 &X-Amz-Date=20140429T175351Z
 &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
 &X-Amz-Signature=9164337efa99caf850e874a1cb7ef62f3cea29d0b448b9e0e7c53b288ddffed2
```

Example From Encrypted, Across Regions

The following code creates a copy of a snapshot, with the new name `mydbsnapshotcopy`, in the us-east-1 region.

```
https://rds.us-east-1.amazonaws.com/
    ?Action=CopyDBSnapshot
    &KmsKeyId=my-us-east-1-key
    &OptionGroupName=custom-option-group-name
    &PreSignedUrl=https%253A%252F%252Frds.us-west-2.amazonaws.com%252F
         %253FAction%253DCopyDBSnapshot
         %2526DestinationRegion%253Dus-east-1
         %2526KmsKeyId%253Dmy-us-east-1-key
         %2526SourceDBSnapshotIdentifier%253Darn%25253Aaws%25253Ards%25253Aus-
west-2%25253A123456789012%25253Asnapshot%25253Amysql-instance1-snapshot-20161115
         %2526SignatureMethod%253DHmacSHA256
         %2526SignatureVersion%253D4
         %2526Version%253D2014-10-31
         %2526X-Amz-Algorithm%253DAWS4-HMAC-SHA256
         %2526X-Amz-Credential%253DAKIADQKE4SARGYLE%252F20161117%252Fus-west-2%252Frds
%252Faws4_request
         %2526X-Amz-Date%253D20161117T215409Z
         %2526X-Amz-Expires%253D3600
         %2526X-Amz-SignedHeaders%253Dcontent-type%253Bhost%253Buser-agent%253Bx-amz-
content-sha256%253Bx-amz-date
         %2526X-Amz-Signature
%253D255a0f17b4e717d3b67fad163c3ec26573b882c03a65523522cf890a67fca613
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &SourceDBSnapshotIdentifier=arn%3Aaws%3Ards%3Aus-west-2%3A123456789012%3Asnapshot
%3Amysql-instance1-snapshot-20161115
    &TargetDBSnapshotIdentifier=mydbsnapshotcopy
    &Version=2014-10-31
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20161117/us-east-1/rds/aws4_request
    &X-Amz-Date=20161117T221704Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=da4f2da66739d2e722c85fcfd225dc27bba7e2b8dbea8d8612434378e52adccf
```

Copying a DB Cluster Snapshot

Use the procedures in this topic to copy a DB cluster snapshot. If your source database engine is Aurora, then your snapshot is a DB cluster snapshot. If your source database engine is MariaDB, Microsoft SQL Server, MySQL, Oracle, or PostgreSQL, then your snapshot is a DB snapshot. For instructions on how to copy a DB snapshot, see Copying a DB Snapshot (p. 215).

For each AWS account, you can copy up to five DB cluster snapshots at a time from one AWS Region to another. Copying both encrypted and unencrypted DB cluster snapshots is supported. If you copy a DB cluster snapshot to another AWS Region, you create a manual DB cluster snapshot that is retained in that AWS Region. Copying a DB cluster snapshot out of the source AWS Region incurs Amazon RDS data transfer charges.

For more information about data transfer pricing, see Amazon RDS Pricing.

After the DB cluster snapshot copy has been created in the new AWS Region, the DB cluster snapshot copy behaves the same as all other DB cluster snapshots in that AWS Region.

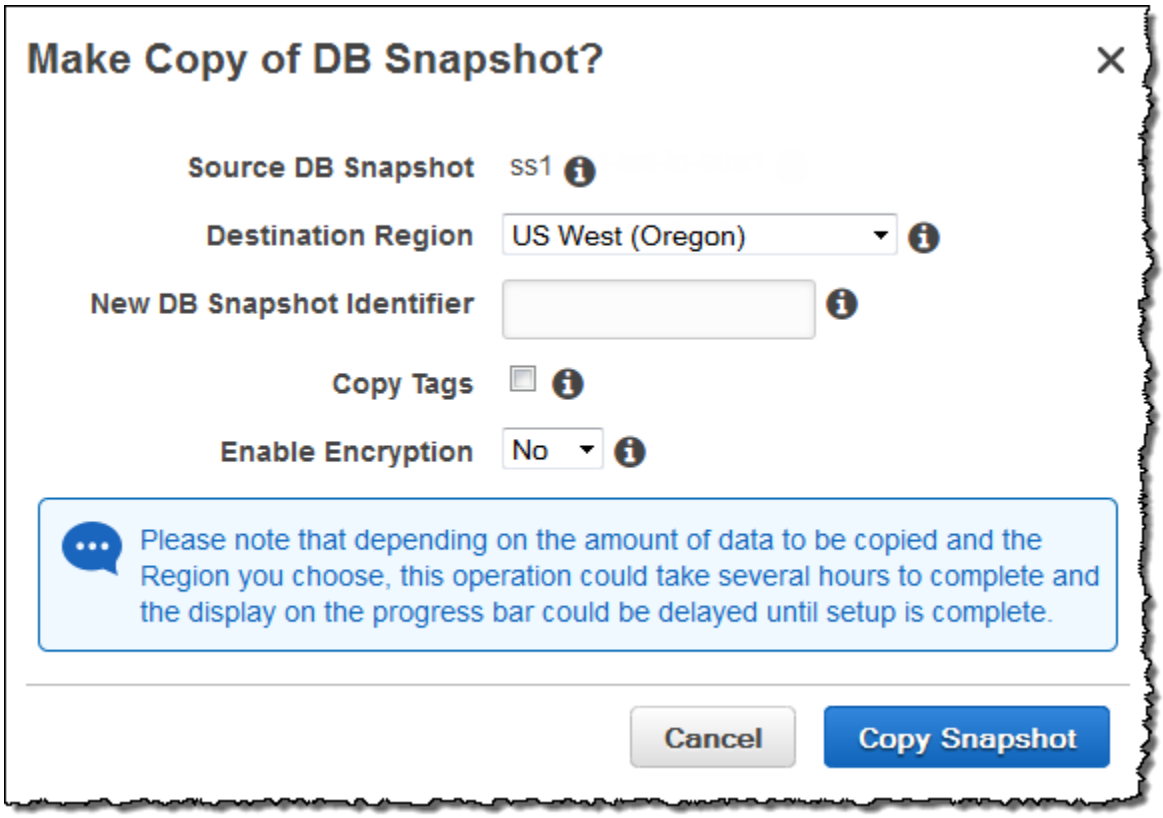
AWS Management Console

This procedure works for copying encrypted or unencrypted DB cluster snapshots, in the same AWS Region or across regions.

To cancel a copy operation once it is in progress, delete the target DB cluster snapshot while that DB cluster snapshot is in **copying** status.

To copy a DB cluster snapshot

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Snapshots**.
3. Select the check box for the DB snapshot you want to copy.
4. Choose **Snapshot Actions**, and then choose **Copy Snapshot**. The **Make Copy of DB Snapshot** page appears.



5. (Optional) To copy the DB cluster snapshot to a different AWS Region, choose that AWS Region for **Destination Region**.
6. Type the name of the DB cluster snapshot copy in **New DB Snapshot Identifier**.
7. To copy tags and values from the snapshot to the copy of the snapshot, choose **Copy Tags**.
8. For **Enable Encryption**, choose one of the following options:

   - Choose **No** if the DB cluster snapshot isn't encrypted and you don't want to encrypt the copy.
   - Choose **Yes** if the DB cluster snapshot isn't encrypted but you want to encrypt the copy. In this case, for **Master Key**, specify the KMS key identifier to use to encrypt the DB cluster snapshot copy.
   - Choose **Yes** if the DB cluster snapshot is encrypted. In this case, you must encrypt the copy, so **Yes** is already selected. For **Master Key**, specify the KMS key identifier to use to encrypt the DB cluster snapshot copy.

9. Choose **Copy Snapshot**.

Copying an Unencrypted DB Cluster Snapshot by Using the AWS CLI or Amazon RDS API

Use the procedures in the following sections to copy an unencrypted DB cluster snapshot by using the AWS CLI or Amazon RDS API.

To cancel a copy operation once it is in progress, delete the target DB cluster snapshot identified by `--target-db-cluster-snapshot-identifier` or `TargetDBClusterSnapshotIdentifier` while that DB cluster snapshot is in **copying** status.

CLI

To copy a DB cluster snapshot, use the AWS CLI `copy-db-cluster-snapshot` command. If you are copying the snapshot to another AWS Region, run the command in the AWS Region to which the snapshot will be copied.

The following options are used to copy an unencrypted DB cluster snapshot:

- `--source-db-cluster-snapshot-identifier` – The identifier for the DB cluster snapshot to be copied. If you are copying the snapshot to another AWS Region, this identifier must be in the ARN format for the source AWS Region.
- `--target-db-cluster-snapshot-identifier` – The identifier for the new copy of the DB cluster snapshot.

The following code creates a copy of DB cluster snapshot `arn:aws:rds:us-east-1:123456789012:cluster-snapshot:aurora-cluster1-snapshot-20130805` named `myclustersnapshotcopy` in the AWS Region in which the command is run. When the copy is made, all tags on the original snapshot are copied to the snapshot copy.

Example

For Linux, OS X, or Unix:

```
aws rds copy-db-cluster-snapshot \
  --source-db-cluster-snapshot-identifier arn:aws:rds:us-east-1:123456789012:cluster-snapshot:aurora-cluster1-snapshot-20130805 \
  --target-db-cluster-snapshot-identifier myclustersnapshotcopy \
  --copy-tags
```

For Windows:

```
aws rds copy-db-cluster-snapshot ^
  --source-db-cluster-snapshot-identifier arn:aws:rds:us-east-1:123456789012:cluster-snapshot:aurora-cluster1-snapshot-20130805 ^
  --target-db-cluster-snapshot-identifier myclustersnapshotcopy ^
  --copy-tags
```

API

To copy a DB cluster snapshot, use the Amazon RDS API `CopyDBClusterSnapshot` action. If you are copying the snapshot to another AWS Region, perform the action in the AWS Region to which the snapshot will be copied.

The following parameters are used to copy an unencrypted DB cluster snapshot:

- `SourceDBClusterSnapshotIdentifier` – The identifier for the DB cluster snapshot to be copied. If you are copying the snapshot to another AWS Region, this identifier must be in the ARN format for the source AWS Region.
- `TargetDBClusterSnapshotIdentifier` – The identifier for the new copy of the DB cluster snapshot.

The following code creates a copy of a snapshot `arn:aws:rds:us-east-1:123456789012:cluster-snapshot:aurora-cluster1-snapshot-20130805` named `myclustersnapshotcopy` in the us-west-1 region. When the copy is made, all tags on the original snapshot are copied to the snapshot copy.

Example

```
https://rds.us-west-1.amazonaws.com/
   ?Action=CopyDBClusterSnapshot
   &CopyTags=true
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &SourceDBSnapshotIdentifier=arn%3Aaws%3Ards%3Aus-east-1%3A123456789012%3Acluster-
snapshot%3Aaurora-cluster1-snapshot-20130805
   &TargetDBSnapshotIdentifier=myclustersnapshotcopy
   &Version=2013-09-09
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20140429/us-west-1/rds/aws4_request
   &X-Amz-Date=20140429T175351Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
   &X-Amz-Signature=9164337efa99caf850e874a1cb7ef62f3cea29d0b448b9e0e7c53b288ddffed2
```

Copying an Encrypted DB Cluster Snapshot by Using the AWS CLI or Amazon RDS API

Use the procedures in the following sections to copy an encrypted DB cluster snapshot by using the AWS CLI or Amazon RDS API.

To cancel a copy operation once it is in progress, delete the target DB cluster snapshot identified by `--target-db-cluster-snapshot-identifier` or `TargetDBClusterSnapshotIdentifier` while that DB cluster snapshot is in **copying** status.

CLI

To copy a DB cluster snapshot, use the AWS CLI `copy-db-cluster-snapshot` command. If you are copying the snapshot to another AWS Region, run the command in the AWS Region to which the snapshot will be copied.

The following options are used to copy an encrypted DB cluster snapshot:

- `--source-region` – If you are copying the snapshot to another AWS Region, specify the AWS Region that the encrypted DB cluster snapshot will be copied from.

  If you are copying the snapshot to another AWS Region and you don't specify `source-region`, you must specify the `pre-signed-url` option instead. The `pre-signed-url` value must be a URL that contains a Signature Version 4 signed request for the `CopyDBClusterSnapshot` action to be called in the source AWS Region where the DB cluster snapshot is copied from. To learn more about the `pre-signed-url`, see `copy-db-cluster-snapshot`.
- `--source-db-cluster-snapshot-identifier` – The identifier for the encrypted DB cluster snapshot to be copied. If you are copying the snapshot to another AWS Region, this identifier must be

in the ARN format for the source AWS Region. If that is the case, the AWS Region specified in `source-db-cluster-snapshot-identifier` must match the AWS Region specified for `--source-region`.

- `--target-db-cluster-snapshot-identifier` – The identifier for the new copy of the encrypted DB cluster snapshot.
- `--kms-key-id` – The KMS key identifier for the key to use to encrypt the copy of the DB cluster snapshot.

  You can optionally use this option if the DB cluster snapshot is encrypted, you are copying the snapshot in the same AWS Region, and you want to specify a new KMS encryption key to use to encrypt the copy. Otherwise, the copy of the DB cluster snapshot is encrypted with the same KMS key as the source DB cluster snapshot.

  You must use this option if the DB cluster snapshot is encrypted and you are copying the snapshot to another AWS Region. In that case, you must specify a KMS key for the destination AWS Region.

The following code example copies the encrypted DB cluster snapshot from the us-west-2 region to the us-east-1 region. The command is called in the us-east-1 region.

Example

For Linux, OS X, or Unix:

```
aws rds copy-db-cluster-snapshot \
  --source-db-cluster-snapshot-identifier arn:aws:rds:us-west-2:123456789012:cluster-snapshot:aurora-cluster1-snapshot-20161115 \
  --target-db-cluster-snapshot-identifier myclustersnapshotcopy \
  --source-region us-west-2 \
  --kms-key-id my-us-east-1-key
```

For Windows:

```
aws rds copy-db-cluster-snapshot ^
  --source-db-cluster-snapshot-identifier arn:aws:rds:us-west-2:123456789012:cluster-snapshot:aurora-cluster1-snapshot-20161115 ^
  --target-db-cluster-snapshot-identifier myclustersnapshotcopy ^
  --source-region us-west-2 ^
  --kms-key-id my-us-east-1-key
```

API

To copy a DB cluster snapshot, use the Amazon RDS API `CopyDBClusterSnapshot` action. If you are copying the snapshot to another AWS Region, perform the action in the AWS Region to which the snapshot will be copied.

The following parameters are used to copy an encrypted DB cluster snapshot:

- `SourceDBClusterSnapshotIdentifier` – The identifier for the encrypted DB cluster snapshot to be copied. If you are copying the snapshot to another AWS Region, this identifier must be in the ARN format for the source AWS Region.
- `TargetDBClusterSnapshotIdentifier` – The identifier for the new copy of the encrypted DB cluster snapshot.
- `KmsKeyId` – The KMS key identifier for the key to use to encrypt the copy of the DB cluster snapshot.

  You can optionally use this parameter if the DB cluster snapshot is encrypted, you are copying the snapshot in the same AWS Region, and you want to specify a new KMS encryption key to use to

encrypt the copy. Otherwise, the copy of the DB cluster snapshot is encrypted with the same KMS key as the source DB cluster snapshot.

You must use this parameter if the DB cluster snapshot is encrypted and you are copying the snapshot to another AWS Region. In that case, you must specify a KMS key for the destination AWS Region.

- `PreSignedUrl` – If you are copying the snapshot to another AWS Region, you must specify the `PreSignedUrl` parameter. The `PreSignedUrl` value must be a URL that contains a Signature Version 4 signed request for the `CopyDBClusterSnapshot` action to be called in the source AWS Region where the DB cluster snapshot is copied from. To learn more about using a presigned URL, see `CopyDBClusterSnapshot`.

  To automatically rather than manually generate a presigned URL, use the AWS CLI `copy-db-cluster-snapshot` command with the `--source-region` option instead.

The following code example copies the encrypted DB cluster snapshot from the us-west-2 region to the us-east-1 region. The action is called in the us-east-1 region.

Example

```
https://rds.us-east-1.amazonaws.com/
    ?Action=CopyDBClusterSnapshot
    &KmsKeyId=my-us-east-1-key
    &PreSignedUrl=https%253A%252F%252Frds.us-west-2.amazonaws.com%252F
         %253FAction%253DCopyDBClusterSnapshot
         %2526DestinationRegion%253Dus-east-1
         %2526KmsKeyId%253Dmy-us-east-1-key
         %2526SourceDBClusterSnapshotIdentifier%253Darn%25253Aaws%25253Ards%25253Aus-
west-2%25253A123456789012%25253Acluster-snapshot%25253Aaurora-cluster1-snapshot-20161115
         %2526SignatureMethod%253DHmacSHA256
         %2526SignatureVersion%253D4
         %2526Version%253D2014-10-31
         %2526X-Amz-Algorithm%253DAWS4-HMAC-SHA256
         %2526X-Amz-Credential%253DAKIADQKE4SARGYLE%252F20161117%252Fus-west-2%252Frds
%252Faws4_request
         %2526X-Amz-Date%253D20161117T215409Z
         %2526X-Amz-Expires%253D3600
         %2526X-Amz-SignedHeaders%253Dcontent-type%253Bhost%253Buser-agent%253Bx-amz-
content-sha256%253Bx-amz-date
         %2526X-Amz-Signature
%253D255a0f17b4e717d3b67fad163c3ec26573b882c03a65523522cf890a67fca613
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &SourceDBClusterSnapshotIdentifier=arn%3Aaws%3Ards%3Aus-
west-2%3A123456789012%3Acluster-snapshot%3Aaurora-cluster1-snapshot-20161115
    &TargetDBClusterSnapshotIdentifier=myclustersnapshotcopy
    &Version=2014-10-31
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20161117/us-east-1/rds/aws4_request
    &X-Amz-Date=20161117T221704Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=da4f2da66739d2e722c85fcfd225dc27bba7e2b8dbea8d8612434378e52adccf
```

Copying a DB Cluster Snapshot Across Accounts

You can enable other AWS accounts to copy DB cluster snapshots that you specify by using the Amazon RDS API `ModifyDBClusterSnapshotAttribute` and `CopyDBClusterSnapshot` actions. You can only copy DB cluster snapshots across accounts in the same AWS Region. The cross-account copying process works as follows, where Account A is making the snapshot available to copy, and Account B is copying it.

1. Using Account A, call `ModifyDBClusterSnapshotAttribute`, specifying **`restore`** for the `AttributeName` parameter, and the ID for Account B for the `ValuesToAdd` parameter.
2. (If the snapshot is encrypted) Using Account A, update the key policy for the KMS key, first adding the ARN of Account B as a `Principal`, and then allow the `kms:CreateGrant` action.
3. (If the snapshot is encrypted) Using Account B, choose or create an IAM user and attach an IAM policy to that user that allows it to copy an encrypted DB snapshot using your KMS key.
4. Using Account B, call `CopyDBClusterSnapshot` and use the `SourceDBClusterSnapshotIdentifier` parameter to specify the ARN of the DB cluster snapshot to be copied, which must include the ID for Account A.

To list all of the AWS accounts permitted to restore a DB snapshot, use the DescribeDBSnapshotAttributes or DescribeDBClusterSnapshotAttributes API action.

To remove sharing permission for an AWS account, use the `ModifyDBSnapshotAttribute` or `ModifyDBClusterSnapshotAttribute` action with `AttributeName` set to `restore` and the ID of the account to remove in the `ValuesToRemove` parameter.

Copying an Unencrypted DB Cluster Snapshot to Another Account

Use the following procedure to copy an unencrypted DB cluster snapshot to another account in the same AWS Region.

1. In the source account for the DB cluster snapshot, call `ModifyDBClusterSnapshotAttribute`, specifying **`restore`** for the `AttributeName` parameter, and the ID for the target account for the `ValuesToAdd` parameter.

   Running the following example using the account `987654321` permits two AWS account identifiers, `123451234512` and `123456789012`, to restore the DB snapshot named `manual-snapshot1`.

```
https://rds.us-west-2.amazonaws.com/
 ?Action=ModifyDBClusterSnapshotAttribute
 &AttributeName=restore
 &DBClusterSnapshotIdentifier=manual-snapshot1
 &SignatureMethod=HmacSHA256&SignatureVersion=4
 &ValuesToAdd.member.1=123451234512
 &ValuesToAdd.member.2=123456789012
 &Version=2014-10-31
 &X-Amz-Algorithm=AWS4-HMAC-SHA256
 &X-Amz-Credential=AKIADQKE4SARGYLE/20150922/us-west-2/rds/aws4_request
 &X-Amz-Date=20150922T220515Z
 &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
 &X-Amz-Signature=ef38f1ce3dab4e1dbf113d8d2a265c67d17ece1999ffd36be85714ed36dddbb3
```

2. In the target account, call `CopyDBClusterSnapshot` and use the `SourceDBClusterSnapshotIdentifier` parameter to specify the ARN of the DB cluster snapshot to be copied, which must include the ID for the source account.

   Running the following example using the account `123451234512` copies the DB cluster snapshot `aurora-cluster1-snapshot-20130805` from account `987654321` and creates a DB cluster snapshot named `dbclustersnapshot1`.

```
https://rds.us-west-2.amazonaws.com/
   ?Action=CopyDBClusterSnapshot
   &CopyTags=true
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &SourceDBClusterSnapshotIdentifier=arn:aws:rds:us-west-2:987654321:cluster-
snapshot:aurora-cluster1-snapshot-20130805
   &TargetDBClusterSnapshotIdentifier=dbclustersnapshot1
```

```
   &Version=2013-09-09
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20150922/us-west-2/rds/aws4_request
   &X-Amz-Date=20140429T175351Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
   &X-Amz-Signature=9164337efa99caf850e874a1cb7ef62f3cea29d0b448b9e0e7c53b288ddffed2
```

Copying an Encrypted DB Cluster Snapshot to Another Account

Use the following procedure to copy an encrypted DB cluster snapshot to another account in the same AWS Region.

1. In the source account for the DB cluster snapshot, call `ModifyDBClusterSnapshotAttribute`, specifying **`restore`** for the `AttributeName` parameter, and the ID for the target account for the `ValuesToAdd` parameter.

   Running the following example using the account `987654321` permits two AWS account identifiers, `123451234512` and `123456789012`, to restore the DB snapshot named `manual-snapshot1`.

```
https://rds.us-west-2.amazonaws.com/
 ?Action=ModifyDBClusterSnapshotAttribute
 &AttributeName=restore
 &DBClusterSnapshotIdentifier=manual-snapshot1
 &SignatureMethod=HmacSHA256&SignatureVersion=4
 &ValuesToAdd.member.1=123451234512
 &ValuesToAdd.member.2=123456789012
 &Version=2014-10-31
 &X-Amz-Algorithm=AWS4-HMAC-SHA256
 &X-Amz-Credential=AKIADQKE4SARGYLE/20150922/us-west-2/rds/aws4_request
 &X-Amz-Date=20150922T220515Z
 &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
 &X-Amz-Signature=ef38f1ce3dab4e1dbf113d8d2a265c67d17ece1999ffd36be85714ed36dddbb3
```

2. In the source account for the DB cluster snapshot, update the key policy for the KMS key, first adding the ARN of the target account as a `Principal`, and then allow the `kms:CreateGrant` action. For more information, see Allowing Access to an AWS KMS Encryption Key (p. 231).
3. In the target account, choose or create an IAM user and attach an IAM policy to that user that allows it to copy an encrypted DB snapshot using your KMS key. For more information, see Creating an IAM Policy to Enable Copying of the Encrypted Snapshot (p. 232).
4. In the target account, call `CopyDBClusterSnapshot` and use the `SourceDBClusterSnapshotIdentifier` parameter to specify the ARN of the DB cluster snapshot to be copied, which must include the ID for the source account.

   Running the following example using the account `123451234512` copies the DB cluster snapshot `aurora-cluster1-snapshot-20130805` from account `987654321` and creates a DB cluster snapshot named `dbclustersnapshot1`.

```
https://rds.us-west-2.amazonaws.com/
   ?Action=CopyDBClusterSnapshot
   &CopyTags=true
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &SourceDBClusterSnapshotIdentifier=arn:aws:rds:us-west-2:987654321:cluster-
snapshot:aurora-cluster1-snapshot-20130805
   &TargetDBClusterSnapshotIdentifier=dbclustersnapshot1
   &Version=2013-09-09
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20150922/us-west-2/rds/aws4_request
   &X-Amz-Date=20140429T175351Z
```

```
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=9164337efa99caf850e874a1cb7ef62f3cea29d0b448b9e0e7c53b288ddffed2
```

Related Topics

- Creating a DB Snapshot (p. 207)
- Restoring from a DB Snapshot (p. 209)
- Backing Up and Restoring Amazon RDS DB Instances (p. 200)

Sharing a DB Snapshot or DB Cluster Snapshot

Using Amazon RDS, you can share a manual DB snapshot or DB cluster snapshot. Sharing snapshots works as described following.

- Sharing a manual DB snapshot or DB cluster snapshot, whether encrypted or unencrypted, enables authorized AWS accounts to copy the snapshot.
- Sharing an unencrypted manual DB snapshot enables authorized AWS accounts to directly restore a DB instance from the snapshot instead of taking a copy of it and restoring from that. This isn't supported for encrypted manual DB snapshots.
- Sharing a manual DB cluster snapshot, whether encrypted or unencrypted, enables authorized AWS accounts to directly restore a DB cluster from the snapshot instead of taking a copy of it and restoring from that.

Note
To share an automated DB snapshot or DB cluster snapshot, copy it to make a manual version of it, and then share that copy.

For more information on copying a snapshot, see Copying a DB Snapshot or DB Cluster Snapshot (p. 213). For more information on restoring a DB instance from a DB snapshot, see Restoring from a DB Snapshot (p. 209). For more information on restoring a DB cluster from a DB cluster snapshot, see Backing Up and Restoring an Aurora DB Cluster (p. 468).

You can share a manual snapshot with up to 20 other AWS accounts. You can also share an unencrypted manual snapshot as public, which makes the snapshot available to all AWS accounts. Take care when sharing a snapshot as public so that none of your private information is included in any of your public snapshots.

The following limitations apply when sharing manual snapshots with other AWS accounts:

- When you restore a DB instance or DB cluster from a shared snapshot using the AWS Command Line Interface (AWS CLI) or Amazon RDS API, you must specify the Amazon Resource Name (ARN) of the shared snapshot as the snapshot identifier.
- You cannot share a DB snapshot that uses an option group with permanent or persistent options.

  A *permanent option* cannot be removed from an option group. Option groups with persistent options cannot be removed from a DB instance once the option group has been assigned to the DB instance.

  The following table lists permanent and persistent options and their related DB engines.

Option Name	Persistent	Permanent	DB Engine
TDE	Yes	No	Microsoft SQL Server Enterprise Edition
TDE	Yes	Yes	Oracle Enterprise Edition
TDE_HSM	Yes	Yes	Oracle Enterprise Edition
Timezone	Yes	Yes	Oracle Enterprise Edition Oracle Standard Edition Oracle Standard Edition One

Sharing an Encrypted Snapshot

You can share DB snapshots or DB cluster snapshots that have been encrypted "at rest" using the AES-256 encryption algorithm, as described in Encrypting Amazon RDS Resources (p. 355). To do this, you must take the following steps:

1. Share the AWS Key Management Service (AWS KMS) encryption key that was used to encrypt the snapshot with any accounts that you want to be able to access the snapshot.

   You can share AWS KMS encryption keys with another AWS account by adding the other account to the KMS key policy. For details on updating a key policy, see Key Policies in the *AWS KMS Developer Guide*. For an example of creating a key policy, see Allowing Access to an AWS KMS Encryption Key (p. 231) later in this topic.
2. Use the AWS Management Console, AWS CLI, or Amazon RDS API to share the encrypted snapshot with the other accounts.

These restrictions apply to sharing encrypted snapshots:

- You can't share encrypted snapshots as public.
- You can't share Oracle or Microsoft SQL Server snapshots that are encrypted using Transparent Data Encryption (TDE).
- You can't share a snapshot that has been encrypted using the default AWS KMS encryption key of the AWS account that shared the snapshot.

Allowing Access to an AWS KMS Encryption Key

For another AWS account to copy an encrypted DB snapshot or DB cluster snapshot shared from your account, the account that you share your snapshot with must have access to the KMS key that encrypted the snapshot. To allow another AWS account access to an AWS KMS key, update the key policy for the KMS key with the ARN of the AWS account that you are sharing to as a `Principal` in the KMS key policy, and then allow the `kms:CreateGrant` action.

After you have given an AWS account access to your KMS encryption key, to copy your encrypted snapshot, that AWS account must create an AWS Identity and Access Management (IAM) user if it doesn't already have one. In addition, that AWS account must also attach an IAM policy to that IAM user that allows the IAM user to copy an encrypted DB snapshot using your KMS key. The account must be an IAM user and cannot be a root AWS account identity due to KMS security restrictions.

In the following key policy example, user `111122223333` is the owner of the KMS encryption key, and user `444455556666` is the account that the key is being shared with. This updated key policy gives the AWS account access to the KMS key by including the ARN for the root AWS account identity for user `444455556666` as a `Principal` for the policy, and by allowing the `kms:CreateGrant` action.

```
{
  "Id": "key-policy-1",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Allow use of the key",
      "Effect": "Allow",
      "Principal": {"AWS": [
        "arn:aws:iam::111122223333:user/KeyUser",
        "arn:aws:iam::444455556666:root"
      ]},
      "Action": [
        "kms:CreateGrant",
        "kms:Encrypt",
```

```
        "kms:Decrypt",
        "kms:ReEncrypt",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": "*"
    },
    {
      "Sid": "Allow attachment of persistent resources",
      "Effect": "Allow",
      "Principal": {"AWS": [
        "arn:aws:iam::111122223333:user/KeyUser",
        "arn:aws:iam::444455556666:root"
      ]},
      "Action": [
        "kms:CreateGrant",
        "kms:ListGrants",
        "kms:RevokeGrant"
      ],
      "Resource": "*",
      "Condition": {"Bool": {"kms:GrantIsForAWSResource": true}}
    }
  ]
}
```

Creating an IAM Policy to Enable Copying of the Encrypted Snapshot

Once the external AWS account has access to your KMS key, the owner of that AWS account can create a policy that allows an IAM user created for that account to copy an encrypted snapshot encrypted with that KMS key.

The following example shows a policy that can be attached to an IAM user for AWS account `444455556666` that enables the IAM user to copy a shared snapshot from AWS account `111122223333` that has been encrypted with the KMS key `c989c1dd-a3f2-4a5d-8d96-e793d082ab26` in the `us-west-2` region.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowUseOfTheKey",
            "Effect": "Allow",
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey",
                "kms:CreateGrant",
                "kms:RetireGrant"
            ],
            "Resource": ["arn:aws:kms:us-west-2:111122223333:key/c989c1dd-a3f2-4a5d-8d96-
e793d082ab26"]
        },
        {
            "Sid": "AllowAttachmentOfPersistentResources",
            "Effect": "Allow",
            "Action": [
                "kms:CreateGrant",
                "kms:ListGrants",
                "kms:RevokeGrant"
            ],
```

```
            "Resource": ["arn:aws:kms:us-west-2:111122223333:key/c989c1dd-a3f2-4a5d-8d96-
e793d082ab26"],
            "Condition": {
                "Bool": {
                    "kms:GrantIsForAWSResource": true
                }
            }
        }
    ]
}
```

For details on updating a key policy, see Key Policies in the *AWS KMS Developer Guide*.

AWS Management Console

Using the Amazon RDS console, you can share a manual DB snapshot or DB cluster snapshot with up to 20 AWS accounts. You can also use the console to stop sharing a manual snapshot with one or more accounts.

To share a manual DB snapshot or DB cluster snapshot by using the Amazon RDS console

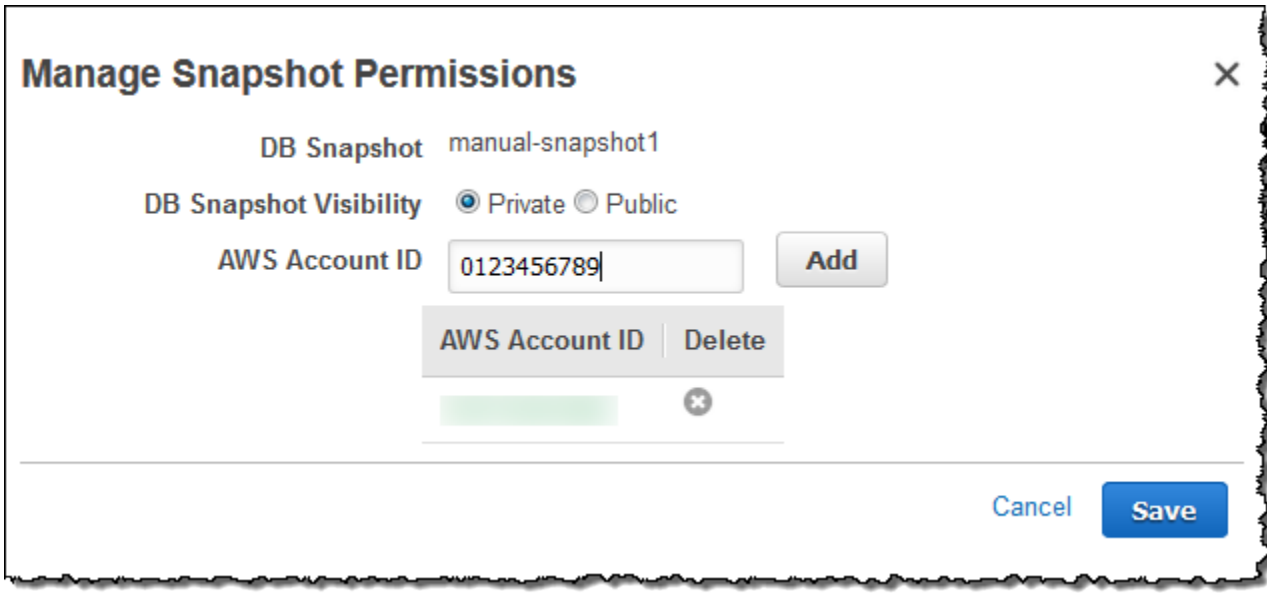
1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Snapshots**.
3. For **Filter**, choose **Manual Snapshots**.
4. Select the check box for the manual snapshot that you want to share.
5. Choose **Snapshot Actions**, and then choose **Share Snapshot**.
6. Choose one of the following options for **DB Snapshot Visibility**.

   - If the source DB cluster is unencrypted, choose **Public** to permit all AWS accounts to restore a DB instance from your manual DB snapshot, or choose **Private** to permit only AWS accounts that you specify to restore a DB instance from your manual DB snapshot.

     Warning
     If you set **DB Snapshot Visibility** to **Public**, all AWS accounts can restore a DB instance from your manual DB snapshot and have access to your data. Do not share any manual DB snapshots that contain private information as **Public**.

   - If the source DB cluster is encrypted, **DB Snapshot Visibility** is set as **Private** because encrypted snapshots can't be shared as public.
7. For **AWS Account ID**, type the AWS account identifier for an account that you want to permit to restore a DB instance or DB cluster from your manual snapshot, and then choose **Add**. Repeat to include additional AWS account identifiers, up to 20 AWS accounts.

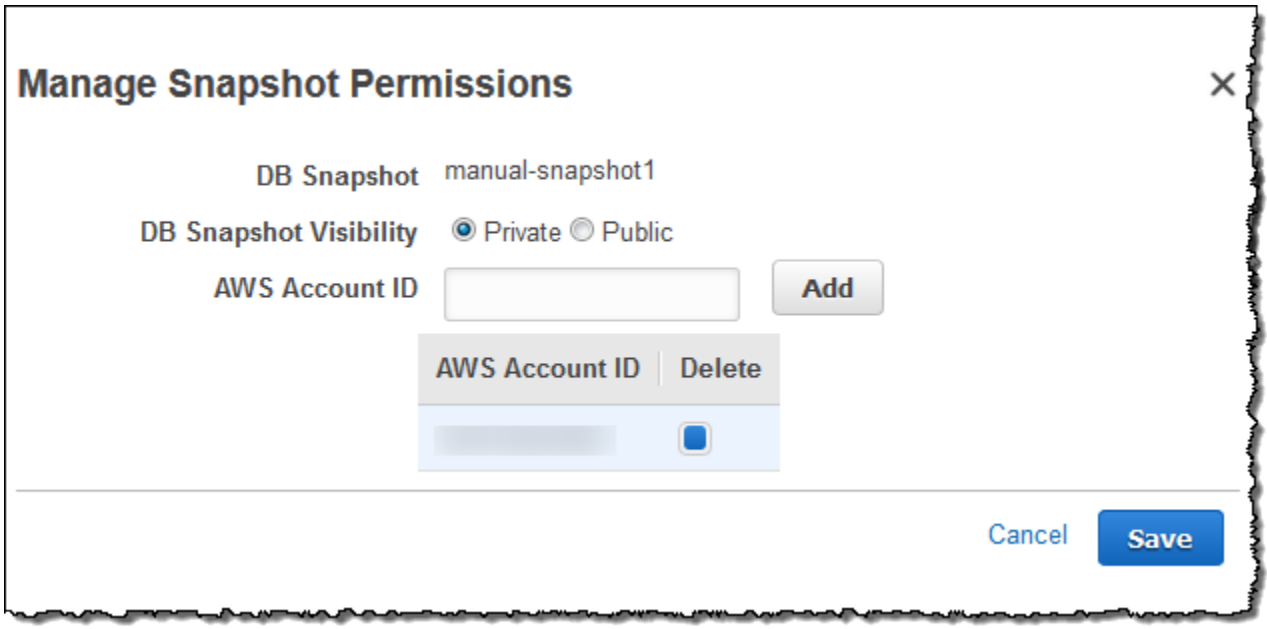
   If you make an error when adding an AWS account identifier to the list of permitted accounts, you can delete it from the list by choosing **Delete** at the right of the incorrect AWS account identifier.



8. After you have added identifiers for all of the AWS accounts that you want to permit to restore the manual snapshot, choose **Save** to save your changes.

To stop sharing a manual DB snapshot or DB cluster snapshot with an AWS account

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Snapshots**.
3. For **Filter**, choose **Manual Snapshots**.
4. Select the check box for the manual snapshot that you want to stop sharing.
5. Choose **Snapshot Actions**, and then choose **Share Snapshot**.
6. To remove permission for an AWS account, choose **Delete** for the AWS account identifier for that account from the list of authorized accounts.



7. Choose **Save** to save your changes.

API

You can also share a manual DB snapshot or DB cluster snapshot with other AWS accounts by using the Amazon RDS API. To do so, call the `ModifyDBSnapshotAttribute` action for DB instances, or the `ModifyDBClusterSnapshotAttribute` action for Amazon Aurora DB clusters. Specify `restore` for `AttributeName`, and use the `ValuesToAdd` parameter to add a list of the IDs for the AWS accounts that are authorized to restore the manual snapshot.

To make a manual snapshot public and restorable by all AWS accounts, use the value `all`. However, take care not to add the `all` value for any manual snapshots that contain private information that you don't want to be available to all AWS accounts. Also, don't specify `all` for encrypted snapshots, because making such snapshots public isn't supported.

To remove sharing permission for an AWS account, use the `ModifyDBSnapshotAttribute` or `ModifyDBClusterSnapshotAttribute` action with `AttributeName` set to `restore` and the `ValuesToRemove` parameter. To mark a manual snapshot as private, remove the value `all` from the values list for the `restore` attribute.

The following example permits two AWS account identifiers, `123451234512` and `123456789012`, to restore the DB snapshot named `manual-snapshot1`, and removes the `all` attribute value to mark the snapshot as private.

```
https://rds.us-west-2.amazonaws.com/
 ?Action=ModifyDBSnapshotAttribute
 &AttributeName=restore
 &DBSnapshotIdentifier=manual-snapshot1
 &SignatureMethod=HmacSHA256&SignatureVersion=4
 &ValuesToAdd.member.1=123451234512
 &ValuesToAdd.member.2=123456789012
 &ValuesToRemove.member.1=all
 &Version=2014-10-31
 &X-Amz-Algorithm=AWS4-HMAC-SHA256
 &X-Amz-Credential=AKIADQKE4SARGYLE/20150922/us-west-2/rds/aws4_request
```

```
&X-Amz-Date=20150922T220515Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=ef38f1ce3dab4e1dbf113d8d2a265c67d17ece1999ffd36be85714ed36dddbb3
```

To list all of the AWS accounts permitted to restore a snapshot, use the `DescribeDBSnapshotAttributes` or `DescribeDBClusterSnapshotAttributes` API action.

Related Topics

- Creating a DB Snapshot (p. 207)
- Copying a DB Snapshot or DB Cluster Snapshot (p. 213)
- Restoring from a DB Snapshot (p. 209)

Restoring a DB Instance to a Specified Time

The Amazon RDS automated backup feature automatically creates a storage volume snapshot of your DB instance, backing up the entire DB instance and not just individual databases. This backup occurs during a daily user-configurable 30 minute period known as the *backup window*. Automated backups are kept for a configurable number of days (called the *backup retention period*). You can restore your DB instance to any specific time during this retention period, creating a new DB instance.

When you restore a DB instance to a point in time, the default DB security group is applied to the new DB instance. If you need custom DB security groups applied to your DB instance, you must apply them explicitly using the AWS Management Console, the Amazon RDS API `ModifyDBInstance` action, or the AWS CLI `modify-db-instance` command once the DB instance is available.

You can restore to any point in time during your backup retention period. To determine the latest restorable time for a DB instance, use the AWS CLI describe-db-instances command and look at the value returned in the **LatestRestorableTime** field for the DB instance. The latest restorable time for a DB instance is typically within 5 minutes of the current time.

The OFFLINE, EMERGENCY, and SINGLE_USER modes are not currently supported. Setting any database into one of these modes will cause the latest restorable time to stop moving ahead for the whole instance.

Several of the database engines used by Amazon RDS have special considerations when restoring from a point in time. When you restore an Oracle DB instance to a point in time, you can specify a different Oracle DB engine, license model, and DBName (SID) to be used by the new DB instance. When you restore a SQL Server DB instance to a point in time, each database within that instance is restored to a point in time within 1 second of each other database within the instance. Transactions that span multiple databases within the instance may be restored inconsistently.

Some actions, such as changing the recovery model of a SQL Server database, can break the sequence of logs that are use for point-in-time recovery. In some cases, Amazon RDS can detect this issue and the latest restorable time is prevented from moving forward; in other cases, such as when a SQL Server database uses the BULK_LOGGED recovery model, the break in log sequence is not detected. It may not be possible to restore a SQL Server DB instance to a point in time if there is a break in the log sequence. For these reasons, Amazon RDS does not support changing the recovery model of SQL Server databases.

AWS Management Console

To restore a DB instance to a specified time

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, click **DB Instances**.
3. Click **Instance Actions**, and then click **Restore To Point In Time**.

   The **Restore DB Instance** window appears.
4. Click on the **Use Custom Restore Time** radio button.
5. Enter the date and time that you wish to restore to in the **Use Custom Restore Time** text boxes.
6. Type the name of the restored DB instance in the **DB Instance Identifier** text box.
7. Click the **Launch DB Instance** button.

CLI

To restore a DB instance to a specified time, use the AWS CLI command `restore-db-instance-to-point-in-time` to create a new database instance.

Example

For Linux, OS X, or Unix:

```
aws rds restore-db-instance-to-point-in-time \
    --source-db-instance-identifier mysourcedbinstance \
    --target-db-instance-identifier mytargetdbinstance \
    --restore-time 2009-10-14T23:45:00.000Z
```

For Windows:

```
aws rds restore-db-instance-to-point-in-time ^
    --source-db-instance-identifier mysourcedbinstance ^
    --target-db-instance-identifier mytargetdbinstance ^
    --restore-time 2009-10-14T23:45:00.000Z
```

API

To restore a DB instance to a specified time, call the Amazon RDS API `RestoreDBInstanceToPointInTime` function with the following parameters:

- `SourceDBInstanceIdentifier = mysourcedbinstance`
- `TargetDBInstanceIdentifier = mytargetdbinstance`
- `RestoreTime = 2013-10-14T23:45:00.000Z`

Example

```
https://rds.us-east-1.amazonaws.com/
   ?Action=RestoreDBInstanceToPointInTime
   &RestoreTime=2013-10-14T23%3A45%3A00.000Z
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &SourceDBInstanceIdentifier=mysourcedbinstance
   &TargetDBInstanceIdentifier=mytargetdbinstance
   &Version=2013-09-09
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20131016/us-east-1/rds/aws4_request
   &X-Amz-Date=20131016T233051Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
   &X-Amz-Signature=087a8eb41cb1ab0fc9ec1575f23e73757ffc6a1e42d7d2b30b9cc0be988cff97
```

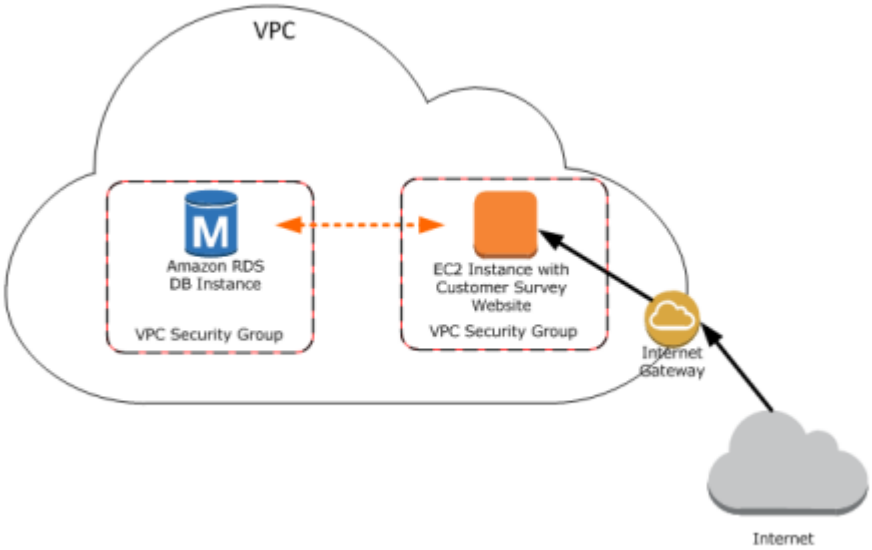
Related Topics

- Creating a DB Snapshot (p. 207)
- Restoring from a DB Snapshot (p. 209)
- Copying a DB Snapshot or DB Cluster Snapshot (p. 213)

Tutorial: Restore a DB Instance from a DB Snapshot

A common scenario when working with Amazon RDS is to have a DB instance that you work with occasionally but that you don't need full time. For example, you might have a quarterly customer survey that uses an Amazon Elastic Compute Cloud (Amazon EC2) instance to host a customer survey website and a DB instance that is used to store the survey results. One way to save money on such a scenario is to take a DB snapshot of the DB instance after the survey is completed, delete the DB instance, and then restore the DB instance when you need to conduct the survey again.

In the following illustration, you can see a possible scenario where an EC2 instance hosting a customer survey website is in the same Amazon Virtual Private Cloud (Amazon VPC) as a DB instance that retains the customer survey data. Note that each instance has its own security group; the EC2 instance security group allows access from the Internet while the DB instance security group allows access only to and from the EC2 instance. When the survey is done, the EC2 instance can be stopped and the DB instance can be deleted after a final DB snapshot is created. When you need to conduct another survey, you can restart the EC2 instance and restore the DB instance from the DB snapshot.



For information about how to set up the needed VPC security groups for this scenario that allows the EC2 instance to connect with the DB instance, see A DB Instance in a VPC Accessed by an EC2 Instance in the Same VPC (p. 392).

You must create a DB snapshot before you can restore a DB instance from one. When you restore the DB instance, you provide the name of the DB snapshot to restore from, and then provide a name for the new DB instance that is created from the restore operation. You cannot restore from a DB snapshot to an existing DB instance; a new DB instance is created when you restore.

Prerequisites for Restoring a DB Instance from a DB Snapshot

Some settings on the restored DB instance are reset when the instance is restored, so you must retain the original resources to be able to restore the DB instance to its previous settings. For example, when you restore a DB instance from a DB snapshot, the default DB parameter and a default security group are

associated with the restored instance. That association means that the default security group does not allow access to the DB instance, and no custom parameter settings are available in the default parameter group. You need to retain the DB parameter group and security group associated with the DB instance that was used to create the DB snapshot.

The following are required before you can restore a DB instance from a DB snapshot:

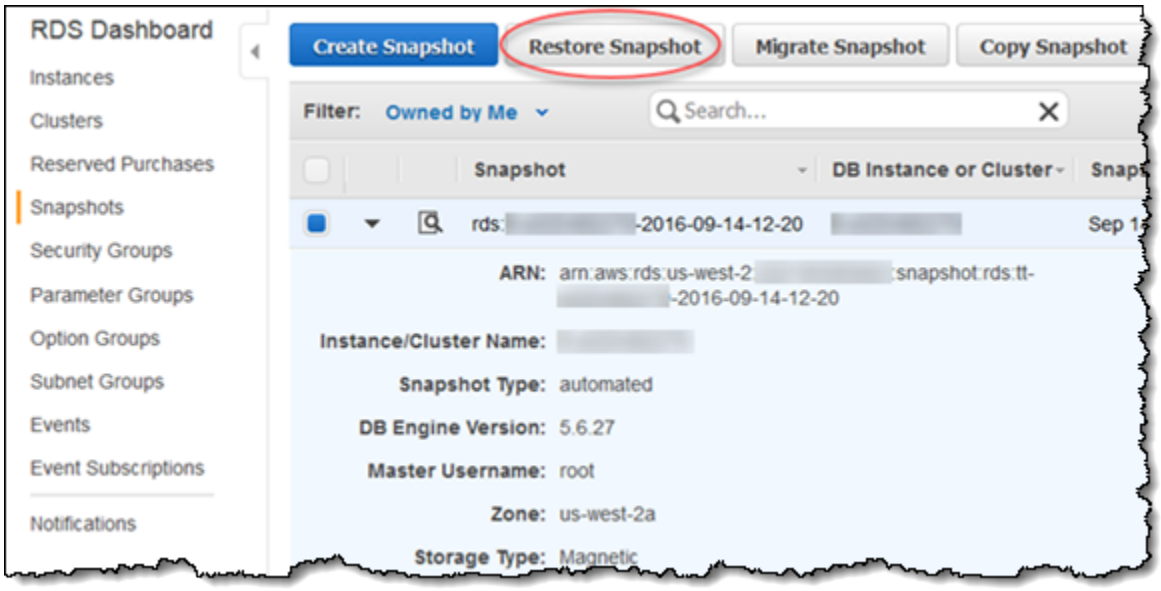
- You must have created a DB snapshot of a DB instance before you can restore a DB instance from that DB snapshot. For more information about creating a DB snapshot, see Creating a DB Snapshot (p. 207).
- You must retain the parameter group and security group associated with the DB instance you created the DB snapshot from.
- You must retain the VPC where the DB instance you made the DB snapshot from was located.
- You need to determine the correct option group for the restored DB instance:
  - The option group associated with the DB snapshot that you restore from is associated with the restored DB instance once it is created. For example, if the DB snapshot you restore from uses Oracle Transparent Data Encryption (TDE), the restored DB instance uses the same option group, which had the TDE option.
  - You cannot use the option group associated with the original DB instance if you attempt to restore that instance into a different VPC or into a different platform. This restriction occurs because when an option group is assigned to a DB instance, it is also linked to the platform that the DB instance is on, either VPC or EC2-Classic (non-VPC). If a DB instance is in a VPC, the option group associated with the instance is linked to that VPC.
  - If you restore a DB instance into a different VPC or onto a different platform, you must either assign the default option group to the instance, assign an option group that is linked to that VPC or platform, or create a new option group and assign it to the DB instance. Note that with persistent or permanent options, such as Oracle TDE, you must create a new option group that includes the persistent or permanent option when restoring a DB instance into a different VPC. For more information about working with option groups, see Working with Option Groups (p. 153).

Restoring a DB Instance from a DB Snapshot

You can use the procedure following to restore from a snapshot in the AWS Management Console.

To restore a DB instance from a DB snapshot

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Snapshots**.
3. Choose the DB snapshot that you want to restore from.



4. Choose **Restore Snapshot**.

   The **Restore DB Instance** window appears.
5. For **DB Instance Identifier**, type the name you want to use for the restored DB instance. If you are restoring from a DB instance that you deleted after you made the DB snapshot, you can use the name of that DB instance.
6. Choose **Restore DB Instance**.

Modifying a Restored DB Instance

As soon as the restore operation is complete, you should associate the custom security group used by the instance you restored from with any applicable custom DB parameter group that you might have. Only the default DB parameter and security groups are associated with the restored instance. If you want to restore the functionality of the DB instance to that of the DB instance that the snapshot was created from, you must modify the DB instance to use the security group and parameter group used by the previous DB instance.

You must apply any changes explicitly using the RDS console's **Modify** command, the `ModifyDBInstance` API, or the `aws rds modify-db-instance` command line tool, once the DB instance is available. We recommend that you retain parameter groups for any DB snapshots you have so that you can associate a restored instance with the correct parameter file.

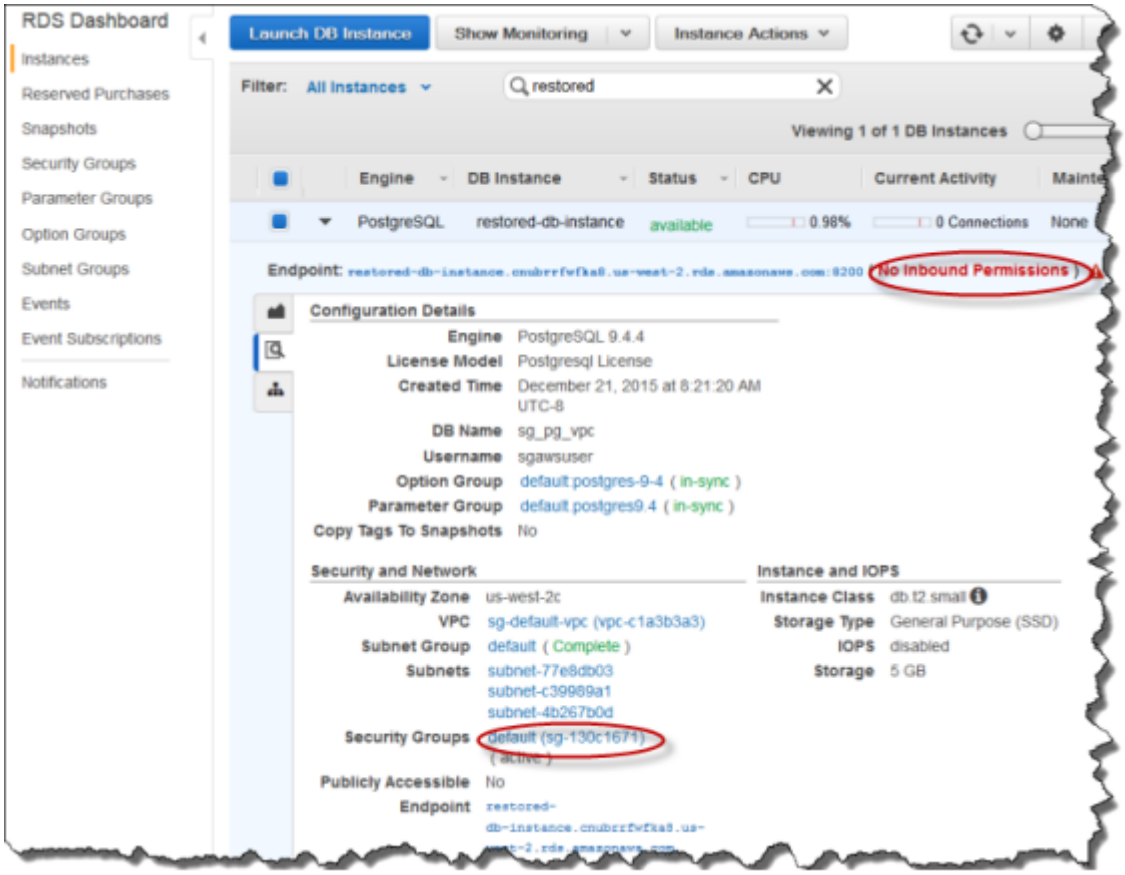
You can modify other settings on the restored DB instance. For example, you can use a different storage type than the source DB snapshot. In this case the restoration process is slower because of the additional work required to migrate the data to the new storage type. In the case of restoring to or from Magnetic (Standard) storage, the migration process is the slowest, because Magnetic storage does not have the IOPS capability of Provisioned IOPS or General Purpose (SSD) storage.

The next steps assume that your DB instance is in a VPC. If your DB instance is not in a VPC, use the AWS Management Console to locate the DB security group you need for the DB instance.

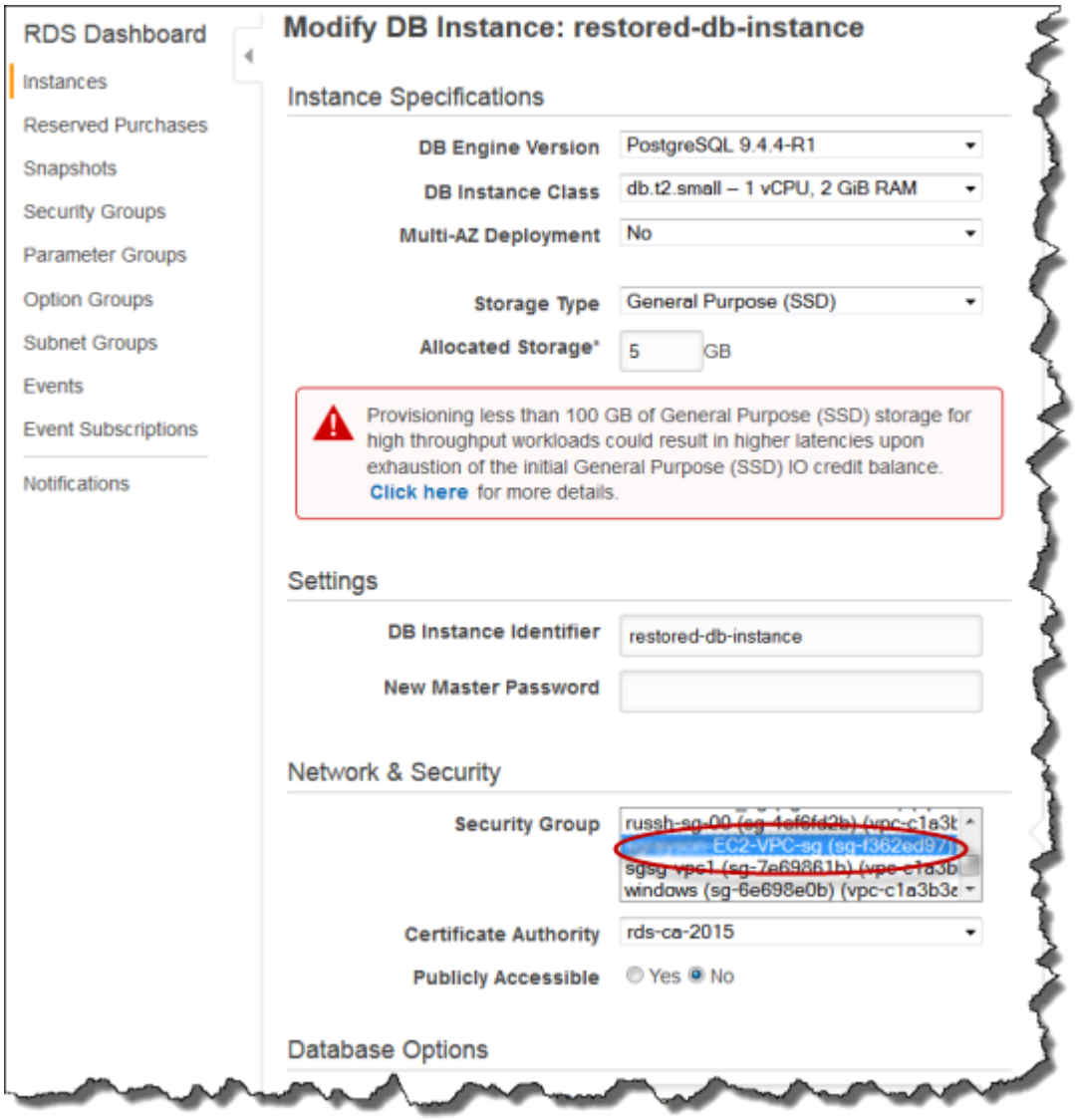
To modify a restored DB instance to have the settings of the original DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.

2. In the navigation pane, choose **Instances**.
3. Select the DB instance created when you restored from the DB snapshot. There are two things to notice here: The security group assigned to the DB instance is the default security group that allows no access, and the warning message shows that there are currently no permissions that allow inbound access.

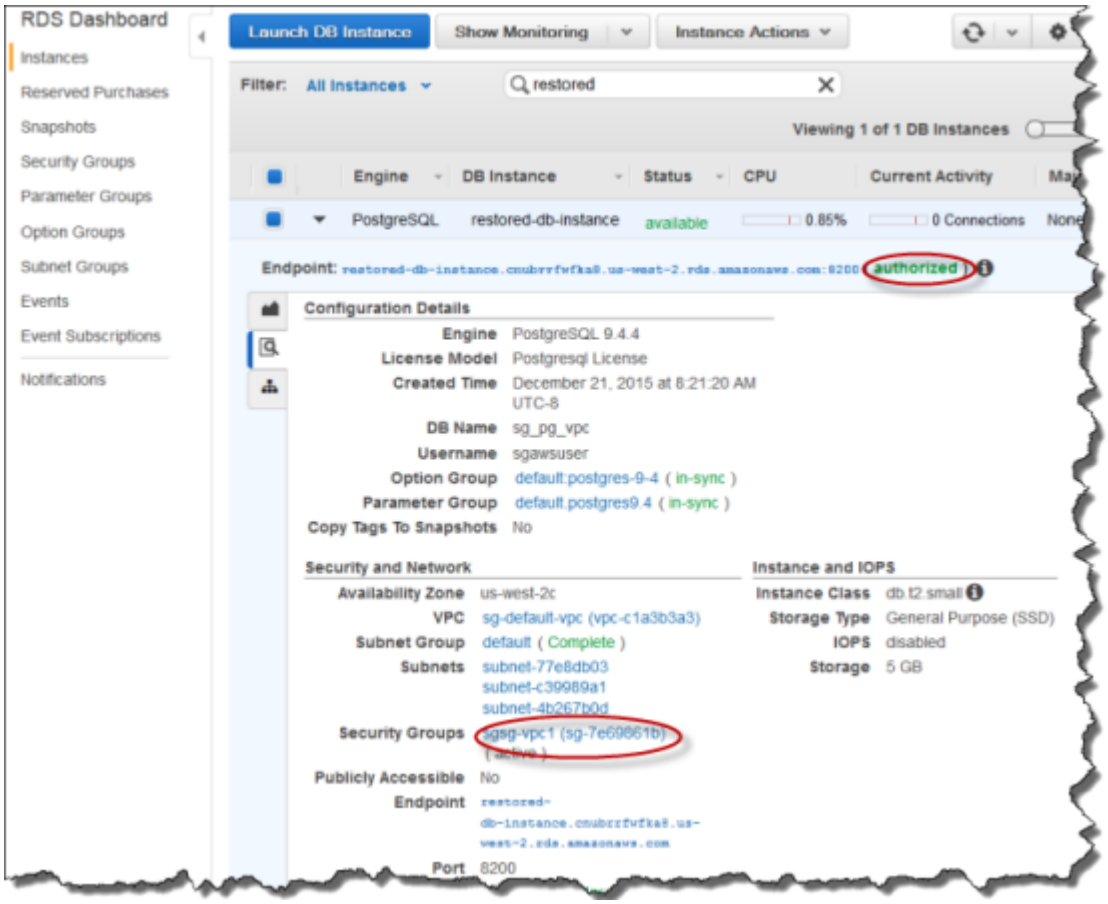


4. Choose **Instance Actions**, and then choose **Modify**.
5. Select the security group that you want to use for your DB instance. If you need to add rules to create a new security group to use with an EC2 instance, see A DB Instance in a VPC Accessed by an EC2 Instance in the Same VPC (p. 392) for more information.



6. Choose **Apply Immediately** (at the bottom of the page).
7. Choose **Continue**, and then choose **Modify DB Instance**.

   Notice that the new security group has been applied, and that the DB instance is now authorized for access.



Related Topics

- Restoring from a DB Snapshot (p. 209)

Monitoring Amazon RDS

Monitoring is an important part of maintaining the reliability, availability, and performance of Amazon RDS and your AWS solutions. You should collect monitoring data from all of the parts of your AWS solution so that you can more easily debug a multi-point failure if one occurs. Before you start monitoring Amazon RDS, we recommend that you create a monitoring plan that includes answers to the following questions:

- What are your monitoring goals?
- What resources will you monitor?
- How often will you monitor these resources?
- What monitoring tools will you use?
- Who will perform the monitoring tasks?
- Who should be notified when something goes wrong?

The next step is to establish a baseline for normal Amazon RDS performance in your environment, by measuring performance at various times and under different load conditions. As you monitor Amazon RDS, you should consider storing historical monitoring data. This stored data will give you a baseline to compare against with current performance data, identify normal performance patterns and performance anomalies, and devise methods to address issues.

For example, with Amazon RDS, you can monitor network throughput, I/O for read, write, and/or metadata operations, client connections, and burst credit balances for your DB instances. When performance falls outside your established baseline, you might need change the instance class of your DB instance or the number of DB instances and Read Replicas that are available for clients in order to optimize your database availability for your workload.

In general, acceptable values for performance metrics depend on what your baseline looks like and what your application is doing. Investigate consistent or trending variances from your baseline. Advice about specific types of metrics follows:

- **High CPU or RAM consumption** – High values for CPU or RAM consumption might be appropriate, provided that they are in keeping with your goals for your application (like throughput or concurrency) and are expected.
- **Disk space consumption** – Investigate disk space consumption if space used is consistently at or above 85 percent of the total disk space. See if it is possible to delete data from the instance or archive data to a different system to free up space.
- **Network traffic** – For network traffic, talk with your system administrator to understand what expected throughput is for your domain network and Internet connection. Investigate network traffic if throughput is consistently lower than expected.
- **Database connections** – Consider constraining database connections if you see high numbers of user connections in conjunction with decreases in instance performance and response time. The best number of user connections for your DB instance will vary based on your instance class and the complexity of the operations being performed. You can determine the number of database connections by associating your DB instance with a parameter group where the `User Connections` parameter is set to a value other than 0 (unlimited). You can either use an existing parameter group or create a new one. For more information, see Working with DB Parameter Groups (p. 170).
- **IOPS metrics** – The expected values for IOPS metrics depend on disk specification and server configuration, so use your baseline to know what is typical. Investigate if values are consistently different than your baseline. For best IOPS performance, make sure your typical working set will fit into memory to minimize read and write operations.

Monitoring Tools

AWS provides various tools that you can use to monitor Amazon RDS. You can configure some of these tools to do the monitoring for you, while some of the tools require manual intervention. We recommend that you automate monitoring tasks as much as possible.

Automated Monitoring Tools

You can use the following automated monitoring tools to watch Amazon RDS and report when something is wrong:

- **Amazon CloudWatch Alarms** – Watch a single metric over a time period that you specify, and perform one or more actions based on the value of the metric relative to a given threshold over a number of time periods. The action is a notification sent to an Amazon Simple Notification Service (Amazon SNS) topic or Auto Scaling policy. CloudWatch alarms do not invoke actions simply because they are in a particular state; the state must have changed and been maintained for a specified number of periods. For more information, see Monitoring with Amazon CloudWatch (p. 247).
- **Amazon CloudWatch Logs** – Monitor, store, and access your log files from AWS CloudTrail or other sources. For more information, see Monitoring Log Files in the *Amazon CloudWatch User Guide*.
- **Amazon RDS Enhanced Monitoring** — provides metrics in real time for the operating system that your DB instance or DB cluster runs on. For more information, see Enhanced Monitoring (p. 258).
- **Amazon CloudWatch Events** – Match events and route them to one or more target functions or streams to make changes, capture state information, and take corrective action. For more information, see What is Amazon CloudWatch Events in the *Amazon CloudWatch User Guide*.
- **AWS CloudTrail Log Monitoring** – Share log files between accounts, monitor CloudTrail log files in real time by sending them to CloudWatch Logs, write log processing applications in Java, and validate that your log files have not changed after delivery by CloudTrail. For more information, see Working with CloudTrail Log Files in the *AWS CloudTrail User Guide*.

  For information on using AWS CloudTrail Log Monitoring with Amazon RDS, see Logging Amazon RDS API Calls Using AWS CloudTrail (p. 324).
- **Amazon RDS Events** – Subscribe to Amazon RDS events to be notified when changes occur with a DB instance, DB cluster, DB snapshot, DB cluster snapshot, DB parameter group, or DB security group. For more information, see Using Amazon RDS Event Notification (p. 279).
- **Database log files** – View, download, or watch database log files using the Amazon RDS console or Amazon RDS APIs. You can also query some database log files that are loaded into database tables. For more information, see Amazon RDS Database Log Files (p. 303).

Manual Monitoring Tools

Another important part of monitoring Amazon RDS involves manually monitoring those items that the CloudWatch alarms don't cover. The Amazon RDS, CloudWatch, AWS Trusted Advisor and other AWS console dashboards provide an at-a-glance view of the state of your AWS environment. We recommend that you also check the log files on your DB instance.

- From the Amazon RDS console, you can monitor the following items for your resources:
  - The number of connections to a DB instance
  - The amount of read and write operations to a DB instance
  - The amount of storage that a DB instance is currently utilizing
  - The amount of memory and CPU being utilized for a DB instance
  - The amount of network traffic to and from a DB instance

- From the AWS Trusted Advisor dashboard, you can review the following cost optimization, security, fault tolerance, and performance improvement checks:
  - Amazon RDS Idle DB Instances
  - Amazon RDS Security Group Access Risk
  - Amazon RDS Backups
  - Amazon RDS Multi-AZ
  - Amazon Aurora DB Instance Accessibility

  For more information on these checks, see Trusted Advisor Best Practices (Checks).
- CloudWatch home page shows:
  - Current alarms and status
  - Graphs of alarms and resources
  - Service health status

  In addition, you can use CloudWatch to do the following:
  - Create customized dashboards to monitor the services you care about
  - Graph metric data to troubleshoot issues and discover trends
  - Search and browse all your AWS resource metrics
  - Create and edit alarms to be notified of problems

Monitoring with Amazon CloudWatch

You can monitor DB instance using CloudWatch, which collects and processes raw data from Amazon RDS into readable, near real-time metrics. These statistics are recorded for a period of two weeks, so that you can access historical information and gain a better perspective on how your web application or service is performing. By default, Amazon RDS metric data is automatically sent to Amazon CloudWatch in 1-minute periods. For more information about Amazon CloudWatch, see What Are Amazon CloudWatch, Amazon CloudWatch Events, and Amazon CloudWatch Logs? in the *Amazon CloudWatch User Guide*.

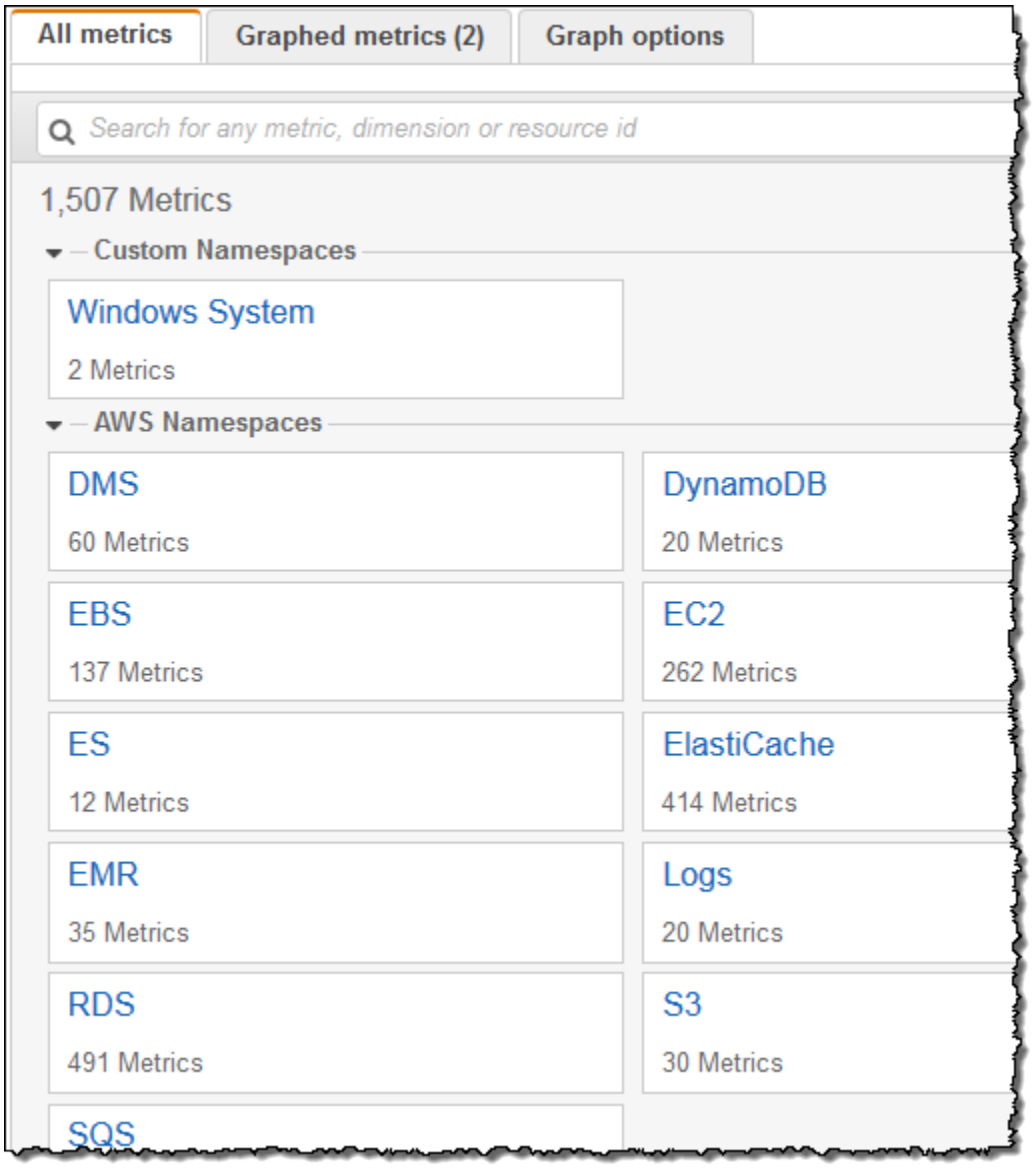
Amazon RDS Metrics and Dimensions

When you use Amazon RDS resources, Amazon RDS sends metrics and dimensions to Amazon CloudWatch every minute. You can use the following procedures to view the metrics for Amazon RDS.

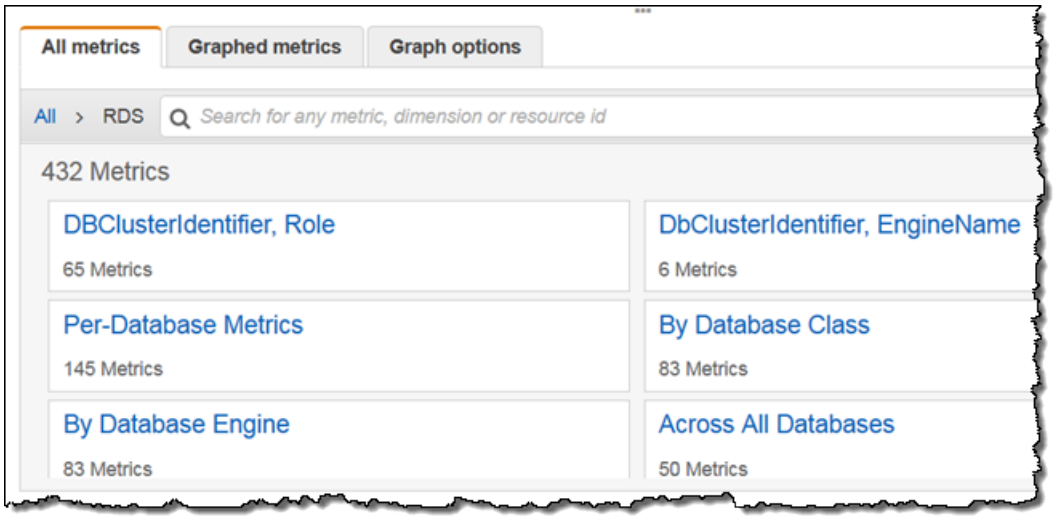
To view metrics using the Amazon CloudWatch console

Metrics are grouped first by the service namespace, and then by the various dimension combinations within each namespace.

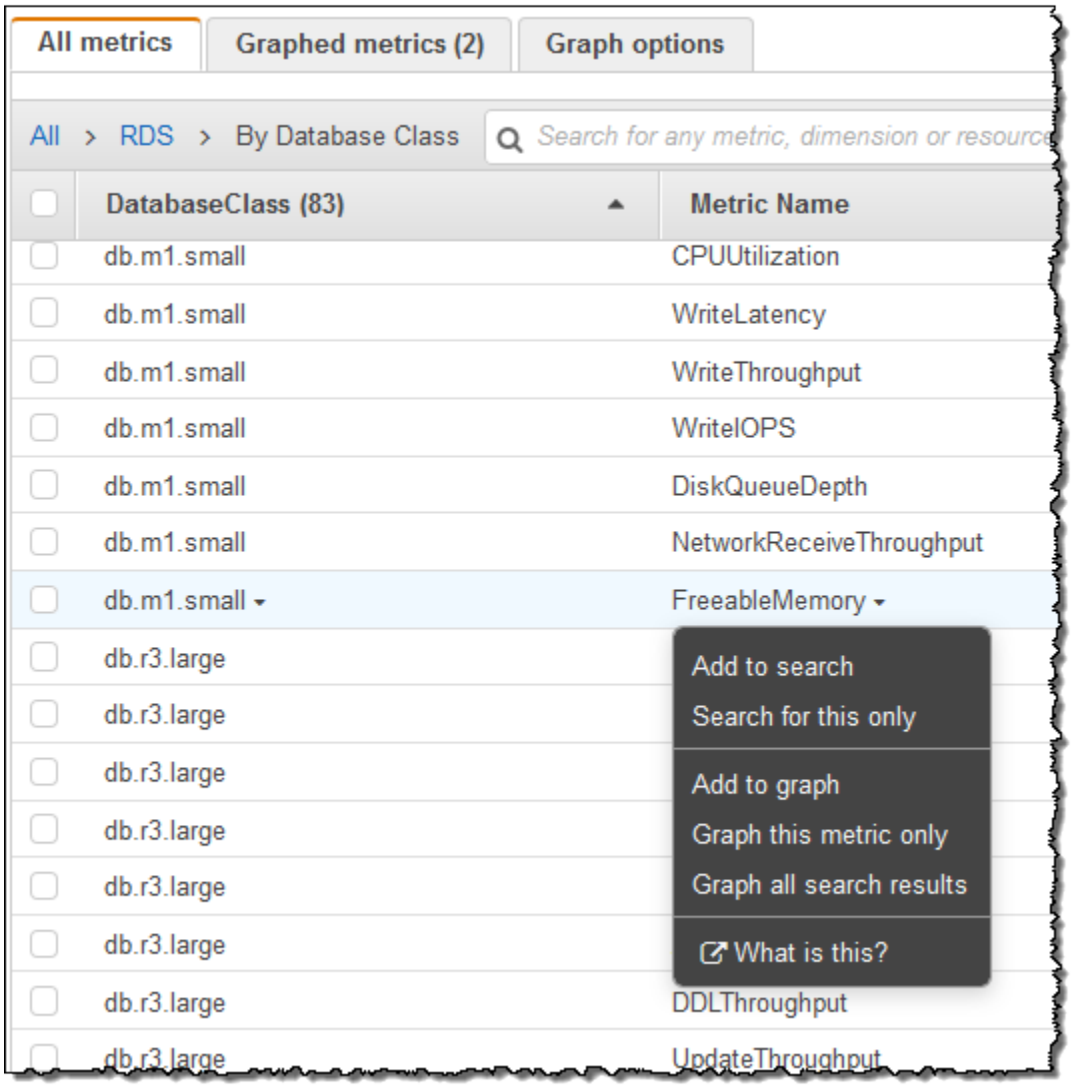
1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. If necessary, change the region. From the navigation bar, select the region where your AWS resources reside. For more information, see Regions and Endpoints.
3. In the navigation pane, choose **Metrics**. Choose the **RDS** metric namespace.



4. Select a metric dimension, for example, **By Database Class**.



5. To sort the metrics, use the column heading. To graph a metric, select the check box next to the metric. To filter by resource, choose the resource ID and then choose **Add to search**. To filter by metric, choose the metric name and then choose **Add to search**.



To view metrics using the AWS CLI

- At a command prompt, use the following command:

```
aws cloudwatch list-metrics --namespace AWS/RDS
```

Amazon RDS Metrics

The `AWS/RDS` namespace includes the following metrics.

Metric	Description
`BinLogDiskUsage`	The amount of disk space occupied by binary logs on the master. Applies to MySQL read replicas. Units: Bytes

Metric	Description
`BurstBalance`	The percent of General Purpose SSD (gp2) burst-bucket I/O credits available. Units: Percent
`CPUUtilization`	The percentage of CPU utilization. Units: Percent
`CPUCreditUsage`	[T2 instances] The number of CPU credits used by the instance for CPU utilization. One CPU credit equals one vCPU running at 100% utilization for one minute or an equivalent combination of vCPUs, utilization, and time (for example, one vCPU running at 50% utilization for two minutes or two vCPUs running at 25% utilization for two minutes). CPU credit metrics are available at a five-minute frequency only. If you specify a period greater than five minutes, use the `Sum` statistic instead of the `Average` statistic. Units: Credits (vCPU-minutes)
`CPUCreditBalance`	[T2 instances] The number of earned CPU credits accumulated since the instance was launched, less the credits used, up to a maximum number based on the instance size. Credits are stored in the credit balance after they are earned, and removed from the credit balance when they are used. The credit balance has a maximum limit, determined by the instance size. If the credit balance has reached the limit, additional earned credits are not added to the balance. The credits in the `CPUCreditBalance` are available for the instance to use to burst beyond its baseline CPU utilization. Credits on a running instance do not expire. However, if you stop an instance, it loses all the credits in the credit balance. CPU credit metrics are available at a five-minute frequency only. Units: Credits (vCPU-minutes)
`CPUSurplusCreditBalance`	[T2 instances] The number of surplus credits that have been used by a T2 Unlimited instance when its `CPUCreditBalance` is zero. The `CPUSurplusCreditBalance` is paid down by earned CPU credits. Units: Credits (vCPU-minutes)
`CPUSurplusCreditsCharged`	[T2 instances] The number of surplus credits that have been used by a T2 Unlimited instance that are not offset by earned CPU credits. `CPUSurplusCreditsCharged` tracks the surplus credits that incur an additional charge, and represents the difference between `CPUSurplusCreditBalance` and `CPUCreditBalance`. Units: Credits (vCPU-minutes)

Metric	Description
DatabaseConnections	The number of database connections in use. Units: Count
DiskQueueDepth	The number of outstanding IOs (read/write requests) waiting to access the disk. Units: Count
FreeableMemory	The amount of available random access memory. Units: Bytes
FreeStorageSpace	The amount of available storage space. Units: Bytes
MaximumUsedTransactionIDs	The maximum transaction ID that has been used. Applies to PostgreSQL. Units: Count
NetworkReceiveThroughput	The incoming (Receive) network traffic on the DB instance, including both customer database traffic and Amazon RDS traffic used for monitoring and replication. Units: Bytes/second
NetworkTransmitThroughput	The outgoing (Transmit) network traffic on the DB instance, including both customer database traffic and Amazon RDS traffic used for monitoring and replication. Units: Bytes/second
OldestReplicationSlotLag	The lagging size of the replica lagging the most in terms of WAL data received. Applies to PostgreSQL. Units: Megabytes
ReadIOPS	The average number of disk I/O operations per second during the polling period. Units: Count/Second
ReadLatency	The average amount of time taken per disk I/O operation. Units: Seconds
ReadThroughput	The average number of bytes read from disk per second. Units: Bytes/Second
ReplicaLag	The amount of time a Read Replica DB instance lags behind the source DB instance. Applies to MySQL, MariaDB, and PostgreSQL Read Replicas. Units: Seconds

Metric	Description
`ReplicationSlotDiskUsage`	The disk space used by replication slot files. Applies to PostgreSQL. Units: Megabytes
`SwapUsage`	The amount of swap space used on the DB instance. Units: Bytes
`TransactionLogsDiskUsage`	The disk space used by transaction logs. Applies to PostgreSQL. Units: Megabytes
`TransactionLogsGeneration`	The size of transaction logs generated per second. Applies to PostgreSQL. Units: Megabytes/second
`WriteIOPS`	The average number of disk I/O operations per second. Units: Count/Second
`WriteLatency`	The average amount of time taken per disk I/O operation. Units: Seconds
`WriteThroughput`	The average number of bytes written to disk per second. Units: Bytes/Second

Amazon RDS Dimensions

Amazon RDS metrics data can be filtered by using any of the dimensions in the following table:

Dimension	Description
`DBInstanceIdentifier`	This dimension filters the data you request for a specific DB instance.
`DBClusterIdentifier`	This dimension filters the data you request for a specific Amazon Aurora DB cluster.
`DBClusterIdentifier, Role`	This dimension filters the data you request for a specific Amazon Aurora DB cluster, aggregating the metric by instance role (WRITER/READER). For example, you can aggregate metrics for all READER instances that belong to a cluster.
`DatabaseClass`	This dimension filters the data you request for all instances in a database class. For example, you can aggregate metrics for all instances that belong to the database class `db.m1.small`
`EngineName`	This dimension filters the data you request for the identified engine name only. For example, you can aggregate metrics for all instances that have the engine name `mysql`.

Creating CloudWatch Alarms to Monitor Amazon RDS

You can create a CloudWatch alarm that sends an Amazon SNS message when the alarm changes state. An alarm watches a single metric over a time period you specify, and performs one or more actions based on the value of the metric relative to a given threshold over a number of time periods. The action is a notification sent to an Amazon SNS topic or Auto Scaling policy.

Alarms invoke actions for sustained state changes only. CloudWatch alarms will not invoke actions simply because they are in a particular state, the state must have changed and been maintained for a specified number of periods. The following procedures outlines how to create alarms for Amazon RDS.

To set alarms using the CloudWatch console

1. Sign in to the AWS Management Console and open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. Choose **Alarms** and then choose **Create Alarm**. This launches the **Create Alarm Wizard**.
3. Choose **RDS Metrics** and scroll through the Amazon RDS metrics to locate the metric you want to place an alarm on. To display just the Amazon RDS metrics in this dialog box, search for the identifier of your resource. Select the metric to create an alarm on and then choose **Next**.
4. Fill in the **Name**, **Description**, **Whenever** values for the metric.
5. If you want CloudWatch to send you an email when the alarm state is reached, in the **Whenever this alarm:** field, choose **State is ALARM**. In the **Send notification to:** field, choose an existing SNS topic. If you select **Create topic**, you can set the name and email addresses for a new email subscription list. This list is saved and appears in the field for future alarms.

   Note
   If you use **Create topic** to create a new Amazon SNS topic, the email addresses must be verified before they receive notifications. Emails are only sent when the alarm enters an alarm state. If this alarm state change happens before the email addresses are verified, they do not receive a notification.
6. At this point, the **Alarm Preview** area gives you a chance to preview the alarm you're about to create. Choose **Create Alarm**.

To set an alarm using the AWS CLI

- Call `put-metric-alarm`. For more information, see *AWS Command Line Interface Reference*.

To set an alarm using the CloudWatch API

- Call `PutMetricAlarm`. For more information, see *Amazon CloudWatch API Reference*

Viewing DB Instance Metrics

Amazon RDS provides metrics so that you can monitor the health of your DB instances and DB clusters. You can monitor both DB instance metrics and operating system (OS) metrics.

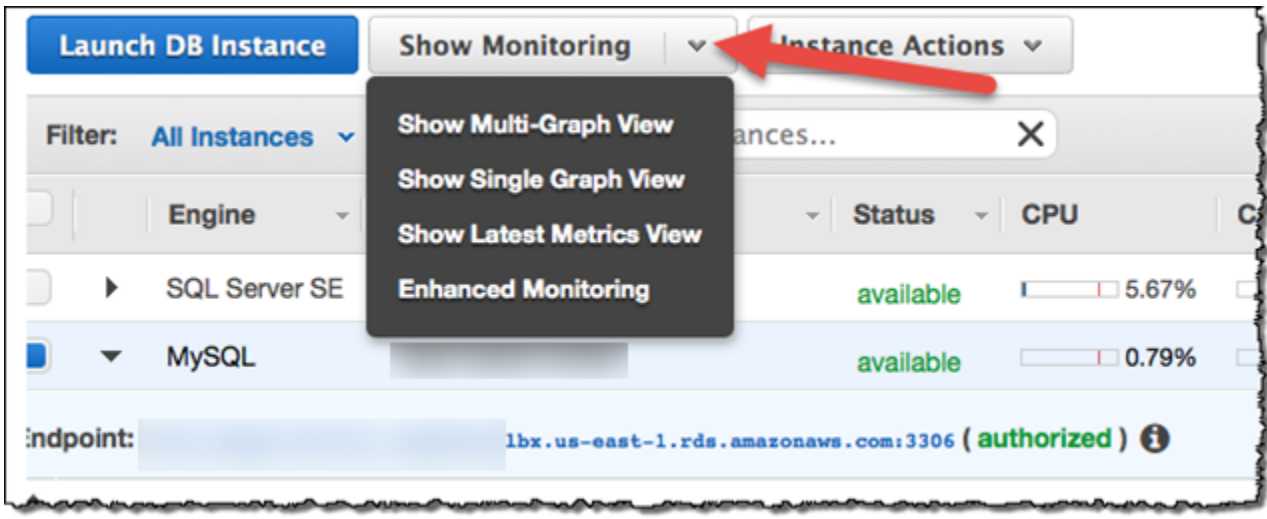
This section provides details on how you can view metrics for your DB instance using the RDS console and CloudWatch. For information on monitoring metrics for the operating system of your DB instance in real time using CloudWatch Logs, see Enhanced Monitoring (p. 258).

Viewing Metrics by Using the Console

To view DB and OS metrics for a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **DB Instances**.
3. Select the check box to the left of the DB cluster you need information about. For **Show Monitoring**, choose the option for how you want to view your metrics from these:

   - **Show Multi-Graph View** – Shows a summary of DB instance metrics available from Amazon CloudWatch. Each metric includes a graph showing the metric monitored over a specific time span.
   - **Show Single Graph View** – Shows a single metric at a time with more detail. Each metric includes a graph showing the metric monitored over a specific time span.
   - **Show Latest Metrics View** – Shows a summary of DB instance metrics without graphs. **Full Monitoring View** includes an option for full-screen viewing.
   - **Enhanced Monitoring** – Shows a summary of OS metrics available for a DB instance with Enhanced Monitoring enabled. Each metric includes a graph showing the metric monitored over a specific time span.



Tip
To select the time range of the metrics represented by the graphs, use **Time Range**.
You can choose any graph to bring up a more detailed view of the graph, using which you can apply metric-specific filters to the metric data.
Time Range is not available for the Enhanced Monitoring Dashboard.

DB Instance Metrics

Amazon RDS integrates with CloudWatch metrics to provide a variety of DB instance metrics. You can view CloudWatch metrics using the RDS console, AWS CLI, or API.

For a complete list of Amazon RDS metrics, go to Amazon RDS Dimensions and Metrics in the *Amazon CloudWatch User Guide*.

Viewing DB Metrics by Using the CloudWatch CLI

Note
The following CLI example requires the CloudWatch command line tools. For more information on CloudWatch and to download the developer tools, go to the Amazon CloudWatch product page. Note that the `StartTime` and `EndTime` values supplied in this example are for illustrative purposes. You must substitute appropriate start and end time values for your DB instance.

To view usage and performance statistics for a DB instance

- Use the CloudWatch command `mon-get-stats` with the following parameters:

```
PROMPT>mon-get-stats FreeStorageSpace --dimensions="DBInstanceIdentifier=mydbinstance"
 --statistics= Average
  --namespace="AWS/RDS" --start-time 2009-10-16T00:00:00 --end-time 2009-10-16T00:02:00
```

Viewing DB Metrics by Using the CloudWatch API

Note that the `StartTime` and `EndTime` values supplied in this example are for illustrative purposes. You must substitute appropriate start and end time values for your DB instance.

To view usage and performance statistics for a DB instance

- Call the CloudWatch API `GetMetricStatistics` with the following parameters:
  - `Statistics.member.1` = `Average`
  - `Namespace` = `AWS/RDS`
  - `StartTime` = `2009-10-16T00:00:00`
  - `EndTime` = `2009-10-16T00:02:00`
  - `Period` = `60`
  - `MeasureName` = `FreeStorageSpace`

  Example

```
http://monitoring.amazonaws.com/
 ?SignatureVersion=2
 &Action=GetMetricStatistics
 &Version=2009-05-15
 &StartTime=2009-10-16T00:00:00
 &EndTime=2009-10-16T00:02:00
 &Period=60
 &Statistics.member.1=Average
 &Dimensions.member.1="DBInstanceIdentifier=mydbinstance"
 &Namespace=AWS/RDS
 &MeasureName=FreeStorageSpace
 &Timestamp=2009-10-15T17%3A48%3A21.746Z
 &AWSAccessKeyId=<AWS Access Key ID>
 &Signature=<Signature>
```

Related Topics

- Using Amazon RDS Event Notification (p. 279)
- Viewing Amazon RDS Events (p. 301)
- Amazon RDS Database Log Files (p. 303)
- Logging Amazon RDS API Calls Using AWS CloudTrail (p. 324)
- What Is Amazon Relational Database Service (Amazon RDS)? (p. 1)

Enhanced Monitoring

Amazon RDS provides metrics in real time for the operating system (OS) that your DB instance runs on. You can view the metrics for your DB instance using the console, or consume the Enhanced Monitoring JSON output from CloudWatch Logs in a monitoring system of your choice.

The cost for using Enhanced Monitoring varies depends on several factors:

- You are only charged for Enhanced Monitoring that exceeds the free tier provided by Amazon CloudWatch Logs.

  For more information about pricing, see Amazon CloudWatch Pricing.
- A smaller monitoring interval results in more frequent reporting of OS metrics and increases your monitoring cost.
- Usage costs for Enhanced Monitoring are applied for each DB instance that Enhanced Monitoring is enabled for. Monitoring a large number of DB instances is more expensive than monitoring only a few.
- DB instances that support a more compute-intensive workload have more OS process activity to report and higher costs for Enhanced Monitoring.

Enhanced Monitoring Availability

- Enhanced Monitoring is available for the following database engines:
  - Amazon Aurora
  - MariaDB
  - Microsoft SQL Server
  - MySQL version 5.5 or later
  - Oracle
  - PostgreSQL
- Enhanced monitoring is available for all DB instance classes except for `db.m1.small`.

Differences Between CloudWatch and Enhanced Monitoring Metrics

CloudWatch gathers metrics about CPU utilization from the hypervisor for a DB instance, and Enhanced Monitoring gathers its metrics from an agent on the instance. As a result, you might find differences between the measurements, because the hypervisor layer performs a small amount of work. The differences can be greater if your DB instances use smaller instance classes, because then there are likely more virtual machines (VMs) that are managed by the hypervisor layer on a single physical instance. Enhanced Monitoring metrics are useful when you want to see how different processes or threads on a DB instance use the CPU.

Setting Up for and Enabling Enhanced Monitoring

Before You Begin

Enhanced Monitoring requires permission to act on your behalf to send OS metric information to CloudWatch Logs. You grant Enhanced Monitoring the required permissions using an AWS Identity and Access Management (IAM) role.

The first time that you enable Enhanced Monitoring in the console, you can select the **Default** option for the **Monitoring Role** property to have RDS create the required IAM role. RDS then automatically creates a role named `rds-monitoring-role` for you, and uses it for the specified DB instance or Read Replica.

You can also create the required role before you enable Enhanced Monitoring, and then specify your new role's name when you enable Enhanced Monitoring. You must create this required role if you enable Enhanced Monitoring using the AWS CLI or the RDS API.

To create the appropriate IAM role to permit Amazon RDS to communicate with the Amazon CloudWatch Logs service on your behalf, take the following steps.

To create an IAM role for Amazon RDS Enhanced Monitoring

1. Open the IAM Console at https://console.aws.amazon.com.
2. In the navigation pane, choose **Roles**.
3. Choose **Create New Role**.
4. Choose the **AWS service** tab and then choose **RDS** from the list of services.
5. Choose **RDS Role for Enhanced Monitoring**, and then choose **Next: Permissions**.
6. On the **Attached permissions policy** page, choose **AmazonRDSEnhancedMonitoringRole**, and then choose **Next: Review**.
7. For **Role Name**, type a name for your role, for example **`emaccess`**, and then choose **Create role**.

Enabling and Disabling Enhanced Monitoring

You can enable Enhanced Monitoring when you create a DB instance or Read Replica, or when you modify a DB instance. If you modify a DB instance to enable Enhanced Monitoring, you do not need to reboot your DB instance for the change to take effect.

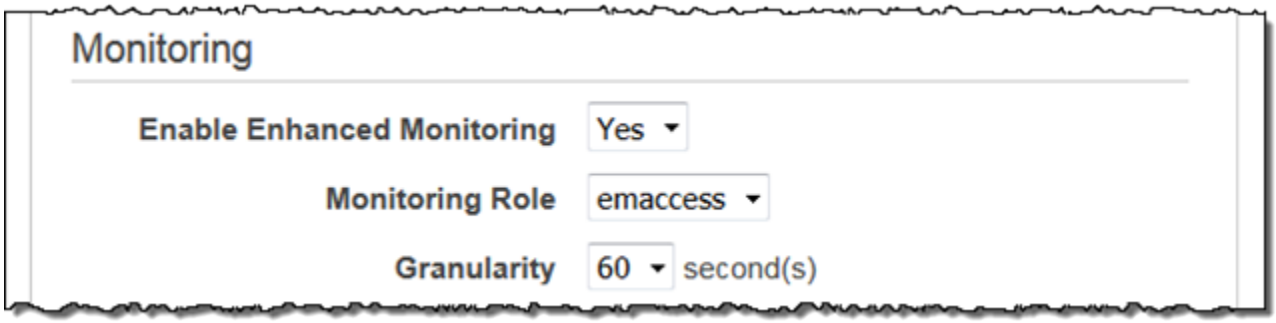
You can enable Enhanced Monitoring in the RDS console when you do one of the following actions:

- **Launch a DB Instance** – You can enable Enhanced Monitoring in the **Configure Advanced Settings** page.
- **Create Read Replica** – You can enable Enhanced Monitoring in the **Configure Advanced Settings** page.
- **Modify a DB Instance** – You can enable Enhanced Monitoring in the **Modify DB Instance** page.

To enable Enhanced Monitoring by using the RDS console, scroll to the **Monitoring** section and do the following:

1. Set the **Enable Enhanced Monitoring** property for your DB instance or Read Replica to **Yes**.
2. Set the **Monitoring Role** property to the IAM role that you created to permit Amazon RDS to communicate with Amazon CloudWatch Logs for you, or choose **Default** to have RDS create a role for you named `rds-monitoring-role`.
3. Set the **Granularity** property to the interval, in seconds, between points when metrics are collected for your DB instance or Read Replica. The **Granularity** property can be set to one of the following values: `1`, `5`, `10`, `15`, `30`, or `60`.

To disable Enhanced Monitoring, set the **Enable Enhanced Monitoring** property to **No**.



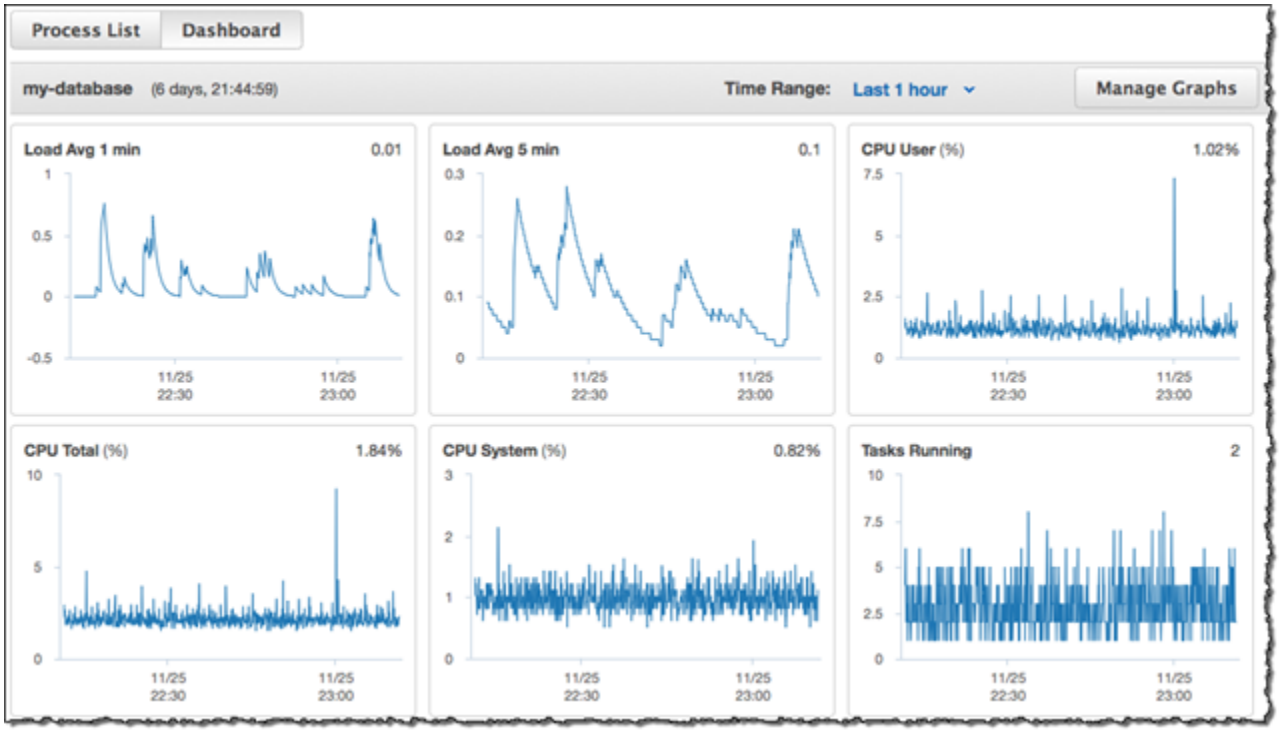
Enabling Enhanced Monitoring does not require your DB instance to restart.

Note
The fastest that the RDS console refreshes is every 5 seconds. If you set the granularity to 1 second in the RDS console, you still see updated metrics only every 5 seconds. You can retrieve 1 second metric updates by using CloudWatch Logs.

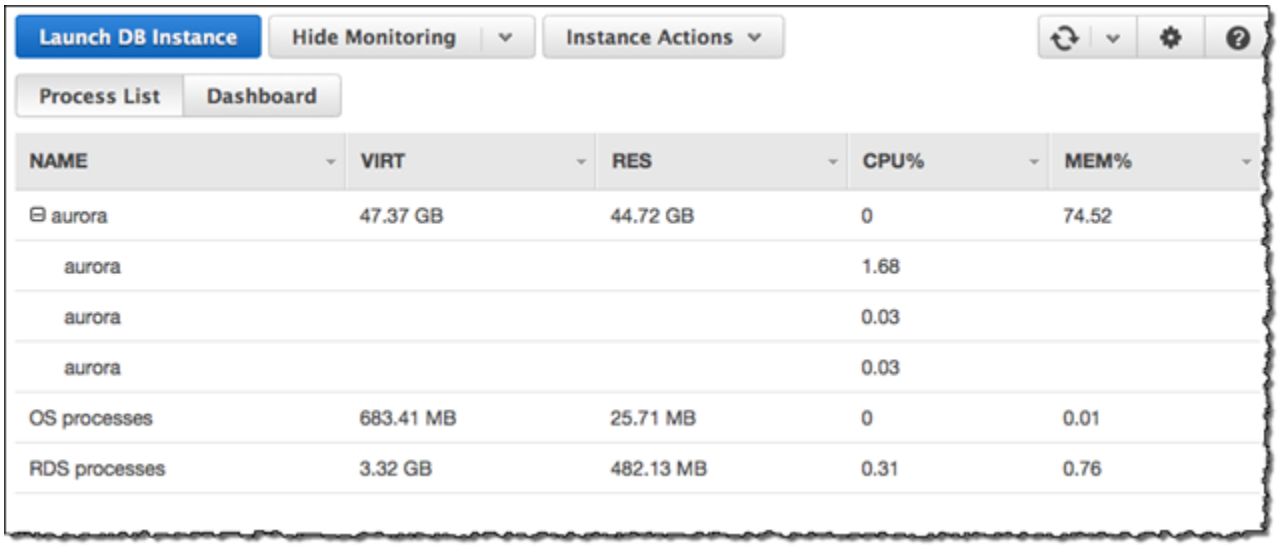
Viewing Enhanced Monitoring

You can view OS metrics reported by Enhanced Monitoring in the RDS console by choosing the **Enhanced Monitoring Dashboard** view for **Show Monitoring**. Two views are available: **Dashboard** view, which shows graphs of the OS metrics, and **Process List** view, which shows the processes running on the DB instance and their related metrics including CPU percentage, memory usage, and so on.

Dashboard view is shown following.



Process List view is shown following.



The Enhanced Monitoring metrics shown in the Process List view are organized as follows:

- **RDS child processes** – Shows a summary of the RDS processes that support the DB instance, for example `aurora` for Amazon Aurora DB clusters and `mysqld` for MySQL DB instances. Process threads appear nested beneath the parent process. Process threads show CPU utilization only as other metrics are the same for all threads for the process. The console displays a maximum of 100 processes and threads. The results are a combination of the top CPU consuming and memory consuming processes and threads. If there are more than 50 processes and more than 50 threads, the console displays the top 50 consumers in each category. This display helps you identify which processes are having the greatest impact on performance.
- **RDS processes** – Shows a summary of the resources used by the RDS management agent, diagnostics monitoring processes, and other AWS processes that are required to support RDS DB instances.
- **OS processes** – Shows a summary of the kernel and system processes, which generally have minimal impact on performance.

The items listed for each process are:

- **VIRT** – Displays the virtual size of the process.
- **RES** – Displays the actual physical memory being used by the process.
- **CPU%** – Displays the percentage of the CPU bandwith consumed by the process.
- **MEM%** – Displays the percentage of the total memory consumed by the process.

The monitoring data that is shown in the RDS console is retrieved from Amazon CloudWatch Logs. You can also retrieve the metrics for a DB instance as a log stream from CloudWatch Logs. For more information, see Viewing Enhanced Monitoring by Using CloudWatch Logs (p. 262).

Enhanced Monitoring metrics are not returned during the following:

- A failover of the DB instance.
- Changing the instance class of the DB instance (scale compute).

Enhanced Monitoring metrics are returned during a reboot of a DB instance because only the database engine is rebooted. Metrics for the operating system are still reported.

Viewing Enhanced Monitoring by Using CloudWatch Logs

After you have enabled Enhanced Monitoring for your DB instance, you can view the metrics for your DB instance using CloudWatch Logs, with each log stream representing a single DB instance being monitored. The log stream identifier is the resource identifier (`DbiResourceId`) for the DB instance.

To view Enhanced Monitoring log data

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. If necessary, choose the region that your DB instance is in. For more information, go to Regions and Endpoints in the *Amazon Web Services General Reference*.
3. Choose **Logs** in the navigation pane.
4. Choose **RDSOSMetrics** from the list of log groups.
5. Choose the log stream that you want to view from the list of log streams.

Available OS Metrics

The following tables list the OS metrics available using Amazon CloudWatch Logs.

Metrics for Amazon Aurora, MariaDB, MySQL, Oracle, and PostgreSQL DB instances

Group	Metrics	Description
`General`	`engine`	The database engine for the DB instance.
	`instanceID`	The DB instance identifier.
	`instanceResourceID`	A region-unique, immutable identifier for the DB instance, also used as the log stream identifier.
	`numVCPUs`	The number of virtual CPUs for the DB instance.
	`timestamp`	The time at which the metrics were taken.
	`uptime`	The amount of time that the DB instance has been active.
	`version`	The version of the OS metrics' stream JSON format.
`cpuUtilization`	`guest`	The percentage of CPU in use by guest programs.
	`idle`	The percentage of CPU that is idle.
	`irq`	The percentage of CPU in use by software interrupts.
	`nice`	The percentage of CPU in use by programs running at lowest priority.
	`steal`	The percentage of CPU in use by other virtual machines.
	`system`	The percentage of CPU in use by the kernel.
	`total`	The total percentage of the CPU in use. This value includes the `nice` value.

Group	Metrics	Description
	user	The percentage of CPU in use by user programs.
	wait	The percentage of CPU unused while waiting for I/O access.
diskIO	avgQueueLen	The number of requests waiting in the I/O device's queue. This metric is not available for Amazon Aurora.
	avgReqSz	The average request size, in kilobytes. This metric is not available for Amazon Aurora.
	await	The number of milliseconds required to respond to requests, including queue time and service time. This metric is not available for Amazon Aurora.
	device	The identifier of the disk device in use. This metric is not available for Amazon Aurora.
	readIOsPS	The number of read operations per second. This metric is not available for Amazon Aurora.
	readKb	The total number of kilobytes read. This metric is not available for Amazon Aurora.
	readKbPS	The number of kilobytes read per second. This metric is not available for Amazon Aurora.
	rrqmPS	The number of merged read requests queued per second. This metric is not available for Amazon Aurora.
	tps	The number of I/O transactions per second. This metric is not available for Amazon Aurora.
	util	The percentage of CPU time during which requests were issued. This metric is not available for Amazon Aurora.
	writeIOsPS	The number of write operations per second. This metric is not available for Amazon Aurora.
	writeKb	The total number of kilobytes written. This metric is not available for Amazon Aurora.
	writeKbPS	The number of kilobytes written per second. This metric is not available for Amazon Aurora.

Group	Metrics	Description
	wrqmPS	The number of merged write requests queued per second. This metric is not available for Amazon Aurora.
fileSys	maxFiles	The maximum number of files that can be created for the file system.
	mountPoint	The path to the file system.
	name	The name of the file system.
	total	The total number of disk space available for the file system, in kilobytes.
	used	The amount of disk space used by files in the file system, in kilobytes.
	usedFilePercent	The percentage of available files in use.
	usedFiles	The number of files in the file system.
	usedPercent	The percentage of the file-system disk space in use.
loadAverageMinute	fifteen	The number of processes requesting CPU time over the last 15 minutes.
	five	The number of processes requesting CPU time over the last 5 minutes.
	one	The number of processes requesting CPU time over the last minute.
memory	active	The amount of assigned memory, in kilobytes.
	buffers	The amount of memory used for buffering I/O requests prior to writing to the storage device, in kilobytes.
	cached	The amount of memory used for caching file system–based I/O.
	dirty	The amount of memory pages in RAM that have been modified but not written to their related data block in storage, in kilobytes.
	free	The amount of unassigned memory, in kilobytes.
	hugePagesFree	The number of free huge pages. Huge pages are a feature of the Linux kernel.
	hugePagesRsvd	The number of committed huge pages.
	hugePagesSize	The size for each huge pages unit, in kilobytes.
	hugePagesSurp	The number of available surplus huge pages over the total.
	hugePagesTotal	The total number of huge pages for the system.
	inactive	The amount of least-frequently used memory pages, in kilobytes.

Group	Metrics	Description
	mapped	The total amount of file-system contents that is memory mapped inside a process address space, in kilobytes.
	pageTables	The amount of memory used by page tables, in kilobytes.
	slab	The amount of reusable kernel data structures, in kilobytes.
	total	The total amount of memory, in kilobytes.
	writeback	The amount of dirty pages in RAM that are still being written to the backing storage, in kilobytes.
network	interface	The identifier for the network interface being used for the DB instance.
	rx	The number of bytes received per second.
	tx	The number of bytes uploaded per second.
processList	cpuUsedPc	The percentage of CPU used by the process.
	id	The identifier of the process.
	memoryUsedPc	The amount of memory used by the process, in kilobytes.
	name	The name of the process.
	parentID	The process identifier for the parent process of the process.
	rss	The amount of RAM allocated to the process, in kilobytes.
	tgid	The thread group identifier, which is a number representing the process ID to which a thread belongs. This identifier is used to group threads from the same process.
	vss	The amount of virtual memory allocated to the process, in kilobytes.
swap	cached	The amount of swap memory, in kilobytes, used as cache memory.
	free	The total amount of swap memory free, in kilobytes.
	total	The total amount of swap memory available, in kilobytes.
tasks	blocked	The number of tasks that are blocked.
	running	The number of tasks that are running.
	sleeping	The number of tasks that are sleeping.
	stopped	The number of tasks that are stopped.
	total	The total number of tasks.
	zombie	The number of child tasks that are inactive with an active parent task.

Metrics for Microsoft SQL Server DB instances

Group	Metrics	Description
General	engine	The database engine for the DB instance.
	instanceID	The DB instance identifier.
	instanceResourceID	A region-unique, immutable identifier for the DB instance, also used as the log stream identifier.
	numVCPUs	The number of virtual CPUs for the DB instance.
	timestamp	The time at which the metrics were taken.
	uptime	The amount of time that the DB instance has been active.
	version	The version of the OS metrics' stream JSON format.
cpuUtilization	idle	The percentage of CPU that is idle.
	kern	The percentage of CPU in use by the kernel.
	user	The percentage of CPU in use by user programs.
disks	name	The identifier for the disk.
	totalKb	The total space of the disk, in kilobytes.
	usedKb	The amount of space used on the disk, in kilobytes.
	usedPc	The percentage of space used on the disk.
	availKb	The space available on the disk, in kilobytes.
	availPc	The percentage of space available on the disk.
	rdCountPS	The number of read operations per second
	rdBytesPS	The number of bytes read per second.
	wrCountPS	The number of write operations per second.
	wBytesPS	The amount of bytes written per second.
memory	commitToKb	The amount of pagefile-backed virtual address space in use, that is, the current commit charge. This value is composed of main memory (RAM) and disk (pagefiles).
	commitLimitKb	The maximum possible value for the commitTotKb metric. This value is the sum of the current pagefile size plus the physical memory available for pageable contents–excluding RAM that is assigned to non-pageable areas.
	commitPeakKb	The largest value of the commitTotKb metric since the operating system was last started.
	kernTotKb	The sum of the memory in the paged and non-paged kernel pools, in kilobytes.

Group	Metrics	Description
	kernPagedKb	The amount of memory in the paged kernel pool, in kilobytes.
	kernNonpagedKb	The amount of memory in the non-paged kernel pool, in kilobytes.
	pageSize	The size of a page, in bytes.
	physTotKb	The amount of physical memory, in kilobytes.
	physAvailKb	The amount of available physical memory, in kilobytes.
	sqlServerTotKb	The amount of memory committed to Microsoft SQL Server, in kilobytes.
	sysCacheKb	The amount of system cache memory, in kilobytes.
network	interface	The identifier for the network interface being used for the DB instance.
	rdBytesPS	The number of bytes received per second.
	wrBytesPS	The number of bytes sent per second.
processList	cpuUsedPc	The percentage of CPU used by the process.
	memUsedPc	The amount of memory used by the process, in kilobytes.
	name	The name of the process.
	pid	The identifier of the process. This value is not present for processes that are owned by Amazon RDS.
	ppid	The process identifier for the parent of this process. This value is only present for child processes.
	tid	The thread identifier. This value is only present for threads. The owning process can be identified by using the pid value.
	workingSetKb	The amount of memory in the private working set plus the amount of memory that is in use by the process and can be shared with other processes, in kilobytes.
	workingSetPrivKb	The amount of memory that is in use by a process, but can't be shared with other processes, in kilobytes.
	workingSetShareableKb	The amount of memory that is in use by a process and can be shared with other processes, in kilobytes.
	virtKb	The amount of virtual address space the process is using, in kilobytes. Use of virtual address space does not necessarily imply corresponding use of either disk or main memory pages.
system	handles	The number of handles that the system is using.
	processes	The number of processes running on the system.
	threads	The number of threads running on the system.

Related Topics

- Using Amazon RDS Event Notification (p. 279)
- Amazon RDS Database Log Files (p. 303)

Preview: Using Amazon Performance Insights

Amazon RDS Performance Insights monitors your Amazon RDS DB instance load so that you can analyze and troubleshoot your database performance. Amazon RDS Performance Insights is currently available only for use with Amazon Aurora (PostgreSQL).

Performance Insights expands on existing Amazon RDS monitoring features to illustrate your database's performance and help you analyze any issues that affect it. With the Performance Insights dashboard, you can visualize the database load and filter the load by waits, SQL statements, hosts, or users. Performance Insights is on by default for the Aurora PostgreSQL database engine. If you have more than one database on the DB instance, performance data for all of the databases is aggregated for the DB instance. Database performance data is kept for 24 hours.

The central metric for Performance Insights is **DB Load**, which represents the average number of active sessions for the database engine. An *active session* is a connection that has submitted work to the database engine and is waiting for a response from it. For example, if you submit a SQL query to the database engine, the database session is active while the database engine is processing that query.

Session information is collected, aggregated, and displayed in the dashboard as the **Average Active Sessions** chart. The **Average Active Sessions** chart displays the **Max CPU** value as a line, so you can see if active sessions are exceeding it or not. The **Max CPU** value is determined by the number of **vCPU** (virtual CPU) cores for your DB instance.

If you find that the load in the **Average Active Sessions** chart is often above the **Max CPU** line and the primary wait state is CPU, the system CPU is overloaded. In these cases, you might want to throttle connections to the instance, tune any SQL queries with a high CPU load, or consider a larger instance class. High and consistent instances of any wait state indicate that there might be bottlenecks or resource contention issues that you should resolve, even if the load does not cross the **Max CPU** line.

The following video provides an overview of Performance Insights.

Using Performance Insights to Analyze Performance of Amazon Aurora with PostgreSQL Compatibility

Topics

Using the Performance Insights Dashboard

The Performance Insights dashboard contains database performance information to help you analyze and troubleshoot performance issues. On the main dashboard page, you can view information about the database load, and drill into details for a particular wait state, SQL query, host, or user.

Topics

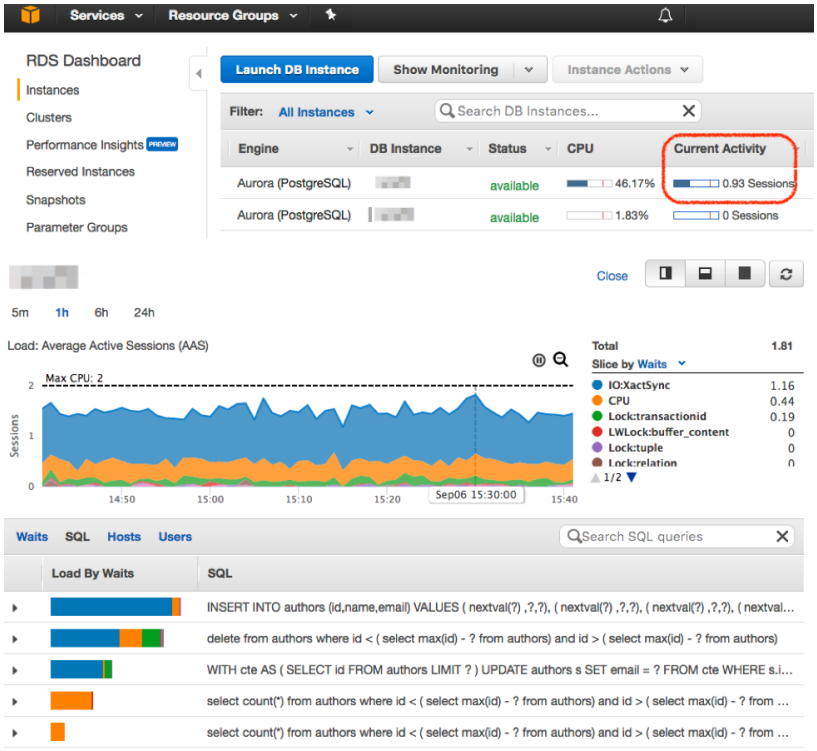
Opening the Performance Insights Dashboard

To see the Performance Insights dashboard, use the following procedure.

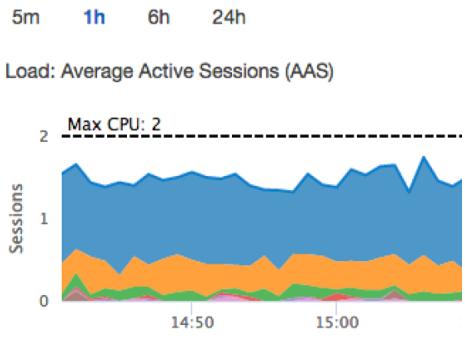
To view the Performance Insights dashboard in the AWS Management Console

1. Open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Performance Insights**.
3. Choose a DB instance. The Performance Insights dashboard is displayed for that instance.

   You can also reach the dashboard by choosing the **Current Activity** widget in the instance listing.



By default, the Performance Insights dashboard shows data for the last 60 minutes. You can modify it to display data for the last 5 minutes, 60 minutes, 6 hours, or 24 hours.



Performance Insights Dashboard Components

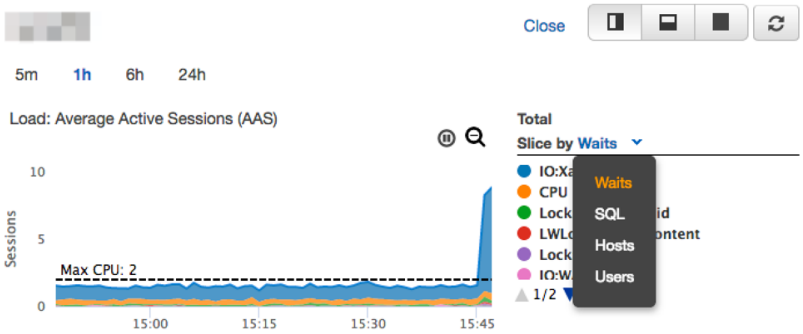
The dashboard is divided into two parts:

1. **Average Active Sessions chart** – Shows how the database load compares to DB instance capacity as represented by the **Max CPU** line.

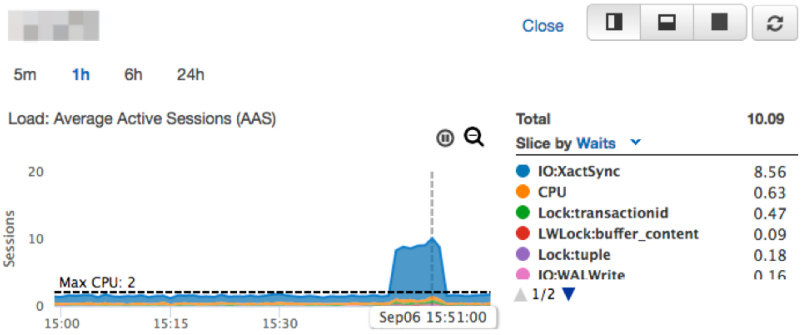
2. **Top load items table** – Shows the top items contributing to database load.

Average Active Sessions Chart

The **Average Active Sessions** chart shows how the database load compares to DB instance capacity as represented by the **Max CPU** line. By default, load is shown as active sessions grouped by wait states. You can also choose to display load as active sessions grouped by SQL queries, hosts, or users instead.



To see details for any item for the selected time period in the legend, hover over that item on the **Average Active Sessions** chart.

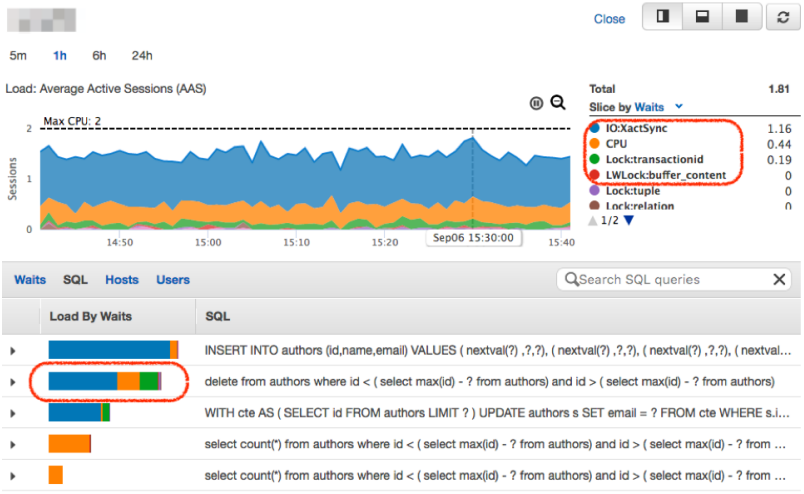


Top Load Items Table

The **Top Load Items** table shows the top items contributing to database load. By default, the top SQL queries that are contributing to the database load are shown. Queries are displayed as digests of multiple actual queries that are structurally similar but that possibly have different parameters. You can choose to display top wait states, hosts, or users instead.

The percentage of the database load associated with each top load item is illustrated in the **DB Load by Waits** column. This column reflects the load for that item by whatever grouping is currently selected in the **Average Active Sessions** chart. Take the case where the **Average Active Sessions** chart is grouping by hosts and you are looking at SQL queries in the top load items table. In this case, the **DB Load by Waits** bar reflects the load that query represents on the related host. Here it's colored-coded to map to the representation of that host in the **Average Active Sessions** chart.

For another example, suppose that the **Average Active Sessions** chart is grouping by wait states and you are looking at SQL queries in the top load items table. In this case, the **DB Load by Waits** bar is sized, segmented, and color-coded to show how much of a given wait state that query is contributing to. It also shows what wait states are affecting that query.



Analyzing Database Load Using the Performance Insights Dashboard

If the **Average Active Sessions** chart shows a bottleneck, you can find out where the load is coming from. To do so, look at the top load items table below the **Average Active Sessions** chart. Choose a particular item, like a SQL query or a user, to drill down into that item and see details about it.

DB load grouped by waits and top SQL queries is the default Performance Insights dashboard view, because this is the combination that typically provides the most insight into performance issues. DB load grouped by waits shows if there are any resource or concurrency bottlenecks in the database. In this case, the **SQL** tab of the top load items table shows which queries are driving that load.

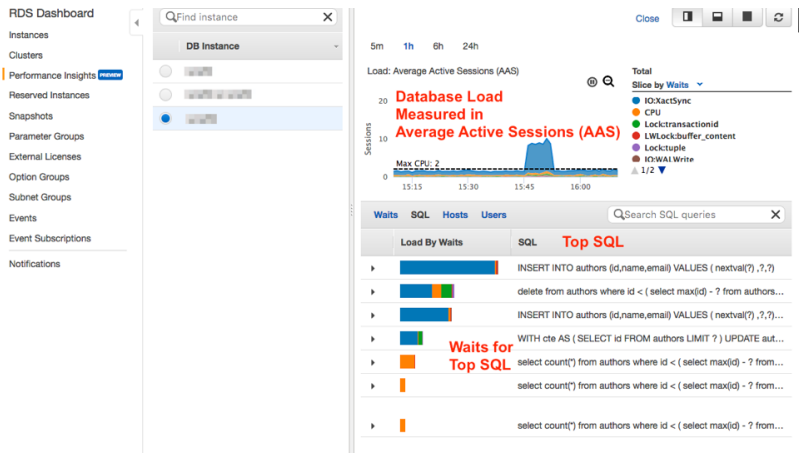
Your typical workflow for diagnosing performance issues is as follows:

1. Review the **Average Active Sessions** chart and see if there are any incidents of database load exceeding the **Max CPU** line.
2. If there is, look at the **Average Active Sessions** chart and identify which wait state or states are primarily responsible.
3. Identify the digest queries causing the load by seeing which of the queries the **SQL** tab on the top load items table are contributing most to those wait states. You can identify these by the **DB Load by Wait** column.
4. Choose one of these digest queries in the **SQL** tab to expand it and see the child queries that it is composed of.

For example, in the dashboard following, **IO:XactSync** waits are a frequent issue. **CPU** wait is less, but it still contributes to load.

The first four roll-up queries in the **SQL** tab of the top load items table correlate strongly to the first state. Thus, those are the ones to drill into and examine the child queries of. You do so to determine how they are contributing to the performance issue.

The last three roll-up queries are the major contributors to CPU. These are the queries to investigate if CPU load is an issue.

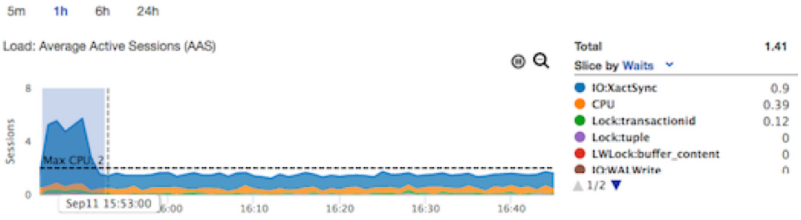


Additional User Interface Features

You can use other features of the Performance Insights user interface to help analyze performance data.

Click-and-Drag Zoom In

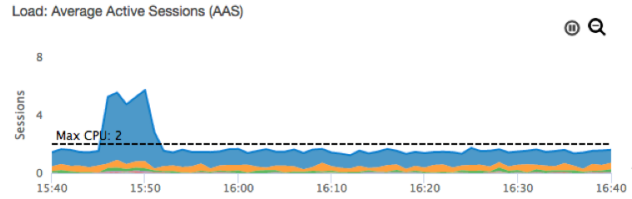
In the Performance Insights interface, you can select a small portion of the load chart and zoom in on the detail.



To zoom in on a portion of the load chart, choose the start time and drag to the end of the time period you want. When you do this, the selected area is highlighted. When you release the mouse, the load chart zooms in on the selected region, and the **Top N** table is recalculated.

Pause and Zoom Out

In the upper-right corner of the load chart, you can find the **Pause** and **Zoom out** tools.



When you choose **Pause**, the load chart stops autorefreshing. When you choose **Pause** again, the chart resumes autorefreshing.

When you choose **Zoom out**, the load chart zooms out to the next largest time interval.

Related Topics

- Using Amazon RDS Event Notification (p. 279)

- Amazon RDS Database Log Files (p. 303)

Access Control for Performance Insights

To access Performance Insights, you must have the appropriate permissions from AWS Identity and Access Management (IAM). There are two options available for granting access:

1. Attach the `AmazonRDSFullAccess` managed policy to an IAM user or role.
2. Create a custom IAM policy and attach it to an IAM user or role.

AmazonRDSFullAccess Managed Policy

`AmazonRDSFullAccess` is an AWS-managed policy that grants access to all of the Amazon RDS API actions. The policy also grants access to related services that are used by the Amazon RDS console—for example, event notifications using Amazon SNS.

In addition, `AmazonRDSFullAccess` contains all the permissions needed for using Performance Insights. If you attach this policy to an IAM user or role, the recipient can use Performance Insights, in addition to all of the other features of the Amazon RDS console.

Using a Custom IAM Policy

You can grant access to Performance Insights by creating or modifying a user-managed IAM policy. When you attach the policy to an IAM user or role, the recipient can use Performance Insights.

To create a custom policy

1. Open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the left navigation pane, choose **Policies**.
3. Choose **Create policy**.
4. On the **Create Policy** page, go to **Create Your Own Policy** and choose **Select**.
5. On the **Review Policy** page, set the following values:

   - **Policy Name:** Type a name for the policy, for example: `PerformanceInsightsFullAccess`
   - **Description:** (Optional) Type a short description for the policy.
   - **Policy Document:** Copy and paste the following:

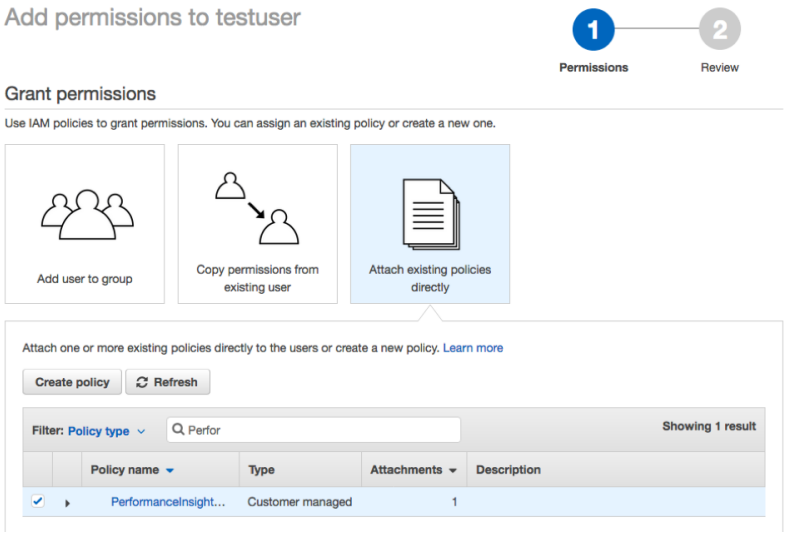
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "pi:*",
            "Resource": "arn:aws:pi:*:*:metrics/rds/*"
        }
    ]
}
```

   When the settings are as you want them, choose **Create Policy**.

You can now attach the policy to an IAM user or role. The following procedure assumes that you already have an IAM user available for this purpose.

To attach the policy to an IAM user

1. Open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the left navigation pane, choose **Users**.
3. Choose an existing user from the list.
4. On the **Summary** page, choose **Add permissions**.
5. Choose **Attach existing policies directly**. For **Search**, type the first few characters of your policy name, as shown following.



6. Choose your policy, and then choose **Next: Review**.
7. Choose **Add permissions**.

Frequently Asked Questions

Q: On which instance sizes will Performance Insights be available?

A: All nonmicro instance sizes. As RDS introduces new instance sizes, Performance Insights will be made available on those, unless they have limited performance such as current *.nano and *.micro instance sizes.

Q: When will Performance Insights be available for RDS for PostgreSQL, Aurora MySQL, RDS for MySQL, RDS for Oracle, RDS for SQL Server, and RDS for MariaDB?

A: Performance Insights will be initially available on Aurora PostgreSQL, followed soon after by Aurora MySQL. Additional engines will be added over time.

Q: How does Performance Insights show the cause of performance problems?

A: Performance problems appear in the Performance Insights section of the RDS console as spikes in the database load graph. One look at this graph can quickly tell you what type of resources your application has been spending time and resources on in the database. Using the console, you can zoom in to any period within the retention time. By selecting the periods of high load, customers can display a list of SQL statements, ordered by overall contribution to load.

Q: How does Performance Insights know about load in my RDS DB instance?

A: Performance Insights collects the state of all of the connected sessions in your DB instance. If a session is spending time in a database-related operation, Performance Insights records the time, the

type of operation (I/O, CPU, locking, and so on), the current SQL statement and several other session attributes. Over periods of time, this data is used to characterize how sessions are contributing to load in your database instance.

Q: Can performance data be queried from inside the RDS instance?

A: No. Performance Insights does not populate any tables in the database, nor does it present data to be retrieved from within the database via SQL.

Q: Can I see what is happening on my instance in real time?

A: Yes. By default, Performance Insights displays a moving one hour window of performance data. The feature is designed to present the latest performance information within a few seconds of real time.

Q: How much does Performance Insights cost?

A: Performance Insights will include 24 hours of retained data and console access only). Performance Insights while in preview, offers a free tier that includes a trailing 24 hours of performance data retention. Pricing for longer term data retention will be published at GA.

Q: How far back can I look at performance data stored in Performance Insights?

A: You can see 24 hours of performance history. Options for longer retention will be made available in the months to come.

Q: Can I turn off Performance Insights on new instances, even though it is enabled by default?

A: Yes. The option for Performance Insights is selected by default in the AWS Management Console when you use the instance creation wizard. You can de-select the option in the wizard to prevent Performance Insights from being enabled. You can also disable Performance Insights in an enabled instance by modifying the instance.

Q: Does Performance Insights work on RDS database instances using encrypted storage?

A: Yes. Performance Insights doesn't read the data you store in your database.

Q: What is DB load and why is it the primary measure used in Performance Insights to detect performance issues?

A: DB load is a time series showing how much time a customer's applications are spending in the database, and how they are spending that time. DB load is measured in units of average active sessions (AAS). An *active session* is a connection (session) that has submitted work to the database engine and is waiting for a response from it. For example, if you submit a SQL statement to a database instance, that session is considered "active" during the time that the instance is processing that query. By counting the number of sessions active in an instance at a given moment, we can provide a metric. This metric, averaged over time periods, can show how busy an instance is, and how much time sessions are spending waiting for the instance to respond. We call this metric DB load. Performance Insights collects DB load time series data from the monitored instance, encrypts and aggregates it to be displayed in the **DB Load** chart. In the console, you can select the time frame you want to view.

Q: Do I have to do anything special to my database to enable Performance Insights?

A: No. However, Performance Insights works even better on some database engines when additional performance tracking is enabled. For instance, when the `pg_stat_statement` extension is enabled on RDS PostgreSQL or Aurora PostgreSQL, Performance Insights uses the PostgreSQL-native SQL identifier to identify the statement. It can then collect the full text of longer statements. In MySQL, enabling the Performance Schema allows Performance Insights to collect much richer and deeper detail on wait events affecting the database.

Q: Does enabling Performance Insights affect my database performance?

A: The Performance Insights agent is designed to stay out of the way of your database workloads. Performance Insights runs at a lower priority than the other processes on your instance, and monitors the health of the host and database. When Performance Insights detects heavy load or depleted resources, it backs off from the usual frequency of data gathering. It still collects data, but only when it is safe to do so. Database options, such as `pg_stat_statement` in RDS PostgreSQL and Aurora PostgreSQL, and Performance Schema in MySQL can use some database resources and potentially affect performance. Whether enabling these options affects a particular system depends on the application workload. AWS recommends testing any database options against your workload before enabling them on a production system.

Q: Should I keep using Enhanced Monitoring or just use Performance Insights?

A: Customers using Enhanced Monitoring to monitor operating system metrics should continue to obtain that data by using Enhanced Monitoring. In the months to come, that data, and also an extensive collection of database metrics, will also become available through the Performance Insights console and an API. At that point, customers will be able to obtain all performance data from Performance Insights. Enhanced Monitoring will remain available for those customers who prefer to use it, but we will encourage customers to standardize their database monitoring on Performance Insights.

Q: Is the data stored in Performance Insights encrypted?

A: Yes. Performance Insights encrypts all potentially sensitive data using your own AWS Key Management Service (AWS KMS) key. Data is encrypted in flight and at rest. AWS personnel cannot access or see any potentially sensitive performance data. Only your users on your AWS account with full access to RDS can view Performance Insights. You can revoke RDS's grant for your KMS key, which enables us to process and display your performance data, at any time.

Q: If I turn off Performance Insights, does AWS retain the data or is it deleted?

A: Performance data retention free tier is restricted to one day. Disabling Performance Insights on an instance causes your performance data for that instance to be deleted.

Q: What happens to Performance Insights data retention when I stop my RDS database instance?

A: Stopping an RDS instance that has Performance Insights enabled has no effect on retention or visibility of historical data for that instance. The period during which the instance was stopped will simply contain no data.

Q: How would I interface Performance Insights with my existing performance tools?

A: In the coming months, Performance Insights will expose a publicly available API designed to enable customers and third parties to take advantage of the valuable data in Performance Insights.

Q: Is there any way for third-party performance tools to integrate with Performance Insights?

A: In the coming months, Performance Insights will expose a publicly available API designed to enable customers and third parties to take advantage of the valuable data in Performance Insights.

Q: Will Performance Insights be available in all the AWS Regions that RDS is?

A: Yes. Performance Insights will initially be available in four AWS Regions: US East (N. Virginia, Ohio), US West (Oregon), and EU (Ireland). Over time, the feature will be made available in all AWS Regions where RDS is supported.

Q: Can I turn on Performance Insights on existing instances or only on new ones?

A: Performance Insights can be enabled on existing instances by modifying the instance to enable Performance Insights. It can be enabled on new instances by specifying that Performance Insights should be enabled when creating the instance.

Q: Does Performance Insights use any of the storage on my database instance?

A: No, Performance Insights doesn't consume storage space on your RDS instances.

Q: How will Performance Insights be different, if at all, when running against different database engines?

A: Performance Insights is designed to present a common approach, look, and feel to tuning across all database engines in RDS. Because certain attributes like wait events and SQL identifiers vary by engine type, these also vary in Performance Insights when working with different database engines. One of the core tenets of Performance Insights is that existing concepts, identifiers, and attributes in a database engine should remain intact. Performance Insights generally doesn't re-interpret or rename wait events and other engine-specific attributes. Instead, it presents them faithfully as reported by the database engine.

Q: Does Performance Insights work on Multi-AZ instances and Read Replica instances?

A: Yes. If an RDS database uses Multi-AZ and has Performance Insights enabled, then Performance Insights remains enabled when and if that instance fails over to the other Availability Zone. Because Read Replicas are independent instances, customers can either enable or disable Performance Insights on those instances.

Q: Can I export my data from Performance Insights?

A: In the coming months, Performance Insights will be adding functionality to export data.

Q: Can I re-import my data to Performance Insights later in order to do performance analysis?

A: No. Performance Insights only shows data that has been collected directly from an instance. Data obtained through Performance Insights is available in the months to come by using an API. At that point, you will be able to perform analysis using one of the analytics-oriented services in AWS, such as Amazon Athena, Amazon Redshift, Amazon Redshift Spectrum, and Amazon QuickSight.

Using Amazon RDS Event Notification

Topics

Amazon RDS uses the Amazon Simple Notification Service (Amazon SNS) to provide notification when an Amazon RDS event occurs. These notifications can be in any notification form supported by Amazon SNS for an AWS region, such as an email, a text message, or a call to an HTTP endpoint.

Amazon RDS groups these events into categories that you can subscribe to so that you can be notified when an event in that category occurs. You can subscribe to an event category for a DB instance,DB cluster, DB snapshot, DB cluster snapshot, DB security group, or for a DB parameter group. For example, if you subscribe to the Backup category for a given DB instance, you will be notified whenever a backup-related event occurs that affects the DB instance. If you subscribe to a Configuration Change category for a DB security group, you will be notified when the DB security group is changed. You will also receive notification when an event notification subscription changes.

Note that for Amazon Aurora, events occur at the cluster rather than instance level, so you won't receive events if you subscribe to an Aurora DB instance. Subscribe to the DB cluster instead.

Event notifications are sent to the addresses you provide when you create the subscription. You may want to create several different subscriptions, such as one subscription receiving all event notifications and another subscription that includes only critical events for your production DB instances. You can easily turn off notification without deleting a subscription by setting the **Enabled** radio button to `No` in the Amazon RDS console or by setting the `Enabled` parameter to `false` using the CLI or Amazon RDS API.

> **Note**
> Amazon RDS event notifications using SMS text messages are currently available for topic ARNs and Amazon RDS resources in the US-East (Northern Virginia) Region. For more information on using text messages with SNS, see Sending and Receiving SMS Notifications Using Amazon SNS.

Amazon RDS uses the Amazon Resource Name (ARN) of an Amazon SNS topic to identify each subscription. The Amazon RDS console will create the ARN for you when you create the subscription. If you use the CLI or API, you have to create the ARN by using the Amazon SNS console or the Amazon SNS API when you create a subscription.

Billing for Amazon RDS event notification is through the Amazon Simple Notification Service (Amazon SNS). Amazon SNS fees apply when using event notification; for more information on Amazon SNS billing, see Amazon Simple Notification Service Pricing.

The process for subscribing to Amazon RDS event notification is as follows:

1. Create an Amazon RDS event notification subscription by using the Amazon RDS console, AWS CLI, or API.
2. Amazon RDS sends an approval email or SMS message to the addresses you submitted with your subscription. To confirm your subscription, click the link in the notification you were sent.

3. When you have confirmed the subscription, the status of your subscription is updated in the Amazon RDS console's **My Event Subscriptions** section.
4. You will begin to receive event notifications.

The following section lists all categories and events that you can be notified of. It also provides information about subscribing to and working with Amazon RDS event subscriptions.

Amazon RDS Event Categories and Event Messages

Amazon RDS generates a significant number of events in categories that you can subscribe to using the Amazon RDS Console, AWS CLI, or the API. Each category applies to a source type, which can be a DB instance, DB snapshot, DB security group, or DB parameter group.

The following table shows the event category and a list of events when a DB instance is the source type.

Categories and Events for the DB Instance Source Type

Category	Amazon RDS Event ID	Description
availability	RDS-EVENT-0006	The DB instance is restarting and will be unavailable until the restart is complete.
availability	RDS-EVENT-0004	The DB instance has shut down.
availability	RDS-EVENT-0022	An error has occurred while restarting MySQL or MariaDB.
backup	RDS-EVENT-0001	A backup of the DB instance has started.
backup	RDS-EVENT-0002	A backup of the DB instance is complete.
configuration change	RDS-EVENT-0009	The DB instance has been added to a security group.
configuration change	RDS-EVENT-0024	The DB instance is being converted to a Multi-AZ DB instance.
configuration change	RDS-EVENT-0030	The DB instance is being converted to a Single-AZ DB instance.
configuration change	RDS-EVENT-0012	The DB instance class for this DB instance is being changed.
configuration change	RDS-EVENT-0018	The current storage settings for this DB instance is being changed.
configuration change	RDS-EVENT-0011	A parameter group for this DB instance has changed.
configuration change	RDS-EVENT-0028	Automatic backups for this DB instance have been disabled.
configuration change	RDS-EVENT-0032	Automatic backups for this DB instance have been enabled.
configuration change	RDS-EVENT-0033	There are [count] users that match the master user name. Users not tied to a specific host have been reset.

Category	Amazon RDS Event ID	Description
configuration change	RDS-EVENT-0025	The DB instance has been converted to a Multi-AZ DB instance.
configuration change	RDS-EVENT-0029	The DB instance has been converted to a Single-AZ DB instance.
configuration change	RDS-EVENT-0014	The DB instance class for this DB instance has changed.
configuration change	RDS-EVENT-0017	The storage settings for this DB instance has changed.
configuration change	RDS-EVENT-0010	The DB instance has been removed from a security group.
configuration change	RDS-EVENT-0016	The master password for the DB instance has been reset.
configuration change	RDS-EVENT-0067	An attempt to reset the master password for the DB instance has failed.
configuration change	RDS-EVENT-0078	The Enhanced Monitoring configuration has been changed.
creation	RDS-EVENT-0005	A DB instance is being created.
deletion	RDS-EVENT-0003	The DB instance is being deleted.
failover	RDS-EVENT-0034	Amazon RDS is not attempting a requested failover because a failover recently occurred on the DB instance.
failover	RDS-EVENT-0013	A Multi-AZ failover that resulted in the promotion of a standby instance has started.
failover	RDS-EVENT-0015	A Multi-AZ failover that resulted in the promotion of a standby instance is complete. It may take several minutes for the DNS to transfer to the new primary DB instance.
failover	RDS-EVENT-0065	The instance has recovered from a partial failover.
failover	RDS-EVENT-0049	A Multi-AZ failover has completed.
failover	RDS-EVENT-0050	A Multi-AZ activation has started after a successful instance recovery.
failover	RDS-EVENT-0051	A Multi-AZ activation is complete. Your database should be accessible now.
failure	RDS-EVENT-0031	The DB instance has failed. We recommend that you begin a point-in-time-restore for the DB instance.
failure	RDS-EVENT-0036	The DB instance is in an incompatible network. Some of the specified subnet IDs are invalid or do not exist.

Category	Amazon RDS Event ID	Description
failure	RDS-EVENT-0035	The DB instance has invalid parameters. For example, MySQL could not start because a memory-related parameter is set too high for this instance class, so the customer action would be to modify the memory parameter and reboot the DB instance.
failure	RDS-EVENT-0058	Error while creating Statspack user account PERFSTAT. Please drop the account before adding the Statspack option.
failure	RDS-EVENT-0079	Enhanced Monitoring cannot be enabled without the enhanced monitoring IAM role. For information on creating the enhanced monitoring IAM role, see To create an IAM role for Amazon RDS Enhanced Monitoring (p. 259).
failure	RDS-EVENT-0080	Enhanced Monitoring was disabled due to an error making the configuration change. It is likely that the enhanced monitoring IAM role is configured incorrectly. For information on creating the enhanced monitoring IAM role, see To create an IAM role for Amazon RDS Enhanced Monitoring (p. 259).
failure	RDS-EVENT-0081	The IAM role that you use to access your Amazon S3 bucket for SQL Server native backup and restore is configured incorrectly. For more information, see Setting Up for Native Backup and Restore (p. 770).
failure	RDS-EVENT-0082	Amazon Aurora was unable to copy backup data from an Amazon S3 bucket. It is likely that the permissions for Aurora to access the Amazon S3 bucket are configured incorrectly. For more information, see Migrating Data from MySQL by Using an Amazon S3 Bucket (p. 488).
low storage	RDS-EVENT-0089	The DB instance has consumed more than 90% of its allocated storage. You can monitor the storage space for a DB instance using the **Free Storage Space** metric. For more information, see Viewing DB Instance Metrics (p. 254).
low storage	RDS-EVENT-0007	The allocated storage for the DB instance has been exhausted. To resolve this issue, you should allocate additional storage for the DB instance. For more information, see the RDS FAQ. You can monitor the storage space for a DB instance using the **Free Storage Space** metric. For more information, see Viewing DB Instance Metrics (p. 254).
maintenance	RDS-EVENT-0026	Offline maintenance of the DB instance is taking place. The DB instance is currently unavailable.
maintenance	RDS-EVENT-0027	Offline maintenance of the DB instance is complete. The DB instance is now available.
notification	RDS-EVENT-0044	Operator-issued notification. For more information, see the event message.

Category	Amazon RDS Event ID	Description
notification	RDS-EVENT-0047	Patching of the DB instance has completed.
notification	RDS-EVENT-0048	Patching of the DB instance has been delayed.
notification	RDS-EVENT-0054	The MySQL storage engine you are using is not InnoDB, which is the recommended MySQL storage engine for Amazon RDS. For information about MySQL storage engines, see Supported Storage Engines for MySQL on Amazon RDS (p. 824).
notification	RDS-EVENT-0055	The number of tables you have for your DB instance exceeds the recommended best practices for Amazon RDS. Please reduce the number of tables on your DB instance. For information about recommended best practices, see Amazon RDS Basic Operational Guidelines (p. 80).
notification	RDS-EVENT-0056	The number of databases you have for your DB instance exceeds the recommended best practices for Amazon RDS. Please reduce the number of databases on your DB instance. For information about recommended best practices, see Amazon RDS Basic Operational Guidelines (p. 80).
notification	RDS-EVENT-0064	The TDE key has been rotated. For more information about Oracle TDE, see Oracle Transparent Data Encryption (p. 1036). For more information about SQL Server TDE, see Microsoft SQL Server Transparent Data Encryption Support (p. 797).
notification	RDS-EVENT-0084	You attempted to convert a DB instance to Multi-AZ, but it contains in-memory file groups which are not supported for Multi-AZ. For more information, see Multi-AZ Deployments for Microsoft SQL Server with Database Mirroring (p. 787).
notification	RDS-EVENT-0087	The DB instance has been stopped.
notification	RDS-EVENT-0088	The DB instance has been started.
read replica	RDS-EVENT-0045	An error has occurred in the read replication process. For more information, see the event message. For information on troubleshooting Read Replica errors, see Troubleshooting a MySQL or MariaDB Read Replica Problem (p. 150).
read replica	RDS-EVENT-0046	The Read Replica has resumed replication. This message appears when you first create a Read Replica, or as a monitoring message confirming that replication is functioning properly. If this message follows an RDS-EVENT-0045 notification, then replication has resumed following an error or after replication was stopped.
read replica	RDS-EVENT-0057	Replication on the Read Replica was terminated.

Category	Amazon RDS Event ID	Description
read replica	RDS-EVENT-0062	Replication on the Read Replica was manually stopped.
read replica	RDS-EVENT-0063	Replication on the Read Replica was reset.
recovery	RDS-EVENT-0020	Recovery of the DB instance has started. Recovery time will vary with the amount of data to be recovered.
recovery	RDS-EVENT-0021	Recovery of the DB instance is complete.
recovery	RDS-EVENT-0023	A manual backup has been requested but Amazon RDS is currently in the process of creating a DB snapshot. Submit the request again after Amazon RDS has completed the DB snapshot.
recovery	RDS-EVENT-0052	Recovery of the Multi-AZ instance has started. Recovery time will vary with the amount of data to be recovered.
recovery	RDS-EVENT-0053	Recovery of the Multi-AZ instance is complete.
recovery	RDS-EVENT-0066	The SQL Server DB instance is re-establishing its mirror. Performance will be degraded until the mirror is reestablished. A database was found with non-FULL recovery model. The recovery model was changed back to FULL and mirroring recovery was started. (<dbname>: <recovery model found>[,...])"
restoration	RDS-EVENT-0008	The DB instance has been restored from a DB snapshot.
restoration	RDS-EVENT-0019	The DB instance has been restored from a point-in-time backup.
security	RDS-EVENT-0068	The CloudHSM Classic partition password was decrypted by the system.

The following table shows the event category and a list of events when a DB parameter group is the source type.

Categories and Events for the DB Parameter Group Source Type

Category	RDS Event ID	Description
configuration change	RDS-EVENT-0037	The parameter group was modified.

The following tables shows the event category and a list of events when a DB security group is the source type.

Categories and Events for the DB Security Group Source Type

Category	RDS Event ID	Description
configuration change	RDS-EVENT-0038	The security group has been modified.
failure	RDS-EVENT-0039	The Amazon EC2 security group owned by [user] does not exist; authorization for the security group has been revoked.

The following tables shows the event category and a list of events when a DB snapshot is the source type.

Categories and Events for the DB Snapshot Source Type

Category	RDS Event ID	Description
creation	RDS-EVENT-0040	A manual DB snapshot is being created.
deletion	RDS-EVENT-0041	A DB snapshot has been deleted.
creation	RDS-EVENT-0042	A manual DB snapshot has been created.
restoration	RDS-EVENT-0043	A DB instance is being restored from a DB snapshot.
notification	RDS-EVENT-0059	Started the copy of the cross region DB snapshot [DB snapshot name] from source region [region name].
notification	RDS-EVENT-0060	Finished the copy of the cross region DB snapshot [DB snapshot name] from source region [region name] in [time] minutes.
notification	RDS-EVENT-0061	The copy of a cross region DB snapshot failed.
creation	RDS-EVENT-0090	An automated DB snapshot is being created.
creation	RDS-EVENT-0091	An automated DB snapshot has been created.

The following tables shows the event category and a list of events when a DB cluster is the source type.

Categories and Events for the DB Cluster Source Type

Category	RDS Event ID	Description
failover	RDS-EVENT-0069	A failover for the DB cluster has failed.
failover	RDS-EVENT-0070	A failover for the DB cluster has restarted.
failover	RDS-EVENT-0071	A failover for the DB cluster has finished.
failover	RDS-EVENT-0072	A failover for the DB cluster has begun within the same Availability Zone.
failover	RDS-EVENT-0073	A failover for the DB cluster has begun across Availability Zones.
failure	RDS-EVENT-0083	Amazon Aurora was unable to copy backup data from an Amazon S3 bucket. It is likely that the permissions

Category	RDS Event ID	Description
		for Aurora to access the Amazon S3 bucket are configured incorrectly. For more information, see Migrating Data from MySQL by Using an Amazon S3 Bucket (p. 488).
migration	RDS-EVENT-0076	Migration to an Amazon Aurora DB cluster failed.
migration	RDS-EVENT-0077	An attempt to convert a table from the source database to InnoDB failed during the migration to an Amazon Aurora DB cluster.

The following tables shows the event category and a list of events when a DB cluster snapshot is the source type.

Categories and Events for the DB Cluster Snapshot Source Type

Category	RDS Event ID	Description
backup	RDS-EVENT-0074	Creation of a manual DB cluster snapshot has started.
backup	RDS-EVENT-0075	A manual DB cluster snapshot has been created.

Subscribing to Amazon RDS Event Notification

You can create an Amazon RDS event notification subscription so you can be notified when an event occurs for a given DB instance, DB snapshot, DB security group, or DB parameter group. The simplest way to create a subscription is with the RDS console. If you choose to create event notification subscriptions using the CLI or API, you must create an Amazon Simple Notification Service topic and subscribe to that topic with the Amazon SNS console or Amazon SNS API. You will also need to retain the Amazon Resource Name (ARN) of the topic because it is used when submitting CLI commands or API actions. For information on creating an SNS topic and subscribing to it, see Getting Started with Amazon SNS.

You can specify the type of source you want to be notified of and the Amazon RDS source that triggers the event. These are defined by the **SourceType** (type of source) and the **SourceIdentifier** (the Amazon RDS source generating the event). If you specify both the **SourceType** and **SourceIdentifier**, such as `SourceType = db-instance` and `SourceIdentifier = myDBInstance1`, you will receive all the DB_Instance events for the specified source. If you specify a **SourceType** but do not specify a **SourceIdentifier**, you will receive notice of the events for that source type for all your Amazon RDS sources. If you do not specify either the **SourceType** nor the **SourceIdentifier**, you will be notified of events generated from all Amazon RDS sources belonging to your customer account.

AWS Management Console

To subscribe to RDS event notification

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the Amazon RDS Console navigation pane, click **Event Subscriptions**.
3. In the **Event Subscriptions** pane, click **Create Event Subscription**.
4. In the **Create Event Subscription** dialog box, do the following:

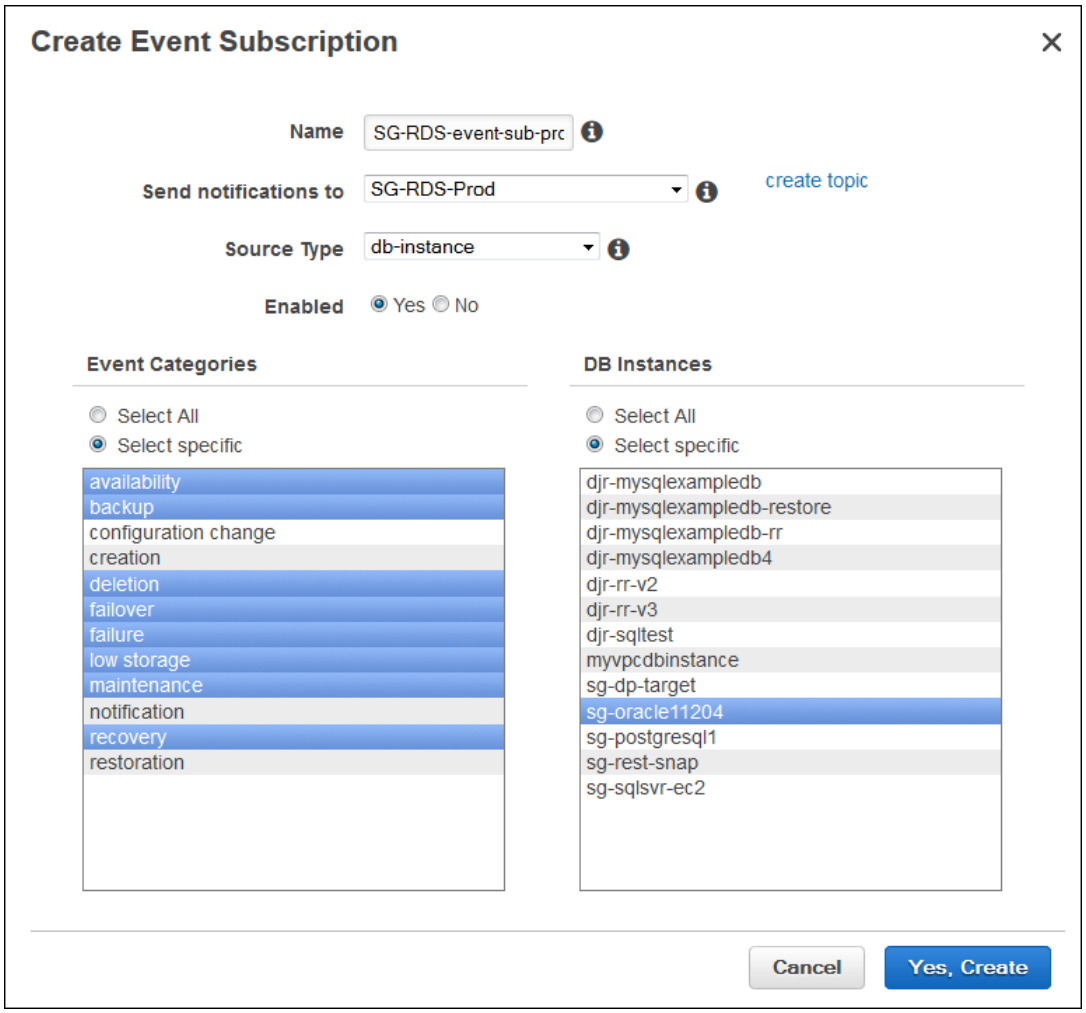
    a. Type a name for the event notification subscription in the **Name** text box.

b. Select an existing Amazon SNS Amazon Resource Name (ARN) for an Amazon SNS topic in the **Send notifications to** dropdown menu or click **create topic** to enter the name of a topic and a list of recipients.

c. Select a source type from the **Source Type** dropdown menu.

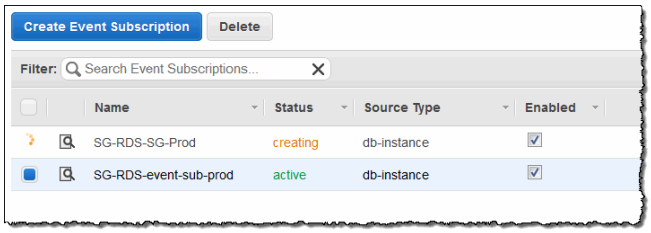
d. Select **Yes** to enable the subscription. If you want to create the subscription but to not have notifications sent yet, select **No**.

e. Depending on the source type you selected, select the event categories and sources you want to receive event notifications for.



f. Click **Yes, Create**.

5. The Amazon RDS console indicates that the subscription is being created.



CLI

To subscribe to RDS Event Notification, use the AWS CLI `create-event-subscription` command. Include the following required parameters:

- `--subscription-name`
- `--sns-topic-arn`

Example

For Linux, OS X, or Unix:

```
aws rds create-event-subscription \
    --subscription-name myeventsubscription \
    --sns-topic-arn arn:aws:sns:us-east-1:802#########:myawsuser-RDS \
    --enabled
```

For Windows:

```
aws rds create-event-subscription ^
    --subscription-name myeventsubscription ^
    --sns-topic-arn arn:aws:sns:us-east-1:802#########:myawsuser-RDS ^
    --enabled
```

API

To subscribe to Amazon RDS Event Notification call the Amazon RDS API function `CreateEventSubscription`. Include the following required parameters:

- `SubscriptionName`
- `SnsTopicArn`

Example

```
https://rds.us-east-1.amazonaws.com/
   ?Action=CreateEventSubscription
   &Enabled=true
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &SnsTopicArn=arn%3Aaws%3Asns%3Aus-east-1%3A802#########%3Amyawsuser-RDS
   &SourceType=db-security-group
   &SubscriptionName=myeventsubscription
   &Version=2014-09-01
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20140425/us-east-1/rds/aws4_request
   &X-Amz-Date=20140425T214325Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
   &X-Amz-Signature=7045960f6ab15609571fb05278004256e186b7633ab2a3ae46826d7713e0b461
```

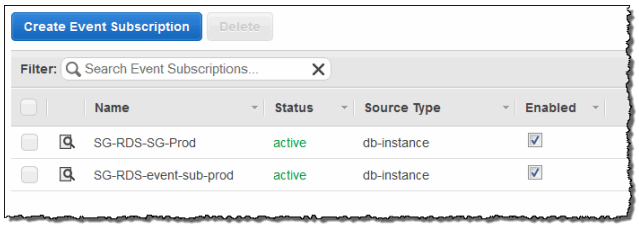
Listing Your Amazon RDS Event Notification Subscriptions

You can list your current Amazon RDS event notification subscriptions.

AWS Management Console

To list your current Amazon RDS event notification subscriptions

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the Amazon RDS Console navigation pane, click **Event Subscriptions**. The Event Subscriptions pane shows all your event notification subscriptions.



CLI

To list your current Amazon RDS event notification subscriptions, use the AWS CLI `describe-event-subscriptions` command.

Example

The following example describes all event subscriptions.

```
aws rds describe-event-subscriptions
```

The following example describes the `myfirsteventsubscription`.

```
aws rds describe-event-subscriptions --subscription-name myfirsteventsubscription
```

API

To list your current Amazon RDS event notification subscriptions, call the Amazon RDS API `DescribeEventSubscriptions` action.

Example

The following code example lists up to 100 event subscriptions.

```
https://rds.us-east-1.amazonaws.com/
   ?Action=DescribeEventSubscriptions
   &MaxRecords=100
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &Version=2014-09-01
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
```

```
   &X-Amz-Credential=AKIADQKE4SARGYLE/20140428/us-east-1/rds/aws4_request
   &X-Amz-Date=20140428T161907Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
   &X-Amz-Signature=4208679fe967783a1a149c826199080a066085d5a88227a80c6c0cadb3e8c0d4
```

The following example describes the `myfirsteventsubscription`.

```
https://rds.us-east-1.amazonaws.com/
   ?Action=DescribeEventSubscriptions
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &SubscriptionName=myfirsteventsubscription
   &Version=2014-09-01
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20140428/us-east-1/rds/aws4_request
   &X-Amz-Date=20140428T161907Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
   &X-Amz-Signature=4208679fe967783a1a149c826199080a066085d5a88227a80c6c0cadb3e8c0d4
```

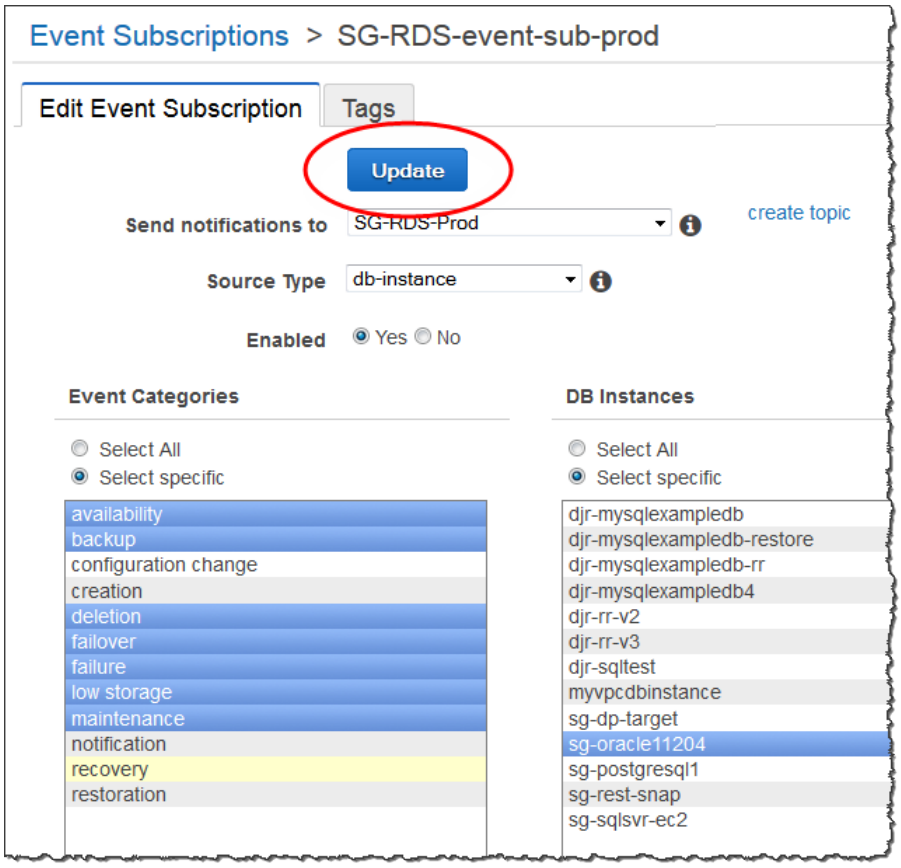
Modifying an Amazon RDS Event Notification Subscription

After you have created a subscription, you can change the subscription name, source identifier, categories, or topic ARN.

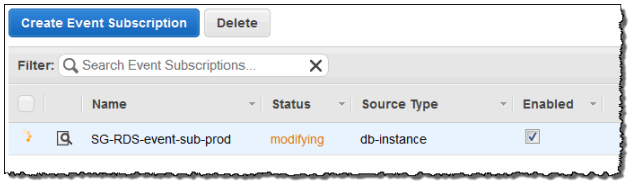
AWS Management Console

To modify an Amazon RDS event notification subscription

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the Amazon RDS Console navigation pane, click **Event Notification**.
3. In the **DB Event Notifications** pane, select the subscription that you want to modify.
4. Make your changes to the subscription in the lower pane.



5. Click **Update**. The Amazon RDS console indicates that the subscription is being modified.



CLI

To modify an Amazon RDS event notification subscription, use the AWS CLI `modify-event-subscription` command. Include the following required parameter:

- `--subscription-name`

Example

The following code enables `myeventsubscription`.

For Linux, OS X, or Unix:

```
aws rds modify-event-subscription \
    --subscription-name myeventsubscription \
    --enabled
```

For Windows:

```
aws rds modify-event-subscription ^
    --subscription-name myeventsubscription ^
    --enabled
```

API

To modify an Amazon RDS Event, call the Amazon RDS API action `ModifyEventSubscription`. Include the following required parameter:

- `SubscriptionName`

Example

The following code enables `myeventsubscription`.

```
https://rds.us-west-2.amazonaws.com/
   ?Action=ModifyEventSubscription
   &Enabled=true
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &SnsTopicArn=arn%3Aaws%3Asns%3Aus-west-2%3A802#########%3Amy-rds-events
   &SubscriptionName=myeventsubscription
   &Version=2014-09-01
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20140428/us-west-2/rds/aws4_request
   &X-Amz-Date=20140428T183020Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
   &X-Amz-Signature=3d85bdfaf13861e93a9528824d9876ed87e6e01aaf43a962ce6f2a39247cf33a
```

Adding a Source Identifier to an Amazon RDS Event Notification Subscription

You can add a source identifier (the Amazon RDS source generating the event) to an existing subscription.

AWS Management Console

You can easily add or remove source identifiers using the Amazon RDS console by selecting or deselecting them when modifying a subscription. See the topic Modifying an Amazon RDS Event Notification Subscription (p. 291) for more information.

CLI

To add a source identifier to an Amazon RDS event notification subscription, use the AWS CLI `add-source-identifier-to-subscription` command. Include the following required parameters:

- `--subscription-name`
- `--source-identifier`

Example

The following example adds the source identifier `mysqldb` to the `myrdseventsubscription.` subscription.

For Linux, OS X, or Unix:

```
aws rds add-source-identifier-to-subscription \
    --subscription-name myrdseventsubscription \
    --source-identifier mysqldb
```

For Windows:

```
aws rds add-source-identifier-to-subscription ^
    --subscription-name myrdseventsubscription ^
    --source-identifier mysqldb
```

API

To add a source identifier to an Amazon RDS event notification subscription, call the Amazon RDS API `AddSourceIdentifierToSubscription`. Include the following required parameters:

- `SubscriptionName`
- `SourceIdentifier`

Example

```
https://rds.us-east-1.amazonaws.com/
   ?Action=AddSourceIdentifierToSubscription
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &SourceIdentifier=mysqldb
   &SubscriptionName=myrdseventsubscription
```

```
&Version=2014-09-01
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20140422/us-east-1/rds/aws4_request
&X-Amz-Date=20140422T230442Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=347d5e788e809cd06c50214b12750a3c39716bf65b239bb6f7ee8ff5374e2df9
```

Removing a Source identifier from an Amazon RDS Event Notification Subscription

You can remove a source identifier (the Amazon RDS source generating the event) from a subscription if you no longer want to be notified of events for that source.

AWS Management Console

You can easily add or remove source identifiers using the Amazon RDS console by selecting or deselecting them when modifying a subscription. See the topic Modifying an Amazon RDS Event Notification Subscription (p. 291) for more information.

CLI

To remove a source identifier from an Amazon RDS event notification subscription, use the AWS CLI `remove-source-identifier-from-subscription` command. Include the following required parameters:

- `--subscription-name`
- `--source-identifier`

Example

The following example removes the source identifier `mysqldb` from the `myrdseventsubscription` subscription.

For Linux, OS X, or Unix:

```
aws rds remove-source-identifier-from-subscription \
    --subscription-name myrdseventsubscription \
    --source-identifier mysqldb
```

For Windows:

```
aws rds remove-source-identifier-from-subscription ^
    --subscription-name myrdseventsubscription ^
    --source-identifier mysqldb
```

API

To remove a source identifier from an Amazon RDS event notification subscription, use the Amazon RDS API `RemoveSourceIdentifierFromSubscription` command. Include the following required parameters:

- `SubscriptionName`
- `SourceIdentifier`

Example

The following example removes the source identifier `mysqldb` from the `myrdseventsubscription` subscription.

```
https://rds.us-east-1.amazonaws.com/
```

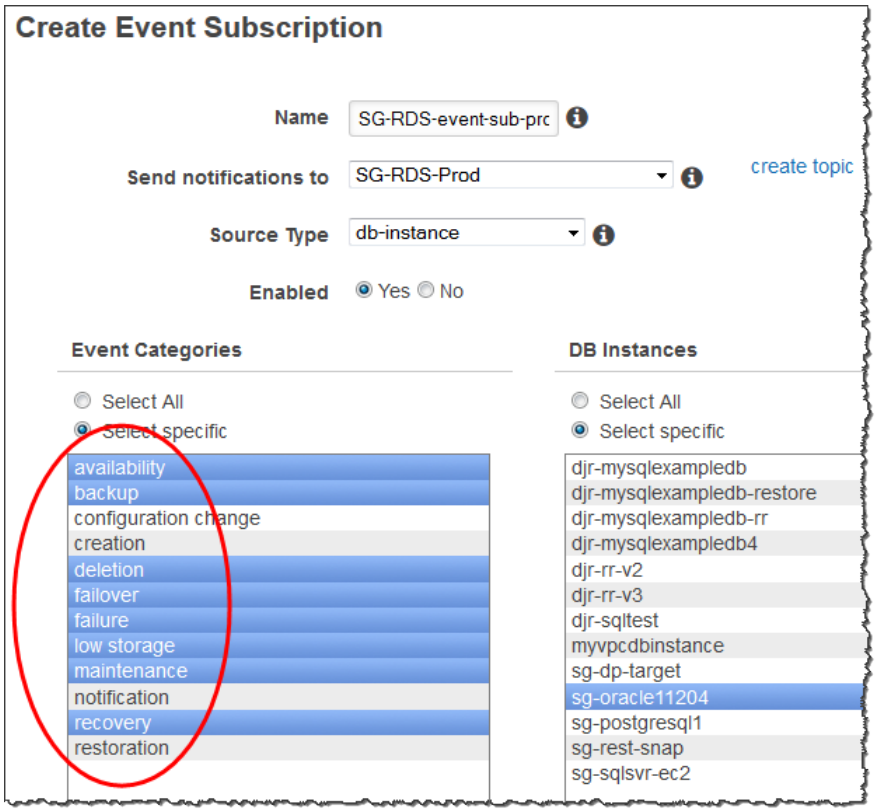
```
?Action=RemoveSourceIdentifierFromSubscription
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&SourceIdentifier=mysqldb
&SubscriptionName=myrdseventsubscription
&Version=2014-09-01
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20140428/us-east-1/rds/aws4_request
&X-Amz-Date=20140428T222718Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=4419f0015657ee120d781849ffdc6642eeafeee42bf1d18c4b2ed8eb732f7bf8
```

Listing the Amazon RDS Event Notification Categories

All events for a resource type are grouped into categories. To view the list of categories available, use the following procedures.

AWS Management Console

When you create or modify an event notification subscription, the event categories are displayed in the Amazon RDS console. See the topic Modifying an Amazon RDS Event Notification Subscription (p. 291) for more information.



CLI

To list the Amazon RDS event notification categories, use the AWS CLI `describe-event-categories` command. This command has no required parameters.

Example

```
aws rds describe-event-categories
```

API

To list the Amazon RDS event notification categories, use the Amazon RDS API `DescribeEventCategories` command. This command has no required parameters.

Example

```
https://rds.us-west-2.amazonaws.com/
   ?Action=DescribeEventCategories
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &Version=2014-09-01
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20140421/us-west-2/rds/aws4_request
   &X-Amz-Date=20140421T194732Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
   &X-Amz-Signature=6e25c542bf96fe24b28c12976ec92d2f856ab1d2a158e21c35441a736e4fde2b
```

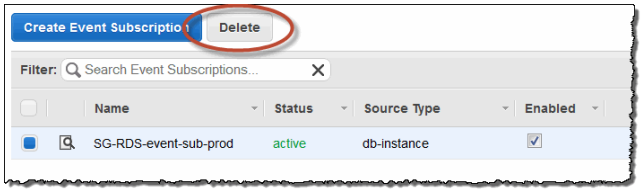
Deleting an Amazon RDS Event Notification Subscription

You can delete a subscription when you no longer need it. All subscribers to the topic will no longer receive event notifications specified by the subscription.

AWS Management Console

To delete an Amazon RDS event notification subscription

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the Amazon RDS Console navigation pane, click **DB Event Subscriptions**.
3. In the **My DB Event Subscriptions** pane, click the subscription that you want to delete.
4. Click **Delete**.
5. The Amazon RDS console indicates that the subscription is being deleted.



CLI

To delete an Amazon RDS event notification subscription, use the AWS CLI `delete-event-subscription` command. Include the following required parameter:

- `--subscription-name`

Example

The following example deletes the subscription `myrdssubscription`.

```
delete-event-subscription --subscription-name myrdssubscription
```

API

To delete an Amazon RDS event notification subscription, use the RDS API `DeleteEventSubscription` command. Include the following required parameter:

- `SubscriptionName`

Example

The following example deletes the subscription `myrdssubscription`.

```
https://rds.us-east-1.amazonaws.com/
   ?Action=DeleteEventSubscription
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
```

```
&SubscriptionName=myrdssubscription
&Version=2014-09-01
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20140423/us-east-1/rds/aws4_request
&X-Amz-Date=20140423T203337Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=05aa834e364a9e1a279d44cc955694518fc96fff638c74faa2be45783102e785
```

Viewing Amazon RDS Events

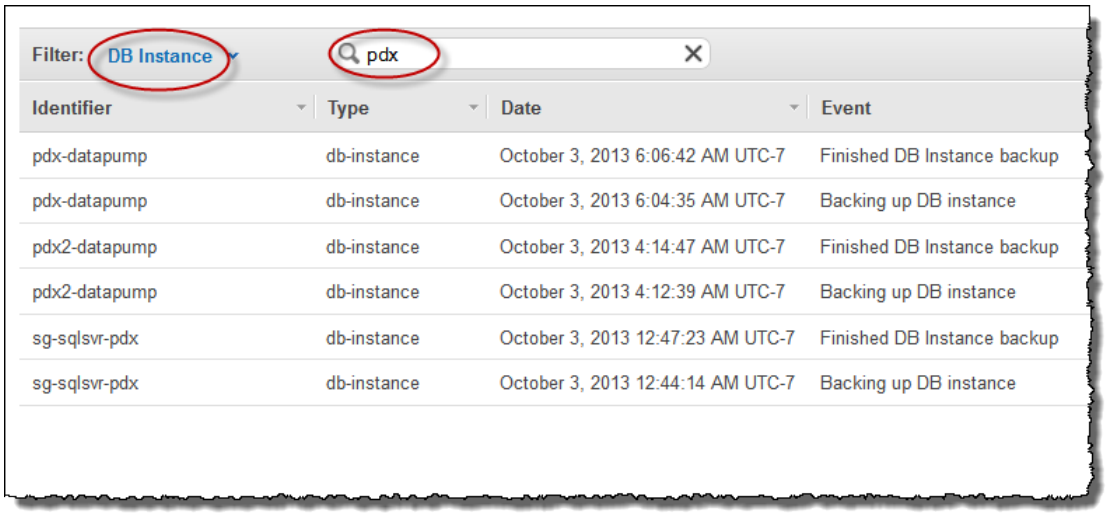
Amazon RDS keeps a record of events that relate to your DB instances, DB snapshots, DB security groups, and DB parameter groups. This information includes the date and time of the event, the source name and source type of the event, and a message associated with the event.

You can retrieve events for your RDS resources through the AWS Management Console, which shows events from the past 24 hours. You can also retrieve events for your RDS resources by using the describe-events AWS CLI command, or the DescribeEvents RDS API action. If you use the AWS CLI or the RDS API to view events, you can retrieve events for up to the past 14 days.

AWS Management Console

To view all Amazon RDS instance events for the past 24 hours

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Events**. The available events appear in a list.
3. Use the **Filter** list to filter the events by type, and use the text box to the right of the **Filter** list to further filter your results. For example, the following screenshot shows a list of events filtered by the DB instance event type and containing the letters **pdx**.



CLI

To view all Amazon RDS instance events for the past 7 days

You can view all Amazon RDS instance events for the past 7 days by calling the describe-events AWS CLI command and setting the `--duration` parameter to `10080`.

```
aws rds describe-events --duration 10080
```

API

To view all Amazon RDS instance events for the past 14 days

You can view all Amazon RDS instance events for the past 14 days by calling the DescribeEvents RDS API action and setting the `Duration` parameter to `20160`.

```
https://rds.us-west-2.amazonaws.com/
   ?Action=DescribeEvents
   &Duration=20160
   &MaxRecords=100
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &Version=2014-09-01
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20140421/us-west-2/rds/aws4_request
   &X-Amz-Date=20140421T194733Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
   &X-Amz-Signature=8e313cabcdbd9766c56a2886b5b298fd944e0b7cfa248953c82705fdd0374f27
```

Related Topics

- Using Amazon RDS Event Notification (p. 279)

Amazon RDS Database Log Files

You can view, download, and watch database logs using the Amazon RDS console, the AWS Command Line Interface (AWS CLI), or the Amazon RDS API. Viewing, downloading, or watching transaction logs is not supported.

For engine-specific documentation, see the following:

Database Engine	Relevant Documentation
MariaDB	You can access the error log, the slow query log, and the general log. For more information, see MariaDB Database Log Files (p. 306).
Microsoft SQL Server	You can access SQL Server error logs, agent logs, and trace files. For more information, see Microsoft SQL Server Database Log Files (p. 312).
MySQL	You can access the error log, the slow query log, and the general log. For more information, see MySQL Database Log Files (p. 313).
Oracle	You can access Oracle alert logs, audit files, and trace files. For more information, see Oracle Database Log Files (p. 318).
PostgreSQL	You can access query logs and error logs. Error logs can contain auto-vacuum and connection information, as well as rds_admin actions. For more information, see PostgreSQL Database Log Files (p. 322).

Viewing and Listing Database Log Files

You can view database log files for your DB engine by using the Amazon RDS console. You can list what log files are available for download or monitoring by using the Amazon RDS CLI or APIs.

> **Note**
> If you cannot view the list of log files for an existing Oracle DB instance, reboot the instance to view the list.

AWS Management Console

To view a database log file

1. Open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Instances**.
3. Choose the DB instance that has the log file you want to view, and then choose **Instance Actions | See Details**.
4. Choose the **Recent Events & Logs** tab.
5. In the **Logs** pane, choose the **View** button next to the log you want to view.

CLI

To list the available database log files for a DB instance use the AWS CLI `describe-db-log-files` command.

The following example directs a list of log files for a DB instance named `my-db-instance` to a text file called log_file_list.txt.

Example

```
aws rds describe-db-log-files --db-instance-identifier my-db-instance > log_file_list.txt
```

API

To list the available database log files for a DB instance call the Amazon RDS API `DescribeDBLogFiles` action.

Downloading a Database Log File

You can use the Amazon RDS console or the AWS CLI to download a database log file.

You can download a complete log file by using the DownloadCompleteDBLogFile (p. 1243) REST API. You can also download complete log files by using the `rds-download-db-logfile` RDS CLI command. For more information, see The `rds-download-db-logfile` Command (p. 1244).

AWS Management Console

To download a database log file

1. Open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Instances**.
3. Choose the DB instance that has the log file you want to view, and then choose **Instance Actions | See Details**.
4. Choose the **Recent Events & Logs** tab.
5. In the **Logs** pane, choose the **Download** button next to the log you want to download.
6. Right-click the link provided, and then choose **Save Link As...** from the dropdown menu. Type the location where you want the log file to be saved, then choose **Save**.

CLI

To download a database log file use the command `download-db-log-file-portion`.

The following example shows how to download the contents of a log file called log/ERROR.4 and store it in a local file called errorlog.txt.

Example

For Linux, OS X, or Unix:

```
aws rds download-db-log-file-portion \
```

```
    --db-instance-identifier myexampledb \
    --no-paginate \
    --log-file-name log/ERROR.4 > errorlog.txt
```

For Windows:

```
aws rds download-db-log-file-portion ^
    --db-instance-identifier myexampledb ^
    --no-paginate ^
    --log-file-name log/ERROR.4 > errorlog.txt
```

Watching a Database Log File

You can monitor the contents of a log file by using the Amazon RDS console.

AWS Management Console

To watch a database log file

1. Open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Instances**.
3. Choose the DB instance that has the log file you want to watch, and then choose **Instance Actions | See Details**.
4. Choose the **Recent Events & Logs** tab.
5. In the **Logs** pane, choose the **Watch** button next to the log you want to watch.

Related Topics

- Monitoring Amazon RDS (p. 245)
- Using Amazon RDS Event Notification (p. 279)

MariaDB Database Log Files

You can monitor the MariaDB error log, slow query log, and the general log. The MariaDB error log is generated by default; you can generate the slow query and general logs by setting parameters in your DB parameter group. Amazon RDS rotates all of the MariaDB log files; the intervals for each type are given following.

You can monitor the MariaDB logs directly through the Amazon RDS console, Amazon RDS API, Amazon RDS CLI, or AWS SDKs. You can also access MariaDB logs by directing the logs to a database table in the main database and querying that table. You can use the mysqlbinlog utility to download a binary log.

For more information about viewing, downloading, and watching file-based database logs, see Amazon RDS Database Log Files (p. 303).

Accessing MariaDB Error Logs

The MariaDB error log is written to the `<host-name>.err` file. You can view this file by using the Amazon RDS console or by retrieving the log using the Amazon RDS API, Amazon RDS CLI, or AWS SDKs. The `<host-name>.err` file is flushed every 5 minutes, and its contents are appended to `mysql-error-running.log`. The `mysql-error-running.log` file is then rotated every hour and the hourly files generated during the last 24 hours are retained. Each log file has the hour it was generated (in UTC) appended to its name. The log files also have a timestamp that helps you determine when the log entries were written.

MariaDB writes to the error log only on startup, shutdown, and when it encounters errors. A DB instance can go hours or days without new entries being written to the error log. If you see no recent entries, it's because the server did not encounter an error that resulted in a log entry.

Accessing the MariaDB Slow Query and General Logs

The MariaDB slow query log and the general log can be written to a file or a database table by setting parameters in your DB parameter group. For information about creating and modifying a DB parameter group, see Working with DB Parameter Groups (p. 170). You must set these parameters before you can view the slow query log or general log in the Amazon RDS console or by using the Amazon RDS API, Amazon RDS CLI, or AWS SDKs.

You can control MariaDB logging by using the parameters in this list:

- `slow_query_log`: To create the slow query log, set to 1. The default is 0.
- `general_log`: To create the general log, set to 1. The default is 0.
- `long_query_time`: To prevent fast-running queries from being logged in the slow query log, specify a value for the shortest query execution time to be logged, in seconds. The default is 10 seconds; the minimum is 0. If log_output = FILE, you can specify a floating point value that goes to microsecond resolution. If log_output = TABLE, you must specify an integer value with second resolution. Only queries whose execution time exceeds the `long_query_time` value are logged. For example, setting `long_query_time` to 0.1 prevents any query that runs for less than 100 milliseconds from being logged.
- `log_queries_not_using_indexes`: To log all queries that do not use an index to the slow query log, set this parameter to 1. The default is 0. Queries that do not use an index are logged even if their execution time is less than the value of the `long_query_time` parameter.
- `log_output` *`option`*: You can specify one of the following options for the `log_output` parameter:
  - **TABLE** (default)– Write general queries to the `mysql.general_log` table, and slow queries to the `mysql.slow_log` table.
  - **FILE**– Write both general and slow query logs to the file system. Log files are rotated hourly.

- **NONE**– Disable logging.

When logging is enabled, Amazon RDS rotates table logs or deletes log files at regular intervals. This measure is a precaution to reduce the possibility of a large log file either blocking database use or affecting performance. `FILE` and `TABLE` logging approach rotation and deletion as follows:

- When `FILE` logging is enabled, log files are examined every hour and log files older than 24 hours are deleted. If the remaining combined log file size after the deletion exceeds a threshold of 2 percent of a DB instance's allocated space, then the largest log files are deleted until the log file size no longer exceeds the threshold.
- When `TABLE` logging is enabled, log tables are rotated every 24 hours if the space used by the table logs is more than 20 percent of the allocated storage space or the size of all logs combined is greater than 10 GB. If the amount of space used for a DB instance is greater than 90 percent of the DB instance's allocated storage space, then the thresholds for log rotation are reduced. Log tables are then rotated if the space used by the table logs is more than 10 percent of the allocated storage space or the size of all logs combined is greater than 5 GB.

  When log tables are rotated, the current log table is copied to a backup log table and the entries in the current log table are removed. If the backup log table already exists, then it is deleted before the current log table is copied to the backup. You can query the backup log table if needed. The backup log table for the `mysql.general_log` table is named `mysql.general_log_backup`. The backup log table for the `mysql.slow_log` table is named `mysql.slow_log_backup`.

  You can rotate the `mysql.general_log` table by calling the `mysql.rds_rotate_general_log` procedure. You can rotate the `mysql.slow_log` table by calling the `mysql.rds_rotate_slow_log` procedure.

  Table logs are rotated during a database version upgrade.

Amazon RDS records both `TABLE` and `FILE` log rotation in an Amazon RDS event and sends you a notification.

To work with the logs from the Amazon RDS console, Amazon RDS API, Amazon RDS CLI, or AWS SDKs, set the `log_output` parameter to FILE. Like the MariaDB error log, these log files are rotated hourly. The log files that were generated during the previous 24 hours are retained.

For more information about the slow query and general logs, go to the following topics in the MariaDB documentation:

- Slow Query Log
- General Query Log

Log File Size

The MariaDB slow query log, error log, and the general log file sizes are constrained to no more than 2 percent of the allocated storage space for a DB instance. To maintain this threshold, logs are automatically rotated every hour and log files older than 24 hours are removed. If the combined log file size exceeds the threshold after removing old log files, then the largest log files are deleted until the log file size no longer exceeds the threshold.

Managing Table-Based MariaDB Logs

You can direct the general and slow query logs to tables on the DB instance by creating a DB parameter group and setting the `log_output` server parameter to `TABLE`. General queries are then logged to the `mysql.general_log` table, and slow queries are logged to the `mysql.slow_log` table. You can query

the tables to access the log information. Enabling this logging increases the amount of data written to the database, which can degrade performance.

Both the general log and the slow query logs are disabled by default. In order to enable logging to tables, you must also set the `general_log` and `slow_query_log` server parameters to `1`.

Log tables keep growing until the respective logging activities are turned off by resetting the appropriate parameter to `0`. A large amount of data often accumulates over time, which can use up a considerable percentage of your allocated storage space. Amazon RDS does not allow you to truncate the log tables, but you can move their contents. Rotating a table saves its contents to a backup table and then creates a new empty log table. You can manually rotate the log tables with the following command line procedures, where the command prompt is indicated by `PROMPT>`:

```
PROMPT> CALL mysql.rds_rotate_slow_log;
PROMPT> CALL mysql.rds_rotate_general_log;
```

To completely remove the old data and reclaim the disk space, call the appropriate procedure twice in succession.

Binary Logging Format

MariaDB on Amazon RDS supports the *row-based* and *mixed* binary log formats, and does not support the *statement-based* binary log format. The default binary logging format is *mixed*. For details on the different MariaDB binary log formats, see Binary Log Formats in the MariaDB documentation.

> **Important**
> Setting the binary logging format to row-based can result in very large binary log files. Large binary log files reduce the amount of storage available for a DB instance and can increase the amount of time to perform a restore operation of a DB instance.

To set the MariaDB binary logging format

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. Create a new parameter group, following the instructions in Creating a DB Parameter Group (p. 171).
3. Choose the new parameter group, and then choose **Go to Details Page**.
4. Choose **Edit Parameters** to modify the parameters in the DB parameter group.
5. Set the `binlog_format` parameter to the binary logging format of your choice, **MIXED** or **ROW**.
6. Choose **Save Changes** to save the updates to the DB parameter group.

For more information on DB parameter groups, see Working with DB Parameter Groups (p. 170).

Accessing MariaDB Binary Logs

You can use the mysqlbinlog utility to download binary logs in text format from MariaDB DB instances. The binary log is downloaded to your local computer. For more information about using the mysqlbinlog utility, go to Using mysqlbinlog in the MariaDB documentation.

To run the mysqlbinlog utility against an Amazon RDS instance, use the following options:

- Specify the `--read-from-remote-server` option.
- `--host`: Specify the DNS name from the endpoint of the instance.
- `--port`: Specify the port used by the instance.
- `--user`: Specify a MariaDB user that has been granted the replication slave permission.

- `--password`: Specify the password for the user, or omit a password value so the utility prompts you for a password.
- `--result-file`: Specify the local file that receives the output.
- Specify the names of one or more binary log files. To get a list of the available logs, use the SQL command SHOW BINARY LOGS.

For more information about mysqlbinlog options, go to mysqlbinlog Options in the MariaDB documentation.

The following is an example:

For Linux, OS X, or Unix:

```
mysqlbinlog \
    --read-from-remote-server \
    --host=mariadbinstance1.1234abcd.region.rds.amazonaws.com \
    --port=3306  \
    --user ReplUser \
    --password <password> \
    --result-file=/tmp/binlog.txt
```

For Windows:

```
mysqlbinlog ^
    --read-from-remote-server ^
    --host=mariadbinstance1.1234abcd.region.rds.amazonaws.com ^
    --port=3306  ^
    --user ReplUser ^
    --password <password> ^
    --result-file=/tmp/binlog.txt
```

Amazon RDS normally purges a binary log as soon as possible, but the binary log must still be available on the instance to be accessed by mysqlbinlog. To specify the number of hours for RDS to retain binary logs, use the `mysql.rds_set_configuration` stored procedure and specify a period with enough time for you to download the logs. After you set the retention period, monitor storage usage for the DB instance to ensure that the retained binary logs do not take up too much storage.

The following example sets the retention period to 1 day:

```
call mysql.rds_set_configuration('binlog retention hours', 24);
```

To display the current setting, use the `mysql.rds_show_configuration` stored procedure:

```
call mysql.rds_show_configuration;
```

Binary Log Annotation

In a MariaDB DB instance, you can use the `Annotate_rows` event to annotate a row event with a copy of the SQL query that caused the row event. This approach provides similar functionality to enabling the `binlog_rows_query_log_events` parameter on a DB instance on MySQL version 5.6 or later.

You can enable binary log annotations globally by creating a custom parameter group and setting the `binlog_annotate_row_events` parameter to **1**. You can also enable annotations at the session level, by calling `SET SESSION binlog_annotate_row_events = 1`. Use the `replicate_annotate_row_events` to replicate binary log annotations to the slave instance if binary logging is enabled on it. No special privileges are required to use these settings.

The following is an example of a row-based transaction in MariaDB. The use of row-based logging is triggered by setting the transaction isolation level to read-committed.

```
CREATE DATABASE IF NOT EXISTS test;
USE test;
CREATE TABLE square(x INT PRIMARY KEY, y INT NOT NULL) ENGINE = InnoDB;
SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
BEGIN
INSERT INTO square(x, y) VALUES(5, 5 * 5);
COMMIT;
```

Without annotations, the binary log entries for the transaction look like the following:

```
BEGIN
/*!*/;
# at 1163
# at 1209
#150922  7:55:57 server id 1855786460  end_log_pos 1209         Table_map: `test`.`square`
 mapped to number 76
#150922  7:55:57 server id 1855786460  end_log_pos 1247         Write_rows: table id 76
 flags: STMT_END_F
### INSERT INTO `test`.`square`
### SET
###   @1=5
###   @2=25
# at 1247
#150922  7:56:01 server id 1855786460  end_log_pos 1274         Xid = 62
COMMIT/*!*/;
```

The following statement enables session-level annotations for this sames transaction, and disables them after committing the transaction:

```
CREATE DATABASE IF NOT EXISTS test;
USE test;
CREATE TABLE square(x INT PRIMARY KEY, y INT NOT NULL) ENGINE = InnoDB;
SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
SET SESSION binlog_annotate_row_events = 1;
BEGIN;
INSERT INTO square(x, y) VALUES(5, 5 * 5);
COMMIT;
SET SESSION binlog_annotate_row_events = 0;
```

With annotations, the binary log entries for the transaction look like the following:

```
BEGIN
/*!*/;
# at 423
# at 483
# at 529
#150922  8:04:24 server id 1855786460  end_log_pos 483  Annotate_rows:
#Q> INSERT INTO square(x, y) VALUES(5, 5 * 5)
#150922  8:04:24 server id 1855786460  end_log_pos 529  Table_map: `test`.`square` mapped
 to number 76
#150922  8:04:24 server id 1855786460  end_log_pos 567  Write_rows: table id 76 flags:
 STMT_END_F
### INSERT INTO `test`.`square`
### SET
###   @1=5
###   @2=25
# at 567
#150922  8:04:26 server id 1855786460  end_log_pos 594  Xid = 88
```

```
COMMIT/*!*/;
```

Microsoft SQL Server Database Log Files

You can access Microsoft SQL Server error logs, agent logs, trace files, and dump files by using the Amazon RDS console or APIs. For more information about viewing, downloading, and watching file-based database logs, see Amazon RDS Database Log Files (p. 303).

Retention Schedule

Log files are rotated each day and whenever your DB instance is restarted. The following is the retention schedule for Microsoft SQL Server logs on Amazon RDS.

Log Type	Retention Schedule
Error logs	A maximum of 30 error logs are retained. Amazon RDS may delete error logs older than 7 days.
Agent logs	A maximum of 10 agent logs are retained. Amazon RDS may delete agent logs older than 7 days.
Trace files	Trace files are retained according to the trace file retention period of your DB instance. The default trace file retention period is 7 days. To modify the trace file retention period for your DB instance, see Setting the Retention Period for Trace and Dump Files (p. 810).
Dump files	Dump files are retained according to the dump file retention period of your DB instance. The default dump file retention period is 7 days. To modify the dump file retention period for your DB instance, see Setting the Retention Period for Trace and Dump Files (p. 810).

Viewing the SQL Server Error Log by Using the rds_read_error_log Procedure

You can use the Amazon RDS stored procedure `rds_read_error_log` to view error logs and agent logs. For more information, see Using the rds_read_error_log Procedure (p. 809).

Related Topics

- Using SQL Server Agent (p. 808)
- Working with Microsoft SQL Server Logs (p. 809)
- Working with Trace and Dump Files (p. 810)

MySQL Database Log Files

You can monitor the MySQL error log, slow query log, and the general log. The MySQL error log is generated by default; you can generate the slow query and general logs by setting parameters in your DB parameter group. Amazon RDS rotates all of the MySQL log files; the intervals for each type are given following.

You can monitor the MySQL logs directly through the Amazon RDS console, Amazon RDS API, Amazon RDS CLI, or AWS SDKs. You can also access MySQL logs by directing the logs to a database table in the main database and querying that table. You can use the mysqlbinlog utility to download a binary log.

For more information about viewing, downloading, and watching file-based database logs, see Amazon RDS Database Log Files (p. 303).

Accessing MySQL Error Logs

The MySQL error log is written to the `mysql-error.log` file. You can view `mysql-error.log` by using the Amazon RDS console or by retrieving the log using the Amazon RDS API, Amazon RDS CLI, or AWS SDKs. `mysql-error.log` is flushed every 5 minutes, and its contents are appended to `mysql-error-running.log`. The `mysql-error-running.log` file is then rotated every hour and the hourly files generated during the last 24 hours are retained. Each log file has the hour it was generated (in UTC) appended to its name. The log files also have a timestamp that helps you determine when the log entries were written.

MySQL writes to the error log only on startup, shutdown, and when it encounters errors. A DB instance can go hours or days without new entries being written to the error log. If you see no recent entries, it's because the server did not encounter an error that would result in a log entry.

Accessing the MySQL Slow Query and General Logs

The MySQL slow query log and the general log can be written to a file or a database table by setting parameters in your DB parameter group. For information about creating and modifying a DB parameter group, see Working with DB Parameter Groups (p. 170). You must set these parameters before you can view the slow query log or general log in the Amazon RDS console or by using the Amazon RDS API, Amazon RDS CLI, or AWS SDKs.

You can control MySQL logging by using the parameters in this list:

- `slow_query_log`: To create the slow query log, set to 1. The default is 0.
- `general_log`: To create the general log, set to 1. The default is 0.
- `long_query_time`: To prevent fast-running queries from being logged in the slow query log, specify a value for the shortest query execution time to be logged, in seconds. The default is 10 seconds, the minimum is 0. If log_output = FILE, you can specify a floating point value that goes to microsecond resolution. If log_output = TABLE, you must specify an integer value with second resolution. Only queries whose execution time exceeds the `long_query_time` value are logged. For example, setting `long_query_time` to 0.1 prevents any query that runs for less than 100 milliseconds from being logged.
- `log_queries_not_using_indexes`: To log all queries that do not use an index to the slow query log, set to 1. The default is 0. Queries that do not use an index are logged even if their execution time is less than the value of the `long_query_time` parameter.
- `log_output` *`option`*: You can specify one of the following options for the `log_output` parameter.
  - **TABLE** (default)– Write general queries to the `mysql.general_log` table, and slow queries to the `mysql.slow_log` table.
  - **FILE**– Write both general and slow query logs to the file system. Log files are rotated hourly.

- **NONE**– Disable logging.

When logging is enabled, Amazon RDS rotates table logs or deletes log files at regular intervals. This measure is a precaution to reduce the possibility of a large log file either blocking database use or affecting performance. `FILE` and `TABLE` logging approach rotation and deletion as follows:

- When `FILE` logging is enabled, log files are examined every hour and log files older than 24 hours are deleted. If the remaining combined log file size after the deletion exceeds a threshold of 2 percent of a DB instance's allocated space, then the largest log files are deleted until the log file size no longer exceeds the threshold.
- When `TABLE` logging is enabled, log tables are rotated every 24 hours if the space used by the table logs is more than 20 percent of the allocated storage space or the size of all logs combined is greater than 10 GB. If the amount of space used for a DB instance is greater than 90 percent of the DB instance's allocated storage space, then the thresholds for log rotation are reduced. Log tables are then rotated if the space used by the table logs is more than 10 percent of the allocated storage space or the size of all logs combined is greater than 5 GB. You can subscribe to the `low_free_storage` event to be notified when log tables are rotated to free up space. For more information, see Using Amazon RDS Event Notification (p. 279).

  When log tables are rotated, the current log table is copied to a backup log table and the entries in the current log table are removed. If the backup log table already exists, then it is deleted before the current log table is copied to the backup. You can query the backup log table if needed. The backup log table for the `mysql.general_log` table is named `mysql.general_log_backup`. The backup log table for the `mysql.slow_log` table is named `mysql.slow_log_backup`.

  You can rotate the `mysql.general_log` table by calling the `mysql.rds_rotate_general_log` procedure. You can rotate the `mysql.slow_log` table by calling the `mysql.rds_rotate_slow_log` procedure.

  Table logs are rotated during a database version upgrade.

To work with the logs from the Amazon RDS console, Amazon RDS API, Amazon RDS CLI, or AWS SDKs, set the `log_output` parameter to FILE. Like the MySQL error log, these log files are rotated hourly. The log files that were generated during the previous 24 hours are retained.

For more information about the slow query and general logs, go to the following topics in the MySQL documentation:

- The Slow Query Log
- The General Query Log

Log File Size

The MySQL slow query log, error log, and the general log file sizes are constrained to no more than 2 percent of the allocated storage space for a DB instance. To maintain this threshold, logs are automatically rotated every hour and log files older than 24 hours are removed. If the combined log file size exceeds the threshold after removing old log files, then the largest log files are deleted until the log file size no longer exceeds the threshold.

For MySQL version 5.6.20 and later, there is a size limit on BLOBs written to the redo log. To account for this limit, ensure that the `innodb_log_file_size` parameter for your MySQL DB instance is 10 times larger than the largest BLOB data size found in your tables, plus the length of other variable length fields (`VARCHAR`, `VARBINARY`, `TEXT`) in the same tables. For information on how to set parameter values, see Working with DB Parameter Groups (p. 170). For information on the redo log BLOB size limit, go to Changes in MySQL 5.6.20.

Managing Table-Based MySQL Logs

You can direct the general and slow query logs to tables on the DB instance by creating a DB parameter group and setting the `log_output` server parameter to `TABLE`. General queries are then logged to the `mysql.general_log` table, and slow queries are logged to the `mysql.slow_log` table. You can query the tables to access the log information. Enabling this logging increases the amount of data written to the database, which can degrade performance.

Both the general log and the slow query logs are disabled by default. In order to enable logging to tables, you must also set the `general_log` and `slow_query_log` server parameters to `1`.

Log tables will keep growing until the respective logging activities are turned off by resetting the appropriate parameter to `0`. A large amount of data often accumulates over time, which can use up a considerable percentage of your allocated storage space. Amazon RDS does not allow you to truncate the log tables, but you can move their contents. Rotating a table saves its contents to a backup table and then creates a new empty log table. You can manually rotate the log tables with the following command line procedures, where the command prompt is indicated by `PROMPT>`:

```
PROMPT> CALL mysql.rds_rotate_slow_log;
PROMPT> CALL mysql.rds_rotate_general_log;
```

To completely remove the old data and reclaim the disk space, call the appropriate procedure twice in succession.

Binary Logging Format

MySQL on Amazon RDS supports both the *row-based* and *mixed* binary logging formats for MySQL version 5.6 and later. The default binary logging format is mixed. For DB instances running MySQL versions 5.1 and 5.5, only mixed binary logging is supported. For details on the different MySQL binary log formats, see Binary Logging Formats in the *MySQL Reference Manual*.

> **Important**
> Setting the binary logging format to row-based can result in very large binary log files. Large binary log files reduce the amount of storage available for a DB instance and can increase the amount of time to perform a restore operation of a DB instance.

To set the MySQL binary logging format:

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. Click **Parameter Groups** in the left pane.
3. For the `default.mysql5.6` or `default.mysql5.7` DB parameter group, click the **Go to Details Page** icon.
4. Click the **Edit Parameters** button to modify the parameters in the DB parameter group.
5. Set the `binlog_format` parameter to the binary logging format of your choice (**MIXED** or **ROW**).
6. Click the **Save Changes** button to save the updates to the DB parameter group.

> **Important**
> Changing the `default.mysql5.6` or `default.mysql5.7` DB parameter group affects all MySQL version 5.6 DB instances that use that parameter group. If you want to specify different binary logging formats for different MySQL 5.6 or 5.7 DB instances in a region, you will need to create your own DB parameter group that identifies the different logging format and assign that DB parameter group to the intended DB instances.

For more information on DB parameter groups, see Working with DB Parameter Groups (p. 170).

Accessing MySQL Binary Logs

You can use the mysqlbinlog utility to download or stream binary logs from Amazon RDS instances running MySQL 5.6 or later. The binary log is downloaded to your local computer, where you can perform actions such as replaying the log using the mysql utility. For more information about using the mysqlbinlog utility, go to Using mysqlbinlog to Back Up Binary Log Files.

To run the mysqlbinlog utility against an Amazon RDS instance, use the following options:

- Specify the `--read-from-remote-server` option.
- `--host`: Specify the DNS name from the endpoint of the instance.
- `--port`: Specify the port used by the instance.
- `--user`: Specify a MySQL user that has been granted the replication slave permission.
- `--password`: Specify the password for the user, or omit a password value so the utility will prompt you for a password.
- To have the file downloaded in binary format, specify the `--raw` option.
- `--result-file`: Specify the local file that will receive the raw output.
- Specify the names of one or more binary log files. To get a list of the available logs, use the SQL command SHOW BINARY LOGS.
- To stream the binary log files, specify the `--stop-never` option.

For more information about mysqlbinlog options, go to mysqlbinlog - Utility for Processing Binary Log Files.

For example:

For Linux, OS X, or Unix:

```
mysqlbinlog \
    --read-from-remote-server \
    --host=MySQL56Instance1.cg034hpkmmjt.region.rds.amazonaws.com \
    --port=3306  \
    --user ReplUser \
    --password \
    --raw \
    --result-file=/tmp/ \
    binlog.00098
```

For Windows:

```
mysqlbinlog ^
    --read-from-remote-server ^
    --host=MySQL56Instance1.cg034hpkmmjt.region.rds.amazonaws.com ^
    --port=3306  ^
    --user ReplUser ^
    --password ^
    --raw ^
    --result-file=/tmp/ ^
    binlog.00098
```

Amazon RDS normally purges a binary log as soon as possible, but the binary log must still be available on the instance to be accessed by mysqlbinlog. To specify the number of hours for RDS to retain binary logs, use the `mysql.rds_set_configuration` stored procedure and specify a period with enough time for you to download the logs. After you set the retention period, monitor storage usage for the DB instance to ensure that the retained binary logs do not take up too much storage.

Note
The `mysql.rds_set_configuration` stored procedure is only available for MySQL version 5.6 or later.

This example sets the retention period to 1 day:

```
call mysql.rds_set_configuration('binlog retention hours', 24);
```

To display the current setting, use the `mysql.rds_show_configuration` stored procedure:

```
call mysql.rds_show_configuration;
```

Oracle Database Log Files

You can access Oracle alert logs, audit files, and trace files by using the Amazon RDS console or APIs. For more information about viewing, downloading, and watching file-based database logs, see Amazon RDS Database Log Files (p. 303).

The Oracle audit files provided are the standard Oracle auditing files. While Fine Grained Auditing (FGA) is a supported feature, log access does not provide access to FGA events stored in the SYS.FGA_LOG$ table and that are accessible through the DBA_FGA_AUDIT_TRAIL view.

The `DescribeDBLogFiles` API action that lists the Oracle log files that are available for a DB instance ignores the `MaxRecords` parameter and returns up to 1000 records.

Retention Schedule

The Oracle database engine may rotate logs files if they get very large. If you want to retain audit or trace files, you should download them. Storing the files locally reduces your Amazon RDS storage costs and makes more space available for your data.

The following is the retention schedule for Oracle alert logs, audit files, and trace files on Amazon RDS.

Log Type	Retention Schedule
Alert Logs	The default retention period for alert logs is 30 days. Amazon RDS may delete alert logs older than 30 days. Oracle rotates alert logs when they exceed 10MB, at which point they will be unavailable from the Amazon RDS views.
Audit Files	The default retention period for audit files is 7 days. Amazon RDS may delete audit files older than 7 days.
Trace files	The default retention period for trace files is 7 days. Amazon RDS may delete trace files older than 7 days.

Switching Online Log files

You can use the Amazon RDS procedure `rdsadmin.rdsadmin_util.switch_logfile` switch online log files. For more information, see Switching Online Log Files (p. 1067).

Retrieving Archived Redo Logs

You can retain archived redo logs. For more information, see Retaining Archived Redo Logs (p. 1070).

Working with Oracle Trace Files

This section describes Amazon RDS-specific procedures to create, refresh, access, and delete trace files.

Listing Files

Two procedures are available to allow access to any file within the `background_dump_dest`. The first method refreshes a view containing a listing of all files currently in the `background_dump_dest`:

```
exec rdsadmin.manage_tracefiles.refresh_tracefile_listing;
```

Once the view is refreshed, use the following view to access the results.

```
rdsadmin.tracefile_listing
```

An alternative to the previous process is to use "from table" to stream non-table data in a table-like format to list DB directory contents:

```
SELECT * FROM table(rdsadmin.rds_file_util.listdir('BDUMP'));
```

The following query shows text of a log file:

```
SELECT text FROM table(rdsadmin.rds_file_util.read_text_file('BDUMP','alert_xxx.log'));
```

Generating Trace Files and Tracing a Session

Since there are no restrictions on `alter session`, many standard methods to generate trace files in Oracle remain available to an Amazon RDS DB instance. The following procedures are provided for trace files that require greater access.

Oracle Method	Amazon RDS Method
`oradebug hanganalyze 3`	`exec rdsadmin.manage_tracefiles.hanganalyze;`
`oradebug dump systemstate 266`	`exec rdsadmin.manage_tracefiles.dump_systemstate;`

You can use many standard methods to trace individual sessions connected to an Oracle DB instance in Amazon RDS. To enable tracing for a session, you can run subprograms in PL/SQL packages supplied by Oracle, such as the DBMS_SESSION and DBMS_MONITOR packages. For more information, see Enabling Tracing for a Session in the Oracle documentation.

Retrieving Trace Files

You can retrieve any trace file in `background_dump_dest` using a standard SQL query of an Amazon RDS managed external table. To use this method, you must execute the procedure to set the location for this table to the specific trace file.

For example, you can use the `rdsadmin.tracefile_listing` view mentioned above to list the all of the trace files on the system. You can then set the tracefile_table view to point to the intended trace file using the following procedure:

```
exec
 rdsadmin.manage_tracefiles.set_tracefile_table_location('CUST01_ora_3260_SYSTEMSTATE.trc');
```

The following example creates an external table in the current schema with the location set to the file provided. The contents can be retrieved into a local file using a SQL query.

```
# eg: send the contents of the tracefile to a local file:
sqlplus user/password@TNS alias << EOF > /tmp/tracefile.txt
select * from tracefile_table;
```

```
EOF
```

Purging Trace Files

Trace files can accumulate and consume disk space. Amazon RDS purges trace files by default and log files that are older than seven days. You can view and set the trace file retention period using the show_configuration procedure. Note that you should run the command `SET SERVEROUTPUT ON` so that you can view the configuration results.

The following example shows the current trace file retention period, and then sets a new trace file retention period.

```
# Show the current tracefile retention
SQL> exec rdsadmin.rdsadmin_util.show_configuration;
NAME:tracefile retention
VALUE:10080
DESCRIPTION:tracefile expiration specifies the duration in minutes before tracefiles in
 bdump are automatically deleted.

# Set the tracefile retention to 24 hours:
SQL> exec rdsadmin.rdsadmin_util.set_configuration('tracefile retention',1440);

#show the new tracefile retention
SQL> exec rdsadmin.rdsadmin_util.show_configuration;
NAME:tracefile retention
VALUE:1440
DESCRIPTION:tracefile expiration specifies the duration in minutes before tracefiles in
 bdump are automatically deleted.
```

In addition to the periodic purge process, you can manually remove files from the `background_dump_dest`. The following example shows how to purge all files older than five minutes.

```
exec rdsadmin.manage_tracefiles.purge_tracefiles(5);
```

You can also purge all files that match a specific pattern (do not include the file extension such as .trc). The following example shows how to purge all files that start with "SCHPOC1_ora_5935".

```
exec rdsadmin.manage_tracefiles.purge_tracefiles('SCHPOC1_ora_5935');
```

Previous Methods for Accessing Alert Logs and Listener Logs

You can view the alert log using the Amazon RDS console. You can also use the following SQL statement to access the alert log:

```
select message_text from alertlog;
```

To access the listener log, use the following SQL statement:

```
select message_text from listenerlog;
```

Note
Oracle rotates the alert and listener logs when they exceed 10MB, at which point they will be unavailable from the Amazon RDS views.

Related Topics

- Common DBA Log Tasks for Oracle DB Instances (p. 1065)

PostgreSQL Database Log Files

RDS PostgreSQL generates query and error logs. We write auto-vacuum info and rds_admin actions to the error log. Postgres also logs connections/disconnections/checkpoints to the error log. For more information, see http://www.postgresql.org/docs/9.4/static/runtime-config-logging.html

You can set the retention period for system logs using the `rds.log_retention_period` parameter in the DB parameter group associated with your DB instance. The unit for this parameter is minutes; for example, a setting of 1440 would retain logs for one day. The default value is 4320 (three days). The maximum value is 10080 (seven days). Note that your instance must have enough allocated storage to contain the retained log files.

You can enable query logging for your PostgreSQL DB instance by setting two parameters in the DB parameter group associated with your DB instance: `log_statement` and `log_min_duration_statement`. The `log_statement` parameter controls which SQL statements are logged. We recommend setting this parameter to *all* to log all statements; the default value is *none*. Alternatively, you can set this value to *ddl* to log all data definition language (DDL) statements (CREATE, ALTER, DROP, etc.) or to *mod* to log all DDL and data modification language (DML) statements (INSERT, UPDATE, DELETE, etc.).

The `log_min_duration_statement` parameter sets the limit in milliseconds of a statement to be logged. All SQL statements that run longer than the parameter setting are logged. This parameter is disabled and set to minus 1 (-1) by default. Enabling this parameter can help you find unoptimized queries. For more information on these settings, see Error Reporting and Logging in the PostgreSQL documentation.

If you are new to setting parameters in a DB parameter group and associating that parameter group with a DB instance, see Working with DB Parameter Groups (p. 170)

The following steps show how to set up query logging:

1. Set the `log_statement` parameter to *all*. The following example shows the information that is written to the postgres.log file:

```
2013-11-05 16:48:56 UTC::@:[2952]:LOG:  received SIGHUP, reloading configuration files
2013-11-05 16:48:56 UTC::@:[2952]:LOG:  parameter "log_min_duration_statement" changed to
 "1"
```

   Additional information is written to the postgres.log file when you execute a query. The following example shows the type of information written to the file after a query:

```
2013-11-05 16:41:07 UTC::@:[2955]:LOG:  checkpoint starting: time
2013-11-05 16:41:07 UTC::@:[2955]:LOG:  checkpoint complete: wrote 1 buffers (0.3%);
 0 transaction log file(s) added, 0 removed, 1 recycled; write=0.000 s, sync=0.003 s,
 total=0.012 s; sync files=1, longest=0.003 s, average=0.003 s
2013-11-05 16:45:14 UTC:[local]:master@postgres:[8839]:LOG:  statement: SELECT d.datname
 as "Name",
        pg_catalog.pg_get_userbyid(d.datdba) as "Owner",
        pg_catalog.pg_encoding_to_char(d.encoding) as "Encoding",
        d.datcollate as "Collate",
        d.datctype as "Ctype",
        pg_catalog.array_to_string(d.datacl, E'\n') AS "Access privileges"
 FROM pg_catalog.pg_database d
 ORDER BY 1;
2013-11-05 16:45:
```

2. Set the `log_min_duration_statement` parameter. The following example shows the information that is written to the postgres.log file when the parameter is set to *1*:

```
2013-11-05 16:48:56 UTC::@:[2952]:LOG:  received SIGHUP, reloading configuration files
2013-11-05 16:48:56 UTC::@:[2952]:LOG:  parameter "log_min_duration_statement" changed to
 "1"
```

Additional information is written to the postgres.log file when you execute a query that exceeds the duration parameter setting. The following example shows the type of information written to the file after a query:

```
2013-11-05 16:51:10 UTC:[local]:master@postgres:[9193]:LOG:  statement: SELECT
 c2.relname, i.indisprimary, i.indisunique, i.indisclustered, i.indisvalid,
 pg_catalog.pg_get_indexdef(i.indexrelid, 0, true),
   pg_catalog.pg_get_constraintdef(con.oid, true), contype, condeferrable, condeferred,
 c2.reltablespace
 FROM pg_catalog.pg_class c, pg_catalog.pg_class c2, pg_catalog.pg_index i
   LEFT JOIN pg_catalog.pg_constraint con ON (conrelid = i.indrelid AND conindid =
 i.indexrelid AND contype IN ('p','u','x'))
 WHERE c.oid = '1255' AND c.oid = i.indrelid AND i.indexrelid = c2.oid
 ORDER BY i.indisprimary DESC, i.indisunique DESC, c2.relname;
2013-11-05 16:51:10 UTC:[local]:master@postgres:[9193]:LOG:  duration: 3.367 ms
2013-11-05 16:51:10 UTC:[local]:master@postgres:[9193]:LOG:  statement: SELECT
 c.oid::pg_catalog.regclass FROM pg_catalog.pg_class c, pg_catalog.pg_inherits i WHERE
 c.oid=i.inhparent AND i.inhrelid = '1255' ORDER BY inhseqno;
2013-11-05 16:51:10 UTC:[local]:master@postgres:[9193]:LOG:  duration: 1.002 ms
2013-11-05 16:51:10 UTC:[local]:master@postgres:[9193]:LOG:  statement:
 SELECT c.oid::pg_catalog.regclass FROM pg_catalog.pg_class c,
 pg_catalog.pg_inherits i WHERE c.oid=i.inhrelid AND i.inhparent = '1255' ORDER BY
 c.oid::pg_catalog.regclass::pg_catalog.text;
2013-11-05 16:51:18 UTC:[local]:master@postgres:[9193]:LOG:  statement: select proname
 from pg_proc;
2013-11-05 16:51:18 UTC:[local]:master@postgres:[9193]:LOG:  duration: 3.469 ms
```

Logging Amazon RDS API Calls Using AWS CloudTrail

AWS CloudTrail is a service that logs all Amazon RDS API calls made by or on behalf of your AWS account. The logging information is stored in an Amazon S3 bucket. You can use the information collected by CloudTrail to monitor activity for your Amazon RDS DB instances. For example, you can determine whether a request completed successfully and which user made the request. To learn more about CloudTrail, see the AWS CloudTrail User Guide.

If an action is taken on behalf of your AWS account using the Amazon RDS console or the Amazon RDS command line interface, then AWS CloudTrail will log the action as calls made to the Amazon RDS API. For example, if you use the Amazon RDS console to modify a DB instance, or call the AWS CLI modify-db-instance command, then the AWS CloudTrail log will show a call to the Amazon RDS API ModifyDBInstance action. For a list of the Amazon RDS API actions that are logged by AWS CloudTrail, go to Amazon RDS API Reference.

Note
AWS CloudTrail only logs events for Amazon RDS API calls. If you want to audit actions taken on your database that are not part of the Amazon RDS API, such as when a user connects to your database or when a change is made to your database schema, then you will need to use the monitoring capabilities provided by your DB engine.

Configuring CloudTrail Event Logging

CloudTrail creates audit trails in each region separately and stores them in an Amazon S3 bucket. You can configure CloudTrail to use Amazon SNS to notify you when a log file is created, but that is optional. CloudTrail will notify you frequently, so we recommend that you use Amazon SNS in conjunction with an Amazon SQS queue and handle notifications programmatically.

You can enable CloudTrail using the AWS Management Console, AWS CLI, or API. When you enable CloudTrail logging, you can have the CloudTrail service create an Amazon S3 bucket for you to store your log files. For details, see Creating and Updating Your Trail in the *AWS CloudTrail User Guide*. The *AWS CloudTrail User Guide* also contains information on how to aggregate CloudTrail logs from multiple regions into a single Amazon S3 bucket.

There is no cost to use the CloudTrail service. However, standard rates for Amazon S3 usage apply as well as rates for Amazon SNS usage should you include that option. For pricing details, see the Amazon S3 and Amazon SNS pricing pages.

Amazon RDS Event Entries in CloudTrail Log Files

CloudTrail log files contain event information formatted using JSON. An event record represents a single AWS API call and includes information about the requested action, the user that requested the action, the date and time of the request, and so on.

CloudTrail log files include events for all AWS API calls for your AWS account, not just calls to the Amazon RDS API. However, you can read the log files and scan for calls to the Amazon RDS API using the `eventName` element.

The following example shows a CloudTrail log for a user that created a snapshot of a DB instance and then deleted that instance using the Amazon RDS console. The console is identified by the `userAgent` element. The requested API calls made by the console (`CreateDBSnapshot` and `DeleteDBInstance`) are found in the `eventName` element for each record. Information about the user (`Alice`) can be found in the `userIdentity` element.

```
{
```

```
  Records:[
  {
    "awsRegion":"us-west-2",
    "eventName":"CreateDBSnapshot",
    "eventSource":"rds.amazonaws.com",
    "eventTime":"2014-01-14T16:23:49Z",
    "eventVersion":"1.0",
    "sourceIPAddress":"192.0.2.01",
    "userAgent":"AWS Console, aws-sdk-java\/unknown-version Linux\/2.6.18-kaos_fleet-1108-
prod.2 Java_HotSpot(TM)_64-Bit_Server_VM\/24.45-b08",
    "userIdentity":
    {
      "accessKeyId":"AKIADQKE4SARGYLE",
      "accountId":"123456789012",
      "arn":"arn:aws:iam::123456789012:user/Alice",
      "principalId":"AIDAI2JXM4FBZZEXAMPLE",
      "sessionContext":
      {
        "attributes":
        {
          "creationDate":"2014-01-14T15:55:59Z",
          "mfaAuthenticated":false
        }
      },
      "type":"IAMUser",
      "userName":"Alice"
    }
  },
  {
    "awsRegion":"us-west-2",
    "eventName":"DeleteDBInstance",
    "eventSource":"rds.amazonaws.com",
    "eventTime":"2014-01-14T16:28:27Z",
    "eventVersion":"1.0",
    "sourceIPAddress":"192.0.2.01",
    "userAgent":"AWS Console, aws-sdk-java\/unknown-version Linux\/2.6.18-kaos_fleet-1108-
prod.2 Java_HotSpot(TM)_64-Bit_Server_VM\/24.45-b08",
    "userIdentity":
    {
      "accessKeyId":"AKIADQKE4SARGYLE",
      "accountId":"123456789012",
      "arn":"arn:aws:iam::123456789012:user/Alice",
      "principalId":"AIDAI2JXM4FBZZEXAMPLE",
      "sessionContext":
      {
        "attributes":
        {
          "creationDate":"2014-01-14T15:55:59Z",
          "mfaAuthenticated":false
        }
      },
      "type":"IAMUser",
      "userName":"Alice"
    }
  }
  ]
}
```


For more information about the different elements and values in CloudTrail log files, see CloudTrail Event Reference in the *AWS CloudTrail User Guide*.

You may also want to make use of one of the Amazon partner solutions that integrate with CloudTrail to read and analyze your CloudTrail log files. For options, see the AWS partners page.

Security in Amazon RDS

You can manage access to your Amazon Relational Database Service (Amazon RDS) resources and your databases on a DB instance. The method you use to manage access depends on what type of task the user needs to perform with Amazon RDS:

- Run your DB instance in an Amazon Virtual Private Cloud (VPC) for the greatest possible network access control. For more information about creating a DB instance in a VPC, see Using Amazon RDS with Amazon Virtual Private Cloud (VPC).
- Use AWS Identity and Access Management (IAM) policies to assign permissions that determine who is allowed to manage RDS resources. For example, you can use IAM to determine who is allowed to create, describe, modify, and delete DB instances, tag resources, or modify DB security groups. For information on setting up a IAM user, see Create an IAM User (p. 5)
- Use security groups to control what IP addresses or Amazon EC2 instances can connect to your databases on a DB instance. When you first create a DB instance, its firewall prevents any database access except through rules specified by an associated security group.
- Use Secure Socket Layer (SSL) connections with DB instances running the MySQL, Amazon Aurora, MariaDB, PostgreSQL, Oracle, or Microsoft SQL Server database engines. For more information on using SSL with a DB instance, see Using SSL to Encrypt a Connection to a DB Instance (p. 358).
- Use RDS encryption to secure your RDS instances and snapshots at rest. RDS encryption uses the industry standard AES-256 encryption algorithm to encrypt your data on the server that hosts your RDS instance. For more information, see Encrypting Amazon RDS Resources (p. 355).
- Use network encryption and transparent data encryption with Oracle DB instances; for more information, see Oracle Native Network Encryption (p. 1003) and Oracle Transparent Data Encryption (p. 1036)
- Use the security features of your DB engine to control who can log in to the databases on a DB instance, just as you do if the database was on your local network.

> **Note**
> You only have to configure security for your use cases. You don't have to configure security access for processes that Amazon RDS manages, such as creating backups, replicating data between a master and a Read Replica, or other processes.

For more information on managing access to Amazon RDS resources and your databases on a DB instance, see the following topics.

Topics

Authentication and Access Control for Amazon RDS

Access to Amazon RDS requires credentials that AWS can use to authenticate your requests. Those credentials must have permissions to access AWS resources, such as an Amazon RDS DB instance. The following sections provide details on how you can use AWS Identity and Access Management (IAM) and Amazon RDS to help secure your resources by controlling who can access them:

Authentication

You can access AWS as any of the following types of identities:

- **AWS account root user** – When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the best practice of using the root user only to create your first IAM user. Then securely lock away the root user credentials and use them to perform only a few account and service management tasks.
- **IAM user** – An IAM user is an identity within your AWS account that has specific custom permissions (for example, permissions to create a DB instance in Amazon RDS). You can use an IAM user name and password to sign in to secure AWS webpages like the AWS Management Console, AWS Discussion Forums, or the AWS Support Center.

  In addition to a user name and password, you can also generate access keys for each user. You can use these keys when you access AWS services programmatically, either through one of the several SDKs or by using the AWS Command Line Interface (CLI). The SDK and CLI tools use the access keys to cryptographically sign your request. If you don't use AWS tools, you must sign the request yourself. Amazon RDS supports *Signature Version 4*, a protocol for authenticating inbound API requests. For more information about authenticating requests, see Signature Version 4 Signing Process in the *AWS General Reference*.
- **IAM role** – An IAM role is an IAM identity that you can create in your account that has specific permissions. It is similar to an *IAM user*, but it is not associated with a specific person. An IAM role enables you to obtain temporary access keys that can be used to access AWS services and resources. IAM roles with temporary credentials are useful in the following situations:
  - **Federated user access** – Instead of creating an IAM user, you can use existing user identities from AWS Directory Service, your enterprise user directory, or a web identity provider. These are known as *federated users*. AWS assigns a role to a federated user when access is requested through an identity provider. For more information about federated users, see Federated Users and Roles in the *IAM User Guide*.
  - **AWS service access** – You can use an IAM role in your account to grant an AWS service permissions to access your account's resources. For example, you can create a role that allows Amazon Redshift to access an Amazon S3 bucket on your behalf and then load data from that bucket into an Amazon Redshift cluster. For more information, see Creating a Role to Delegate Permissions to an AWS Service in the *IAM User Guide*.
  - **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An

instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Using Roles for Applications on Amazon EC2 in the *IAM User Guide*.

Access Control

You can have valid credentials to authenticate your requests, but unless you have permissions you cannot create or access Amazon RDS resources. For example, you must have permissions to create an Amazon RDS DB instance, create a DB snapshot, add an event subscription, and so on.

The following sections describe how to manage permissions for Amazon RDS. We recommend that you read the overview first.

- Overview of Managing Access Permissions to Your Amazon RDS Resources (p. 328)
- Using Identity-Based Policies (IAM Policies) for Amazon RDS (p. 332)

Overview of Managing Access Permissions to Your Amazon RDS Resources

Every AWS resource is owned by an AWS account, and permissions to create or access the resources are governed by permissions policies. An account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles), and some services (such as AWS Lambda) also support attaching permissions policies to resources.

Note
An *account administrator* (or administrator user) is a user with administrator privileges. For more information, see IAM Best Practices in the *IAM User Guide*.

When granting permissions, you decide who is getting the permissions, the resources they get permissions for, and the specific actions that you want to allow on those resources.

Topics

- Amazon RDS Resources and Operations (p. 328)
- Understanding Resource Ownership (p. 329)
- Managing Access to Resources (p. 329)
- Specifying Policy Elements: Actions, Effects, Resources, and Principals (p. 331)
- Specifying Conditions in a Policy (p. 331)

Amazon RDS Resources and Operations

In Amazon RDS, the primary resource is a *DB instance*. Amazon RDS supports other resources that can be used with the primary resource such as *DB snapshots*, *parameter groups*, and *event subscriptions*. These are referred to as *subresources*.

These resources and subresources have unique Amazon Resource Names (ARNs) associated with them as shown in the following table.

Resource Type	ARN Format
DB cluster	`arn:aws:rds:region:account-id:cluster:db-cluster-name`

Resource Type	ARN Format
DB cluster parameter group	arn:aws:rds:*region*:*account-id*:cluster-pg:*cluster-parameter-group-name*
DB cluster snapshot	arn:aws:rds:*region*:*account-id*:cluster-snapshot:*cluster-snapshot-name*
DB instance	arn:aws:rds:*region*:*account-id*:db:*db-instance-name*
DB option group	arn:aws:rds:*region*:*account-id*:og:*option-group-name*
DB parameter group	arn:aws:rds:*region*:*account-id*:pg:*parameter-group-name*
DB snapshot	arn:aws:rds:*region*:*account-id*:snapshot:*snapshot-name*
DB security group	arn:aws:rds:*region*:*account-id*:secgrp:*security-group-name*
DB subnet group	arn:aws:rds:*region*:*account-id*:subgrp:*subnet-group-name*
Event subscription	arn:aws:rds:*region*:*account-id*:es:*subscription-name*
Read Replica	arn:aws:rds:*region*:*account-id*:db:*db-instance-name*
Reserved DB instance	arn:aws:rds:*region*:*account-id*:ri:*reserved-db-instance-name*

Amazon RDS provides a set of operations to work with the Amazon RDS resources. For a list of available operations, see Actions.

Understanding Resource Ownership

A *resource owner* is the AWS account that created a resource. That is, the resource owner is the AWS account of the *principal entity* (the root account, an IAM user, or an IAM role) that authenticates the request that creates the resource. The following examples illustrate how this works:

- If you use the root account credentials of your AWS account to create an RDS resource, such as a DB instance, your AWS account is the owner of the RDS resource.
- If you create an IAM user in your AWS account and grant permissions to create RDS resources to that user, the user can create RDS resources. However, your AWS account, to which the user belongs, owns the RDS resources.
- If you create an IAM role in your AWS account with permissions to create RDS resources, anyone who can assume the role can create RDS resources. Your AWS account, to which the role belongs, owns the RDS resources.

Managing Access to Resources

A *permissions policy* describes who has access to what. The following section explains the available options for creating permissions policies.

> **Note**
> This section discusses using IAM in the context of Amazon RDS. It doesn't provide detailed information about the IAM service. For complete IAM documentation, see What Is IAM? in the *IAM User Guide*. For information about IAM policy syntax and descriptions, see AWS IAM Policy Reference in the *IAM User Guide*.

Policies attached to an IAM identity are referred to as *identity-based* policies (IAM polices) and policies attached to a resource are referred to as *resource-based* policies. Amazon RDS supports only identity-based policies (IAM policies).

Topics

Identity-Based Policies (IAM Policies)

You can attach policies to IAM identities. For example, you can do the following:

- **Attach a permissions policy to a user or a group in your account** – An account administrator can use a permissions policy that is associated with a particular user to grant permissions for that user to create an Amazon RDS resource, such as a DB instance.
- **Attach a permissions policy to a role (grant cross-account permissions)** – You can attach an identity-based permissions policy to an IAM role to grant cross-account permissions. For example, the administrator in Account A can create a role to grant cross-account permissions to another AWS account (for example, Account B) or an AWS service as follows:
  1. Account A administrator creates an IAM role and attaches a permissions policy to the role that grants permissions on resources in Account A.
  2. Account A administrator attaches a trust policy to the role identifying Account B as the principal who can assume the role.
  3. Account B administrator can then delegate permissions to assume the role to any users in Account B. Doing this allows users in Account B to create or access resources in Account A. The principal in the trust policy can also be an AWS service principal if you want to grant an AWS service permissions to assume the role.

  For more information about using IAM to delegate permissions, see Access Management in the *IAM User Guide*.

The following is an example policy that allows the user with the ID `123456789012` to create DB instances for your AWS account. The policy requires that the name of the new DB instance begin with `test`. The new DB instance must also use the MySQL database engine and the `db.t2.micro` DB instance class. In addition, the new DB instance must use an option group and a DB parameter group that starts with `default`, and it must use the `default` subnet group.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCreateDBInstanceOnly",
            "Effect": "Allow",
            "Action": [
                "rds:CreateDBInstance"
            ],
            "Resource": [
                "arn:aws:rds:*:123456789012:db:test*",
                "arn:aws:rds:*:123456789012:og:default*",
                "arn:aws:rds:*:123456789012:pg:default*",
                "arn:aws:rds:*:123456789012:subgrp:default"
            ],
            "Condition": {
                "StringEquals": {
                    "rds:DatabaseEngine": "mysql",
                    "rds:DatabaseClass": "db.t2.micro"
                }
            }
        }
    ]
```

```
}
```

For more information about using identity-based policies with Amazon RDS, see Using Identity-Based Policies (IAM Policies) for Amazon RDS (p. 332). For more information about users, groups, roles, and permissions, see Identities (Users, Groups, and Roles) in the *IAM User Guide*.

Resource-Based Policies

Other services, such as Amazon S3, also support resource-based permissions policies. For example, you can attach a policy to an S3 bucket to manage access permissions to that bucket. Amazon RDS doesn't support resource-based policies.

Specifying Policy Elements: Actions, Effects, Resources, and Principals

For each Amazon RDS resource (see Amazon RDS Resources and Operations (p. 328)), the service defines a set of API operations (see Actions). To grant permissions for these API operations, Amazon RDS defines a set of actions that you can specify in a policy. Performing an API operation can require permissions for more than one action.

The following are the basic policy elements:

- **Resource** – In a policy, you use an Amazon Resource Name (ARN) to identify the resource to which the policy applies. For more information, see Amazon RDS Resources and Operations (p. 328).
- **Action** – You use action keywords to identify resource operations that you want to allow or deny. For example, the `rds:DescribeDBInstances` permission allows the user permissions to perform the Amazon RDS `DescribeDBInstances` operation.
- **Effect** – You specify the effect when the user requests the specific action—this can be either allow or deny. If you don't explicitly grant access to (allow) a resource, access is implicitly denied. You can also explicitly deny access to a resource, which you might do to make sure that a user cannot access it, even if a different policy grants access.
- **Principal** – In identity-based policies (IAM policies), the user that the policy is attached to is the implicit principal. For resource-based policies, you specify the user, account, service, or other entity that you want to receive permissions (applies to resource-based policies only). Amazon RDS doesn't support resource-based policies.

To learn more about IAM policy syntax and descriptions, see AWS IAM Policy Reference in the *IAM User Guide*.

For a table showing all of the Amazon RDS API actions and the resources that they apply to, see Amazon RDS API Permissions: Actions, Resources, and Conditions Reference (p. 335).

You can test IAM policies with the IAM policy simulator. It automatically provides a list of resources and parameters required for each AWS action, including Amazon RDS actions. The IAM policy simulator determines the permissions required for each of the actions that you specify. For information about the IAM policy simulator, see Testing IAM Policies with the IAM Policy Simulator in the *IAM User Guide*.

Specifying Conditions in a Policy

When you grant permissions, you can use the access policy language to specify the conditions when a policy should take effect. For example, you might want a policy to be applied only after a specific date. For more information about specifying conditions in a policy language, see Condition in the *IAM User Guide*.

To express conditions, you use predefined condition keys. There are AWS-wide condition keys and RDS-specific keys that you can use as appropriate. For a complete list of AWS-wide keys, see Available Keys

for Conditions in the *IAM User Guide*. For a complete list of RDS-specific keys, see Using IAM Policy Conditions for Fine-Grained Access Control (p. 349).

Using Identity-Based Policies (IAM Policies) for Amazon RDS

This topic provides examples of identity-based policies in which an account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles).

> **Important**
> We recommend that you first review the introductory topics that explain the basic concepts and options available for you to manage access to your Amazon RDS resources. For more information, see Overview of Managing Access Permissions to Your Amazon RDS Resources (p. 328).

The sections in this topic cover the following:

- Permissions Required to Use the Amazon RDS Console (p. 333)
- AWS Managed (Predefined) Policies for Amazon RDS (p. 333)
- Customer Managed Policy Examples (p. 334)

The following is an example of an IAM policy:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCreateDBInstanceOnly",
            "Effect": "Allow",
            "Action": [
                "rds:CreateDBInstance"
            ],
            "Resource": [
                "arn:aws:rds:*:123456789012:db:test*",
                "arn:aws:rds:*:123456789012:og:default*",
                "arn:aws:rds:*:123456789012:pg:default*",
                "arn:aws:rds:*:123456789012:subgrp:default"
            ],
            "Condition": {
                "StringEquals": {
                    "rds:DatabaseEngine": "mysql",
                    "rds:DatabaseClass": "db.t2.micro"
                }
            }
        }
    ]
}
```

The policy includes a single statement that specifies the following permissions for the IAM user:

- The policy allows the IAM user to create a DB instance using the CreateDBInstance API action (this also applies to the create-db-instance AWS CLI command and the AWS Management Console).
- The `Resource` element specifies that the user can perform actions on or with resources. You specify resources using an Amazon Resources Name (ARN). This ARN includes the name of the service that the resource belongs to (`rds`), the AWS Region (`*` indicates any region in this example), the user account number (`123456789012` is the user ID in this example), and the type of resource. For more

information about creating ARNs, see Working with Amazon Resource Names (ARNs) in Amazon RDS (p. 184).

The `Resource` element in the example specifies the following policy constraints on resources for the user:

- The DB instance identifier for the new DB instance must begin with `test` (for example, `testCustomerData1`, `test-region2-data`).
- The option group for the new DB instance must begin with `default`.
- The DB parameter group for the new DB instance must begin with `default`.
- The subnet group for the new DB instance must be the `default` subnet group.

- The `Condition` element specifies that the DB engine must be MySQL and the DB instance class must be `db.t2.micro`. The `Condition` element specifies the conditions when a policy should take effect. You can add additional permissions or restrictions by using the `Condition` element. For more information about specifying conditions, see Using IAM Policy Conditions for Fine-Grained Access Control (p. 349).

The policy doesn't specify the `Principal` element because in an identity-based policy you don't specify the principal who gets the permission. When you attach policy to a user, the user is the implicit principal. When you attach a permission policy to an IAM role, the principal identified in the role's trust policy gets the permissions.

For a table showing all of the Amazon RDS API actions and the resources that they apply to, see Amazon RDS API Permissions: Actions, Resources, and Conditions Reference (p. 335).

Permissions Required to Use the Amazon RDS Console

For a user to work with the Amazon RDS console, that user must have a minimum set of permissions. These permissions allow the user to describe the Amazon RDS resources for their AWS account and to provide other related information, including Amazon EC2 security and network information.

If you create an IAM policy that is more restrictive than the minimum required permissions, the console won't function as intended for users with that IAM policy. To ensure that those users can still use the Amazon RDS console, also attach the `AmazonRDSReadOnlyAccess` managed policy to the user, as described in AWS Managed (Predefined) Policies for Amazon RDS (p. 333).

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the Amazon RDS API.

AWS Managed (Predefined) Policies for Amazon RDS

AWS addresses many common use cases by providing standalone IAM policies that are created and administered by AWS. Managed policies grant necessary permissions for common use cases so you can avoid having to investigate what permissions are needed. For more information, see AWS Managed Policies in the *IAM User Guide*.

The following AWS managed policies, which you can attach to users in your account, are specific to Amazon RDS:

- **AmazonRDSReadOnlyAccess** – Grants read-only access to all Amazon RDS resources for the root AWS account.
- **AmazonRDSFullAccess** – Grants full access to all Amazon RDS resources for the root AWS account.

You can also create custom IAM policies that allow users to access the required Amazon RDS API actions and resources. You can attach these custom policies to the IAM users or groups that require those permissions.

Customer Managed Policy Examples

In this section, you can find example user policies that grant permissions for various Amazon RDS actions. These policies work when you are using RDS API actions, AWS SDKs, or the AWS CLI. When you are using the console, you need to grant additional permissions specific to the console, which is discussed in Permissions Required to Use the Amazon RDS Console (p. 333).

> **Note**
> All examples use the US West (Oregon) Region (`us-west-2`) and contain fictitious account IDs.

Examples

Example 1: Allow a User to Perform Any Describe Action on Any RDS Resource

The following permissions policy grants permissions to a user to run all of the actions that begin with `Describe`. These actions show information about an RDS resource, such as a DB instance. The wildcard character (*) in the `Resource` element indicates that the actions are allowed for all Amazon RDS resources owned by the account.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowRDSDescribe",
         "Effect":"Allow",
         "Action":"rds:Describe*",
         "Resource":"*"
      }
   ]
}
```

Example 2: Allow a User to Create a DB Instance That Uses the Specified DB Parameter and Security Groups

The following permissions policy grants permissions to allow a user to only create a DB instance that must use the `mysql-production` DB parameter group and the `db-production` DB security group.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowMySQLProductionCreate",
         "Effect":"Allow",
         "Action":"rds:CreateDBInstance",
         "Resource":[
            "arn:aws:rds:us-west-2:123456789012:pg:mysql-production",
            "arn:aws:rds:us-west-2:123456789012:secgrp:db-production"
         ]
      }
   ]
}
```

Example 3: Prevent a User from Deleting a DB Instance

The following permissions policy grants permissions to prevent a user from deleting a specific DB instance. For example, you might want to deny the ability to delete your production instances to any user that is not an administrator.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"DenyDelete1",
         "Effect":"Deny",
         "Action":"rds:DeleteDBInstance",
         "Resource":"arn:aws:rds:us-west-2:123456789012:db:my-mysql-instance"
      }
   ]
}
```

Amazon RDS API Permissions: Actions, Resources, and Conditions Reference

When you set up Access Control (p. 328) and write permissions policies that you can attach to an IAM identity (identity-based policies), you can use the following as a reference.

The following lists each Amazon RDS API operation. Included in the list are the corresponding actions for which you can grant permissions to perform the action, the AWS resource that you can grant the permissions for, and condition keys that you can include for fine-grained access control. You specify the actions in the policy's `Action` field, the resource value in the policy's `Resource` field, and conditions in the policy's `Condition` field. For more information about conditions, see Using IAM Policy Conditions for Fine-Grained Access Control (p. 349).

You can use AWS-wide condition keys in your Amazon RDS policies to express conditions. For a complete list of AWS-wide keys, see Available Keys in the *IAM User Guide*.

You can test IAM policies with the IAM policy simulator. It automatically provides a list of resources and parameters required for each AWS action, including Amazon RDS actions. The IAM policy simulator determines the permissions required for each of the actions that you specify. For information about the IAM policy simulator, see Testing IAM Policies with the IAM Policy Simulator in the *IAM User Guide*.

> **Note**
> To specify an action, use the `rds:` prefix followed by the API operation name (for example, `rds:CreateDBInstance`).

The following lists RDS API operations and their related actions, resources, and condition keys.

Topics

- Amazon RDS Actions That Support Resource-Level Permissions (p. 335)
- Amazon RDS Actions That Don't Support Resource-Level Permissions (p. 349)
- Related Topics (p. 349)

Amazon RDS Actions That Support Resource-Level Permissions

Resource-level permissions refers to the ability to specify the resources on which users are allowed to perform actions. Amazon RDS has partial support for resource-level permissions. This means that for certain Amazon RDS actions, you can control when users are allowed to use those actions based on conditions that have to be fulfilled, or specific resources that users are allowed to use. For example, you can grant users permission to modify only specific DB instances.

The following lists RDS API operations and their related actions, resources, and condition keys.

RDS API Operations and Actions	Resources	Condition Keys
AddRoleToDBCluster rds:AddRoleToDBCluster	DB cluster arn:aws:rds:*region*:*account-id*:cluster:*db-cluster-name*	rds:cluster-tag
	IAM role arn:aws:iam::*account-id*:role/*role-name*	N/A
AddSourceIdentifierToSubscription rds:AddSourceIdentifierToSubscription	Event subscription arn:aws:rds:*region*:*account-id*:es:*subscription-name*	rds:es-tag
AddTagsToResource rds:AddTagsToResource	DB instance arn:aws:rds:*region*:*account-id*:db:*db-instance-name*	rds:db-tag
	DB option group arn:aws:rds:*region*:*account-id*:og:*option-group-name*	rds:og-tag
	DB parameter group arn:aws:rds:*region*:*account-id*:pg:*parameter-group-name*	rds:pg-tag
	DB security group arn:aws:rds:*region*:*account-id*:secgrp:*security-group-name*	rds:secgrp-tag
	DB subnet group arn:aws:rds:*region*:*account-id*:subgrp:*subnet-group-name*	rds:subgrp-tag
	DB snapshot arn:aws:rds:*region*:*account-id*:snapshot:*snapshot-name*	rds:snapshot-tag
	Event subscription arn:aws:rds:*region*:*account-id*:es:*subscription-name*	rds:es-tag
	Reserved DB instance arn:aws:rds:*region*:*account-id*:ri:*reserved-db-instance-name*	rds:ri-tag
ApplyPendingMaintenanceAction	DB instance	rds:db-tag

RDS API Operations and Actions	Resources	Condition Keys
rds:ApplyPendingMaintenanceAction	arn:aws:rds:*region*:*account-id*:db:*db-instance-name*	
AuthorizeDBSecurityGroupIngress rds:AuthorizeDBSecurityGroupIngress	DB security group arn:aws:rds:*region*:*account-id*:secgrp:*security-group-name*	rds:secgrp-tag
CopyDBClusterSnapshot rds:CopyDBClusterSnapshot	DB cluster snapshot arn:aws:rds:*region*:*account-id*:cluster-snapshot:*cluster-snapshot-name*	rds:cluster-snapshot-tag
CopyDBParameterGroup rds:CopyDBParameterGroup	DB parameter group arn:aws:rds:*region*:*account-id*:pg:*parameter-group-name*	rds:pg-tag
CopyDBSnapshot rds:CopyDBSnapshot	DB snapshot arn:aws:rds:*region*:*account-id*:snapshot:*snapshot-name*	rds:snapshot-tag
CopyOptionGroup rds:CopyOptionGroup	DB option group arn:aws:rds:*region*:*account-id*:og:*option-group-name*	rds:og-tag
CreateDBCluster rds:CreateDBCluster	DB cluster arn:aws:rds:*region*:*account-id*:cluster:*db-cluster-name*	rds:DatabaseEngine rds:DatabaseName rds:cluster-tag
	DB option group arn:aws:rds:*region*:*account-id*:og:*option-group-name*	rds:og-tag
	DB cluster parameter group arn:aws:rds:*region*:*account-id*:cluster-pg:*cluster-parameter-group-name*	rds:cluster-pg-tag
	DB subnet group arn:aws:rds:*region*:*account-id*:subgrp:*subnet-group-name*	rds:subgrp-tag
CreateDBClusterParameterGroup rds:CreateDBClusterParameterGroup	DB cluster parameter group arn:aws:rds:*region*:*account-id*:cluster-pg:*cluster-parameter-group-name*	rds:cluster-pg-tag

RDS API Operations and Actions	Resources	Condition Keys
CreateDBClusterSnapshot rds:CreateDBClusterSnapshot	DB cluster arn:aws:rds:*region*:*account-id*:cluster:*db-cluster-name*	rds:cluster-tag
	DB cluster snapshot arn:aws:rds:*region*:*account-id*:cluster-snapshot:*cluster-snapshot-name*	rds:cluster-snapshot-tag
CreateDBInstance rds:CreateDBInstance	DB instance arn:aws:rds:*region*:*account-id*:db:*db-instance-name*	rds:DatabaseClass rds:DatabaseEngine rds:DatabaseName rds:MultiAz rds:Piops rds:StorageSize rds:Vpc rds:db-tag
	DB option group arn:aws:rds:*region*:*account-id*:og:*option-group-name*	rds:og-tag
	DB parameter group arn:aws:rds:*region*:*account-id*:pg:*parameter-group-name*	rds:pg-tag
	DB security group arn:aws:rds:*region*:*account-id*:secgrp:*security-group-name*	rds:secgrp-tag
	DB subnet group arn:aws:rds:*region*:*account-id*:subgrp:*subnet-group-name*	rds:subgrp-tag
	DB cluster arn:aws:rds:*region*:*account-id*:cluster:*db-cluster-name*	rds:cluster-tag
CreateDBInstanceReadReplica rds:CreateDBInstanceReadReplica	DB instance arn:aws:rds:*region*:*account-id*:db:*db-instance-name*	rds:DatabaseClass rds:Piops rds:db-tag

RDS API Operations and Actions	Resources	Condition Keys
	DB option group arn:aws:rds:*region*:*account-id*:og:*option-group-name*	rds:og-tag
	DB subnet group arn:aws:rds:*region*:*account-id*:subgrp:*subnet-group-name*	rds:subgrp-tag
CreateDBParameterGroup rds:CreateDBParameterGroup	DB parameter group arn:aws:rds:*region*:*account-id*:pg:*parameter-group-name*	rds:pg-tag
CreateDBSecurityGroup rds:CreateDBSecurityGroup	DB security group arn:aws:rds:*region*:*account-id*:secgrp:*security-group-name*	rds:secgrp-tag
CreateDBSnapshot rds:CreateDBSnapshot	DB instance arn:aws:rds:*region*:*account-id*:db:*db-instance-name*	rds:db-tag
	DB snapshot arn:aws:rds:*region*:*account-id*:snapshot:*snapshot-name*	rds:snapshot-tag
CreateDBSubnetGroup rds:CreateDBSubnetGroup	DB subnet group arn:aws:rds:*region*:*account-id*:subgrp:*subnet-group-name*	rds:subgrp-tag
CreateEventSubscription rds:CreateEventSubscription	Event subscription arn:aws:rds:*region*:*account-id*:es:*subscription-name*	rds:es-tag
CreateOptionGroup rds:CreateOptionGroup	DB option group arn:aws:rds:*region*:*account-id*:og:*option-group-name*	rds:og-tag
DeleteDBCluster rds:DeleteDBCluster	DB cluster arn:aws:rds:*region*:*account-id*:cluster:*db-cluster-name*	rds:cluster-tag
	DB cluster snapshot arn:aws:rds:*region*:*account-id*:cluster-snapshot:*cluster-snapshot-name*	rds:cluster-snapshot-tag

RDS API Operations and Actions	Resources	Condition Keys
DeleteDBClusterParameterGroup rds:DeleteDBClusterParameterGroup	DB cluster parameter group arn:aws:rds:*region*:*account-id*:cluster-pg:*cluster-parameter-group-name*	rds:cluster-pg-tag
DeleteDBClusterSnapshot rds:DeleteDBClusterSnapshot	DB cluster snapshot arn:aws:rds:*region*:*account-id*:cluster-snapshot:*cluster-snapshot-name*	rds:cluster-snapshot-tag
DeleteDBInstance rds:DeleteDBInstance	DB instance arn:aws:rds:*region*:*account-id*:db:*db-instance-name*	rds:db-tag
DeleteDBParameterGroup rds:DeleteDBParameterGroup	DB parameter group arn:aws:rds:*region*:*account-id*:pg:*parameter-group-name*	rds:pg-tag
DeleteDBSecurityGroup rds:DeleteDBSecurityGroup	DB security group arn:aws:rds:*region*:*account-id*:secgrp:*security-group-name*	rds:secgrp-tag
DeleteDBSnapshot rds:DeleteDBSnapshot	DB snapshot arn:aws:rds:*region*:*account-id*:snapshot:*snapshot-name*	rds:snapshot-tag
DeleteDBSubnetGroup rds:DeleteDBSubnetGroup	DB subnet group arn:aws:rds:*region*:*account-id*:subgrp:*subnet-group-name*	rds:subgrp-tag
DeleteEventSubscription rds:DeleteEventSubscription	Event subscription arn:aws:rds:*region*:*account-id*:es:*subscription-name*	rds:es-tag
DeleteOptionGroup rds:DeleteOptionGroup	DB option group arn:aws:rds:*region*:*account-id*:og:*option-group-name*	rds:og-tag
DescribeDBClusterParameterGroups rds:DescribeDBClusterParameterGroups	DB cluster parameter group arn:aws:rds:*region*:*account-id*:cluster-pg:*cluster-parameter-group-name*	rds:cluster-pg-tag

RDS API Operations and Actions	Resources	Condition Keys
DescribeDBClusterParameters rds:DescribeDBClusterParameters	DB cluster parameter group arn:aws:rds:*region*:*account-id*:cluster-pg:*cluster-parameter-group-name*	rds:cluster-pg-tag
DescribeDBClusters rds:DescribeDBClusters	DB cluster arn:aws:rds:*region*:*account-id*:cluster:*db-cluster-instance-name*	rds:cluster-tag
DescribeDBClusterSnapshotAttributes rds:DescribeDBClusterSnapshotAttributes	DB cluster snapshot arn:aws:rds:*region*:*account-id*:cluster-snapshot:*cluster-snapshot-name*	rds:cluster-snapshot-tag
DescribeDBEngineVersions rds:DescribeDBEngineVersions	DB parameter group arn:aws:rds:*region*:*account-id*:pg:*parameter-group-name*	rds:pg-tag
DescribeDBLogFiles rds:DescribeDBLogFiles	DB instance arn:aws:rds:*region*:*account-id*:db:*db-instance-name*	rds:db-tag
DescribeDBParameterGroups rds:DescribeDBParameterGroups	DB parameter group arn:aws:rds:*region*:*account-id*:pg:*parameter-group-name*	rds:pg-tag
DescribeDBParameters rds:DescribeDBParameters	DB parameter group arn:aws:rds:*region*:*account-id*:pg:*parameter-group-name*	rds:pg-tag
DescribeDBSecurityGroups rds:DescribeDBSecurityGroups	DB security group arn:aws:rds:*region*:*account-id*:secgrp:*security-group-name*	rds:secgrp-tag
DescribeDBSnapshotAttributes rds:DescribeDBSnapshotAttributes	DB snapshot arn:aws:rds:*region*:*account-id*:snapshot:*snapshot-name*	rds:snapshot-tag
DescribeDBSnapshots rds:DescribeDBSnapshots	DB instance arn:aws:rds:*region*:*account-id*:db:*db-instance-name*	rds:db-tag
	DB snapshot arn:aws:rds:*region*:*account-id*:snapshot:*snapshot-name*	rds:snapshot-tag

RDS API Operations and Actions	Resources	Condition Keys
DescribeDBSubnetGroups rds:DescribeDBSubnetGroups	DB subnet group arn:aws:rds:*region*:*account-id*:subgrp:*subnet-group-name*	rds:subgrp-tag
DescribeEvents rds:DescribeEvents	Event subscription arn:aws:rds:*region*:*account-id*:es:*subscription-name*	rds:es-tag
DescribeEventSubscriptions rds:DescribeEventSubscriptions	Event subscription arn:aws:rds:*region*:*account-id*:es:*subscription-name*	rds:es-tag
DescribeOptionGroupOptions rds:DescribeOptionGroupOptions	DB option group arn:aws:rds:*region*:*account-id*:og:*option-group-name*	rds:og-tag
DescribeOptionGroups rds:DescribeOptionGroups	DB option group arn:aws:rds:*region*:*account-id*:og:*option-group-name*	rds:og-tag
DescribePendingMaintenanceActions rds:DescribePendingMaintenanceActions	DB instance arn:aws:rds:*region*:*account-id*:db:*db-instance-name*	rds:DatabaseClass rds:DatabaseEngine rds:DatabaseName rds:MultiAz rds:Piops rds:StorageSize rds:Vpc rds:db-tag
DescribeReservedDBInstances rds:DescribeReservedDBInstances	Reserved DB instance arn:aws:rds:*region*:*account-id*:ri:*reserved-db-instance-name*	rds:DatabaseClass rds:MultiAz rds:ri-tag
DescribeReservedDBInstancesOfferings rds:DescribeReservedDBInstancesOfferings	DB instance arn:aws:rds:*region*:*account-id*:db:*db-instance-name*	rds:DatabaseClass rds:MultiAz
DownloadDBLogFilePortion rds:DownloadDBLogFilePortion	DB instance arn:aws:rds:*region*:*account-id*:db:*db-instance-name*	rds:db-tag

RDS API Operations and Actions	Resources	Condition Keys
FailoverDBCluster rds:FailoverDBCluster	DB cluster arn:aws:rds:*region*:*account-id*:cluster:*db-cluster-instance-name*	rds:cluster-tag
ListTagsForResource rds:ListTagsForResource	DB instance arn:aws:rds:*region*:*account-id*:db:*db-instance-name*	rds:db-tag
	DB option group arn:aws:rds:*region*:*account-id*:og:*option-group-name*	rds:og-tag
	DB parameter group arn:aws:rds:*region*:*account-id*:pg:*parameter-group-name*	rds:pg-tag
	DB security group arn:aws:rds:*region*:*account-id*:secgrp:*security-group-name*	rds:secgrp-tag
	DB subnet group arn:aws:rds:*region*:*account-id*:subgrp:*subnet-group-name*	rds:subgrp-tag
	DB snapshot arn:aws:rds:*region*:*account-id*:snapshot:*snapshot-name*	rds:snapshot-tag
	Event subscription arn:aws:rds:*region*:*account-id*:es:*subscription-name*	rds:es-tag
	Reserved DB instance arn:aws:rds:*region*:*account-id*:ri:*reserved-db-instance-name*	rds:ri-tag
ModifyDBCluster rds:ModifyDBCluster	DB cluster arn:aws:rds:*region*:*account-id*:cluster:*db-cluster-name*	rds:cluster-tag
	DB option group arn:aws:rds:*region*:*account-id*:og:*option-group-name*	rds:og-tag

RDS API Operations and Actions	Resources	Condition Keys
	DB cluster parameter group arn:aws:rds:*region*:*account-id*:cluster-pg:*cluster-parameter-group-name*	rds:cluster-pg-tag
ModifyDBClusterParameterGroup rds:ModifyDBClusterParameterGroup	DB cluster parameter group arn:aws:rds:*region*:*account-id*:cluster-pg:*cluster-parameter-group-name*	rds:cluster-pg-tag
ModifyDBClusterSnapshotAttribute rds:ModifyDBClusterSnapshotAttribute	DB cluster snapshot arn:aws:rds:*region*:*account-id*:cluster-snapshot:*cluster-snapshot-name*	rds:cluster-snapshot-tag
ModifyDBInstance rds:ModifyDBInstance	DB instance arn:aws:rds:*region*:*account-id*:db:*db-instance-name*	rds:DatabaseClass rds:MultiAz rds:Piops rds:StorageSize rds:Vpc rds:db-tag
	DB option group arn:aws:rds:*region*:*account-id*:og:*option-group-name*	rds:og-tag
	DB parameter group arn:aws:rds:*region*:*account-id*:pg:*parameter-group-name*	rds:pg-tag
	DB security group arn:aws:rds:*region*:*account-id*:secgrp:*security-group-name*	rds:secgrp-tag
ModifyDBParameterGroup rds:ModifyDBParameterGroup	DB parameter group arn:aws:rds:*region*:*account-id*:pg:*parameter-group-name*	rds:pg-tag
ModifyDBSnapshotAttribute rds:ModifyDBSnapshotAttribute	DB snapshot arn:aws:rds:*region*:*account-id*:snapshot:*snapshot-name*	rds:snapshot-tag

RDS API Operations and Actions	Resources	Condition Keys
ModifyDBSubnetGroup rds:ModifyDBSubnetGroup	DB subnet group arn:aws:rds:*region*:*account-id*:subgrp:*subnet-group-name*	rds:subgrp-tag
ModifyEventSubscription rds:ModifyEventSubscription	Event subscription arn:aws:rds:*region*:*account-id*:es:*subscription-name*	rds:es-tag
ModifyOptionGroup rds:ModifyOptionGroup	DB option group arn:aws:rds:*region*:*account-id*:og:*option-group-name*	rds:og-tag
PromoteReadReplica rds:PromoteReadReplica	DB instance arn:aws:rds:*region*:*account-id*:db:*db-instance-name*	rds:db-tag
PromoteReadReplicaDBCluster rds:PromoteReadReplicaDBCluster	DB cluster arn:aws:rds:*region*:*account-id*:cluster:*db-cluster-name*	
RebootDBInstance rds:RebootDBInstance	DB instance arn:aws:rds:*region*:*account-id*:db:*db-instance-name*	rds:db-tag
RemoveSourceIdentifierFromSubscription rds:RemoveSourceIdentifierFromSubscription	Event subscription arn:aws:rds:*region*:*account-id*:es:*subscription-name*	rds:es-tag
RemoveTagsFromResource rds:RemoveTagsFromResource	DB instance arn:aws:rds:*region*:*account-id*:db:*db-instance-name*	rds:db-tag
	DB option group arn:aws:rds:*region*:*account-id*:og:*option-group-name*	rds:og-tag
	DB parameter group arn:aws:rds:*region*:*account-id*:pg:*parameter-group-name*	rds:pg-tag
	DB security group arn:aws:rds:*region*:*account-id*:secgrp:*security-group-name*	rds:secgrp-tag

RDS API Operations and Actions	Resources	Condition Keys
	DB subnet group arn:aws:rds:*region*:*account-id*:subgrp:*subnet-group-name*	rds:subgrp-tag
	DB snapshot arn:aws:rds:*region*:*account-id*:snapshot:*snapshot-name*	rds:snapshot-tag
	Event subscription arn:aws:rds:*region*:*account-id*:es:*subscription-name*	rds:es-tag
	Reserved DB instance arn:aws:rds:*region*:*account-id*:ri:*reserved-db-instance-name*	rds:ri-tag
ResetDBClusterParameterGroup rds:ResetDBClusterParameterGroup	DB cluster parameter group arn:aws:rds:*region*:*account-id*:cluster-pg:*cluster-parameter-group-name*	rds:cluster-pg-tag
ResetDBParameterGroup rds:ResetDBParameterGroup	DB parameter group arn:aws:rds:*region*:*account-id*:pg:*parameter-group-name*	rds:pg-tag
RestoreDBClusterFromS3 rds:RestoreDBClusterFromS3	DB cluster arn:aws:rds:*region*:*account-id*:cluster:*db-cluster-instance-name*	rds:DatabaseEngine rds:DatabaseName rds:cluster-tag
	DB cluster parameter group arn:aws:rds:*region*:*account-id*:cluster-pg:*cluster-parameter-group-name*	rds:cluster-pg-tag
	DB option group arn:aws:rds:*region*:*account-id*:og:*option-group-name*	rds:og-tag
	DB subnet group arn:aws:rds:*region*:*account-id*:subgrp:*subnet-group-name*	rds:subgrp-tag

RDS API Operations and Actions	Resources	Condition Keys
RestoreDBClusterFromSnapshot rds:RestoreDBClusterFromSnapshot	DB cluster arn:aws:rds:*region*:*account-id*:cluster:*db-cluster-instance-name*	rds:DatabaseEngine rds:DatabaseName rds:cluster-tag
	DB option group arn:aws:rds:*region*:*account-id*:og:*option-group-name*	rds:og-tag
	DB cluster snapshot arn:aws:rds:*region*:*account-id*:cluster-snapshot:*cluster-snapshot-name*	rds:cluster-snapshot-tag
RestoreDBClusterToPointInTime rds:RestoreDBClusterToPointInTime	DB cluster arn:aws:rds:*region*:*account-id*:cluster:*db-cluster-instance-name*	rds:cluster-tag
	DB option group arn:aws:rds:*region*:*account-id*:og:*option-group-name*	rds:og-tag
	DB subnet group arn:aws:rds:*region*:*account-id*:subgrp:*subnet-group-name*	rds:subgrp-tag
RestoreDBInstanceFromDBSnapshot rds:RestoreDBInstanceFromDBSnapshot	DB instance arn:aws:rds:*region*:*account-id*:db:*db-instance-name*	rds:DatabaseClass rds:DatabaseEngine rds:DatabaseName rds:MultiAz rds:Piops rds:Vpc rds:db-tag
	DB option group arn:aws:rds:*region*:*account-id*:og:*option-group-name*	rds:og-tag
	DB snapshot arn:aws:rds:*region*:*account-id*:snapshot:*snapshot-name*	rds:snapshot-tag

RDS API Operations and Actions	Resources	Condition Keys
	DB subnet group arn:aws:rds:*region*:*account-id*:subgrp:*subnet-group-name*	rds:subgrp-tag
RestoreDBInstanceToPointInTime rds:RestoreDBInstanceToPointInTime	DB instance arn:aws:rds:*region*:*account-id*:db:*db-instance-name*	rds:DatabaseClass rds:DatabaseEngine rds:DatabaseName rds:MultiAz rds:Piops rds:Vpc rds:db-tag
	DB option group arn:aws:rds:*region*:*account-id*:og:*option-group-name*	rds:og-tag
	DB snapshot arn:aws:rds:*region*:*account-id*:snapshot:*snapshot-name*	rds:snapshot-tag
	DB subnet group arn:aws:rds:*region*:*account-id*:subgrp:*subnet-group-name*	rds:subgrp-tag
RevokeDBSecurityGroupIngress rds:RevokeDBSecurityGroupIngress	DB security group arn:aws:rds:*region*:*account-id*:secgrp:*security-group-name*	rds:secgrp-tag
StartDBInstance rds:StartDBInstance	DB instance arn:aws:rds:*region*:*account-id*:db:*db-instance-name*	rds:DatabaseClass rds:DatabaseEngine rds:DatabaseName rds:MultiAz rds:Piops rds:Vpc rds:db-tag

RDS API Operations and Actions	Resources	Condition Keys

StopDBInstance
 	
`rds:StopDBInstance`
	DB instance
	
	`arn:aws:rds:`*`region`*`:`*`account-id`*`:db:`*`db-instance-name`*
		`rds:DatabaseClass`
		
		`rds:DatabaseEngine`
		
		`rds:DatabaseName`
		
		`rds:MultiAz`
		
		`rds:Piops`
		
		`rds:Vpc`
		
		`rds:db-tag`

Amazon RDS Actions That Don't Support Resource-Level Permissions

You can use all Amazon RDS actions in an IAM policy to either grant or deny users permission to use that action. However, not all Amazon RDS actions support resource-level permissions, which enable you to specify the resources on which an action can be performed. The following Amazon RDS API actions currently don't support resource-level permissions. Therefore, to use these actions in an IAM policy, you must grant users permission to use all resources for the action by using a `*` wildcard for the `Resource` element in your statement.

- DescribeAccountAttributes (Action: `rds:DescribeAccountAttributes`)
- DescribeCertificates (Action: `rds:DescribeCertificates`)
- DescribeDBClusterSnapshots (Action: `rds:DescribeDBClusterSnapshots`)
- DescribeDBInstances (Action: `rds:DescribeDBInstances`)
- DescribeEngineDefaultClusterParameters (Action: `rds:DescribeEngineDefaultClusterParameters`)
- DescribeEngineDefaultParameters (Action: `rds:DescribeEngineDefaultParameters`)
- DescribeEventCategories (Action: `rds:DescribeEventCategories`)
- DescribeOrderableDBInstanceOptions (Action: `rds:DescribeOrderableDBInstanceOptions`)
- DownloadCompleteDBLogFile (p. 1243) (Action: `rds:DownloadCompleteDBLogFile`)
- PurchaseReservedDBInstancesOffering (Action: `rds:PurchaseReservedDBInstancesOffering`)

Related Topics

- Access Control (p. 328)
- Using IAM Policy Conditions for Fine-Grained Access Control (p. 349)
- Security in Amazon RDS (p. 326)

Using IAM Policy Conditions for Fine-Grained Access Control

When you grant permissions in Amazon RDS, you can specify conditions that determine how a permissions policy takes effect.

Overview

In Amazon RDS, you have the option to specify conditions when granting permissions using an IAM policy (see Access Control (p. 328)). For example, you can:

- Allow users to create a DB instance only if they specify a particular database engine.
- Allow users to modify RDS resources that are tagged with a particular tag name and tag value.

There are two ways to specify conditions in an IAM policy for Amazon RDS:

- Using Condition Keys
- Using Custom Tags

Specifying Conditions: Using Condition Keys

AWS provides a set of predefined condition keys (AWS-wide condition keys) for all AWS services that support IAM for access control. For example, you can use the `aws:userid` condition key to require a specific AWS ID when requesting an action. For more information and a list of the AWS-wide condition keys, see Available Keys for Conditions in the *IAM User Guide*.

Note
Condition keys are case sensitive.

In addition Amazon RDS also provides its own condition keys that you can include in `Condition` elements in an IAM permissions policy. The following table shows the RDS condition keys that apply to RDS resources.

RDS Condition Key	Description	Value Type
`rds:DatabaseClass`	A type of DB instance class.	String
`rds:DatabaseEngine`	A database engine, such as MySQL.	String
`rds:DatabaseName`	The user-defined name of the database on the DB instance.	String
`rds:MultiAz`	A value that specifies whether the DB instance runs in multiple Availability Zones. To indicate that the DB instance is using Multi-AZ, specify **`true`**.	Boolean
`rds:Piops`	A value that contains the number of Provisioned IOPS (PIOPS) that the instance supports. To indicate a DB instance that does not have PIOPS enabled, specify 0.	Integer
`rds:StorageSize`	The storage volume size (in GB).	Integer
`rds:Vpc`	A value that specifies whether the DB instance runs in an Amazon Virtual Private Cloud (Amazon VPC). To indicate that the DB instance runs in an Amazon VPC, specify **`true`**.	Boolean

For example, the following `Condition` element uses a condition key and specifies the MySQL database engine. You could apply this to an IAM policy that allows permission to the `rds:CreateDBInstance` action to enable users to only create DB instances with the MySQL database engine. For an example of an IAM policy that uses this condition, see Example Policies: Using Condition Keys (p. 351).

```
"Condition":{"StringEquals":{"rds:DatabaseEngine": "mysql" } }
```

For a list of all of the RDS condition key identifiers and the RDS actions and resources that they apply to, see Amazon RDS API Permissions: Actions, Resources, and Conditions Reference (p. 335).

Example Policies: Using Condition Keys

Following are examples of how you can use condition keys in Amazon RDS IAM permissions policies.

Example 1: Grant Permission to Create a DB Instance that Uses a Specific DB Engine and Isn't MultiAZ

The following policy uses an RDS condition key and allows a user to create only DB instances that use the MySQL database engine and don't use MultiAZ. The `Condition` element indicates the requirement that the database engine is MySQL.

```
{

   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowMySQLCreate",
         "Effect":"Allow",
         "Action":"rds:CreateDBInstance",
         "Resource":"*",
         "Condition":{
            "StringEquals":{
               "rds:DatabaseEngine":"mysql"
            },
            "Bool":{
                    "rds:MultiAz": false
            }
         }
      }
   ]
}
```

Example 2: Explicitly Deny Permission to Create DB Instances for Certain DB Instance Classes and Create DB Instances that Use Provisioned IOPS

The following policy explicitly denies permission to create DB instances that use the DB instance classes `r3.8xlarge` and `m4.10xlarge`, which are the largest and most expensive instances. This policy also prevents users from creating DB instances that use Provisioned IOPS, which incurs an additional cost.

Explicitly denying permission supersedes any other permissions granted. This ensures that identities to not accidentally get permission that you never want to grant.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"DenyLargeCreate",
         "Effect":"Deny",
         "Action":"rds:CreateDBInstance",
         "Resource":"*",
         "Condition":{
            "StringEquals":{
               "rds:DatabaseClass":[
                  "db.r3.8xlarge",
                  "db.m4.10xlarge"
               ]
            }
         }
```

```
      },
      {
         "Sid":"DenyPIOPSCreate",
         "Effect":"Deny",
         "Action":"rds:CreateDBInstance",
         "Resource":"*",
         "Condition":{
            "NumericNotEquals":{
               "rds:Piops":"0"
            }
         }
      }
   ]
}
```

Specifying Conditions: Using Custom Tags

RDS supports specifying conditions in an IAM policy using custom tags.

For example, if you add a tag named `environment` to your DB instances with values such as `beta`, `staging`, `production`, and so on, you can create a policy that restricts certain users to DB instances based on the `environment` tag value.

Note
Custom tag identifiers are case-sensitive.

The following table lists the RDS tag identifiers that you can use in a `Condition` element.

RDS Tag Identifier	Applies To
`db-tag`	DB instances, including Read Replicas
`snapshot-tag`	DB snapshots
`ri-tag`	Reserved DB instances
`secgrp-tag`	DB security groups
`og-tag`	DB option groups
`pg-tag`	DB parameter groups
`subgrp-tag`	DB subnet groups
`es-tag`	Event subscriptions
`cluster-tag`	DB clusters
`cluster-pg-tag`	DB cluster parameter groups
`cluster-snapshot-tag`	DB cluster snapshots

The syntax for a custom tag condition is as follows:

```
"Condition":{"StringEquals":{"rds:rds-tag-identifier/tag-name": ["value"]} }
```

For example, the following `Condition` element applies to DB instances with a tag named `environment` and a tag value of `production`.

```
"Condition":{"StringEquals":{"rds:db-tag/environment": ["production"]} }
```

For information about creating tags, see Tagging Amazon RDS Resources (p. 129).

Important
If you manage access to your RDS resources using tagging, we recommend that you secure access to the tags for your RDS resources. You can manage access to tags by creating policies for the `AddTagsToResource` and `RemoveTagsFromResource` actions. For example, the following policy denies users the ability to add or remove tags for all resources. You can then create policies to allow specific users to add or remove tags.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"DenyTagUpdates",
         "Effect":"Deny",
         "Action":[
            "rds:AddTagsToResource",
            "rds:RemoveTagsFromResource"
         ],
         "Resource":"*"
      }
   ]
}
```

For a list of all of the condition key values, and the RDS actions and resources that they apply to, see Amazon RDS API Permissions: Actions, Resources, and Conditions Reference (p. 335).

Example Policies: Using Custom Tags

Following are examples of how you can use custom tags in Amazon RDS IAM permissions policies. For more information about adding tags to an Amazon RDS resource, see Working with Amazon Resource Names (ARNs) in Amazon RDS (p. 184).

Note
All examples use the us-west-2 region and contain fictitious account IDs.

Example 1: Grant Permission for Actions on a Resource with a Specific Tag with Two Different Values

The following policy allows permission to perform the `ModifyDBInstance` and `CreateDBSnapshot` APIs on instances with either the `stage` tag set to `development` or `test`.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowDevTestCreate",
         "Effect":"Allow",
         "Action":[
            "rds:ModifyDBInstance",
            "rds:CreateDBSnapshot"
         ],
         "Resource":"*",
         "Condition":{
            "StringEquals":{
               "rds:db-tag/stage":[
                  "development",
                  "test"
               ]
            }
         }
      }
```

```
    ]
}
```

Example 2: Explicitly Deny Permission to Create a DB Instance that Uses Specified DB Parameter Groups

The following policy explicitly denies permission to create a DB instance that uses DB parameter groups with specific tag values. You might apply this policy if you require that a specific customer-created DB parameter group always be used when creating DB instances. Note that policies that use `Deny` are most often used to restrict access that was granted by a broader policy.

Explicitly denying permission supersedes any other permissions granted. This ensures that identities to not accidentally get permission that you never want to grant.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"DenyProductionCreate",
         "Effect":"Deny",
         "Action":"rds:CreateDBInstance",
         "Resource":"*",
         "Condition":{
            "StringEquals":{
               "rds:pg-tag/usage":"prod"
            }
         }
      }
   ]
}
```

Example 3: Grant Permission for Actions on a DB Instance with an Instance Name that is Prefixed with a User Name

The following policy allows permission to call any API (except to `AddTagsToResource` or `RemoveTagsFromResource`) on a DB instance that has a DB instance name that is prefixed with the user's name and that has a tag called `stage` equal to `devo` or that has no tag called `stage`.

The `Resource` line in the policy identifies a resource by its Amazon Resource Name (ARN). For more information about using ARNs with Amazon RDS resources, see Working with Amazon Resource Names (ARNs) in Amazon RDS (p. 184).

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowFullDevAccessNoTags",
         "Effect":"Allow",
         "NotAction":[
            "rds:AddTagsToResource",
            "rds:RemoveTagsFromResource"
         ],
         "Resource":"arn:aws:rds:*:123456789012:db:${aws:username}*",
         "Condition":{
            "StringEqualsIfExists":{
               "rds:db-tag/stage":"devo"
            }
         }
      }
   ]
}
```

Related Topics

- Access Control (p. 328)
- Amazon RDS API Permissions: Actions, Resources, and Conditions Reference (p. 335)
- Security in Amazon RDS (p. 326)

Encrypting Amazon RDS Resources

You can encrypt your Amazon RDS instances and snapshots at rest by enabling the encryption option for your Amazon RDS DB instance. Data that is encrypted at rest includes the underlying storage for a DB instance, its automated backups, Read Replicas, and snapshots.

Amazon RDS encrypted instances use the industry standard AES-256 encryption algorithm to encrypt your data on the server that hosts your Amazon RDS instance. Once your data is encrypted, Amazon RDS handles authentication of access and decryption of your data transparently with a minimal impact on performance. You don't need to modify your database client applications to use encryption.

Topics

- Enabling Amazon RDS Encryption for a DB Instance (p. 355)
- Availability of Amazon RDS Encrypted Instances (p. 356)
- Managing Amazon RDS Encryption Keys (p. 357)
- Limitations of Amazon RDS Encrypted Instances (p. 358)

Amazon RDS encrypted instances provide an additional layer of data protection by securing your data from unauthorized access to the underlying storage. You can use Amazon RDS encryption to increase data protection of your applications deployed in the cloud, and to fulfill compliance requirements for data-at-rest encryption.

Amazon RDS also supports encrypting an Oracle or SQL Server DB instance with Transparent Data Encryption (TDE). TDE can be used with encryption at rest, although using TDE and encryption at rest simultaneously might slightly affect the performance of your database. You must manage different keys for each encryption method. For more information on TDE, see Oracle Transparent Data Encryption (p. 1036), Using AWS CloudHSM Classic to Store Amazon RDS Oracle TDE Keys (p. 1086), or Microsoft SQL Server Transparent Data Encryption Support (p. 797).

To manage the keys used for encrypting and decrypting your Amazon RDS resources, you use the AWS Key Management Service (AWS KMS). AWS KMS combines secure, highly available hardware and software to provide a key management system scaled for the cloud. Using AWS KMS, you can create encryption keys and define the policies that control how these keys can be used. AWS KMS supports CloudTrail, so you can audit key usage to verify that keys are being used appropriately. Your AWS KMS keys can be used in combination with Amazon RDS and supported AWS services such as Amazon Simple Storage Service (Amazon S3), Amazon Elastic Block Store (Amazon EBS), and Amazon Redshift. For a list of services that support AWS KMS, go to Supported Services in the *AWS Key Management Service Developer Guide*.

All logs, backups, and snapshots are encrypted for an Amazon RDS encrypted instance. A Read Replica of an Amazon RDS encrypted instance is also encrypted using the same key as the master instance when both are in the same region. If the master and Read Replica are in different regions, you encrypt using the encryption key for that region.

Enabling Amazon RDS Encryption for a DB Instance

To enable encryption for a new DB instance, choose **Yes** for **Enable encryption** on the Amazon RDS console. For information on creating a DB instance, see one of the following topics:

- Creating a DB Instance Running the MySQL Database Engine (p. 830)
- Creating a DB Instance Running the Oracle Database Engine (p. 949)
- Creating a DB Instance Running the Microsoft SQL Server Database Engine (p. 738)
- Creating a DB Instance Running the PostgreSQL Database Engine (p. 1172)
- Creating an Amazon Aurora DB Cluster (p. 437)
- Creating a DB Instance Running the MariaDB Database Engine (p. 678)

If you use the create-db-instance AWS CLI command to create an encrypted RDS DB instance, set the `--storage-encrypted` parameter to true. If you use the CreateDBInstance API action, set the `StorageEncrypted` parameter to true.

When you create an encrypted DB instance, you can also supply the AWS KMS key identifier for your encryption key. If you don't specify an AWS KMS key identifier, then Amazon RDS uses your default encryption key for your new DB instance. AWS KMS creates your default encryption key for Amazon RDS for your AWS account. Your AWS account has a different default encryption key for each AWS Region.

Once you have created an encrypted DB instance, you cannot change the encryption key for that instance. Therefore, be sure to determine your encryption key requirements before you create your encrypted DB instance.

If you use the AWS CLI `create-db-instance` command to create an encrypted RDS DB instance, set the `--kms-key-id` parameter to the Amazon Resource Name (ARN) for the AWS KMS encryption key for the DB instance. If you use the Amazon RDS API `CreateDBInstance` action, set the `KmsKeyId` parameter to the ARN for your AWS KMS key for the DB instance.

You can use the ARN of a key from another account to encrypt an RDS DB instance. Or you might create a DB instance with the same AWS account that owns the AWS KMS encryption key used to encrypt that new DB instance. In this case, the AWS KMS key ID that you pass can be the AWS KMS key alias instead of the key's ARN.

Important
If Amazon RDS loses access to the encryption key for a DB instance—for example, when RDS access to a key is revoked—then the encrypted DB instance goes into a terminal state. In this case, you can only restore the DB instance from a backup. We strongly recommend that you always enable backups for encrypted DB instances to guard against the loss of encrypted data in your databases.

Availability of Amazon RDS Encrypted Instances

Amazon RDS encrypted instances are currently available for all database engines and storage types. Amazon RDS encryption is not currently available in the China (Beijing) region.

Amazon RDS encryption is available for the following DB instance classes:

Instance Type	Instance Class
General Purpose (M4)—Current Generation	db.m4.large db.m4.xlarge db.m4.2xlarge db.m4.4xlarge db.m4.10xlarge db.m4.16xlarge

Instance Type	Instance Class
Memory Optimized (R3)—Current Generation	db.r3.large db.r3.xlarge db.r3.2xlarge db.r3.4xlarge db.r3.8xlarge
Memory Optimized (R4)—Next Generation	db.r4.large db.r4.xlarge db.r4.2xlarge db.r4.4xlarge db.r4.8xlarge db.r4.16xlarge
Burst Capable (T2)—Current Generation	db.t2.small db.t2.medium db.t2.large db.t2.xlarge db.t2.2xlarge
General Purpose (M3)—Previous Generation	db.m3.medium db.m3.large db.m3.xlarge db.m3.2xlarge

Note
Encryption at rest is not available for DB instances running SQL Server Express Edition.

Managing Amazon RDS Encryption Keys

You can manage keys used for Amazon RDS encrypted instances using the AWS Key Management Service (AWS KMS) in the IAM console. If you want full control over a key, then you must create a customer-managed key. You cannot delete, revoke, or rotate default keys provisioned by AWS KMS.

You can view audit logs of every action taken with a customer-managed key by using AWS CloudTrail.

Important
If you disable the key for an encrypted DB instance, you cannot read from or write to that DB instance. When Amazon RDS encounters a DB instance encrypted by a key that Amazon RDS doesn't have access to, Amazon RDS puts the DB instance into a terminal state. In this state, the DB instance is no longer available and the current state of the database can't be recovered. To restore the DB instance, you must re-enable access to the encryption key for Amazon RDS, and then restore the DB instance from a backup.

Limitations of Amazon RDS Encrypted Instances

The following limitations exist for Amazon RDS encrypted instances:

- You can only enable encryption for an Amazon RDS DB instance when you create it, not after the DB instance is created.

  However, because you can encrypt a copy of an unencrypted DB snapshot, you can effectively add encryption to an unencrypted DB instance. That is, you can create a snapshot of your DB instance, and then create an encrypted copy of that snapshot. You can then restore a DB instance from the encrypted snapshot, and thus you have an encrypted copy of your original DB instance. For more information, see Copying a DB Snapshot or DB Cluster Snapshot (p. 213). You don't need to encrypt an Amazon Aurora DB cluster snapshot in order to create an encrypted copy of an Aurora DB cluster. If you specify a KMS encryption key when restoring from an unencrypted DB cluster snapshot, the restored DB cluster is encrypted using the specified KMS encryption key.
- DB instances that are encrypted cannot be modified to disable encryption.
- You cannot have an encrypted Read Replica of an unencrypted DB instance or an unencrypted Read Replica of an encrypted DB instance.
- Encrypted Read Replicas must be encrypted with the same key as the source DB instance.
- You cannot restore an unencrypted backup or snapshot to an encrypted DB instance. You can, however, restore an unencrypted Aurora DB cluster snapshot to an encrypted Aurora DB cluster if you specify a KMS encryption key when you restore from the unencrypted DB cluster snapshot.
- To copy an encrypted snapshot from one region to another, you must specify the KMS key identifier of the destination region. This is because KMS encryption keys are specific to the region that they are created in.

The source snapshot remains encrypted throughout the copy process. AWS Key Management Service uses envelope encryption to protect data during the copy process. For more information about envelope encryption, see Envelope Encryption.

Using SSL to Encrypt a Connection to a DB Instance

You can use SSL from your application to encrypt a connection to a DB instance running MySQL, MariaDB, Amazon Aurora, SQL Server, Oracle, or PostgreSQL. Each DB engine has its own process for implementing SSL. To learn how to implement SSL for your DB instance, use the link following that corresponds to your DB engine:

- Securing Aurora Data with SSL (p. 434)
- SSL Support for MariaDB DB Instances (p. 674)
- Using SSL with a Microsoft SQL Server DB Instance (p. 791)
- SSL Support for MySQL DB Instances (p. 826)
- SSL Support for Oracle DB Instances (p. 936)
- Using SSL with a PostgreSQL DB Instance (p. 1170)

A root certificate that works for all regions can be downloaded at https://s3.amazonaws.com/rds-downloads/rds-ca-2015-root.pem. It is the trusted root entity and should work in most cases but might fail if your application doesn't accept certificate chains. If your application doesn't accept certificate chains, download the AWS Region–specific certificate from the list of intermediate certificates found later in this section.

A certificate bundle that contains both the old and new root certificates can be downloaded at https://s3.amazonaws.com/rds-downloads/rds-combined-ca-bundle.pem .

If your application is on the Microsoft Windows platform and requires a PKCS7 file, you can download the PKCS7 certificate bundle that contains both the old and new certificates at https://s3.amazonaws.com/rds-downloads/rds-combined-ca-bundle.p7b .

Intermediate Certificates

You might need to use an intermediate certificate to connect to your region. For example, you must use an intermediate certificate to connect to the AWS GovCloud (US) region using SSL. If you need an intermediate certificate for a particular AWS Region, download the certificate from the following list:

Asia Pacific (Mumbai)

Asia Pacific (Tokyo)

Asia Pacific (Seoul)

Asia Pacific (Singapore)

Asia Pacific (Sydney)

EU (Frankfurt)

EU (Ireland)

South America (São Paulo)

US East (N. Virginia)

US East (Ohio)

US West (N. California)

US West (Oregon)

China (Beijing)

AWS GovCloud (US) (CA-2012; for CA-2017, see following)

GovCloud (US) SSL Certificates 2017

To maintain connectivity, you need to update the CA-2012 SSL certificates your client or application is using to connect to RDS before August 15, 2017, at 20:00 UTC. Follow these steps:

1. Download the new AWS GovCloud Intermediate SSL certificate bundle.
2. Use the new certificates you downloaded in the previous step to update your database client or application by following the steps on the download page. This action is specific to the configuration of your client or application.

Use the Modify operation for your RDS instance on the AWS Management Console (or the ModifyDBInstance API) to change the Certificate Authority (CA) from rds-ca-2012 to rds-ca-2017, and then choose **Apply Immediately**. This operation updates the SSL certificates on the RDS instance and initiates a reboot operation to have the new certificates take effect. This reboot operation typically takes less than two minutes to complete. In some cases, such as when a database has a large number of tables, a reboot can take longer. For more information, see Best Practices for Amazon RDS (p. 80).

IAM Database Authentication for MySQL and Amazon Aurora

With Amazon RDS for MySQL or Aurora with MySQL compatibility, you can authenticate to your DB instance or DB cluster using AWS Identity and Access Management (IAM) database authentication. With this authentication method, you don't need to use a password when you connect to a DB instance. Instead, you use an authentication token.

An *authentication token* is a unique string of characters that Amazon RDS generates on request. Authentication tokens are generated using AWS Signature Version 4. Each token has a lifetime of 15 minutes. You don't need to store user credentials in the database, because authentication is managed externally using IAM. You can also still use standard database authentication.

IAM database authentication provides the following benefits:

- Network traffic to and from the database is encrypted using Secure Sockets Layer (SSL).
- You can use IAM to centrally manage access to your database resources, instead of managing access individually on each DB instance or DB cluster.
- For applications running on Amazon EC2, you can use EC2 instance profile credentials to access the database instead of a password, for greater security.

Topics

Availability for IAM Database Authentication

IAM database authentication is available for the following database engines and instance classes:

- MySQL 5.6, minor version 5.6.34 or higher. All instance classes are supported, except for `db.m1.small`.
- MySQL 5.7, minor version 5.7.16 or higher. All instance classes are supported, except for `db.m1.small`.
- Amazon Aurora 1.10 or higher. All instance classes are supported, except for `db.t2.small`.

Limitations for IAM Database Authentication

With IAM database authentication, you are limited to a maximum of 20 connections per second. If you are using a `db.t2.micro` instance class, the limit is 10 connections per second.

The Amazon RDS for MySQL and Aurora MySQL database engines do not impose any limits on authentication attempts per second. However, when you use IAM database authentication, your application must generate an authentication token. Your application then uses that token to connect to the DB instance or cluster. If you exceed the maximum connection-per-second limit, then the extra overhead of IAM database authentication can cause connection throttling. The extra overhead can even cause existing connections to drop.

We recommend the following:

- Use IAM database authentication as a mechanism for temporary, personal access to databases.
- Don't use IAM database authentication if your application requires more than the maximum number of connections.
- Use IAM database authentication only for workloads that can be easily retried.

Enabling and Disabling IAM Database Authentication

By default, IAM database authentication is disabled on DB instances and DB clusters. You can enable IAM database authentication (or disable it again) using the AWS Management Console, AWS CLI, or the Amazon RDS API.

Topics

AWS Management Console

To create a new DB instance or DB cluster with IAM authentication by using the console, see the following workflows:

- For Amazon RDS for MySQL, see Creating a DB Instance Running the MySQL Database Engine (p. 830).
- For Aurora MySQL, see Creating an Amazon Aurora DB Cluster (p. 437).

Each of these creation workflows has a **Configure Advanced Settings** page, where you can enable IAM DB authentication. In that page's **Database Options** section, choose **Yes** for **Enable IAM DB Authentication**.

To enable or disable IAM authentication for an existing DB instance or cluster

1. Open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. From the dashboard, choose either **Instances** or **Clusters**.
3. Choose the DB instance or DB cluster that you want to modify, and then choose **Instance Actions**, **Modify** or **Modify Cluster** as appropriate.
4. In the **Database Options** section, for **Enable IAM DB Authentication** choose **Yes** (to enable) or **No** (to disable), and then choose **Continue**.
5. Choose **Modify DB Instance** or **Modify Cluster** as appropriate.

To restore a DB instance or cluster

1. Open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. From the dashboard, choose **Snapshots**.
3. Choose the snapshot you want to restore, and then choose **Snapshot Actions**, **Restore Snapshot**.
4. In the **Database Options** section, go to **Enable IAM DB Authentication** and choose **Yes** (to enable) or **No** (to disable).
5. Choose **Restore DB Instance**.

AWS CLI

To create a new DB instance or DB cluster with IAM authentication by using the AWS CLI, use one of the following commands:

- `create-db-instance` for Amazon RDS MySQL
- `create-db-cluster` for Aurora MySQL

Specify the `--enable-iam-database-authentication` option, as shown in the following example.

```
aws rds create-db-instance \
    --db-instance-identifier mydbinstance \
    --db-instance-class db.m3.medium \
    --engine MySQL \
    --allocated-storage 20 \
    --master-username masterawsuser \
    --master-user-password masteruserpassword \
    --enable-iam-database-authentication
```

For an existing DB instance or DB cluster, use one of the following AWS CLI commands:

- `modify-db-instance` for Amazon RDS MySQL
- `modify-db-cluster` for Aurora MySQL

Specify either the `--enable-iam-database-authentication` or `--no-enable-iam-database-authentication` option, as appropriate.

By default, Amazon RDS modifies the DB instance during the next maintenance window. If you want to override this and enable IAM DB authentication as soon as possible, use the `--apply-immediately` parameter.

The following example shows how to immediately enable IAM authentication for an existing DB instance.

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --apply-immediately \
    --enable-iam-database-authentication
```

If you are restoring a DB instance or DB cluster, use one of the following AWS CLI commands:

- `aws rds restore-db-instance-to-point-in-time`
- `aws rds restore-db-instance-from-db-snapshot`

The IAM database authentication setting defaults to that of the source snapshot. To change this setting, set the `--enable-iam-database-authentication` or `--no-enable-iam-database-authentication` option, as appropriate.

Amazon RDS API

For a new DB instance or DB cluster, use one of the following API actions:

- `CreateDBInstance` for Amazon RDS MySQL
- `CreateDBCluster` for Aurora MySQL

Set the `EnableIAMDatabaseAuthentication` parameter to `true`.

For an existing DB instance or DB cluster, use one of the following API actions:

- `ModifyDBInstance` for Amazon RDS MySQL
- `ModifyDBCluster` for Aurora MySQL

Set the `EnableIAMDatabaseAuthentication` to `true` to enable IAM authentication, or `false` to disable it.

If you are restoring a DB instance or DB cluster, use one of the following API actions:

- `RestoreDBInstanceToPointInTime`
- `RestoreDBInstanceFromDBSnapshot`

The IAM database authentication setting defaults to that of the source snapshot. To change this setting, set the `EnableIAMDatabaseAuthentication` to `true` to enable IAM authentication, or `false` to disable it.

Creating and Using an IAM Policy for IAM Database Access

To allow an IAM user or role to connect to your DB instance or DB cluster, you must create an IAM policy. After that, you attach the policy to an IAM user or role.

> **Note**
> To learn more about IAM policies, see Authentication and Access Control for Amazon RDS (p. 327).

The following example policy allows an IAM user to connect to a DB instance using IAM database authentication.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
             "rds-db:connect"
         ],
         "Resource": [
             "arn:aws:rds-db:us-west-2:123456789012:dbuser:db-12ABC34DEFG5HIJ6KLMNOP78QR/
jane_doe"
         ]
      }
   ]
}
```

> **Important**
> Don't confuse the `rds-db:` prefix with other Amazon RDS action prefixes that begin with `rds:`. You use the `rds-db:` prefix and the `rds-db:connect` action only for IAM database authentication. They aren't valid in any other context.

The example policy includes a single statement with the following elements:

- `Effect`—Specify `Allow` to grant access to the DB instance. If you don't explicitly allow access, then access is denied by default.

- `Action`—Specify `rds-db:connect` to allow connection to the DB instance.
- `Resource`—Specify an Amazon Resource Name (ARN) that describes one database account in one DB instance. The ARN format is as follows.

```
arn:aws:rds-db:region:account-id:dbuser:dbi-resource-id/database-user-name
```

  In this format, the following are so:

  - *region* is the AWS Region for the Amazon RDS DB instance. In the example policy, the AWS Region is `us-west-2`.
  - *account-id* is the AWS account number for the DB instance. In the example policy, the account number is `123456789012`.
  - *dbi-resource-id* is the identifier for the DB instance. This identifier is unique to an AWS Region and never changes. In the example policy, the identifier is `db-12ABC34DEFG5HIJ6KLMNOP78QR`.

    To find a DB instance resource ID in the AWS Management Console for Amazon RDS, choose the DB instance you want, and then choose **Instance Actions**, **See Details**. The **Resource ID** is shown in the **Configuration Details** section.

    Alternatively, you can use the AWS CLI command to list the identifiers and resource IDs for all of your DB instances in the current AWS Region, as shown following.

```
aws rds describe-db-instances \
    --query "DBInstances[*].[DBInstanceIdentifier,DbiResourceId]"
```

  - *db-user-name* is the name of the MySQL database account to associate with IAM authentication. In the example policy, the database account is `jane_doe`.

You can construct other ARNs to support various access patterns. The following policy allows access to two different database accounts in a DB instance:

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
             "rds-db:connect"
         ],
         "Resource": [
             "arn:aws:rds-db:us-west-2:123456789012:dbuser:db-12ABC34DEFG5HIJ6KLMNOP78QR/
jane_doe",
             "arn:aws:rds-db:us-west-2:123456789012:dbuser:db-12ABC34DEFG5HIJ6KLMNOP78QR/
mary_roe"
         ]
      }
   ]
}
```

The following IAM policy allows access to a DB cluster, rather than a DB instance. The cluster identifier is `cluster-CO4FHMOYDKJ7CVBEJS2UWDQX7I`.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
```

```
            "Effect": "Allow",
            "Action": [
                "rds-db:connect"
            ],
            "Resource": [
                "arn:aws:rds-db:us-west-2:123456789012:dbuser:cluster-
CO4FHMOYDKJ7CVBEJS2UWDQX7I/jane_doe"
            ]
        }
    ]
}
```

To find a DB cluster resource ID in the AWS Management Console for Amazon RDS, choose the DB cluster you want and expand the selection, and then choose **Instance Actions**, **See Details**. The **Resource ID** is shown in the **DB Cluster Details** section.

Alternatively, you can use the AWS CLI command to list the identifiers and resource IDs for all of your DB clusters in the current AWS Region, as shown following.

```
aws rds describe-db-clusters \
    --query "DBClusters[*].[DBClusterIdentifier,DbClusterResourceId]"
```

The following policy uses the "*" character to match all of the DB instances and DB clusters for a particular AWS account and AWS Region. However, the policy only grants access to DB instances or DB clusters that have a `jane_doe` database account.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
             "rds-db:connect"
         ],
         "Resource": [
             "arn:aws:rds-db:us-west-2:123456789012:dbuser:*/jane_doe"
         ]
      }
   ]
}
```

The IAM user or role has access to only those databases that the database user does. For example, suppose that your DB instance has a database named *dev*, and another database named *test*. If the database user `jane_doe` has access only to *dev*, any IAM users or roles that access that DB instance with the `jane_doe` user also have access only to *dev*. This access restriction is also true for other database objects, such as tables, views, and so on.

Attaching an IAM Policy to an IAM User or Role

After you create an IAM policy to allow database authentication, you need to attach the policy to an IAM user or role. For a tutorial on this topic, see Create and Attach Your First Customer Managed Policy in the *IAM User Guide*.

As you work through the tutorial, you can use one of the policy examples shown in this section as a starting point and tailor it to your needs. At the end of the tutorial, you have an IAM user with an attached policy that can make use of the `rds-db:connect` action.

Note
You can map multiple IAM users or roles to the same database user account. For example, suppose that your IAM policy specified the following resource ARN.

```
arn:aws:rds-db:us-west-2:123456789012:dbuser:db-12ABC34DEFG5HIJ6KLMNOP78QR/jane_doe
```

If you attach the policy to IAM users *Jane*, *Bob*, and *Diego*, then each of those users can connect to the specified DB instance using the `jane_doe` database account.

Creating a Database Account

With IAM database authentication, you don't need to assign database passwords to the MySQL user accounts you create. Instead, authentication is handled by `AWSAuthenticationPlugin`—an AWS-provided plugin that works seamlessly with IAM to authenticate your IAM users.

To create a database account for MySQL, connect to the DB instance or DB cluster and issue the `CREATE USER` statement, as shown in the following example.

```
CREATE USER jane_doe IDENTIFIED WITH AWSAuthenticationPlugin AS 'RDS';
```

The `IDENTIFIED WITH` clause allows MySQL to use the `AWSAuthenticationPlugin` to authenticate the database account (`jane_doe`). The `AS 'RDS'` clause maps the `jane_doe` database account to the corresponding IAM user or role.

Note
If you see the following message, it means that the AWS-provided plugin is not available for the current DB instance or DB cluster.
`ERROR 1524 (HY000): Plugin 'AWSAuthenticationPlugin' is not loaded`
To troubleshoot this error, verify that you are using a supported configuration and that you have enabled IAM database authentication on your DB instance or DB cluster. For more information, see Availability for IAM Database Authentication (p. 360) and Enabling and Disabling IAM Database Authentication (p. 361).

After you create an account using `AWSAuthenticationPlugin`, you manage it in the same way as other database accounts. For example, you can modify account privileges with `GRANT` and `REVOKE` statements, or modify various account attributes with the `ALTER USER` statement.

If you remove an IAM user that is mapped to a database account, you should also remove the database account with the `DROP USER` statement.

Connecting to the DB Instance or DB Cluster

With IAM database authentication, you use an authentication token when you connect to your DB instance or DB cluster. An *authentication token* is a string of characters that you use instead of a password. Once you generate an authentication token, it's valid for 15 minutes before it expires. If you try to connect using an expired token, the connection request is denied.

Every authentication token must be accompanied by a valid signature, using AWS signature version 4. (For more information, see Signature Version 4 Signing Process in the AWS General Reference.) The AWS CLI and the AWS SDK for Java can automatically sign each token you create.

Alternatively, you can use the AWS SDK for Java to manually create and manually sign an authentication token.

Once you have a signed IAM authentication token, you can connect to an Amazon RDS DB instance or Aurora DB cluster. Following, you can find out how to do this using either the `mysql` command line tool or the AWS SDK for Java.

Topics

Command Line: AWS CLI and mysql Client

You can connect from the command line to an RDS DB instance or Aurora DB cluster with the AWS CLI and `mysql` command line tool as described following.

Topics

Generating an Authentication Token

The following example shows how to get a signed authentication token using the AWS CLI.

```
aws rds generate-db-auth-token \
   --hostname rdsmysql.cdgmuqiadpid.us-west-2.rds.amazonaws.com \
   --port 3306 \
   --region us-west-2 \
   --username jane_doe
```

In the example, the parameters are as follows:

- `--hostname` — The host name of the DB instance or DB cluster that you want to access.
- `--port` — The port number used for connecting to the DB instance or DB cluster.
- `--region` — The AWS Region where the DB instance or DB cluster is running.
- `--username` — The database account that you want to access.

The first several characters of the token look like the following.

```
rdsmysql.cdgmuqiadpid.us-west-2.rds.amazonaws.com:3306/?Action=connect&DBUser=jane_doe&X-
Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Expires=900...
```

Connecting to a DB Instance or DB Cluster

The general format for connecting is shown following.

```
mysql --host=hostName --port=portNumber --ssl-ca=[full path]rds-combined-ca-bundle.pem --
enable-cleartext-plugin --user=userName --password=authToken
```

The parameters are as follows:

- `--host` — The host name of the DB instance or DB cluster that you want to access.
- `--port` — The port number used for connecting to the DB instance or DB cluster.
- `--ssl-ca` — The SSL certificate file that contains the public key. For more information, see Using SSL to Encrypt a Connection to a DB Instance (p. 358).
- `--enable-cleartext-plugin` — A value that specifies that `AWSAuthenticationPlugin` must be used for this connection.
- `--user` — The database account that you want to access.
- `--password` — A signed IAM authentication token.

The authentication token consists of several hundred characters. It can be unwieldy on the command line. One way to work around this is to save the token to an environment variable, and then use that variable when you connect. The following example shows one way to perform this workaround.

```
RDSHOST="rdsmysql.cdgmuqiadpid.us-west-2.rds.amazonaws.com"
TOKEN="$(aws rds generate-db-auth-token --hostname $RDSHOST --port 3306 --username
 jane_doe )"

mysql --host=$RDSHOST --port=3306 --ssl-ca=/sample_dir/rds-combined-ca-bundle.pem --enable-
cleartext-plugin --user=jane_doe --password=$TOKEN
```

When you connect using `AWSAuthenticationPlugin`, the connection is secured using SSL. To verify this, type the following at the `mysql>` command prompt.

```
show status like 'Ssl%';
```

The following lines in the output show more details.

```
+---------------+-------------+
| Variable_name | Value

                                                                  |
+---------------+-------------+
| ...           | ...
| Ssl_cipher    | AES256-SHA

                                                                  |
| ...           | ...
| Ssl_version   | TLSv1.1

                                                                  |
| ...           | ...
+-----------------------------+
```

AWS SDK for Java

You can connect from the command line to an RDS DB instance or Aurora DB cluster with the AWS SDK for Java as described following.

Topics

- Generating an Authentication Token (p. 368)
- Manually Constructing an Authentication Token (p. 369)
- Connecting to a DB Instance or DB Cluster (p. 372)

Generating an Authentication Token

If you are writing programs using the AWS SDK for Java, you can get a signed authentication token using the `RdsIamAuthTokenGenerator` class. Using this class requires that you provide AWS credentials. To do this, you create an instance of the `DefaultAWSCredentialsProviderChain` class. `DefaultAWSCredentialsProviderChain` uses the first AWS access key and secret key that it finds in the default credential provider chain. For more information about AWS access keys, see Managing Access Keys for IAM Users.

After you create an instance of `RdsIamAuthTokenGenerator`, you can call the `getAuthToken` method to obtain a signed token. Provide the AWS Region, host name, port number, and user name. The following code example illustrates how to do this.

```
package com.amazonaws.codesamples;

import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.services.rds.auth.GetIamAuthTokenRequest;
import com.amazonaws.services.rds.auth.RdsIamAuthTokenGenerator;

public class GenerateRDSAuthToken {

    public static void main(String[] args) {

     String region = "us-west-2";
     String hostname = "rdsmysql.cdgmuqiadpid.us-west-2.rds.amazonaws.com";
     String port = "3306";
     String username = "jane_doe";

     System.out.println(generateAuthToken(region, hostname, port, username));
    }

    static String generateAuthToken(String region, String hostName, String port, String
 username) {

     RdsIamAuthTokenGenerator generator = RdsIamAuthTokenGenerator.builder()
      .credentials(new DefaultAWSCredentialsProviderChain())
      .region(region)
      .build();

     String authToken = generator.getAuthToken(
      GetIamAuthTokenRequest.builder()
      .hostname(hostName)
      .port(Integer.parseInt(port))
      .userName(username)
      .build());

     return authToken;
    }

}
```

Manually Constructing an Authentication Token

In Java, the easiest way to generate an authentication token is to use `RdsIamAuthTokenGenerator`. This class creates an authentication token for you, and then signs it using AWS signature version 4. For more information, see Signature Version 4 Signing Process in the *AWS General Reference.*

However, you can also construct and sign an authentication token manually, as shown in the following code example.

```
package com.amazonaws.codesamples;

import com.amazonaws.SdkClientException;
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.auth.SigningAlgorithm;
import com.amazonaws.util.BinaryUtils;
import org.apache.commons.lang3.StringUtils;

import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.nio.charset.Charset;
import java.security.MessageDigest;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.SortedMap;
import java.util.TreeMap;
```

```
import static com.amazonaws.auth.internal.SignerConstants.AWS4_TERMINATOR;
import static com.amazonaws.util.StringUtils.UTF8;

public class CreateRDSAuthTokenManually {
    public static String httpMethod = "GET";
    public static String action = "connect";
    public static String canonicalURIParameter = "/";
    public static SortedMap<String, String> canonicalQueryParameters = new TreeMap();
    public static String payload = StringUtils.EMPTY;
    public static String signedHeader = "host";
    public static String algorithm = "AWS4-HMAC-SHA256";
    public static String serviceName = "rds-db";
    public static String requestWithoutSignature;

    public static void main(String[] args) throws Exception {

        String region = "us-west-2";
        String instanceName = "rdsmysql.cdgmuqiadpid.us-west-2rds.amazonaws.com";
        String port = "3306";
        String username = "jane_doe";

        Date now = new Date();
        String date = new SimpleDateFormat("yyyyMMdd").format(now);
        String dateTimeStamp = new SimpleDateFormat("yyyyMMdd'T'HHmmssZ").format(now);
        DefaultAWSCredentialsProviderChain creds = new
 DefaultAWSCredentialsProviderChain();
     String awsAccessKey = creds.getCredentials().getAWSAccessKeyId();
     String awsSecretKey = creds.getCredentials().getAWSSecretKey();
        String expiryMinutes = "900";

        System.out.println("Step 1:  Create a canonical request:");
        String canonicalString = createCanonicalString(username, awsAccessKey, date,
 dateTimeStamp, region, expiryMinutes, instanceName, port);
        System.out.println(canonicalString);
        System.out.println();

        System.out.println("Step 2:  Create a string to sign:");
        String stringToSign = createStringToSign(dateTimeStamp, canonicalString,
 awsAccessKey, date, region);
        System.out.println(stringToSign);
        System.out.println();

        System.out.println("Step 3:  Calculate the signature:");
        String signature = BinaryUtils.toHex(calculateSignature(stringToSign,
 newSigningKey(awsSecretKey, date, region, serviceName)));
        System.out.println(signature);
        System.out.println();

        System.out.println("Step 4:  Add the signing info to the request");
        System.out.println(appendSignature(signature));
        System.out.println();

    }

    //Step 1: Create a canonical request date should be in format YYYYMMDD and dateTime
 should be in format YYYYMMDDTHHMMSSZ
    public static String createCanonicalString(String user, String accessKey, String date,
 String dateTime, String region, String expiryPeriod, String hostName, String port) throws
 Exception {
        canonicalQueryParameters.put("Action", action);
        canonicalQueryParameters.put("DBUser", user);
        canonicalQueryParameters.put("X-Amz-Algorithm", "AWS4-HMAC-SHA256");
        canonicalQueryParameters.put("X-Amz-Credential", accessKey + "%2F" + date + "%2F" +
 region + "%2F" + serviceName + "%2Faws4_request");
        canonicalQueryParameters.put("X-Amz-Date", dateTime);
```

```
        canonicalQueryParameters.put("X-Amz-Expires", expiryPeriod);
        canonicalQueryParameters.put("X-Amz-SignedHeaders", signedHeader);
        String canonicalQueryString = "";
        while(!canonicalQueryParameters.isEmpty()) {
            String currentQueryParameter = canonicalQueryParameters.firstKey();
            String currentQueryParameterValue =
canonicalQueryParameters.remove(currentQueryParameter);
            canonicalQueryString = canonicalQueryString + currentQueryParameter + "=" +
currentQueryParameterValue;
            if (!currentQueryParameter.equals("X-Amz-SignedHeaders")) {
                canonicalQueryString += "&";
            }
        }
        String canonicalHeaders = "host:" + hostName + ":" + port + '\n';
        requestWithoutSignature = hostName + ":" + port + "/?" + canonicalQueryString;

        String hashedPayload = BinaryUtils.toHex(hash(payload));
        return httpMethod + '\n' + canonicalURIParameter + '\n' + canonicalQueryString +
'\n' + canonicalHeaders + '\n' + signedHeader + '\n' + hashedPayload;

    }

    //Step 2: Create a string to sign using sig v4
    public static String createStringToSign(String dateTime, String canonicalRequest,
String accessKey, String date, String region) throws Exception {
        String credentialScope = date + "/" + region + "/" + serviceName + "/aws4_request";
        return algorithm + '\n' + dateTime + '\n' + credentialScope + '\n' +
BinaryUtils.toHex(hash(canonicalRequest));

    }

    //Step 3: Calculate signature
    /**
     * Step 3 of the AWS Signature version 4 calculation. It involves deriving
     * the signing key and computing the signature. Refer to
     * http://docs.aws.amazon
     * .com/general/latest/gr/sigv4-calculate-signature.html
     */
    public static byte[] calculateSignature(String stringToSign,
                                            byte[] signingKey) {
        return sign(stringToSign.getBytes(Charset.forName("UTF-8")), signingKey,
                SigningAlgorithm.HmacSHA256);
    }

    public static byte[] sign(byte[] data, byte[] key,
                              SigningAlgorithm algorithm) throws SdkClientException {
        try {
            Mac mac = algorithm.getMac();
            mac.init(new SecretKeySpec(key, algorithm.toString()));
            return mac.doFinal(data);
        } catch (Exception e) {
            throw new SdkClientException(
                    "Unable to calculate a request signature: "
                            + e.getMessage(), e);
        }
    }

    public static byte[] newSigningKey(String secretKey,
                                       String dateStamp, String regionName, String serviceName)
{
        byte[] kSecret = ("AWS4" + secretKey).getBytes(Charset.forName("UTF-8"));
        byte[] kDate = sign(dateStamp, kSecret, SigningAlgorithm.HmacSHA256);
        byte[] kRegion = sign(regionName, kDate, SigningAlgorithm.HmacSHA256);
        byte[] kService = sign(serviceName, kRegion,
                SigningAlgorithm.HmacSHA256);
        return sign(AWS4_TERMINATOR, kService, SigningAlgorithm.HmacSHA256);
```

```
    }

    public static byte[] sign(String stringData, byte[] key,
                      SigningAlgorithm algorithm) throws SdkClientException {
        try {
            byte[] data = stringData.getBytes(UTF8);
            return sign(data, key, algorithm);
        } catch (Exception e) {
            throw new SdkClientException(
                    "Unable to calculate a request signature: "
                            + e.getMessage(), e);
        }
    }

    //Step 4: append the signature
    public static String appendSignature(String signature) {
        return requestWithoutSignature + "&X-Amz-Signature=" + signature;
    }

    public static byte[] hash(String s) throws Exception {
        try {
            MessageDigest md = MessageDigest.getInstance("SHA-256");
            md.update(s.getBytes(UTF8));
            return md.digest();
        } catch (Exception e) {
            throw new SdkClientException(
                    "Unable to compute hash while signing request: "
                            + e.getMessage(), e);
        }
    }
}
```

Connecting to a DB Instance or DB Cluster

The following code example shows how to generate an authentication token, and then use it to connect to an Amazon RDS instance running MySQL.

To run this code example, you need the AWS SDK for Java (https://aws.amazon.com/sdk-for-java). In addition, you need the following:

- MySQL Connector/J. This code example was tested with `mysql-connector-java-5.1.33-bin.jar`.
- An intermediate certificate for Amazon RDS that is specific to an AWS Region. (For more information, see Using SSL to Encrypt a Connection to a DB Instance (p. 358).) At runtime, the class loader looks for the certificate in the same directory as this Java code example, so that the class loader can find it.
- Modify the values of the following variables as needed:
  - `RDS_INSTANCE_HOSTNAME` – The host name of the DB instance or DB cluster that you want to access.
  - `RDS_INSTANCE_PORT` – The port number used for connecting to the DB instance or DB cluster.
  - `REGION_NAME` – The AWS Region where the DB instance or DB cluster is running.
  - `DB_USER` – The database account that you want to access.
  - `SSL_CERTIFICATE` – An SSL certificate for Amazon RDS that is specific to an AWS Region. To download a certificate for your AWS Region, see Intermediate Certificates (p. 359). Place the SSL certificate in the same directory as this Java program file, so that the class loader can find the certificate at runtime.

This code example obtains AWS credentials from the default credential provider chain.

```
package com.amazonaws.samples;
```

```
import com.amazonaws.services.rds.auth.RdsIamAuthTokenGenerator;
import com.amazonaws.services.rds.auth.GetIamAuthTokenRequest;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.security.KeyStore;
import java.security.cert.CertificateFactory;
import java.security.cert.X509Certificate;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Properties;

import java.net.URL;

public class IAMDatabaseAuthenticationTester {
    //AWS Credentials of the IAM user with policy enabling IAM Database Authenticated
 access to the db by the db user.
    private static final DefaultAWSCredentialsProviderChain creds = new
 DefaultAWSCredentialsProviderChain();
    private static final String AWS_ACCESS_KEY =
 creds.getCredentials().getAWSAccessKeyId();
    private static final String AWS_SECRET_KEY = creds.getCredentials().getAWSSecretKey();

    //Configuration parameters for the generation of the IAM Database Authentication token
    private static final String RDS_INSTANCE_HOSTNAME = "rdsmysql.cdgmuqiadpid.us-
west-2.rds.amazonaws.com";
    private static final int RDS_INSTANCE_PORT = 3306;
    private static final String REGION_NAME = "us-west-2";
    private static final String DB_USER = "jane_doe";
    private static final String JDBC_URL = "jdbc:mysql://" + RDS_INSTANCE_HOSTNAME + ":" +
 RDS_INSTANCE_PORT;

    private static final String SSL_CERTIFICATE = "rds-ca-2015-us-west-2.pem";

    private static final String KEY_STORE_TYPE = "JKS";
    private static final String KEY_STORE_PROVIDER = "SUN";
    private static final String KEY_STORE_FILE_PREFIX = "sys-connect-via-ssl-test-cacerts";
    private static final String KEY_STORE_FILE_SUFFIX = ".jks";
    private static final String DEFAULT_KEY_STORE_PASSWORD = "changeit";

    public static void main(String[] args) throws Exception {
        //get the connection
        Connection connection = getDBConnectionUsingIam();

        //verify the connection is successful
        Statement stmt= connection.createStatement();
        ResultSet rs=stmt.executeQuery("SELECT 'Success!' FROM DUAL;");
        while (rs.next()) {
             String id = rs.getString(1);
            System.out.println(id); //Should print "Success!"
        }

        //close the connection
        stmt.close();
        connection.close();
    }

    /**
```

```
    * This method returns a connection to the db instance authenticated using IAM Database
 Authentication
    * @return
    * @throws Exception
    */
   private static Connection getDBConnectionUsingIam() throws Exception {
       setSslProperties();
       return DriverManager.getConnection(JDBC_URL, setMySqlConnectionProperties());
   }

   /**
    * This method sets the mysql connection properties which includes the IAM Database
 Authentication token
    * as the password. It also specifies that SSL verification is required.
    * @return
    */
   private static Properties setMySqlConnectionProperties() {
       Properties mysqlConnectionProperties = new Properties();
       mysqlConnectionProperties.setProperty("verifyServerCertificate","true");
       mysqlConnectionProperties.setProperty("useSSL", "true");
       mysqlConnectionProperties.setProperty("user",DB_USER);
       mysqlConnectionProperties.setProperty("password",generateAuthToken());
       return mysqlConnectionProperties;
   }

   /**
    * This method generates the IAM Auth Token.
    * An example IAM Auth Token would look like follows:
    * btusi123.cmz7kenwo2ye.rds.cn-north-1.amazonaws.com.cn:3306/?
Action=connect&DBUser=iamtestuser&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Date=20171003T010726Z&X-Amz-SignedHeaders=host&X-Amz-Expires=899&X-Amz-
Credential=AKIAPFXHGVDI5RNFO4AQ%2F20171003%2Fcn-north-1%2Frds-db%2Faws4_request&X-Amz-
Signature=f9f45ef96c1f770cdad11a53e33ffa4c3730bc03fdee820cfdf1322eed15483b
    * @return
    */
   private static String generateAuthToken() {
       BasicAWSCredentials awsCredentials = new BasicAWSCredentials(AWS_ACCESS_KEY,
 AWS_SECRET_KEY);

       RdsIamAuthTokenGenerator generator = RdsIamAuthTokenGenerator.builder()
               .credentials(new
 AWSStaticCredentialsProvider(awsCredentials)).region(REGION_NAME).build();
       return generator.getAuthToken(GetIamAuthTokenRequest.builder()

 .hostname(RDS_INSTANCE_HOSTNAME).port(RDS_INSTANCE_PORT).userName(DB_USER).build());
   }

   /**
    * This method sets the SSL properties which specify the key store file, its type and
 password:
    * @throws Exception
    */
   private static void setSslProperties() throws Exception {
       System.setProperty("javax.net.ssl.trustStore", createKeyStoreFile());
       System.setProperty("javax.net.ssl.trustStoreType", KEY_STORE_TYPE);
       System.setProperty("javax.net.ssl.trustStorePassword", DEFAULT_KEY_STORE_PASSWORD);
   }

   /**
    * This method returns the path of the Key Store File needed for the SSL verification
 during the IAM Database Authentication to
    * the db instance.
    * @return
    * @throws Exception
    */
   private static String createKeyStoreFile() throws Exception {
```

```
        return createKeyStoreFile(createCertificate()).getPath();
    }

    /**
     *  This method generates the SSL certificate
     * @return
     * @throws Exception
     */
    private static X509Certificate createCertificate() throws Exception {
        CertificateFactory certFactory = CertificateFactory.getInstance("X.509");
        URL url = new File(SSL_CERTIFICATE).toURI().toURL();
        if (url == null) {
            throw new Exception();
        }
        try (InputStream certInputStream = url.openStream()) {
            return (X509Certificate) certFactory.generateCertificate(certInputStream);
        }
    }

    /**
     * This method creates the Key Store File
     * @param rootX509Certificate - the SSL certificate to be stored in the KeyStore
     * @return
     * @throws Exception
     */
    private static File createKeyStoreFile(X509Certificate rootX509Certificate) throws
 Exception {
        File keyStoreFile = File.createTempFile(KEY_STORE_FILE_PREFIX,
 KEY_STORE_FILE_SUFFIX);
        try (FileOutputStream fos = new FileOutputStream(keyStoreFile.getPath())) {
            KeyStore ks = KeyStore.getInstance(KEY_STORE_TYPE, KEY_STORE_PROVIDER);
            ks.load(null);
            ks.setCertificateEntry("rootCaCertificate", rootX509Certificate);
            ks.store(fos, DEFAULT_KEY_STORE_PASSWORD.toCharArray());
        }
        return keyStoreFile;
    }
}
```

Amazon RDS Security Groups

Security groups control the access that traffic has in and out of a DB instance. Three types of security groups are used with Amazon RDS: DB security groups, VPC security groups, and Amazon EC2 security groups. In simple terms, these work as follows:

- A DB security group controls access to EC2-Classic DB instances that are not in a VPC.
- A VPC security group controls access to DB instances and EC2 instances inside a VPC.
- An EC2 security group controls access to an EC2 instance.

By default, network access is turned off to a DB instance. You can specify rules in a security group that allows access from an IP address range, port, or EC2 security group. Once ingress rules are configured, the same rules apply to all DB instances that are associated with that security group. You can specify up to 20 rules in a security group.

DB Security Groups

DB security groups are used with DB instances that are not in a VPC and on the EC2-Classic platform. Each DB security group rule enables a specific source to access a DB instance that is associated with that

DB security group. The source can be a range of addresses (for example, 203.0.113.0/24), or an EC2 security group. When you specify an EC2 security group as the source, you allow incoming traffic from all EC2 instances that use that EC2 security group. DB security group rules apply to inbound traffic only; outbound traffic is not currently permitted for DB instances.

You don't need to specify a destination port number when you create DB security group rules. The port number defined for the DB instance is used as the destination port number for all rules defined for the DB security group. DB security groups can be created using the Amazon RDS API actions or the Amazon RDS page of the AWS Management Console.

For more information about working with DB security groups, see Working with DB Security Groups (EC2-Classic Platform) (p. 380).

VPC Security Groups

Each VPC security group rule enables a specific source to access a DB instance in a VPC that is associated with that VPC security group. The source can be a range of addresses (for example, 203.0.113.0/24), or another VPC security group. By specifying a VPC security group as the source, you allow incoming traffic from all instances (typically application servers) that use the source VPC security group. VPC security groups can have rules that govern both inbound and outbound traffic, though the outbound traffic rules do not apply to DB instances. You must use the Amazon EC2 API or the **Security Group** option on the VPC Console to create VPC security groups.

When you create rules for your VPC security group that allow access to the instances in your VPC, you must specify a port for each range of addresses that the rule allows access for. For example, if you want to enable SSH access to instances in the VPC, then you create a rule allowing access to TCP port 22 for the specified range of addresses.

You can configure multiple VPC security groups that allow access to different ports for different instances in your VPC. For example, you can create a VPC security group that allows access to TCP port 80 for web servers in your VPC. You can then create another VPC security group that allows access to TCP port 3306 for RDS MySQL DB instances in your VPC.

For more information on VPC security groups, see Security Groups in the *Amazon Virtual Private Cloud User Guide*.

DB Security Groups vs. VPC Security Groups

The following table shows the key differences between DB security groups and VPC security groups.

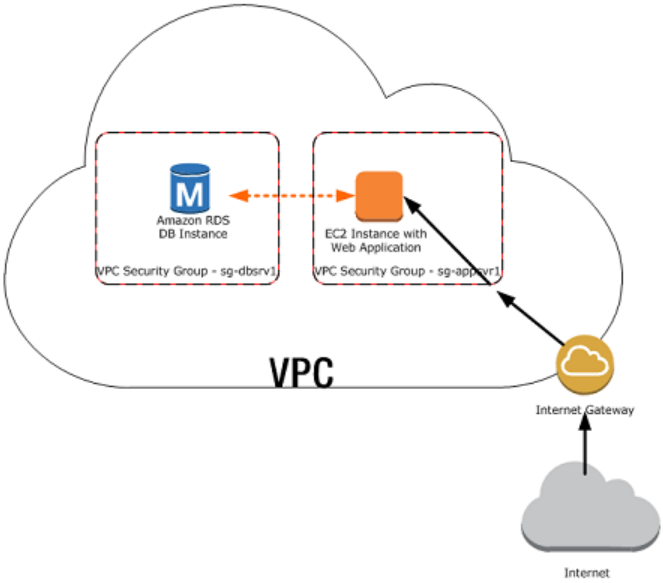
DB Security Group	VPC Security Group
Controls access to DB instances outside a VPC.	Controls access to DB instances in VPC.
Uses Amazon RDS API actions or the Amazon RDS page of the AWS Management Console to create and manage group and rules.	Uses Amazon EC2 API actions or the Amazon VPC page of the AWS Management Console to create and manage group and rules.
When you add a rule to a group, you don't need to specify port number or protocol.	When you add a rule to a group, specify the protocol as TCP. In addition, specify the same port number that you used to create the DB instances (or options) that you plan to add as members to the group.
Groups allow access from EC2 security groups in your AWS account or other accounts.	Groups allow access from other VPC security groups in your VPC only.

Security Group Scenario

A common use of an RDS instance in a VPC is to share data with an application server running in an Amazon EC2 instance in the same VPC, which is accessed by a client application outside the VPC. For this scenario, you use the RDS and VPC pages on the AWS Management Console or the RDS and EC2 API actions to create the necessary instances and security groups:

1. Create a VPC security group (for example, `sg-appsrv1`) and define inbound rules that use the IP addresses of the client application as the source. This security group allows your client application to connect to EC2 instances in a VPC that uses this security group.
2. Create an EC2 instance for the application and add the EC2 instance to the VPC security group (`sg-appsrv1`) that you created in the previous step. The EC2 instance in the VPC shares the VPC security group with the DB instance.
3. Create a second VPC security group (for example, `sg-dbsrv1`) and create a new rule by specifying the VPC security group that you created in step 1 (`sg-appsrv1`) as the source.
4. Create a new DB instance and add the DB instance to the VPC security group (`sg-dbsrv1`) that you created in the previous step. When you create the instance, use the same port number as the one specified for the VPC security group (`sg-dbsrv1`) rule that you created in step 3.

The following diagram shows this scenario.



For more information about using a VPC, see Amazon Virtual Private Cloud (VPCs) and Amazon RDS (p. 390).

Associating a Security Group with a DB Instance

You can associate a security group with a DB instance by using **Modify** on the RDS console, the `ModifyDBInstance` Amazon RDS API, or the AWS CLI `modify-db-instance` command. For information about modifying a DB instance, see Modifying an Amazon RDS DB Instance and Using the Apply Immediately Parameter (p. 114). For security group considerations when you restore a DB instance from a DB snapshot, see Security Group Considerations (p. 209).

Deleting DB VPC Security Groups

DB VPC security groups are an RDS mechanism to synchronize security information with a VPC security group. However, this synchronization is no longer required, because RDS has been updated to use VPC security group information directly.

> **Note**
> DB VPC security groups are deprecated, and they are different from DB security groups, VPC security groups, and EC2 security groups.

We strongly recommend that you delete any DB VPC security groups that you currently use. If you don't delete your DB VPC security groups, you might encounter unintended behaviors with your RDS DB instances, which can be as severe as losing access to a DB instance. The unintended behaviors are a result of an action such as an update to a DB instance, an option group, or similar. Such updates cause RDS to resynchronize the DB VPC security group with the VPC security group. This resynchronization can result in your security information being overwritten with incorrect and outdated security information. This result can have a severe impact on your access to your RDS DB instances.

How Can I Determine If I Have a DB VPC Security Group?

Because DB VPC security groups have been deprecated, they don't appear in the RDS console. However, you can call the describe-db-security-groups AWS CLI command or the DescribeDBSecurityGroups API action to determine if you have any DB VPC security groups.

In this case, you can call the `describe-db-security-groups` AWS CLI command with JSON specified as the output format. If you do, you can identify DB VPC security groups by the VPC identifier on the second line of the output for the security group as shown in the following example.

```
{
    "DBSecurityGroups": [
        {
            "VpcId": "vpc-abcd1234",
            "DBSecurityGroupDescription": "default:vpc-abcd1234",
            "IPRanges": [
                {
                    "Status": "authorized",
                    "CIDRIP": "xxx.xxx.xxx.xxx/n"
                },
                {
                    "Status": "authorized",
                    "CIDRIP": "xxx.xxx.xxx.xxx/n "
                }
            ],
            "OwnerId": "123456789012",
            "EC2SecurityGroups": [],
            "DBSecurityGroupName": "default:vpc-abcd1234"
        }
    ]
}
```

If you run the `DescribeDBSecurityGroups` API action, then you can identify DB VPC security groups using the <VpcId> response element as shown in the following example.

```
<DBSecurityGroup>
  <EC2SecurityGroups/>
  <DBSecurityGroupDescription>default:vpc-abcd1234</DBSecurityGroupDescription>
  <IPRanges>
    <IPRange>
      <CIDRIP>xxx.xxx.xxx.xxx/n</CIDRIP>
      <Status>authorized</Status>
```

```
    </IPRange>
    <IPRange>
      <CIDRIP>xxx.xxx.xxx.xxx/n</CIDRIP>
      <Status>authorized</Status>
    </IPRange>
  </IPRanges>
  <VpcId>vpc-abcd1234</VpcId>
  <OwnerId>123456789012</OwnerId>
  <DBSecurityGroupName>default:vpc-abcd1234</DBSecurityGroupName>
</DBSecurityGroup>
```

How Do I Delete a DB VPC Security Group?

Because DB VPC security groups don't appear in the RDS console, you must call the delete-db-security-group AWS CLI command or the DeleteDBSecurityGroup API action to delete a DB VPC security group.

After you delete a DB VPC security group, your DB instances in your VPC continue to be secured by the VPC security group for that VPC. The DB VPC security group that was deleted was merely a copy of the VPC security group information.

Review Your AWS CloudFormation Templates

Older versions of AWS CloudFormation templates can contain instructions to create a DB VPC security group. Because DB VPC security groups are not yet fully deprecated, they can still be created. Make sure that any AWS CloudFormation templates that you use to provision a DB instance with security settings don't also create a DB VPC security group. Don't use AWS CloudFormation templates that create an RDS `DBSecurityGroup` with an `EC2VpcId` as shown in the following example.

```
"DbSecurityByEC2SecurityGroup" : {
   Type" : "AWS::RDS::DBSecurityGroup",
   "Properties" : {
      "GroupDescription" : "Ingress for Amazon EC2 security group",
      "EC2VpcId" : { "MyVPC" },
      "DBSecurityGroupIngress" : [ {
         "EC2SecurityGroupId" : "sg-b0ff1111",
         "EC2SecurityGroupOwnerId" : "111122223333"
      }, {
         "EC2SecurityGroupId" : "sg-ffd722222",
         "EC2SecurityGroupOwnerId" : "111122223333"
      } ]
   }
}
```

Instead, add security information for your RDS DB instances in a VPC using VPC security groups, as shown in the following example.

```
"DBInstance" : {
  "Type": "AWS::RDS::DBInstance",
  "Properties": {
    "DBName"            : { "Ref" : "DBName" },
    "Engine"            : "MySQL",
    "MultiAZ"           : { "Ref": "MultiAZDatabase" },
    "MasterUsername"    : { "Ref" : "<master_username>" },
    "DBInstanceClass"   : { "Ref" : "DBClass" },
    "AllocatedStorage"  : { "Ref" : "DBAllocatedStorage" },
    "MasterUserPassword": { "Ref" : "<master_password>" },
    "VPCSecurityGroups" : [ { "Fn::GetAtt": [ "VPCSecurityGroup", "GroupId" ] } ]
}
```

Working with DB Security Groups (EC2-Classic Platform)

By default, network access is turned off to a DB instance. You can specify rules in a *security group* that allows access from an IP address range, port, or EC2 security group. Once ingress rules are configured, the same rules apply to all DB instances that are associated with that security group. You can specify up to 20 rules in a security group.

Amazon RDS supports two different kinds of security groups. The one you use depends on which Amazon RDS platform you are on:

- **VPC security groups** – for the EC2-VPC platform.
- **DB security groups** – for the EC2-Classic platform.

You are most likely on the EC2-VPC platform (and must use VPC security groups) if any of the following are true:

- If you are a new Amazon RDS customer.
- If you have never created a DB instance before.
- If you are creating a DB instance in an AWS Region you have not used before.

Otherwise, if you are on the EC2-Classic platform, you use DB security groups to manage access to your Amazon RDS DB instances. For more information about the differences between DB security groups and VPC security groups, see Amazon RDS Security Groups (p. 375).

> **Note**
> To determine which platform you are on, see Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform (p. 391).
> If you are on the EC2-VPC platform, you must use VPC security groups instead of DB security groups. For more information about using a VPC, see Amazon Virtual Private Cloud (VPCs) and Amazon RDS (p. 390).

Topics

Creating a DB Security Group

To create a DB security group, you need to provide a name and a description.

AWS Management Console

To create a DB security group

1. Open the Amazon RDS console at https://console.aws.amazon.com/rds/.

2. Choose **Security Groups** in the navigation pane on the left side of the window.
3. Choose **Create DB Security Group**.
4. Type the name and description of the new DB security group in the **Name** and **Description** text boxes. The security group name can't contain spaces and can't start with a number.
5. Choose **Yes, Create**.

   The DB security group is created.

   A newly created DB security group doesn't provide access to a DB instance by default. You must specify a range of IP addresses or an Amazon EC2 security group that can have access to the DB instance. To specify IP addresses or an Amazon EC2 security group for a DB security group, see Authorizing Network Access to a DB Security Group from an IP Range (p. 383).

CLI

To create a DB security group, use the AWS CLI command `create-db-security-group`.

Example

For Linux, OS X, or Unix:

```
aws rds create-db-security-group \
    --db-security-group-name mydbsecuritygroup \
    --db-security-group-description "My new security group"
```

For Windows:

```
aws rds create-db-security-group ^
    --db-security-group-name mydbsecuritygroup ^
    --db-security-group-description "My new security group"
```

A newly created DB security group doesn't provide access to a DB instance by default. You must specify a range of IP addresses or an Amazon EC2 security group that can have access to the DB instance. To specify IP addresses or an Amazon EC2 security group for a DB security group, see Authorizing Network Access to a DB Security Group from an IP Range (p. 383).

API

To create a DB security group, call the Amazon RDS function `CreateDBSecurityGroup` with the following parameters:

- `DBSecurityGroupName = mydbsecuritygroup`
- `Description = "My new security group"`

Example

```
https://rds.amazonaws.com/
    ?Action=CreateDBSecurityGroup
    &DBSecurityGroupName=mydbsecuritygroup
    &Description=My%20new%20db%20security%20group
    &Version=2012-01-15
    &SignatureVersion=2
    &SignatureMethod=HmacSHA256
    &Timestamp=2012-01-20T22%3A06%3A23.624Z
    &AWSAccessKeyId=<AWS Access Key ID>
```

```
    &Signature=<Signature>
```

A newly created DB security group doesn't provide access to a DB instance by default. You must specify a range of IP addresses or an Amazon EC2 security group that can have access to the DB instance. To specify IP addresses or an Amazon EC2 security group for a DB security group, see Authorizing Network Access to a DB Security Group from an IP Range (p. 383).

Listing Available DB Security Groups

You can list which DB security groups have been created for your AWS account.

AWS Management Console

To list all available DB security groups for an AWS account

1. Open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. Choose **Security Groups** in the navigation pane on the left side of the window.

   The available DB security groups appear in the **DB Security Groups** list.

CLI

To list all available DB security groups for an AWS account, Use the AWS CLI command `describe-db-security-groups` with no parameters.

Example

```
aws rds describe-db-security-groups
```

API

To list all available DB security groups for an AWS account, call `DescribeDBSecurityGroups` with no parameters.

Example

```
https://rds.amazonaws.com/
    ?Action=DescribeDBSecurityGroups
    &MaxRecords=100
    &Version=2009-10-16
    &SignatureVersion=2
    &SignatureMethod=HmacSHA256
    &AWSAccessKeyId=<AWS Access Key ID>
    &Signature=<Signature>
```

Viewing a DB security group

You can view detailed information about your DB security group to see what IP ranges have been authorized.

AWS Management Console

To view properties of a specific DB security group

1. Open the Amazon RDS console at https://console.aws.amazon.com/rds/.

2. Choose **Security Groups** in the navigation pane on the left side of the window.
3. Select the details icon for the DB security group you want to view. The detailed information for the DB security group is displayed.

CLI

To view the properties of a specific DB security group use the AWS CLI `describe-db-security-groups`. Specify the DB security group you want to view.

Example

For Linux, OS X, or Unix:

```
aws rds describe-db-security-groups \
    --db-security-group-name mydbsecuritygroup
```

For Windows:

```
aws rds describe-db-security-groups ^
    --db-security-group-name mydbsecuritygroup
```

API

To view properties of a specific DB security group, call `DescribeDBSecurityGroups` with the following parameters:

- `DBSecurityGroupName=mydbsecuritygroup`

Example

```
https://rds.amazonaws.com/
    ?Action=DescribeDBSecurityGroups
    &DBSecurityGroupName=mydbsecuritygroup
    &Version=2009-10-16
    &SignatureVersion=2
    &SignatureMethod=HmacSHA256
    &Timestamp=2009-10-16T22%3A23%3A07.107Z
    &AWSAccessKeyId=<AWS Access Key ID>
    &Signature=<Signature>
```

Associating a DB Security Group with a DB Instance

You can associate a DB security group with a DB instance using the RDS console's **Modify** option, the `ModifyDBInstance` Amazon RDS API, or the AWS CLI `modify-db-instance` command. For information about modifying a DB instance, see Modifying an Amazon RDS DB Instance and Using the Apply Immediately Parameter (p. 114).

Authorizing Network Access to a DB Security Group from an IP Range

By default, network access is turned off to a DB instance. If you want to access a DB instance that is not in a VPC, you must set access rules for a DB security group to allow access from specific EC2 security groups or CIDR IP ranges. You then must associate that DB instance with that DB security group. This

process is called *ingress*. Once ingress is configured for a DB security group, the same ingress rules apply to all DB instances associated with that DB security group.

Warning
Talk with your network administrator if you are intending to access a DB instance behind a firewall to determine the IP addresses you should use.

In following example, you configure a DB security group with an ingress rule for a CIDR IP range.

AWS Management Console

To configure a DB security group with an ingress rule for a CIDR IP range

1. Open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. Select **Security Groups** from the navigation pane on the left side of the console window.
3. Select the details icon for the DB security group you want to authorize.
4. In the details page for your security group, select *CIDR/IP* from the **Connection Type** drop-down list, type the CIDR range for the ingress rule you want to add to this DB security group into the **CIDR** text box, and choose **Authorize**.

   Tip
   The AWS Management Console displays a CIDR IP based on your connection below the CIDR text field. If you are not accessing the DB instance from behind a firewall, you can use this CIDR IP.
5. The status of the ingress rule is **authorizing** until the new ingress rule has been applied to all DB instances that are associated with the DB security group that you modified. After the ingress rule has been successfully applied, the status changes to **authorized**.

CLI

To configure a DB security group with an ingress rule for a CIDR IP range, use the AWS CLI command `authorize-db-security-group-ingress`.

Example

For Linux, OS X, or Unix:

```
aws rds authorize-db-security-group-ingress \
    --db-security-group-name mydbsecuritygroup \
    --cidrip 192.168.1.10/27
```

For Windows:

```
aws rds authorize-db-security-group-ingress ^
    --db-security-group-name mydbsecuritygroup ^
    --cidrip 192.168.1.10/27
```

The command should produce output similar to the following.

```
SECGROUP  mydbsecuritygroup  My new DBSecurityGroup
IP-RANGE  192.168.1.10/27  authorizing
```

API

To configure a DB security group with an ingress rule for a CIDR IP range, call the Amazon RDS API `AuthorizeDBSecurityGroupIngress` with the following parameters:

- `DBSecurityGroupName = mydbsecuritygroup`
- `CIDRIP = 192.168.1.10/27`

Example

```
https://rds.amazonaws.com/
 ?Action=AuthorizeDBSecurityGroupIngress
 &CIDRIP=192.168.1.10%2F27
 &DBSecurityGroupName=mydbsecuritygroup
 &Version=2009-10-16
 &Action=AuthorizeDBSecurityGroupIngress
 &SignatureVersion=2
 &SignatureMethod=HmacSHA256
 &Timestamp=2009-10-22T17%3A10%3A50.274Z
 &AWSAccessKeyId=<AWS Access Key ID>
 &Signature=<Signature>
```

Authorizing Network Access to a DB Instance from an Amazon EC2 Instance

If you want to access your DB instance from an Amazon EC2 instance, you must first determine if your EC2 instance and DB instance are in a VPC. If you are using a default VPC, you can assign the same EC2 or VPC security group that you used for your EC2 instance when you create or modify the DB instance that the EC2 instance accesses.

If your DB instance and EC2 instance are not in a VPC, you must configure the DB instance's security group with an ingress rule that allows traffic from the Amazon EC2 instance. You do this by adding the Amazon EC2 security group for the EC2 instance to the DB security group for the DB instance. In this example, you add an ingress rule to a DB security group for an Amazon EC2 security group.

Important

- Adding an ingress rule to a DB security group for an Amazon EC2 security group only grants access to your DB instances from Amazon EC2 instances associated with that Amazon EC2 security group.
- You can't authorize an Amazon EC2 security group that is in a different AWS Region than your DB instance. You can authorize an IP range, or specify an Amazon EC2 security group in the same AWS Region that refers to IP address in another AWS Region. If you specify an IP range, we recommend that you use the private IP address of your Amazon EC2 instance, which provides a more direct network route from your Amazon EC2 instance to your Amazon RDS DB instance, and doesn't incur network charges for data sent outside of the Amazon network.

AWS Management Console

To add an EC2 security group to a DB security group

1. Open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. From the navigation pane, choose **Security Groups**.
3. Select the details icon for the DB security group you want to grant access.
4. In the details page for your security group, choose **EC2 Security Group** for **Connection Type**, and then select the Amazon EC2 security group you want to use. Then choose **Authorize**.
5. The status of the ingress rule is **authorizing** until the new ingress rule has been applied to all DB instances that are associated with the DB security group that you modified. After the ingress rule has been successfully applied, the status changes to **authorized**.

CLI

To grant access to an Amazon EC2 security group, use the AWS CLI command `authorize-db-security-group-ingress`.

Example

For Linux, OS X, or Unix:

```
aws rds authorize-db-security-group-ingress \
    --db-security-group-name default  \
    --ec2-security-group-name myec2group \
    --ec2-security-group-owner-id 987654321021
```

For Windows:

```
aws rds authorize-db-security-group-ingress ^
    --db-security-group-name default  ^
    --ec2-security-group-name myec2group ^
    --ec2-security-group-owner-id 987654321021
```

The command should produce output similar to the following:

```
SECGROUP  Name     Description
SECGROUP  default  default
      EC2-SECGROUP  myec2group  987654321021  authorizing
```

API

To authorize network access to an Amazon EC2 security group, call that Amazon RDS API function, http://docs.aws.amazon.com/AmazonRDS/latest/APIReference/API_AuthorizeDBSecurityGroupIngress.html`AuthorizeDBSecurityGroupIngress` with the following parameters:

- `EC2SecurityGroupName = myec2group`
- `EC2SecurityGroupOwnerId = 987654321021`

Example

```
https://rds.amazonaws.com/
    ?Action=AuthorizeDBSecurityGroupIngress
    &EC2SecurityGroupOwnerId=987654321021
    &EC2SecurityGroupName=myec2group
    &Version=2009-10-16
    &SignatureVersion=2
    &SignatureMethod=HmacSHA256
    &Timestamp=2009-10-22T17%3A10%3A50.274Z
    &AWSAccessKeyId=<AWS Access Key ID>
    &Signature=<Signature>
```

Revoking Network Access to a DB Instance from an IP Range

You can easily revoke network access from a CIDR IP range to DB Instances belonging to a DB security group by revoking the associated CIDR IP ingress rule.

In this example, you revoke an ingress rule for a CIDR IP on a DB Security Group.

AWS Management Console

To revoke an ingress rule for a CIDR IP range on a DB Security Group

1. Open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. From the navigation pane, choose **Security Groups**.
3. Select the details icon for the DB security group that has the ingress rule you want to revoke.
4. In the details page for your security group, choose **Remove** next to the ingress rule you want to revoke.
5. The status of the ingress rule is **revoking** until the ingress rule has been removed from all DB instances that are associated with the DB security group that you modified. After the ingress rule has been successfully removed, the ingress rule is removed from the DB security group.

CLI

To revoke an ingress rule for a CIDR IP range on a DB security group, use the AWS CLI command `revoke-db-security-group-ingress`.

Example

For Linux, OS X, or Unix:

```
aws rds revoke-db-security-group-ingress \
    --db-security-group-name mydbsecuritygroup \
    --cidrip 192.168.1.1/27
```

For Windows:

```
aws rds revoke-db-security-group-ingress ^
    --db-security-group-name mydbsecuritygroup ^
    --cidrip 192.168.1.1/27
```

The command should produce output similar to the following.

```
SECGROUP  mydbsecuritygroup  My new DBSecurityGroup
  IP-RANGE  192.168.1.1/27  revoking
```

API

To revoke an ingress rule for a CIDR IP range on a DB security group, call the Amazon RDS API action http://docs.aws.amazon.com/AmazonRDS/latest/APIReference/API_RevokeDBSecurityGroupIngress.html`RevokeDBSecurityGroupIngress` with the following parameters:

- `DBSecurityGroupName = mydbsecuritygroup`
- `CIDRIP = 192.168.1.10/27`

Example

```
https://rds.amazonaws.com/
    ?Action=RevokeDBSecurityGroupIngress
    &DBSecurityGroupName=mydbsecuritygroup
```

```
&CIDRIP=192.168.1.10%2F27
&Version=2009-10-16
&SignatureVersion=2&SignatureMethod=HmacSHA256
&Timestamp=2009-10-22T22%3A32%3A12.515Z
&AWSAccessKeyId=<AWS Access Key ID>
&Signature=<Signature>
```

Related Topics

- Amazon RDS Security Groups (p. 375)

Master User Account Privileges

When you create a new DB instance, the default master user that you use gets certain privileges for that DB instance. The following table shows the privileges the master user gets for each of the database engines.

Note
If you accidentally delete the permissions for the master user you can restore them by resetting the password for the account.

Database Engine	System Privilege	Role
MySQL and MariaDB	SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD, PROCESS, REFERENCES, INDEX, ALTER, SHOW DATABASES, CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE, REPLICATION CLIENT, CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, CREATE USER, EVENT, TRIGGER ON *.* WITH GRANT OPTION, REPLICATION SLAVE (Only For Amazon RDS MySQL versions 5.6 and 5.7, Amazon RDS MariaDB)	
Amazon Aurora MySQL	CREATE, DROP, GRANT OPTION, REFERENCES, EVENT, ALTER, DELETE, INDEX, INSERT, SELECT, UPDATE, CREATE TEMPORARY TABLES, LOCK TABLES, TRIGGER, CREATE VIEW, SHOW VIEW, LOAD FROM S3, SELECT INTO S3, ALTER ROUTINE, CREATE ROUTINE, EXECUTE, CREATE USER, PROCESS, SHOW DATABASES , RELOAD, REPLICATION CLIENT, REPLICATION SLAVE	
Amazon Aurora PostgreSQL	LOGIN, NOSUPERUSER, INHERIT, CREATEDB, CREATEROLE, NOREPLICATION, VALID UNTIL 'infinity'	RDS_SUPERUSER
PostgreSQL	CREATE ROLE, CREATE DB, PASSWORD VALID UNTIL INFINITY, CREATE EXTENSION, ALTER EXTENSION, DROP EXTENSION, CREATE TABLESPACE, ALTER < OBJECT> OWNER, CHECKPOINT, PG_CANCEL_BACKEND(), PG_TERMINATE_BACKEND(), SELECT PG_STAT_REPLICATION, EXECUTE PG_STAT_STATEMENTS_RESET(), OWN POSTGRES_FDW_HANDLER(), OWN POSTGRES_FDW_VALIDATOR(), OWN POSTGRES_FDW, EXECUTE PG_BUFFERCACHE_PAGES(), SELECT PG_BUFFERCACHE	RDS_SUPERUSER

Database Engine	System Privilege	Role
Oracle	ALTER DATABASE LINK, ALTER PUBLIC DATABASE LINK, DROP ANY DIRECTORY, EXEMPT ACCESS POLICY, EXEMPT IDENTITY POLICY, GRANT ANY OBJECT PRIVILEGE, RESTRICTED SESSION, EXEMPT REDACTION POLICY	AQ_ADMINISTRATOR_ROLE, AQ_USER_ROLE, CONNECT, CTXAPP, DBA, EXECUTE_CATALOG_ROLE, RECOVERY_CATALOG_OWNER, RESOURCE, SELECT_CATALOG_ROLE
Microsoft SQL Server	ALTER ANY CONNECTION, ALTER ANY LINKED SERVER, ALTER ANY LOGIN, ALTER SERVER STATE, ALTER TRACE, CONNECT SQL, CREATE ANY DATABASE, VIEW ANY DATABASE, VIEW ANY DEFINITION, VIEW SERVER STATE, ALTER ANY SERVER ROLE, ALTER ANY USER	DB_OWNER (Database Level Role) PROCESSADMIN(Server Level Role) SETUPADMIN(Server Level Role) SQLAgentUserRole(Server Level Role)

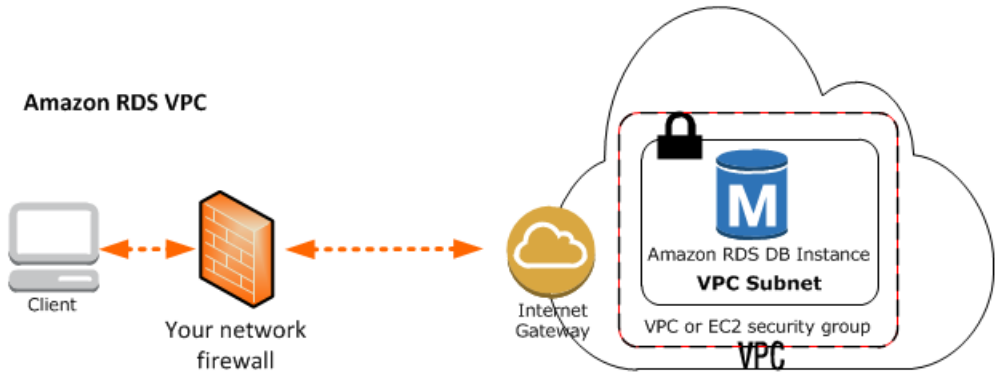
Related Topics

- Working with DB Security Groups (EC2-Classic Platform) (p. 380)

Amazon Virtual Private Cloud (VPCs) and Amazon RDS

There are two Amazon Elastic Compute Cloud (EC2) platforms that host Amazon RDS DB instances, *EC2-VPC* and *EC2-Classic*. Amazon Virtual Private Cloud (Amazon VPC) lets you launch AWS resources, such as Amazon Relational Database Service (Amazon RDS) DB instances, into a virtual private cloud (VPC).

When you use an Amazon VPC, you have control over your virtual networking environment: you can select your own IP address range, create subnets, and configure routing and access control lists. The basic functionality of Amazon RDS is the same whether your DB instance is running in an Amazon VPC or not: Amazon RDS manages backups, software patching, automatic failure detection, and recovery. There is no additional cost to run your DB instance in Amazon VPC.



Accounts that support only the *EC2-VPC* platform have a default VPC. All new DB instances are created in the default VPC unless you specify otherwise. If you are a new Amazon RDS customer, if you have never created a DB instance before, or if you are creating a DB instance in a region you have not used before, you are most likely on the *EC2-VPC* platform and have a default VPC.

Some legacy DB instances on the *EC2-Classic* platform are not in a VPC. The legacy *EC2-Classic* platform does not have a default VPC, but as is true for either platform, you can create your own VPC and specify that a DB instance be located in that VPC.

Topics

This documentation only discusses VPC functionality relevant to Amazon RDS DB instances. For more information about Amazon VPC, see Amazon VPC Getting Started Guide and Amazon VPC User Guide.

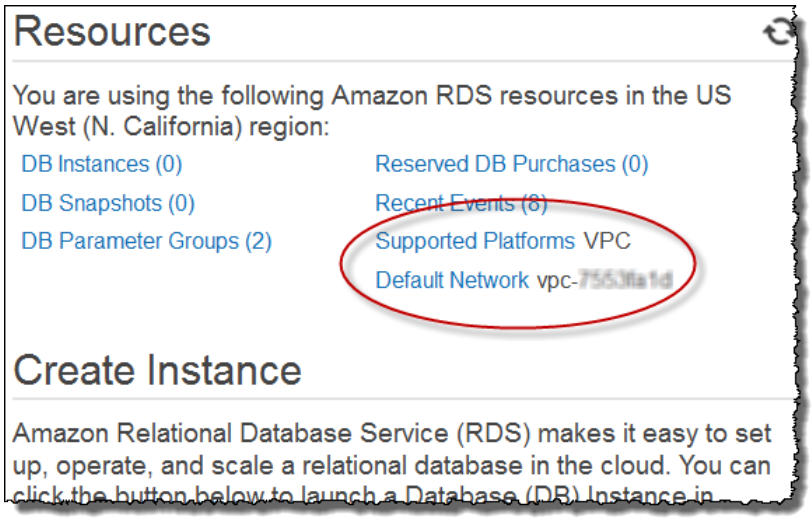
Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform

Your AWS account and the region you select determines which of the two RDS platforms your DB instance is created on: *EC2-Classic* or *EC2-VPC*. The type of platform determines if you have a default VPC, and which type of security group you use to provide access to your DB instance. The legacy *EC2-Classic* platform is the original platform used by Amazon RDS; if you are on this platform and want to use a VPC, you must create the VPC using the Amazon VPC console or Amazon VPC API. Accounts that only support the *EC2-VPC* platform have a default VPC where all DB instance are created, and you must use either an EC2 or VPC security group to provide access to the DB instance.

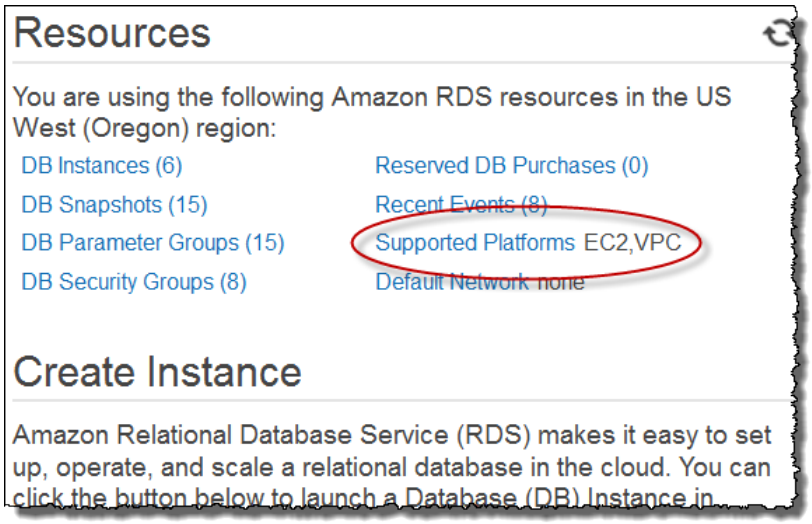
> **Note**
> If you are a new Amazon RDS customer, if you have never created a DB instance before, or if you are creating a DB instance in a region you have not used before, in almost all cases you are on the *EC2-VPC* platform and have a default VPC.

You can tell which platform your AWS account in a given region is using by looking at the RDS console or EC2 console home pages. If you are a new Amazon RDS customer, if you have never created a DB instance before, or if you are creating a DB instance in a region you have not used before, you might be redirected to the first-run console page and will not see the home page following.

If **Supported Platforms** indicates `VPC`, as shown in the screenshot following, your AWS account in the current region uses the *EC2-VPC* platform, and uses a default VPC. The name of the default VPC is shown below the supported platform. To provide access to a DB instance created on the *EC2-VPC* platform, you must create a VPC security group.



If **Supported Platforms** indicates `EC2,VPC`, as shown in the screenshot following, your AWS account in the current region uses the *EC2-Classic* platform, and you do not have a default VPC. To provide access to a DB instance created on the *EC2-Classic* platform, you must create a DB security group. Note that you can create a VPC on the *EC2-Classic* platform, but one is not created for you by default as it is on accounts that support the *EC2-VPC* platform.



Note
If you are interested in moving an existing DB instance into a VPC, you can use the AWS Management Console to do it easily. For more information. see Moving a DB Instance Not in a VPC into a VPC (p. 405).

Scenarios for Accessing a DB Instance in a VPC

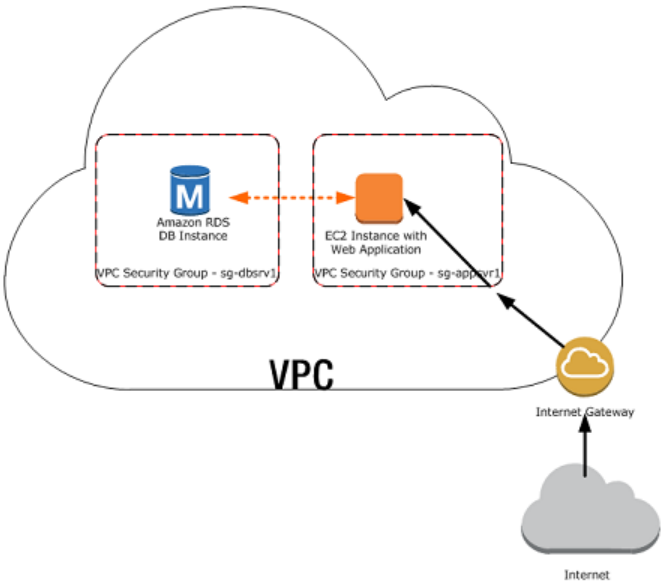
Amazon RDS supports the following scenarios for accessing a DB instance in a VPC:

DB Instance	Accessed By
In a VPC	An EC2 Instance in the Same VPC (p. 392)
	An EC2 Instance in a Different VPC (p. 394)
	An EC2 Instance Not in a VPC (p. 395)
	A Client Application Through the Internet (p. 396)
Not in a VPC	An EC2 Instance in a VPC (p. 396)
	An EC2 Instance Not in a VPC (p. 397)
	A Client Application Through the Internet (p. 398)

A DB Instance in a VPC Accessed by an EC2 Instance in the Same VPC

A common use of an RDS instance in a VPC is to share data with an application server that is running in an EC2 instance in the same VPC. This is the user scenario created if you use AWS Elastic Beanstalk to create an EC2 instance and a DB instance in the same VPC.

The following diagram shows this scenario.



The simplest way to manage access between EC2 instances and DB instances in the same VPC is to do the following:

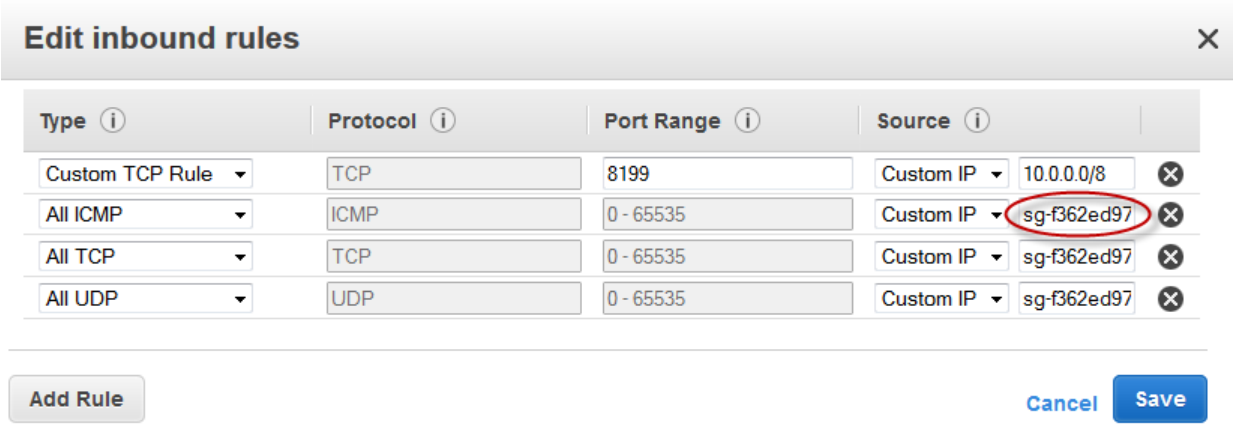
- Create a VPC security group that your DB instances will be in. This security group can be used to restrict access to the DB instances. For example, you can create a custom rule for this security group that allows TCP access using the port you assigned to the DB instance when you created it and an IP address you will use to access the DB instance for development or other purposes.
- Create a VPC security group that your EC2 instances (web servers and clients) will be in. This security group can, if needed, allow access to the EC2 instance from the Internet via the VPC's routing table. For example, you can set rules on this security group to allow TCP access to the EC2 instance over port 22.
- Create custom rules in the security group for your DB instances that allow connections from the security group you created for your EC2 instances. This would allow any member of the security group to access the DB instances.

For a tutorial that shows you how to create a VPC with both public and private subnets for this scenario, see Tutorial: Create an Amazon VPC for Use with an Amazon RDS DB Instance (p. 406).

To create a rule in a VPC security group that allows connections from another security group, do the following:

1. Sign in to the AWS Management Console and open the Amazon VPC console at https://console.aws.amazon.com/vpc.
2. In the navigation pane, choose **Security Groups**.
3. Select or create a security group that you want to allow access to members of another security group. In the scenario above, this would be the security group you will use for your DB instances. Choose **Add Rule**.
4. From **Type**, choose **All ICMP**. In the **Source** box, start typing the ID of the security group; this provides you with a list of security groups. Select the security group with members that you want to have access to the resources protected by this security group. In the scenario above, this would be the security group you will use for your EC2 instance.
5. Repeat the steps for the TCP protocol by creating a rule with **All TCP** as the **Type** and your security group in the **Source** box. If you intend to use the UDP protocol, create a rule with **All UDP** as the **Type** and your security group in the **Source** box.

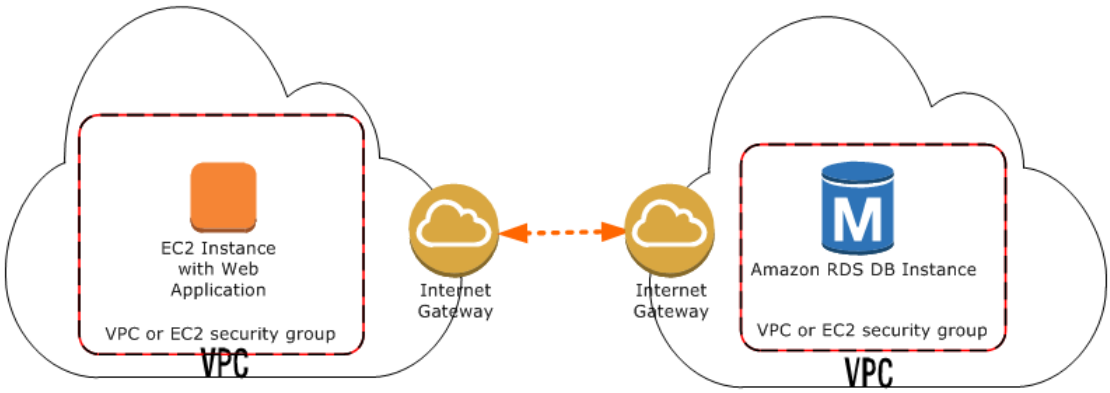
6. Create a custom TCP rule that permits access via the port you used when you created your DB instance, such as port 3306 for MySQL. Enter your security group or an IP address you will use in the **Source** box.
7. Choose **Save** when you are done.



A DB Instance in a VPC Accessed by an EC2 Instance in a Different VPC

When your DB instance is in a different VPC from the EC2 instance you are using to access it, there are several ways to access the DB instance. If the DB instance and EC2 instance are in different VPCs but in the same region, you can use VPC peering. If the DB instance and the EC2 instance are in different regions, you must use the public IP of the DB instance to access it.

The following diagram shows this scenario.



A VPC peering connection is a networking connection between two VPCs that enables you to route traffic between them using private IP addresses. Instances in either VPC can communicate with each other as if they are within the same network. You can create a VPC peering connection between your own VPCs, or with a VPC in another AWS account within a single region. To learn more about VPC peering, see the VPC documentation.

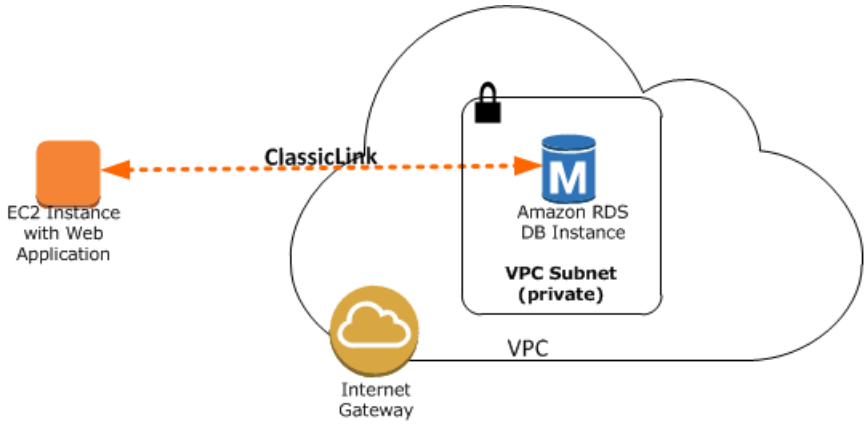
Use the public IP of the DB instance when you need to connect to a DB instance that is in a different VPC and region from your EC2 instance. The DB instance must allow public access, must be in a public subnet,

and the subnet must have an Internet gateway. Amazon RDS automatically creates a public subnet for your DB instance when you set the **VPC** option to **`Create new VPC`** and **Publicly Accessible** option to **`Yes`** when you create the DB instance.

A DB Instance in a VPC Accessed by an EC2 Instance Not in a VPC

You can communicate between an Amazon RDS DB instance that is in a VPC and an EC2 instance that is not in an Amazon VPC by using *ClassicLink*. When you use Classic Link, an application on the EC2 instance can connect to the DB instance by using the RDS endpoint for the DB instance. ClassicLink is available at no charge.

The following diagram shows this scenario.



Using ClassicLink, you can connect an EC2 instance to a logically isolated database where you define the IP address range and control the access control lists (ACLs) to manage network traffic. You don't have to use public IP addresses or tunneling to communicate with the DB instance in the VPC. This arrangement provides you with higher throughput and lower latency connectivity for inter-instance communications.

To enable ClassicLink between a DB instance in a VPC and an EC2 instance not in a VPC

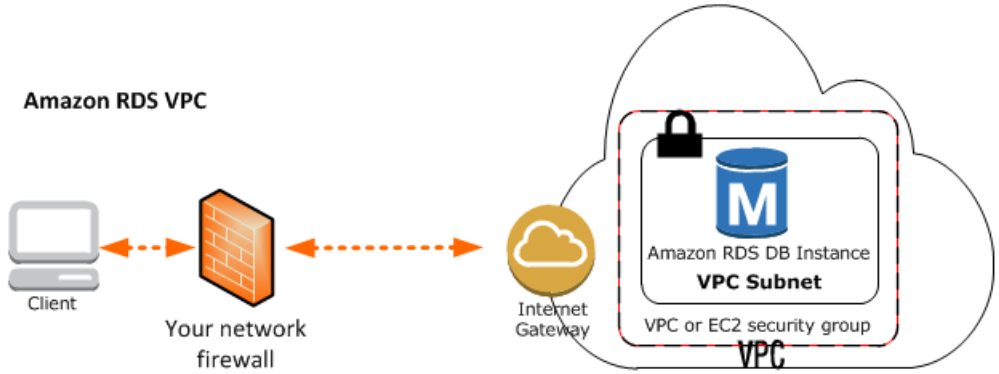
1. Sign in to the AWS Management Console and open the Amazon VPC console at https://console.aws.amazon.com/vpc.
2. In the navigation pane, choose **Your VPCs**.
3. For **VPC**, choose the VPC used by the DB instance.
4. For **Actions** menu, choose **Enable ClassicLink**. In the confirmation dialog box, choose **Yes, Enable**.
5. On the EC2 console, select the EC2 instance you want to connect to the DB instance in the VPC.
6. For **Actions** menu, choose **ClassicLink**, and then choose **Link to VPC**.
7. On the **Link to VPC** page, choose the security group you want to use, and then choose **Link to VPC**.

Note
The ClassicLink features are only visible in the consoles for accounts and regions that support EC2-Classic. For more information, see ClassicLink in the *Amazon EC2 User Guide for Linux Instances.*

A DB Instance in a VPC Accessed by a Client Application Through the Internet

To access a DB instance in a VPC from a client application through the internet, you configure a VPC with a single public subnet, and an Internet gateway to enable communication over the Internet.

The following diagram shows this scenario.



We recommend the following configuration:

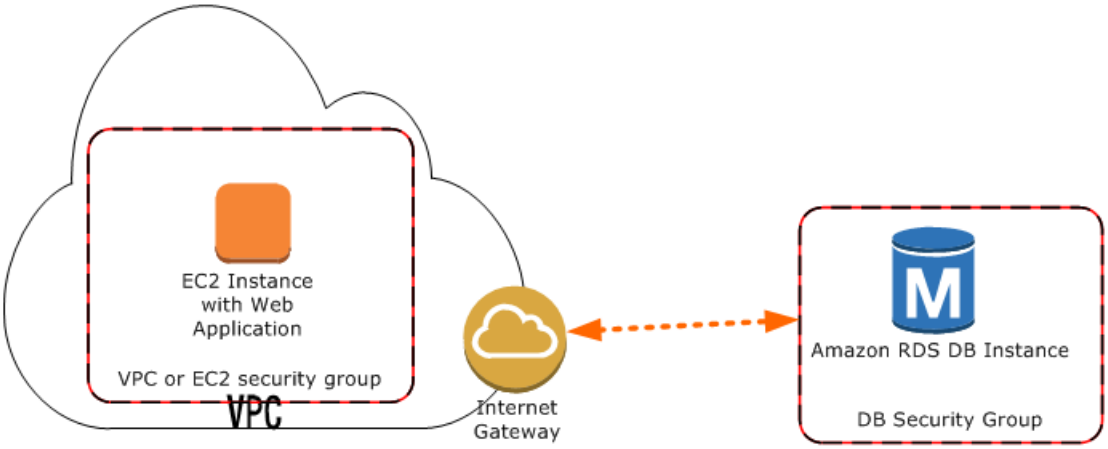
- A VPC of size /16 (for example CIDR: 10.0.0.0/16). This size provides 65,536 private IP addresses.
- A subnet of size /24 (for example CIDR: 10.0.0.0/24). This size provides 256 private IP addresses.
- An Internet gateway which connects the VPC to the Internet and to other AWS products.
- An instance with a private IP address in the subnet range (for example: 10.0.0.6), which enables the instance to communicate with other instances in the VPC, and an Elastic IP address (for example: 198.51.100.2), which enables the instance to be reached from the Internet.
- A route table entry that enables instances in the subnet to communicate with other instances in the VPC, and a route table entry that enables instances in the subnet to communicate directly over the Internet.

For more information, see scenario 1 in the VPC documentation.

A DB Instance Not in a VPC Accessed by an EC2 Instance in a VPC

In the case where you have an EC2 instance in a VPC and an RDS DB instance not in a VPC, you can connect them over the public Internet.

The following diagram shows this scenario.



Note
ClassicLink, as described in A DB Instance in a VPC Accessed by an EC2 Instance Not in a VPC (p. 395), is not available for this scenario.

To connect your DB instance and your EC2 instance over the public Internet, do the following:

- Ensure that the EC2 instance is in a public subnet in the VPC.
- Ensure that the RDS DB instance was marked as publicly accessible.
- A note about network ACLs here. A network ACL is like a firewall for your entire subnet. Therefore, all instances in that subnet are subject to network ACL rules. By default, network ACLs allow all traffic and you generally don't need to worry about them, unless you particularly want to add rules as an extra layer of security. A security group, on the other hand, is associated with individual instances, and you do need to worry about security group rules.
- Add the necessary ingress rules to the DB security group for the RDS DB instance.

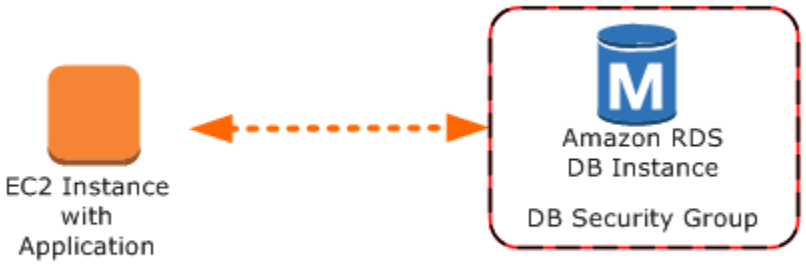
  An ingress rule specifies a network port and a CIDR/IP range. For example, you can add an ingress rule that allows port 3306 to connect to a MySQL RDS DB instance, and a CIDR/IP range of `203.0.113.25/32`. For more information, see Authorizing Network Access to a DB Security Group from an IP Range (p. 383).

  Note
  If you are interested in moving an existing DB instance into a VPC, you can use the AWS Management Console to do it easily. For more information. see Moving a DB Instance Not in a VPC into a VPC (p. 405).

A DB Instance Not in a VPC Accessed by an EC2 Instance Not in a VPC

When neither your DB instance nor an application on an EC2 instance are in a VPC, you can access the DB instance by using its endpoint and port.

The following diagram shows this scenario.



You must create a DB security group for the instance that permits access from the port you specified when creating the instance. For example, you could use a connection string similar to this connection string used with *sqlplus* to access an Oracle DB instance:

```
PROMPT>sqlplus 'mydbusr@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=<endpoint>)
    (PORT=<port number>))(CONNECT_DATA=(SID=<database name>)))'
```

For more information, see the following documentation.

Database Engine	Relevant Documentation
Amazon Aurora	Connecting to an Amazon Aurora DB Cluster (p. 457)
MariaDB	Connecting to a DB Instance Running the MariaDB Database Engine (p. 688)
Microsoft SQL Server	Connecting to a DB Instance Running the Microsoft SQL Server Database Engine (p. 749)
MySQL	Connecting to a DB Instance Running the MySQL Database Engine (p. 840)
Oracle	Connecting to a DB Instance Running the Oracle Database Engine (p. 959)
PostgreSQL	Connecting to a DB Instance Running the PostgreSQL Database Engine (p. 1179)

Note
If you are interested in moving an existing DB instance into a VPC, you can use the AWS Management Console to do it easily. For more information. see Moving a DB Instance Not in a VPC into a VPC (p. 405).

A DB Instance Not in a VPC Accessed by a Client Application Through the Internet

New Amazon RDS customers can only create a DB instance in a VPC. However, you might need to connect to an existing Amazon RDS DB instance that is not in a VPC from a client application through the Internet.

The following diagram shows this scenario.

In this scenario, you must ensure that the DB security group for the RDS DB instance includes the necessary ingress rules for your client application to connect. An ingress rule specifies a network port and a CIDR/IP range. For example, you can add an ingress rule that allows port 3306 to connect to a MySQL RDS DB instance, and a CIDR/IP range of `203.0.113.25/32`. For more information, see Authorizing Network Access to a DB Security Group from an IP Range (p. 383).

Warning
If you intend to access a DB instance behind a firewall, talk with your network administrator to determine the IP addresses you should use.

Note
If you are interested in moving an existing DB instance into a VPC, you can use the AWS Management Console to do it easily. For more information. see Moving a DB Instance Not in a VPC into a VPC (p. 405).

Working with an Amazon RDS DB Instance in a VPC

Unless you are working with a legacy DB instance, your DB instance is in a virtual private cloud (VPC). A virtual private cloud is a virtual network that is logically isolated from other virtual networks in the AWS cloud. Amazon Virtual Private Cloud (Amazon VPC) lets you launch AWS resources, such as an Amazon Relational Database Service (Amazon RDS) or Amazon Elastic Compute Cloud (Amazon EC2) instance, into a VPC. The VPC can either be a default VPC that comes with your account or one that you create. All VPCs are associated with your AWS account.

Your default VPC has three subnets you can use to isolate resources inside the VPC. The default VPC also has an Internet Gateway that can be used to provide access to resources inside the VPC from outside the VPC.

For a list of scenarios involving Amazon RDS DB instances in a VPC and outside of a VPC, see Scenarios for Accessing a DB Instance in a VPC (p. 392).

For a tutorial that shows you how to create a VPC that you can use with a common Amazon RDS scenario, see Tutorial: Create an Amazon VPC for Use with an Amazon RDS DB Instance (p. 406).

To learn how to work with an Amazon RDS DB instances inside a VPC, see the following:

Topics

- Updating the VPC for a DB Instance (p. 404)
- Moving a DB Instance Not in a VPC into a VPC (p. 405)

Working with a DB Instance in a VPC

Here are some tips on working with a DB instance in a VPC:

- Your VPC must have at least one subnet in at least two of the Availability Zones in the region where you want to deploy your DB instance. A subnet is a segment of a VPC's IP address range that you can specify and that lets you group instances based on your security and operational needs.
- If you want your DB instance in the VPC to be publicly accessible, you must enable the VPC attributes *DNS hostnames* and *DNS resolution*.
- Your VPC must have a DB subnet group that you create (for more information, see the next section). You create a DB subnet group by specifying the subnets you created. Amazon RDS uses that DB subnet group and your preferred Availability Zone to select a subnet and an IP address within that subnet to assign to your DB instance.
- Your VPC must have a VPC security group that allows access to the DB instance.
- The CIDR blocks in each of your subnets must be large enough to accommodate spare IP addresses for Amazon RDS to use during maintenance activities, including failover and compute scaling.
- A VPC can have an *instance tenancy* attribute of either *default* or *dedicated*. All default VPCs have the instance tenancy attribute set to default, and a default VPC can support any DB instance class.

  If you choose to have your DB instance in a dedicated VPC where the instance tenancy attribute is set to dedicated, the DB instance class of your DB instance must be one of the approved Amazon EC2 dedicated instance types. For example, the m3.medium EC2 dedicated instance corresponds to the db.m3.medium DB instance class. For information about instance tenancy in a VPC, go to Using EC2 Dedicated Instances in the *Amazon Virtual Private Cloud User Guide*.

  For more information about the instance types that can be in a dedicated instance, see Amazon EC2 Dedicated Instances on the EC2 pricing page.
- When an option group is assigned to a DB instance, it is linked to the supported platform the DB instance is on, either VPC or EC2-Classic (non-VPC). Furthermore, if a DB instance is in a VPC, the option group associated with the instance is linked to that VPC. This linkage means that you cannot use the option group assigned to a DB instance if you attempt to restore the instance into a different VPC or onto a different platform.
- If you restore a DB instance into a different VPC or onto a different platform, you must either assign the default option group to the instance, assign an option group that is linked to that VPC or platform, or create a new option group and assign it to the DB instance. Note that with persistent or permanent options, such as Oracle TDE, you must create a new option group that includes the persistent or permanent option when restoring a DB instance into a different VPC.

Working with DB Subnet Groups

Subnets are segments of a VPC's IP address range that you designate to group your resources based on security and operational needs. A DB subnet group is a collection of subnets (typically private) that you create in a VPC and that you then designate for your DB instances. A DB subnet group allows you to specify a particular VPC when creating DB instances using the CLI or API; if you use the console, you can just select the VPC and subnets you want to use.

Each DB subnet group should have subnets in at least two Availability Zones in a given region. When creating a DB instance in VPC, you must select a DB subnet group. Amazon RDS uses that DB subnet group and your preferred Availability Zone to select a subnet and an IP address within that subnet to

associate with your DB instance. If the primary DB instance of a Multi-AZ deployment fails, Amazon RDS can promote the corresponding standby and subsequently create a new standby using an IP address of the subnet in one of the other Availability Zones.

When Amazon RDS creates a DB instance in a VPC, it assigns a network interface to your DB instance by using an IP address selected from your DB subnet group. However, we strongly recommend that you use the DNS name to connect to your DB instance because the underlying IP address can change during failover.

Note
For each DB instance that you run in a VPC, you should reserve at least one address in each subnet in the DB subnet group for use by Amazon RDS for recovery actions.

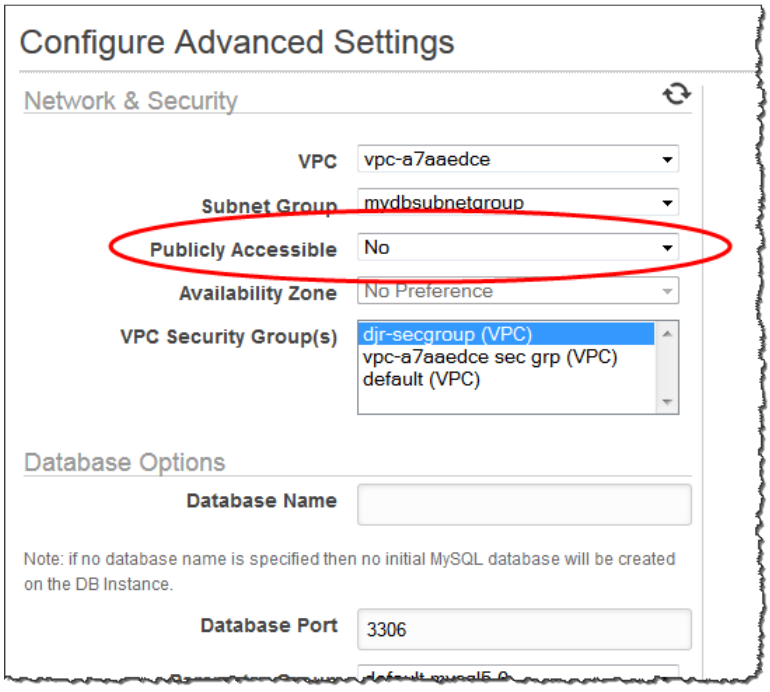
Hiding a DB Instance in a VPC from the Internet

One common Amazon RDS scenario is to have a VPC in which you have an EC2 instance with a public-facing web application and a DB instance with a database that is not publicly accessible. For example, you can create a VPC that has a public subnet and a private subnet. Amazon EC2 instances that function as web servers can be deployed in the public subnet, and the Amazon RDS DB instances are deployed in the private subnet. In such a deployment, only the web servers have access to the DB instances. For an illustration of this scenario, see A DB Instance in a VPC Accessed by an EC2 Instance in the Same VPC (p. 392).

When you launch a DB instance inside a VPC, you can designate whether the DB instance you create has a DNS that resolves to a public IP address by using the *PubliclyAccessible* parameter. This parameter lets you designate whether there is public access to the DB instance. Note that access to the DB instance is ultimately controlled by the security group it uses, and that public access is not permitted if the security group assigned to the DB instance does not permit it.

You can modify a DB instance to turn on or off public accessibility by modifying the *PubliclyAccessible* parameter. This parameter is modified just like any other DB instance parameter. For more information, see the modifying section for your DB engine.

The following illustration shows the **Publicly Accessible** option in the **Launch DB Instance Wizard**.



Creating a DB Instance in a VPC

The following procedures help you create a DB instance in a VPC. If your account has a default VPC, you can begin with step 3 because the VPC and DB subnet group have already been created for you. If your AWS account doesn't have a default VPC, or if you want to create an additional VPC, you can create a new VPC. If you don't know if you have a default VPC, see Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform (p. 391).

> **Note**
> If you want your DB instance in the VPC to be publicly accessible, you must update the DNS information for the VPC by enabling the VPC attributes *DNS hostnames* and *DNS resolution*. For information about updating the DNS information for a VPC instance, see Updating DNS Support for Your VPC.

Follow these steps to create a DB instance in a VPC:

Step 1: Create a VPC

If your AWS account does not have a default VPC or if you want to create an additional VPC, follow the instructions for creating a new VPC. See Create a VPC with Private and Public Subnets (p. 406) in the Amazon RDS documentation, or see Step 1: Create a VPC in the Amazon VPC documentation.

Step 2: Add Subnets to the VPC

Once you have created a VPC, you need to create subnets in at least two Availability Zones. You use these subnets when you create a DB subnet group. Note that if you have a default VPC, a subnet is automatically created for you in each Availability Zone in the region.

For instructions on how to create subnets in a VPC, see Create a VPC with Private and Public Subnets (p. 406) in the Amazon RDS documentation.

Step 3: Create a DB Subnet Group

A DB subnet group is a collection of subnets (typically private) that you create for a VPC and that you then designate for your DB instances. A DB subnet group allows you to specify a particular VPC when you create DB instances using the CLI or API. If you use the Amazon RDS console, you can just select the VPC and subnets you want to use. Each DB subnet group must have at least one subnet in at least two Availability Zones in the region.

> **Note**
> For a DB instance to be publicly accessible, the subnets in the DB subnet group must have an Internet gateway. For more information about Internet gateways for subnets, go to Internet Gateways in the Amazon VPC documentation.

When you create a DB instance in a VPC, you must select a DB subnet group. Amazon RDS then uses that DB subnet group and your preferred Availability Zone to select a subnet and an IP address within that subnet. Amazon RDS creates and associates an Elastic Network Interface to your DB instance with that IP address. For Multi-AZ deployments, defining a subnet for two or more Availability Zones in a region allows Amazon RDS to create a new standby in another Availability Zone should the need arise.

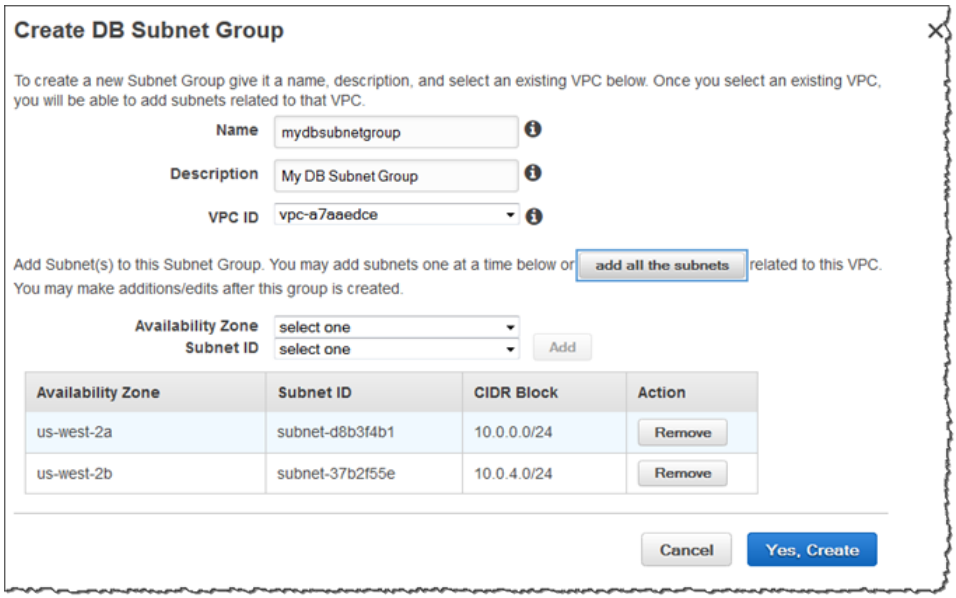
You need to do this even for Single-AZ deployments, just in case you want to convert them to Multi-AZ deployments at some point.

In this step, you create a DB subnet group and add the subnets you created for your VPC.

AWS Management Console

To create a DB subnet group

1. Open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Subnet Groups**.
3. Choose **Create DB Subnet Group**.
4. For **Name**, type the name of your DB subnet group.
5. For **Description**, type a description for your DB subnet group.
6. For **VPC ID**, choose the VPC that you created.
7. In the **Add Subnet(s) to this Subnet Group** section, click the **add all the subnets** link.



8. Choose **Yes, Create**, and then choose **Close**.

   Your new DB subnet group appears in the DB subnet groups list on the RDS console. You can click the DB subnet group to see details, including all of the subnets associated with the group, in the details pane at the bottom of the window.

Step 4: Create a VPC Security Group

Before you create your DB instance, you must create a VPC security group to associate with your DB instance. For instructions on how to create a security group for your DB instance, see Create a VPC Security Group for a Private Amazon RDS DB Instance (p. 409) in the Amazon RDS documentation, or see Security Groups for Your VPC in the Amazon VPC documentation.

Step 5: Create a DB Instance in the VPC

In this step, you create a DB instance and use the VPC name, the DB subnet group, and the VPC security group you created in the previous steps.

Note
If you want your DB instance in the VPC to be publicly accessible, you must enable the VPC attributes *DNS hostnames* and *DNS resolution*. For information on updating the DNS information for a VPC instance, see Updating DNS Support for Your VPC.

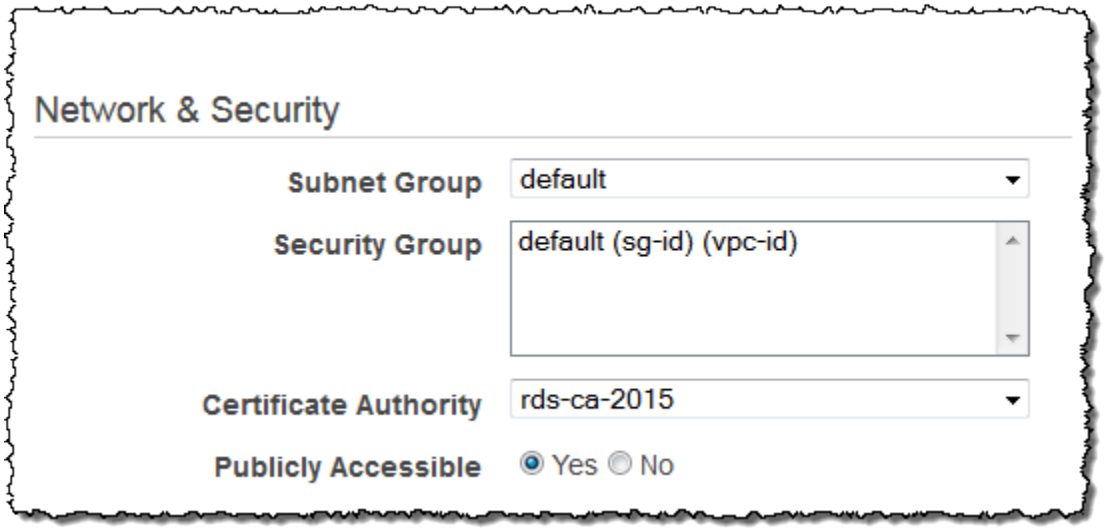
For details on how to create a DB instance for your DB engine, see the topic following that discusses your DB engine. For each engine, when prompted in the **Launch DB Instance Wizard**, enter the VPC name, the DB subnet group, and the VPC security group you created in the previous steps.

Database Engine	Relevant Documentation
Amazon Aurora	Creating an Amazon Aurora DB Cluster (p. 437)
MariaDB	Creating a DB Instance Running the MariaDB Database Engine (p. 678)
Microsoft SQL Server	Creating a DB Instance Running the Microsoft SQL Server Database Engine (p. 738)
MySQL	Creating a DB Instance Running the MySQL Database Engine (p. 830)
Oracle	Creating a DB Instance Running the Oracle Database Engine (p. 949)
PostgreSQL	Creating a DB Instance Running the PostgreSQL Database Engine (p. 1172)

Updating the VPC for a DB Instance

You can use the AWS Management Console to easily move your DB instance to a different VPC.

For details on how to modify a DB instance for your DB engine, see the topic in the table following that discusses your DB engine. In the **Network & Security** section of the modify page, shown following, for **Subnet Group**, enter the new subnet group. The new subnet group must be a subnet group in a new VPC.



Database Engine	Relevant Documentation
MariaDB	Modifying a DB Instance Running the MariaDB Database Engine (p. 691)

Database Engine	Relevant Documentation
Microsoft SQL Server	Modifying a DB Instance Running the Microsoft SQL Server Database Engine (p. 756)
MySQL	Modifying a DB Instance Running the MySQL Database Engine (p. 843)
Oracle	Modifying a DB Instance Running the Oracle Database Engine (p. 967)
PostgreSQL	Modifying a DB Instance Running the PostgreSQL Database Engine (p. 1183)

Note
Updating VPCs is not currently supported for Aurora clusters.

Moving a DB Instance Not in a VPC into a VPC

Some legacy DB instances on the EC2-Classic platform are not in a VPC. If your DB instance is not in a VPC, you can use the AWS Management Console to easily move your DB instance into a VPC. Before you can move a DB instance not in a VPC, into a VPC, you must create the VPC.

Follow these steps to create a VPC for your DB instance.

- Step 1: Create a VPC (p. 402)
- Step 2: Add Subnets to the VPC (p. 402)
- Step 3: Create a DB Subnet Group (p. 402)
- Step 4: Create a VPC Security Group (p. 403)

After you create the VPC, follow these steps to move your DB instance into the VPC.

- Updating the VPC for a DB Instance (p. 404)

The following are some limitations to moving your DB instance into the VPC.

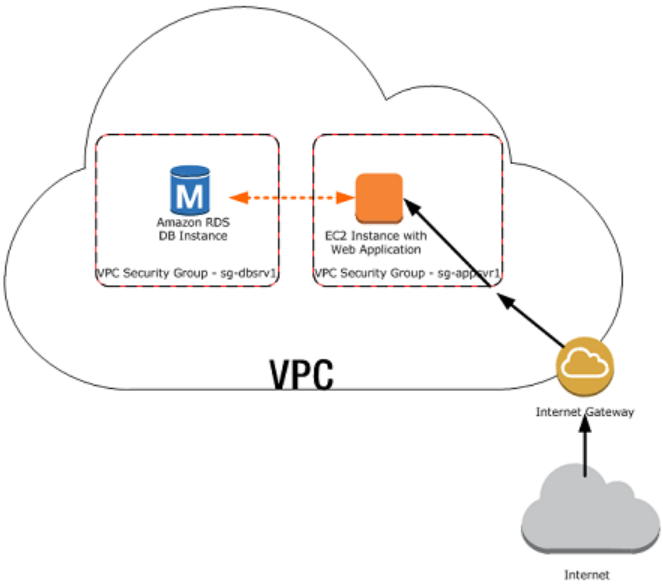
- Moving a Multi-AZ DB instance not in a VPC into a VPC is not currently supported.
- Moving a DB instance with Read Replicas not in a VPC into a VPC is not currently supported.

If you move your DB instance into a VPC, and you are using a custom option group with your DB instance, then you need to change the option group that is associated with your DB instance. Option groups are platform-specific, and moving to a VPC is a change in platform. To use a custom option group in this case, assign the default VPC option group to the DB instance, assign an option group that is used by other DB instances in the VPC you are moving to, or create a new option group and assign it to the DB instance. For more information, see Working with Option Groups (p. 153).

Tutorial: Create an Amazon VPC for Use with an Amazon RDS DB Instance

A common scenario includes an Amazon RDS DB instance in an Amazon VPC, that shares data with a Web server that is running in the same VPC. In this tutorial you create the VPC for this scenario.

The following diagram shows this scenario. For information about other scenarios, see Scenarios for Accessing a DB Instance in a VPC (p. 392).



Because your Amazon RDS DB instance only needs to be available to your web server, and not to the public Internet, you create a VPC with both public and private subnets. The web server is hosted in the public subnet, so that it can reach the public Internet. The Amazon RDS DB instance is hosted in a private subnet. The web server is able to connect to the Amazon RDS DB instance because it is hosted within the same VPC, but the Amazon RDS DB instance is not available to the public Internet, providing greater security.

Create a VPC with Private and Public Subnets

Use the following procedure to create a VPC with both public and private subnets.

To create a VPC and subnets

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the top-right corner of the AWS Management Console, choose the region to create your VPC in. This example uses the US West (Oregon) region.
3. In the upper-left corner, choose **VPC Dashboard**. To begin creating a VPC, choose **Start VPC Wizard**.
4. On the **Step 1: Select a VPC Configuration** page, choose **VPC with Public and Private Subnets**, and then choose **Select**.
5. On the **Step 2: VPC with Public and Private Subnets** page, set these values:
   - **IPv4 CIDR block:** `10.0.0.0/16`
   - **IPv6 CIDR block:** No IPv6 CIDR Block
   - **VPC name:** `tutorial-vpc`

- **Public subnet's IPv4 CIDR:** `10.0.0.0/24`
- **Availability Zone:** `us-west-2a`
- **Public subnet name:** `Tutorial public`
- **Private subnet's IPv4 CIDR:** `10.0.1.0/24`
- **Availability Zone:** `us-west-2a`
- **Private subnet name:** `Tutorial Private 1`
- **Instance type:** `t2.small`

  Important
  If you do not see the **Instance type** box in the console, click **Use a NAT instance instead**. This link is on the right.

  Note
  If the t2.small instance type is not listed, you can select a different instance type.
- **Key pair name:** `No key pair`
- **Service endpoints:** Skip this field.
- **Enable DNS hostnames:** `Yes`
- **Hardware tenancy:** `Default`

6. When you're finished, choose **Create VPC**.

Create Additional Subnets

You must have either two private subnets or two public subnets available to create an Amazon RDS DB subnet group for an RDS DB instance to use in a VPC. Because the RDS DB instance for this tutorial is private, add a second private subnet to the VPC.

To create an additional subnet

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. To add the second private subnet to your VPC, choose **VPC Dashboard**, choose **Subnets**, and then choose **Create Subnet**.
3. On the **Create Subnet** page, set these values:

   - **Name tag:** `Tutorial private 2`
   - **VPC:** Choose the VPC that you created in the previous step, for example: `vpc-f1b76594 (10.0.0.0/16) | tutorial-vpc`
   - **Availability Zone:** `us-west-2b`

     Note
     Choose an Availability Zone different from the one that you chose for the first private subnet.
   - **IPv4 CIDR block:** `10.0.2.0/24`
4. When you're finished, choose **Yes, Create**.
5. To ensure that the second private subnet that you created uses the same route table as the first private subnet, choose **VPC Dashboard**, choose **Subnets**, and then choose the first private subnet that you created for the VPC, `Tutorial private 1`.
6. Below the list of subnets, choose the **Route Table** tab, and note the value for **Route Table**—for example: `rtb-98b613fd`.
7. In the list of subnets, choose the second private subnet `Tutorial private 2`, and choose the **Route Table** tab.
8. If the current route table is not the same as the route table for the first private subnet, choose **Edit**. For **Change to**, choose the route table that you noted earlier—for example: `rtb-98b613fd`.

9. To save your selection, choose **Save**.

Create a VPC Security Group for a Public Web Server

Next you create a security group for public access. To connect to public instances in your VPC, you add inbound rules to your VPC security group that allow traffic to connect from the internet.

To create a VPC security group

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. Choose **VPC Dashboard**, choose **Security Groups**, and then choose **Create Security Group**.
3. On the **Create Security Group** page, set these values:
   - **Name tag:** `tutorial-securitygroup`
   - **Group name:** `tutorial-securitygroup`
   - **Description:** `Tutorial Security Group`
   - **VPC:** Choose the VPC that you created earlier, for example: `vpc-f1b76594 (10.0.0.0/16) | tutorial-vpc`
4. To create the security group, choose **Yes, Create**.

To add inbound rules to the security group

1. Determine the IP address that you will use to connect to instances in your VPC. To determine your public IP address, you can use the service at http://checkip.amazonaws.com. If you are connecting through an Internet service provider (ISP) or from behind your firewall without a static IP address, you need to find out the range of IP addresses used by client computers.

   Warning
   If you use `0.0.0.0/0`, you enable all IP addresses to access your public instances. This approach is acceptable for a short time in a test environment, but it's unsafe for production environments. In production, you'll authorize only a specific IP address or range of addresses to access your instances.
2. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
3. Choose **VPC Dashboard**, choose **Security Groups**, and then choose the `tutorial-securitygroup` security group that you created in the previous procedure.
4. Choose the **Inbound Rules** tab, and then choose **Edit**.
5. Set the following values for your new inbound rule to allow Secure Shell (SSH) access to your EC2 instance. If you do this, you can connect to your EC2 instance to install the web server and other utilities, and to upload content for your web server.
   - **Type:** `SSH (22)`
   - **Source:** The IP address or range from the prior step, for example: `203.0.113.25/32`.
6. Choose **Add another rule**.
7. Set the following values for your new inbound rule to allow HTTP access to your web server.
   - **Type:** `HTTP (80)`
   - **Source:** `0.0.0.0/0`.
8. To save your settings, choose **Save**.

Create a VPC Security Group for a Private Amazon RDS DB Instance

To keep your Amazon RDS DB instance private, create a second security group for private access. To connect to private instances in your VPC, you add inbound rules to your VPC security group that allow traffic from your web server only.

To create a VPC security group

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. Choose **VPC Dashboard**, choose **Security Groups**, and then choose **Create Security Group**.
3. On the **Create Security Group** page, set these values:
   - **Name tag:** `tutorial-db-securitygroup`
   - **Group name:** `tutorial-db-securitygroup`
   - **Description:** `Tutorial DB Instance Security Group`
   - **VPC:** Choose the VPC that you created earlier, for example: `vpc-f1b76594 (10.0.0.0/16) | tutorial-vpc`
4. To create the security group, choose **Yes, Create**.

To add inbound rules to the security group

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. Choose **VPC Dashboard**, choose **Security Groups**, and then choose the `tutorial-db-securitygroup` security group that you created in the previous procedure.
3. Choose the **Inbound Rules** tab, and then choose **Edit**.
4. Set the following values for your new inbound rule to allow MySQL traffic on port 3306 from your EC2 instance. If you do this, you can connect from your web server to your DB instance to store and retrieve data from your web application to your database.
   - **Type:** `MySQL/Aurora (3306)`
   - **Source:** The identifier of the `tutorial-securitygroup` security group that you created previously in this tutorial, for example: `sg-9edd5cfb`.
5. To save your settings, choose **Save**.

Storage for Amazon RDS

Most of Amazon RDS use Amazon Elastic Block Store (Amazon EBS) volumes for database and log storage. The exception is Amazon Aurora, which uses our proprietary storage system. Depending on the amount of storage requested, Amazon RDS automatically stripes across multiple Amazon EBS volumes to enhance IOPS performance. Amazon RDS provides three types of storage with a range of storage and performance options, as described following.

For more information about pricing, see Amazon RDS Pricing.

Amazon RDS Storage Types

Amazon RDS provides three storage types: General Purpose (SSD), Provisioned IOPS (input/output operations per second), and magnetic. They differ in performance characteristics and price, which means that you can tailor your storage performance and cost to the needs of your database workload. When using the Provisioned IOPS and General Purpose (SSD) storage types, you can create MySQL, MariaDB, Microsoft SQL Server, PostgreSQL, and Oracle RDS DB instances with up to 16 TiB of storage.

- **General Purpose (SSD)** – General Purpose (SSD), also called gp2, volumes offer cost-effective storage that is ideal for a broad range of workloads. These volumes deliver single-digit millisecond latencies and the ability to burst to 3,000 IOPS for extended periods of time. Between a minimum of 100 IOPS (at 33.33 GiB and below) and a maximum of 10,000 IOPS (at 3,334 GiB and above), baseline performance scales linearly at 3 IOPS per GiB of volume size. This storage type is excellent for small to medium-sized databases.

  For more information about general purpose (SSD) storage, including the storage size ranges, see General Purpose (SSD) Storage (p. 411).
- **Provisioned IOPS** – Provisioned IOPS storage is designed to meet the needs of I/O-intensive workloads, particularly database workloads, that are sensitive to storage performance and consistency in random access I/O throughput. You specify the amount of storage you want allocated, and then specify the amount of dedicated IOPS you want. Amazon RDS delivers within 10 percent of the provisioned IOPS performance 99.9 percent of the time over a given year.

  For more information about provisioned IOPS storage, including the storage size ranges, see Provisioned IOPS Storage (p. 413).
- **Magnetic** – Amazon RDS also supports magnetic storage for backward compatibility. We recommend that you use General Purpose (SSD) or Provisioned IOPS for any new storage needs. The maximum amount of storage allowed for DB instances on magnetic storage is less than that of the other storage types.

Several factors can affect the performance of Amazon EBS volumes, such as instance configuration, I/O characteristics, and workload demand. For more information about getting the most out of your Provisioned IOPS volumes, see Amazon EBS Volume Performance.

For existing DB instances, you might observe some I/O capacity improvement if you scale up your storage.

Performance Metrics

Amazon RDS provides several metrics that you can use to determine how your DB instance is performing. You can view the metrics in the Amazon RDS Management Console by choosing your DB instance and

then choosing **Show Monitoring**. You can also use Amazon CloudWatch to monitor these metrics. For more information, see Viewing DB Instance Metrics (p. 254). Enhanced Monitoring provides more detailed I/O metrics; for more information, see Enhanced Monitoring (p. 258).

- **IOPS** – The number of I/O operations completed per second. This metric is reported as the average IOPS for a given time interval. Amazon RDS reports read and write IOPS separately on 1-minute intervals. Total IOPS is the sum of the read and write IOPS. Typical values for IOPS range from zero to tens of thousands per second.
- **Latency** – The elapsed time between the submission of an I/O request and its completion. This metric is reported as the average latency for a given time interval. Amazon RDS reports read and write latency separately on 1-minute intervals in units of seconds. Typical values for latency are in the millisecond (ms). For example, Amazon RDS reports 2 ms as 0.002 seconds.
- **Throughput** – The number of bytes per second transferred to or from disk. This metric is reported as the average throughput for a given time interval. Amazon RDS reports read and write throughput separately on 1-minute intervals using units of megabytes per second (MB/s). Typical values for throughput range from zero to the I/O channel's maximum bandwidth.
- **Queue Depth** – The number of I/O requests in the queue waiting to be serviced. These are I/O requests that have been submitted by the application but have not been sent to the device because the device is busy servicing other I/O requests. Time spent waiting in the queue is a component of latency and service time (not available as a metric). This metric is reported as the average queue depth for a given time interval. Amazon RDS reports queue depth in 1-minute intervals. Typical values for queue depth range from zero to several hundred.

General Purpose (SSD) Storage

General purpose (SSD) storage offers cost-effective storage that is ideal for small or medium-sized database workloads. The following are the storage size ranges for General purpose (SSD) DB instances:

- MySQL, MariaDB, and PostgreSQL DB instances: 5 GiB–16 TiB
- Oracle DB instances: 10 GiB–16 TiB
- SQL Server Enterprise and Standard editions: 200 GiB–16 TiB
- SQL Server Web and Express editions: 20 GiB–16 TiB

General purpose (SSD) storage can deliver single-digit millisecond latencies and the ability to burst to 3,000 IOPS for extended periods of time. Between a minimum of 100 IOPS (at 33.33 GiB and below) and a maximum of 10,000 IOPS (at 3,334 GiB and above), baseline performance scales linearly at 3 IOPS per GiB of volume size.

Some workloads can exhaust the 3000 IOPS burst storage, so you need to plan accordingly.

I/O Credits and Burst Performance

General Purpose (SSD) storage performance is governed by volume size, which dictates the base performance level of the volume and how quickly it accumulates I/O credits. Larger volumes have higher base performance levels and accumulate I/O credits faster. *I/O credits* represent the available bandwidth that your General Purpose (SSD) storage can use to burst large amounts of I/O when more than the base level of performance is needed. The more I/O credits your storage has for I/O, the more time it can burst beyond its base performance level and the better it performs when more performance is needed.

When using General Purpose (SSD) storage, your DB instance receives an initial I/O credit balance of 5.4 million I/O credits, which is enough to sustain a burst performance of 3,000 IOPS for 30 minutes. This initial I/O credit balance is designed to provide a fast initial boot cycle for boot volumes and to

provide a good bootstrapping experience for other applications. Volumes earn I/O credits at the baseline performance rate of 3 IOPS per GiB of volume size. For example, a 100 GiB SSD volume has a baseline performance of 300 IOPS.

When your storage requires more than the base performance I/O level, it uses I/O credits in the I/O credit balance to burst to the required performance level, up to a maximum of 3,000 IOPS. Storage larger than 1,000 GiB has a base performance that is equal or greater than the maximum burst performance, so its I/O credit balance never depletes and it can burst indefinitely. When your storage uses fewer I/O credits than it earns in a second, unused I/O credits are added to the I/O credit balance. The maximum I/O credit balance for a DB instance using General Purpose (SSD) storage is equal to the initial I/O credit balance (5.4 million I/O credits).

If your storage uses all of its I/O credit balance, its maximum performance remains at the base performance level until I/O demand drops below the base level and unused I/O credits are added to the I/O credit balance. (The *base performance level* is the rate at which your storage earns I/O credits.) The more storage, the greater the base performance is and the faster it replenishes the I/O credit balance.

Note
Storage conversions between magnetic storage and General Purpose (SSD) storage can potentially deplete the initial 5.4 million I/O credits (3,000 IOPS X 30 Minutes) allocated for General Purpose (SSD) storage. When performing these storage conversions, the first 82 GiB of data is converted at approx. 3,000 IOPS. The remaining data is converted at the base performance rate of 100 IOPS per GiB of allocated General Purpose (SSD) storage. This approach can result in longer conversion times. You can provision more General Purpose (SSD) storage to increase your base I/O performance rate, thus improving the conversion time, but note that you cannot reduce storage size once it has been allocated.

The following table lists several storage sizes. For each, it lists the associated base performance of the storage, which is also the rate at which it accumulates I/O credits. The table also lists the burst duration at the 3,000 IOPS maximum, when starting with a full I/O credit balance. In addition, the table lists the time in seconds that the storage takes to refill an empty I/O credit balance.

Storage size (GiB)	Base Performance (IOPS)	Maximum Burst Duration @ 3,000 IOPS (seconds)	Seconds to Fill Empty I/O Credit Balance
1	100	1,862	54,000
100	300	2,000	18,000
250	750	2,400	7,200
500	1,500	3,600	3,600
750	2,250	7,200	2,400
1,000	3,000	Infinite	N/A

The burst duration of your storage depends on the size of the storage, the burst IOPS required, and the I/O credit balance when the burst begins. This relationship is shown in the equation following.

$$\text{Burst duration} = \frac{\text{(Credit balance)}}{\text{(Burst IOPS)} - 3\text{(Storage size in GiB)}}$$

If you notice that your storage performance is frequently limited to the base level due to an empty I/O credit balance, consider allocating more General Purpose (SSD) storage with a higher base performance

level. Alternatively, you can switch to Provisioned IOPS storage for workloads that require sustained IOPS performance.

For workloads with steady state I/O requirements, provisioning less than 100 GiB of General Purpose (SSD) storage might result in higher latencies if you exhaust your I/O credit balance.

Note
In general, most workloads never exceed the I/O credit balance.

Provisioned IOPS Storage

For any production application that requires fast and consistent I/O performance, we recommend Provisioned IOPS (input/output operations per second) storage. Provisioned IOPS storage is a storage type that delivers fast, predictable, and consistent throughput performance. Provisioned IOPS storage is optimized for online transaction processing (OLTP) workloads that have consistent performance requirements. Provisioned IOPS helps performance tuning.

When you create a DB instance, you specify an IOPS rate and storage space allocation. Amazon RDS provisions that IOPS rate and storage for the lifetime of the DB instance or until you change it.

Note
Your actual realized IOPS might vary from the value that you specify depending on your database workload, DB instance size, and the page size and channel bandwidth that are available for your DB engine. For more information, see Factors That Affect Realized IOPS Rates (p. 417).

The following table shows the IOPS and storage range for each database engine.

	Range of Provisioned IOPS	Range of Storage	Range of IOPS to Storage (GiB) Ratio
MariaDB	1,000–40,000 IOPS	100 GiB–16 TiB	1:1–50:1
Microsoft SQL Server, Enterprise and Standard editions	1000–20,000 IOPS	200 GiB–16 TiB	1:1–50:1
Microsoft SQL Server, Web and Express editions	1000–20,000 IOPS	100 GiB–16 TiB	1:1–50:1
MySQL	1,000–40,000 IOPS	100 GiB–16 TiB	1:1–50:1
Oracle	1,000–40,000 IOPS	100 GiB–16 TiB	1:1–50:1
PostgreSQL	1,000–40,000 IOPS	100 GiB–16 TiB	1:1–50:1

The ratio of the requested IOPS rate to the amount of storage allocated is important, and depends on your database engine. For example, for Oracle, the ratio should be between 1:1 and 50:1. You can start by provisioning an Oracle DB instance with 1,000 IOPS and 200 GiB storage (a ratio of 5:1). You can then scale up to 2,000 IOPS with 200 GiB of storage (a ratio of 10:1). You can scale up to 30,000 IOPS with 6 TiB (6144 GiB) of storage (a ratio of 5:1), and you can scale up further if necessary.

You can modify an existing Oracle, MySQL, or MariaDB DB instance to use Provisioned IOPS storage. You can also modify Provisioned IOPS storage settings.

Using Provisioned IOPS Storage with Multi-AZ, Read Replicas, Snapshots, VPC, and DB Instance Classes

For production OLTP use cases, we recommend that you use Multi-AZ deployments for enhanced fault tolerance and Provisioned IOPS storage for fast and predictable performance. In addition to Multi-AZ deployments, Provisioned IOPS storage complements the following features:

- Amazon VPC for network isolation and enhanced security.
- Read replicas – The type of storage on a read replica is independent of that on the master DB instance. For example, if the master DB instance uses magnetic storage, you can add read replicas that use Provisioned IOPS storage and vice versa. You might use magnetic storage–based read replicas with a master DB instance that uses Provisioned IOPS storage. The performance of your read replicas in this case might differ considerably from that of a configuration where both the master DB instance and read replicas use Provisioned IOPS storage.
- DB snapshots – If you are using a DB instance that uses Provisioned IOPS storage, you can use a DB snapshot to restore an identically configured DB instance. You can do so regardless of whether the target DB instance uses magnetic storage or Provisioned IOPS storage. If your DB instance uses magnetic storage, you can use a DB snapshot to restore only a DB instance that uses magnetic storage.
- You can use Provisioned IOPS storage with any DB instance class. However, smaller DB instance classes don't consistently make the best use of Provisioned IOPS storage. For the best performance, we recommend that you use one of the DB instance types that are optimized for Provisioned IOPS storage.

Provisioned IOPS Storage Costs

Because Provisioned IOPS storage reserves resources for your use, you are charged for the resources whether or not you use them in a given month. When you use Provisioned IOPS storage, you are not charged the monthly Amazon RDS I/O charge. If you prefer to pay only for I/O that you consume, a DB instance that uses magnetic storage might be a better choice.

For more information about pricing, see Amazon RDS Pricing.

Getting the Most Out of Amazon RDS Provisioned IOPS

Using Provisioned IOPS storage increases the number of I/O requests that the system can process concurrently. Increased concurrency allows for decreased latency since I/O requests spend less time in a queue. Decreased latency allows for faster database commits, which improves response time and allows for higher database throughput.

For example, consider a heavily loaded OLTP database provisioned for 10,000 Provisioned IOPS that runs consistently at the channel limit of 105 Mbps throughput for reads. The workload isn't perfectly balanced, so there is some unused write channel bandwidth. The instance would consume less than 10,000 IOPS and but would still benefit from increasing capacity to 20,000 Provisioned IOPS.

Increasing Provisioned IOPS capacity from 10,000 to 20,000 doubles the system's capacity for concurrent I/O. Increased concurrency means decreased latency, which allows transactions to complete faster, so the database transaction rate increases. Read and write latency would improve by different amounts and the system would settle into a new equilibrium based on whichever resource becomes constrained first.

It is possible for Provisioned IOPS consumption to actually *decrease* under these conditions even though the database transaction rate can be much higher. For example, you can see write requests decline

accompanied by an increase in write throughput. That's a good indicator that your database is making better use of group commit. More write throughput and the same write IOPS means log writes have become larger but are still less than 256 KB. More write throughput and fewer write I/O means log writes have become larger and are averaging larger than 32 KB since those I/O requests consume more than one I/O of Provisioned IOPS capacity.

Provisioned IOPS Storage Support in the AWS CLI and Amazon RDS API

The AWS CLI supports Provisioned IOPS storage in the following commands:

- `create-db-snapshot` – The output shows the IOPS value.
- `create-db-instance` – Includes the input parameter `iops`, and the output shows the IOPS rate.
- `modify-db-instance` – Includes the input parameter `iops`, and the output shows the IOPS rate.
- `restore-db-instance-from-db-snapshot` – Includes the input parameter `iops`, and the output shows current IOPS rate. If **Apply Immediately** was specified, the output also shows the pending IOPS rate.
- `restore-db-instance-to-point-in-time` – Includes the input parameter `iops`, and the output shows the IOPS rate.
- `create-db-instance-read-replica` – Includes the input parameter `iops`, and the output shows the IOPS rate.

The Amazon RDS API supports Provisioned IOPS storage in the following actions:

- `CreateDBInstance` – Includes the input parameter `iops`, and the output shows the IOPS rate.
- `CreateDBInstanceReadReplica` – Includes the input parameter `iops`, and the output shows the IOPS rate.
- `CreateDBSnapshot` – The output shows the IOPS rate.
- `ModifyDBInstance` – Includes the input parameter `iops`, and the output shows the IOPS rate.
- `RestoreDBInstanceFromDBSnapshot` – Includes the input parameter `iops`, and the output shows current IOPS rate. If **Apply Immediately** was specified, the output also shows the pending IOPS rate.
- `RestoreDBInstanceToPointInTime` – Includes the input parameter `iops`, and the output shows the IOPS rate.

Adding Storage and Changing Storage Type for MariaDB, MySQL, Oracle, and PostgreSQL

For existing DB instances, you might observe some I/O capacity improvement If you run these types of DB instances on current-generation or next-generation instance classes, scaling up storage typically only takes a few minutes. Similarly, converting to a different storage type completes in a short amount of time after a brief DB instance outage. For more information about current-generation and next-generation instance classes, see DB Instance Class (p. 92).

After you modify the storage for a DB instance, the status of the DB instance is **storage-optimization**. The DB instance is fully operational after a storage modification. However, you can't make further storage modifications for either six hours or while the DB instance status is **storage-optimization**, whichever is longer.

Adding Storage and Changing Storage Type for Microsoft SQL Server

For existing DB instances, you might observe some I/O capacity improvement if you scale up your storage. You can modify a DB instance to use additional storage and you can convert to a different storage type.

When you modify your DB instance to increase the allocated storage, a short outage of a few minutes may occur. After that, the DB instance is online but in the storage-optimization state. Performance may be degraded during storage optimization. The storage optimization process is usually short, but can sometimes take up to and even beyond 24 hours. Once Amazon RDS begins to modify your SQL Server DB instance to increase the storage size or type, you can't submit another request to increase the storage size or type for six hours, or while the status is storage-optimization. There is no impact to other operations, such as backups and snapshots, during storage optimization.

For more information, see Modifying a DB Instance Running the Microsoft SQL Server Database Engine (p. 756).

Facts About Amazon RDS Storage

The following points are important facts you should know about Amazon RDS storage:

- Maximum channel bandwidth depends on the DB instance class.
- I/O size doesn't affect the IOPS values reported by the metrics, which are based solely on the number of I/Os over time. This functionality means that it is possible to consume all of the IOPS provisioned with fewer I/Os than specified if the I/O sizes are larger than 32 KB. For example, a system provisioned for 5,000 IOPS can attain a maximum of 2,500 IOPS with 64 KB I/O or 1,250 IOPS with 128 KB IO.

  Magnetic storage doesn't provision I/O capacity, so all I/O sizes are counted as a single I/O. General purpose storage provisions I/O capacity based on the size of the volume. For more information on general purpose storage throughput, see General Purpose (SSD) Volumes.
- Provisioned IOPS provides a way to reserve I/O capacity by specifying IOPS. Like any other system capacity attribute, maximum throughput under load is constrained by the resource that is consumed first. That resource might be IOPS, channel bandwidth, CPU, memory, or database internal resources.

Other Factors That Impact Storage Performance

All of the following system-related activities consume I/O capacity and might reduce database instance performance while in progress:

- Multi-AZ peer creation
- Read replica creation
- Scaling storage

System resources can constrain the throughput of a DB instance, but there can be other reasons for a bottleneck. If you encounter the following situations, the database might be the issue:

- The channel throughput limit is not reached.
- Queue depths are consistently low.
- CPU utilization is under 80 percent.

- There is free memory available.
- Your application has dozens of threads all submitting transactions as fast as the database takes them, but there is clearly unused I/O capacity.

If there isn't at least one system resource that is at or near a limit, and adding threads doesn't increase the database transaction rate, the bottleneck is most likely contention in the database. The most common forms are row lock and index page lock contention, but there are many other possibilities. If this is your situation, you should seek the advice of a database performance tuning expert.

Factors That Affect Realized IOPS Rates

Your actual realized IOPS rate might vary from the amount that you provision depending on page size and network bandwidth, which are determined in part by your DB engine. It is also affected by DB instance size and database workload.

Page Size and Channel Bandwidth

The theoretical maximum IOPS rate is also a function of database I/O page size and available channel bandwidth. MySQL and MariaDB use a page size of 16 KB. Oracle, PostgreSQL (default), and SQL Server use 8 KB. On a DB instance with a full duplex I/O channel bandwidth of 1000 megabits per second (Mbps), the maximum IOPS for page I/O is about 8,000 IOPS total for both directions (input/output channel) for 16 KB I/O. It is 16,000 IOPS total for both directions for 8 KB I/O.

If traffic on one of the channels reaches capacity, available IOPS on the other channel cannot be reallocated. As a result, the attainable IOPS rate is less than the provisioned IOPS rate.

Each page read or write action constitutes one I/O operation. Database operations that read or write more than a single page use multiple I/O operations for each database operation. I/O requests larger than 32 KB are treated as more than one I/O for the purposes of PIOPS capacity consumption. A 40 KB I/O request consumes 1.25 I/Os, a 48 KB request consumes 1.5 I/Os, a 64 KB request consumes 2 I/Os, and so on. The I/O request isn't split into separate I/Os; all I/O requests are presented to the storage device unchanged. For example, if the database submits a 128 KB I/O request, it goes to the storage device as a single 128 KB I/O request. However, it consumes the same amount of PIOPS capacity as four 32 KB I/O requests.

DB Instance Classes for Provisioned IOPS

If you are using Provisioned IOPS storage, we recommend that you use the M4, M3, R4, R3, and M2 DB instance classes. These instance classes are optimized for Provisioned IOPS storage; other instance classes are not.

DB Instance Classes Optimized for Provisioned IOPS	Dedicated EBS Throughput (Mbps)	Maximum 16k IOPS Rate**	Max Bandwidth (MB/s)**
db.m1.large	500 Mbps	4000	62.5
db.m1.xlarge	1000 Mbps	8000	125
db.m2.2xlarge	500 Mbps	4000	62.5
db.m2.4xlarge	1000 Mbps	8000	125

DB Instance Classes Optimized for Provisioned IOPS	Dedicated EBS Throughput (Mbps)	Maximum 16k IOPS Rate**	Max Bandwidth (MB/s)**
db.m3.xlarge	500 Mbps	4000	62.5
db.m3.2xlarge	1000 Mbps	8000	125
db.r3.xlarge	500 Mbps	4000	62.5
db.r3.2xlarge	1000 Mbps	8000	125
db.r3.4xlarge	2000 Mbps	16000	250
db.r3.8xlarge	*	*	*
db.r4.large	425 Mbps	3000	50
db.r4.xlarge	850 Mbps	6000	100
db.r4.2xlarge	1700 Mbps	12000	200
db.r4.4xlarge	3500 Mbps	18750	437
db.r4.8xlarge	7000 Mbps	37500	875
db.r4.16xlarge	14,000 Mbps	75000	1750
db.m4.large	450 Mbps	3600	56.25
db.m4.xlarge	750 Mbps	6000	93.75
db.m4.2xlarge	1000 Mbps	8000	125
db.m4.4xlarge	2000 Mbps	16000	250
db.m4.10xlarge	4000 Mbps	32000	500
db.m4.16xlarge	10,000 Mbps	65000	1250

* The r3.8xlarge DB instance class has a 10-gigabit network interface that doesn't offer Amazon EBS optimization. Therefore, dedicated EBS bandwidth is not available in the r3.8xlarge DB instance class. However, you can use all of that bandwidth for traffic to EBS if your application isn't pushing other network traffic that contends with EBS.

** This value is a rounded approximation based on a 100 percent read-only workload, and it's provided as a baseline configuration aid. EBS-optimized connections are full-duplex. They can drive more throughput and IOPS in a 50/50 read/write workload where both communication lanes are used. In some cases, network, file system, and EBS encryption overhead can reduce the maximum throughput and IOPS available.

Database Workload

System activities such as automated backups, DB snapshots, and scale storage operations might consume some I/O, which reduces the overall capacity available for normal database operations. If your database design results in concurrency issues, locking, or other forms of database contention, you might not be able to directly use all the bandwidth that you provision.

If you provision IOPS capacity to meet your peak workload demand, during the nonpeak periods your application probably consumes fewer IOPS on average than provisioned.

To help you verify that you are making the best use of your Provisioned IOPS storage, we have added a new CloudWatch Metric called Disk Queue Depth. If your application is maintaining an average queue depth of approximately 5 outstanding I/O operations per 1000 IOPS that you provision, you can assume that you are consuming the capacity that you provisioned. For example, if you provisioned 10,000 IOPS, you should have a minimum of 50 outstanding I/O operations in order to use the capacity you provisioned.

Amazon RDS Storage Limitations

The following are Amazon RDS storage limitations:

- You can't decrease allocated storage for a DB instance.
- If your DB instance is running on a previous generation instance class, adding storage or converting to a different storage type can take time and might slightly reduce the performance of your DB instance. So, you should plan when to make these changes.

  Although your DB instance is available for reads and writes when adding storage, you might experience degraded performance until the process is complete. Adding storage might take several hours; the duration of the process depends on several factors such as database load, storage size, storage type, and amount of IOPS provisioned. Typical scale storage time, depending on the size of the source volume, is between one and two hours, but can take up to several days in some cases. During the scaling process, the DB instance is available for use but might experience performance degradation. While storage is being added, nightly backups are suspended and no other Amazon RDS operations can take place, including modify, reboot, delete, create Read Replica, and create DB Snapshot.

  In general, previous generation DB instance classes don't support storage sizes greater than 6 TiB. The only exception is the db.m3.medium instance class, which supports storage up to a maximum of 16 TiB.

  For more information about DB instance classes, see DB Instance Class (p. 92).

  Note
  These limitations don't apply to current generation and next generation instance classes.
- For any type of instance class (next generation, current generation, or previous generation), storage conversions to or from magnetic storage and any other type of storage can take a long time. Storage conversions between magnetic storage and general purpose (SSD) storage can potentially deplete the initial 5.4 million I/O credits (3,000 IOPS X 30 minutes) allocated for general purpose (SSD) storage. When performing these storage conversions, the first 82 GiB of data is converted at approximately 3,000 IOPS. The remaining data is converted at the base performance rate of 100 IOPS per GiB of allocated general purpose (SSD) storage. This approach can result in longer conversion times.

Working with Storage Types

To specify how you want your data stored in Amazon RDS, you select a storage type and provide a storage size (in gibibytes) when you create or modify a DB instance. You can change the type of storage your instance uses by modifying the DB instance, but changing the storage type results in a short outage for the instance. However, increasing the allocated storage doesn't result in an outage. For more information about Amazon RDS storage types, see Amazon RDS Storage Types (p. 410).

You can't reduce the amount of storage once it has been allocated. The only way to reduce the amount of storage allocated to a DB instance is to dump the data out of the DB instance and create a new DB instance with less storage space. You then load the data into the new DB instance.

When estimating your storage needs, consider that Amazon RDS allocates a minimum amount of storage for file system structures. This reserved space can be up to 3 percent of the allocated storage for a DB instance, though in most cases the reserved space is far less. We recommend that you set up an Amazon CloudWatch alarm for your DB instance's free storage space and react when necessary. For information on setting CloudWatch alarms, see the CloudWatch Getting Started Guide.

Topics

Modifying a DB Instance to Use a Different Storage Type

You can use the Amazon RDS Management Console, the Amazon RDS API, or the AWS Command Line Interface (AWS CLI) to modify a DB instance to use Standard (Magnetic), General Purpose (SSD), or Provisioned IOPS storage. You must specify either a value for allocated storage or specify both allocated storage and IOPS values. You might need to modify the amount of allocated storage in order to maintain the required ratio between IOPS and storage. For more information about the required ratio between IOPS and storage, see Using Provisioned IOPS Storage with Multi-AZ, Read Replicas, Snapshots, VPC, and DB Instance Classes (p. 414).

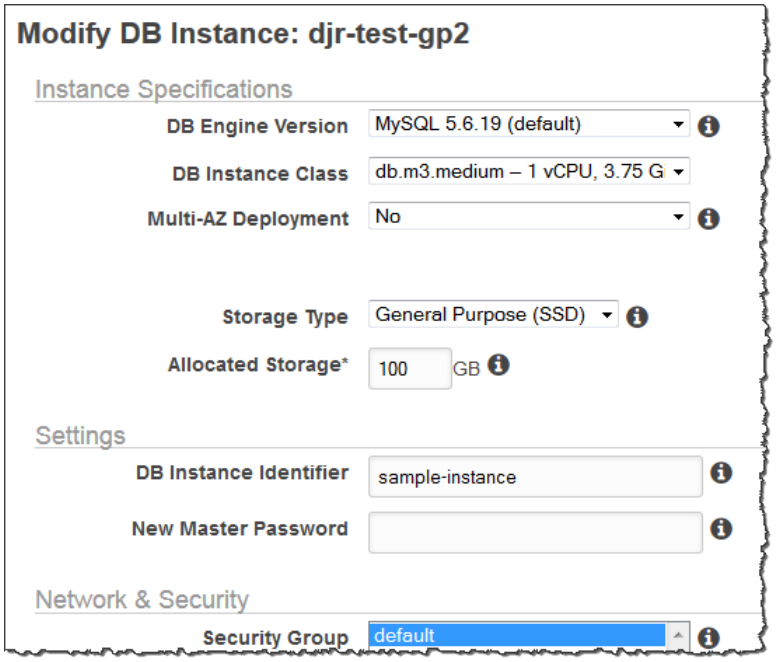
An immediate outage occurs when you convert from one storage type to another. The data for that DB instance is migrated to a new volume. The duration of the migration depends on several factors such as database load, storage size, storage type, and amount of IOPS provisioned (if any). The typical migration time is a few minutes, and the DB instance is available for use during the migration. However, when you are migrating to or from magnetic storage, the migration time usually takes longer, up to several days in some cases. During the migration to or from magnetic storage, the DB instance is available for use, but might experience performance degradation.

For DB instances in a single Availability Zone, the DB instance is unavailable for a few minutes when the conversion is initiated. For Multi-AZ deployments, the time the DB instance is unavailable is limited to the time it takes for a failover operation to complete, which typically takes less than two minutes. Although your DB instance is available for reads and writes during the conversion, you might experience degraded performance until the conversion process is complete. This process can take several hours.

AWS Management Console

To modify a DB instance to use a different storage type

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. On the navigation pane on the Amazon RDS console, choose **DB Instances**.
3. Choose the DB instance that you want to modify.
4. For **Instance Actions**, choose **Modify**.
5. For **Storage Type**, choose a value for the DB instance, and type a value for **Allocated Storage**. If you are modifying your DB instance to use the Provisioned IOPS storage type, then also provide a **Provisioned IOPS** value. For more information, see Modifying IOPS and Storage Settings for a DB Instance That Uses Provisioned IOPS Storage (p. 422).



6. To immediately initiate conversion of the DB instance to use the new storage type, select the **Apply Immediately** check box. If the check box is cleared (the default), the changes are applied during the next maintenance window. An immediate outage occurs when the conversion is applied. For more information about storage, see Storage for Amazon RDS (p. 410).
7. When the settings are as you want them, choose **Continue**.

CLI

To modify a DB instance to use a different storage type, use the AWS CLI `modify-db-instance` command. Set the following parameters:

- `--allocated-storage` – Amount of storage to be allocated for the DB instance, in gibibytes.
- `--storage-type` – The new storage type for the DB instance. You can specify `gp2` for general purpose (SSD), `io1` for Provisioned IOPS), or `standard` for magnetic storage.
- `--apply-immediately` – Use `--apply-immediately` to initiate conversion immediately, or `--no-apply-immediately` (the default) to apply the conversion during the next maintenance window. An immediate outage occurs when the conversion is applied. For more information about storage, see Storage for Amazon RDS (p. 410).

API

Use the Amazon RDS API `ModifyDBInstance` action. Set the following parameters:

- `AllocatedStorage` – Amount of storage to be allocated for the DB instance, in gibibytes.
- `StorageType` – The new storage type for the DB instance. You can specify `gp2` for general purpose (SSD), `io1` for Provisioned IOPS), or `standard` for magnetic storage.
- `ApplyImmediately` – Set to `True` if you want to initiate conversion immediately. If `False` (the default), the conversion is applied during the next maintenance window. An immediate outage occurs when the conversion is applied. For more information about storage, see Storage for Amazon RDS (p. 410).

Modifying IOPS and Storage Settings for a DB Instance That Uses Provisioned IOPS Storage

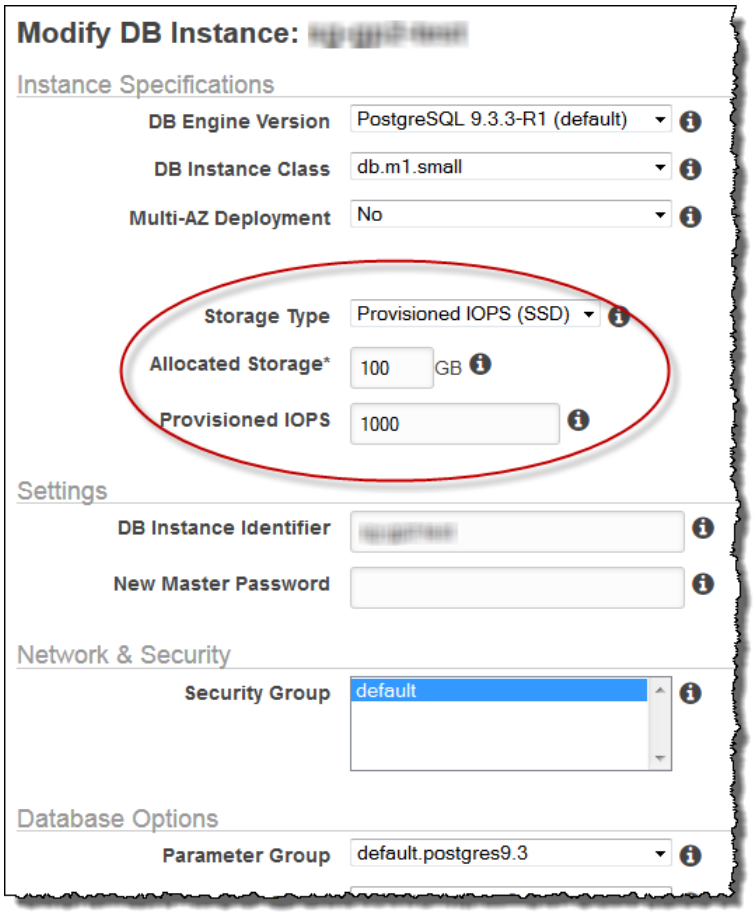
You can modify the settings for an Oracle, PostgreSQL, MySQL, or MariaDB DB instance that uses Provisioned IOPS storage by using the AWS Management Console, the Amazon RDS API, or the AWS Command Line Interface (AWS CLI). You must specify the storage type, allocated storage, and the amount of Provisioned IOPS that you require. You can choose from 1000 IOPS and 100 GiB of storage up to 40,000 IOPS and 16 TiB (16384 GiB) of storage, depending on your database engine. You cannot reduce the amount of allocated storage from the value currently allocated for the DB instance. For more information, see Using Provisioned IOPS Storage with Multi-AZ, Read Replicas, Snapshots, VPC, and DB Instance Classes (p. 414).

AWS Management Console

To modify the Provisioned IOPS settings for a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **DB Instances**.

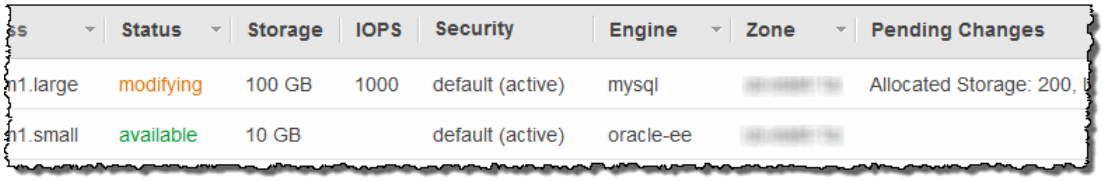
   Note
   To filter the list of DB instances, for **Search DB Instances**, type a text string for Amazon RDS to use to filter the results. Only DB instances whose names contain the string appear.
3. Choose the DB instance with Provisioned IOPS storage that you want to modify.
4. For **Instance Actions**, choose **Modify**.
5. On the **Modify DB Instance** page, type the value that you want for either **Allocated Storage** or **Provisioned IOPS**.



If the value you specify for either **Allocated Storage** or **Provisioned IOPS** is outside the limits supported by the other parameter, a warning message is displayed indicating the range of values required for the other parameter.

6. To apply the changes to the DB instance immediately, select the **Apply Immediately** check box. If you leave the check box cleared, the changes are applied during the next maintenance window.
7. Choose **Continue**.
8. Review the parameters that will be changed, and choose **Modify DB Instance** to complete the modification.

   The new value for allocated storage or for provisioned IOPS appears in the **Pending Values** column.



CLI

To modify the Provisioned IOPS settings for a DB instance use the AWS CLI `modify-db-instance` command. Set the following parameters:

- `--storage-type` – Set to `io1` for Provisioned IOPS.
- `--allocated-storage` – Amount of storage to be allocated for the DB instance, in gibibytes.
- `--iops` – The new amount of Provisioned IOPS for the DB instance, expressed in I/O operations per second.
- `--apply-immediately` – Use `--apply-immediately` to initiate conversion immediately. Use `--no-apply-immediately` (the default) to apply the conversion during the next maintenance window.

API

To modify the Provisioned IOPS settings for a DB instance use the Amazon RDS API `ModifyDBInstance` action. Set the following parameters:

- `StorageType` – Set to `io1` for Provisioned IOPS.
- `AllocatedStorage` – Amount of storage to be allocated for the DB instance, in gibibytes.
- `Iops` – The new IOPS rate for the DB instance, expressed in I/O operations per second.
- `ApplyImmediately` – Set to `True` if you want to initiate conversion immediately. If `False` (the default), the conversion is applied during the next maintenance window.

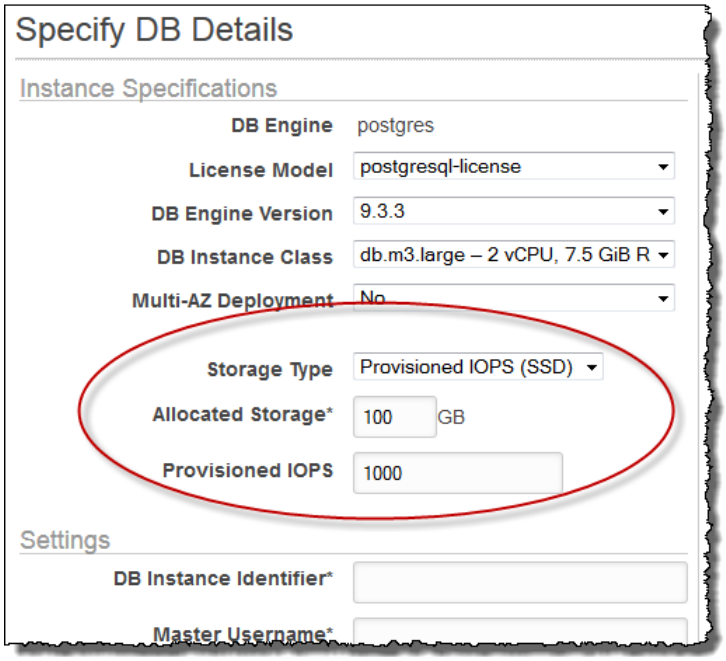
Creating a DB Instance That Uses Provisioned IOPS Storage

You can create a DB instance that uses Provisioned IOPS by setting several parameters when you launch the DB instance. You can use the AWS Management Console, the Amazon RDS API, or the AWS Command Line Interface (AWS CLI). For more information about the settings you should use when creating a DB instance, see Creating a DB Instance Running the MySQL Database Engine (p. 830), Creating a DB Instance Running the MariaDB Database Engine (p. 678), Creating a DB Instance Running the Oracle Database Engine (p. 949), or Creating a DB Instance Running the Microsoft SQL Server Database Engine (p. 738).

AWS Management Console

To create a new DB instance that uses Provisioned IOPS storage

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. From the Amazon RDS console, choose **Launch DB Instance**.
3. In the Launch RDS DB Instance wizard, on the **Engine Selection** page, choose the **Select** button next to the DB engine that you want.
4. On the **Specify DB Details** page, choose **Provisioned IOPS (SSD)** for **Storage Type**.
5. Specify values for **Allocated Storage** and **Provisioned IOPS**. For information about the allowed ranges and ratios, see Provisioned IOPS Storage (p. 413).



6. When the settings are as you want them, choose **Continue**. Type the remaining values to create the DB instance.

CLI

To create a new DB instance that uses Provisioned IOPS storage, use the AWS CLI `create-db-instance` command. Specify the required parameters and include values for the following parameters that apply to Provisioned IOPS storage:

- `--storage-type` – Set to `io1` for Provisioned IOPS.
- `--allocated-storage` - Amount of storage to be allocated for the DB instance, in gibibytes.
- `--iops` - The new IOPS rate for the DB instance, expressed in I/O operations per second.

API

To create a new DB instance that uses Provisioned IOPS storage, use the Amazon RDS API `CreateDBInstance` action. Specify the required parameters and include values for the following parameters that apply to Provisioned IOPS storage:

- `StorageType` – Set to `io1` for Provisioned IOPS.
- `AllocatedStorage` - Amount of storage to be allocated for the DB instance, in gibibytes.
- `Iops` - The new IOPS rate for the DB instance, expressed in I/O operations per second.

Creating a MySQL or MariaDB Read Replica That Uses Provisioned IOPS Storage

You can create a MySQL or MariaDB Read Replica that uses Provisioned IOPS storage. You can create a Read Replica that uses Provisioned IOPS storage by using a source DB instance that uses either standard storage or Provisioned IOPS storage.

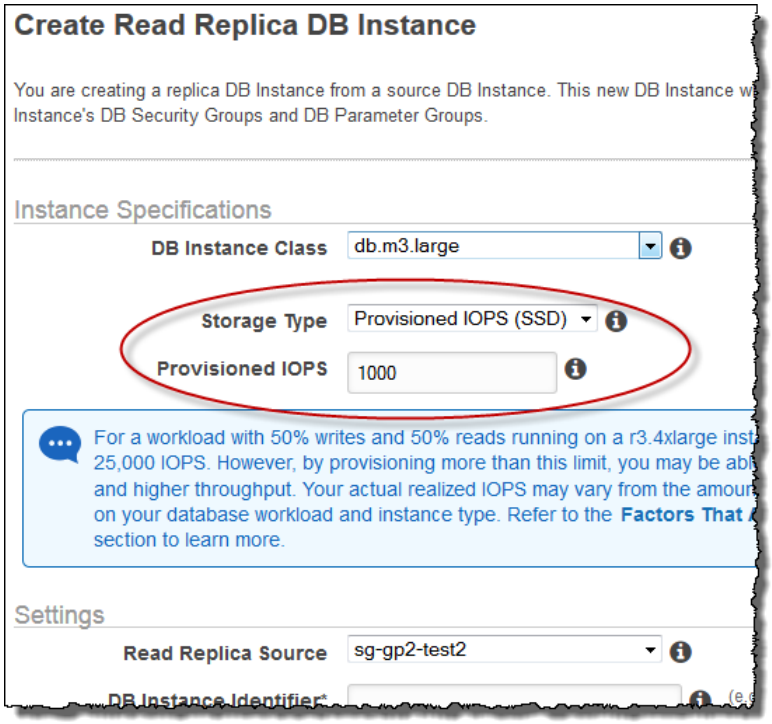
AWS Management Console

For a complete description on how to create a Read Replica, see Creating a Read Replica (p. 139).

To create a Read Replica DB instance that uses Provisioned IOPS storage

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the **Navigation** pane, choose **DB Instances**.
3. Choose the MySQL or MariaDB DB instance with Provisioned IOPS storage that you want to use as the source for the Read Replica, and choose **Instance Actions**, **Create Read Replica**.

   Important
   The DB instance that you are creating a Read Replica for must have allocated storage within the range of storage for MySQL and MariaDB PIOPS (100 GiB–16 TiB). If the allocated storage for that DB instance is not within that range, then the **Provisioned IOPS** storage type isn't available as an option when creating the Read Replica. Instead, you can set only the **GP2** or **Standard** storage types. You can modify the allocated storage for the source DB instance to be within the range of storage for MySQL and MariaDB PIOPS before creating a Read Replica. For more information on the PIOPS range of storage, see Provisioned IOPS Storage (p. 413). For information on modifying a MySQL DB instance, see Modifying a DB Instance Running the MySQL Database Engine (p. 843). For information on modifying a MariaDB DB instance, see Modifying a DB Instance Running the MariaDB Database Engine (p. 691).
4. On the **Create Read Replica DB Instance** page, type a DB instance identifier for the Read Replica.



5. Choose **Yes, Create Read Replica**.

CLI

To create a Read Replica DB instance that uses Provisioned IOPS, use the AWS CLI `create-db-instance-read-replica` command. Specify the required parameters and include values for the following parameters that apply to Provisioned IOPS storage:

- `--allocated-storage` - Amount of storage to be allocated for the DB instance, in gibibytes.
- `--iops` - The new IOPS rate for the DB instance, expressed in I/O operations per second.

API

To create a Read Replica DB instance that uses Provisioned IOPS, use the Amazon RDS API `CreateDBInstanceReadReplica` action. Specify the required parameters and include values for the following parameters that apply to Provisioned IOPS storage:

- `AllocatedStorage` - Amount of storage to be allocated for the DB instance, in gibibytes.
- `Iops` - The new IOPS rate for the DB instance, expressed in I/O operations per second.

Amazon Aurora on Amazon RDS

Amazon Aurora (Aurora) is a fully managed, MySQL- and PostgreSQL-compatible, relational database engine. It combines the speed and reliability of high-end commercial databases with the simplicity and cost-effectiveness of open-source databases.

Aurora makes it simple and cost-effective to set up, operate, and scale your MySQL and PostgreSQL deployments, freeing you to focus on your business and applications. Amazon RDS provides administration for Aurora by handling routine database tasks such as provisioning, patching, backup, recovery, failure detection, and repair. Amazon RDS also provides push-button migration tools to convert your existing Amazon RDS for MySQL and Amazon RDS for PostgreSQL applications to Aurora.

Amazon Aurora is a drop-in replacement for MySQL and PostgreSQL. The code, tools, and applications you use today with your existing MySQL and PostgreSQL databases can be used with Amazon Aurora.

Before using Amazon Aurora, you should complete the steps in Setting Up for Amazon RDS (p. 5), and then review the concepts and features of Aurora in Overview of Amazon Aurora (p. 430)

Common Management Tasks for Amazon Aurora

The following are the common management tasks you perform for an Amazon Aurora DB cluster, with links to relevant documentation for each task.

Task Area	Relevant Documentation
Setting up Amazon RDS for first-time use Set up Amazon RDS so that you can use Amazon Aurora.	Setting Up for Amazon RDS (p. 5)
Understanding Amazon Aurora Learn about basic Amazon Aurora concepts and features, such as DB clusters, primary instances, and Aurora Replicas.	Overview of Amazon Aurora (p. 430)
Create an Amazon Aurora DB Cluster Create Amazon Aurora DB clusters and Aurora Replicas using the AWS Management Console or the AWS Command Line Interface (AWS CLI).	Creating an Amazon Aurora DB Cluster (p. 437)
Migrating from another database or RDS DB instance to an Amazon Aurora DB cluster You have several options for migrating data from an on-premises database or other RDS DB instance to your Aurora DB cluster.	Migrating Data to an Amazon Aurora DB Cluster (p. 466)
Managing access permissions to Amazon RDS resources and databases Learn how to manage access to your Aurora DB clusters and other Amazon RDS capabilities.	Security in Amazon RDS (p. 326) Overview of Managing Access Permissions to Your Amazon RDS Resources (p. 328)

Task Area	Relevant Documentation
Configuring specific Amazon Aurora database parameters If your DB cluster is going to require specific database parameters, you can create a DB parameter group and a DB cluster parameter group before you create the DB cluster.	Amazon Aurora DB Cluster and DB Instance Parameters (p. 469) Working with DB Parameter Groups (p. 170)
Connecting to your Amazon Aurora DB cluster Learn how to connect to your Aurora DB cluster using a standard client application.	Aurora Endpoints (p. 431) Connecting to an Amazon Aurora DB Cluster (p. 457)
Integrating your Amazon Aurora DB cluster with other AWS services Learn how to extend your Aurora DB cluster to use additional capabilities in the AWS Cloud.	Integrating Aurora with Other AWS Services (p. 483)
Configuring database backup and restore Amazon Aurora automatically backs up your data. You can configure how long Aurora keeps backups for, and you can restore from a DB cluster snapshot, from a specific point in time, or from other files.	Backing Up and Restoring an Aurora DB Cluster (p. 468) Migrating Data to an Amazon Aurora DB Cluster (p. 466)
Monitoring an Amazon Aurora DB cluster You can monitor your Aurora DB cluster by using Amazon CloudWatch, RDS events, and Enhanced Monitoring (CloudWatch Logs).	Monitoring an Amazon Aurora DB Cluster (p. 470) Monitoring Amazon RDS (p. 245)
Updating the database engine or operating system for your Amazon Aurora DB cluster Learn how to manage database engine and operating system updates for your Aurora DB cluster.	Amazon Aurora MySQL Database Engine Updates (p. 610) Amazon Aurora PostgreSQL Database Engine Updates (p. 662) DB Instance and DB Cluster Maintenance (p. 102)
Set up replication with an Amazon Aurora DB cluster Learn how to replicate data between your Amazon Aurora DB cluster and an on-premises database, an Aurora DB cluster in a different AWS Region, or another RDS DB instance.	Replication with Amazon Aurora (p. 478)
Copy or share an Amazon Aurora DB cluster snapshot Learn how to copy an Aurora DB cluster snapshot to the same AWS Region, or a different AWS Region. Learn how to share an Aurora DB cluster snapshot with other AWS accounts.	Copying a DB Snapshot or DB Cluster Snapshot (p. 213) Sharing a DB Snapshot or DB Cluster Snapshot (p. 230)

Overview of Amazon Aurora

Amazon Aurora (Aurora) is a fully managed, MySQL- and PostgreSQL-compatible, relational database engine. It combines the speed and reliability of high-end commercial databases with the simplicity and cost-effectiveness of open-source databases. It delivers up to five times the throughput of MySQL and up to three times the throughput of PostgreSQL without requiring changes to most of your existing applications.

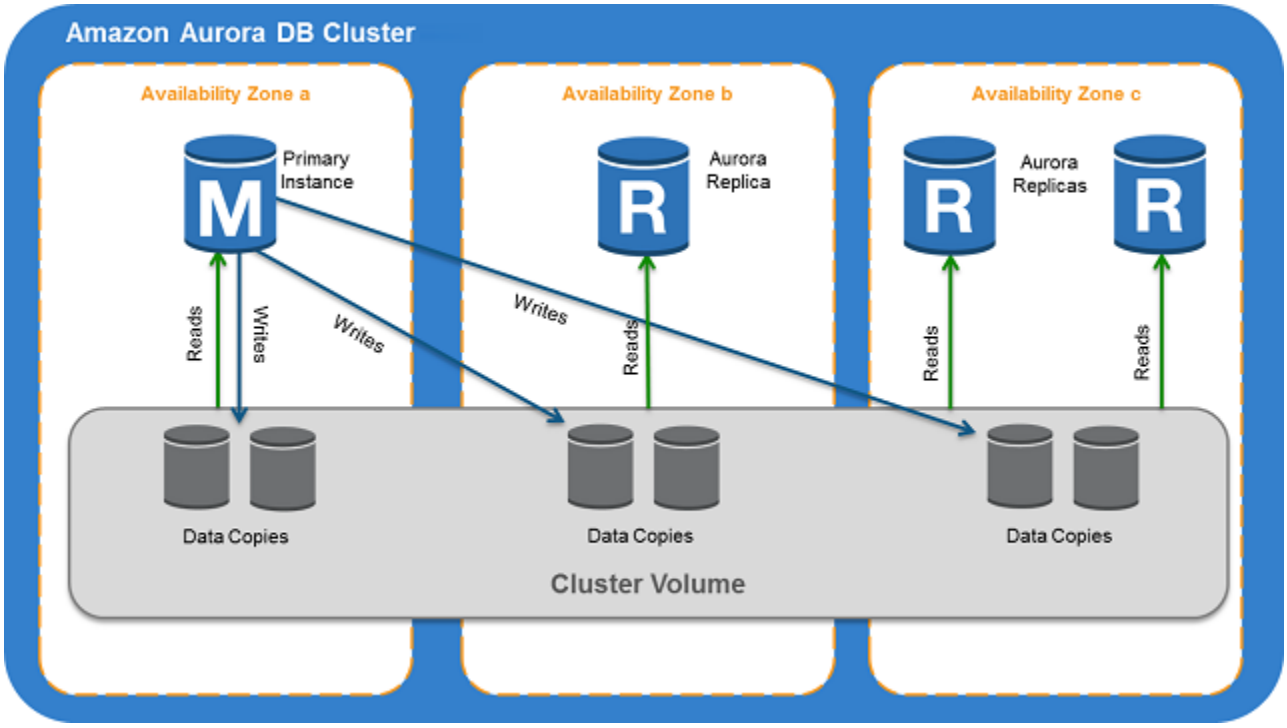
Aurora makes it simple and cost-effective to set up, operate, and scale your MySQL and PostgreSQL deployments, freeing you to focus on your business and applications. Amazon RDS provides administration for Aurora by handling routine database tasks such as provisioning, patching, backup, recovery, failure detection, and repair. Amazon RDS also provides push-button migration tools to convert your existing Amazon RDS for MySQL and Amazon RDS for PostgreSQL applications to Aurora.

Amazon Aurora is a drop-in replacement for MySQL and PostgreSQL. The code, tools and applications you use today with your existing MySQL and PostgreSQL databases can be used with Amazon Aurora.

When you create an Amazon Aurora instance, you create a *DB cluster*. A DB cluster consists of one or more DB instances, and a cluster volume that manages the data for those instances. An Aurora *cluster volume* is a virtual database storage volume that spans multiple Availability Zones, with each Availability Zone having a copy of the DB cluster data. Two types of DB instances make up an Aurora DB cluster:

- **Primary instance** – Supports read and write operations, and performs all of the data modifications to the cluster volume. Each Aurora DB cluster has one primary instance.
- **Aurora Replica** – Supports only read operations. Each Aurora DB cluster can have up to 15 Aurora Replicas in addition to the primary instance. Multiple Aurora Replicas distribute the read workload, and by locating Aurora Replicas in separate Availability Zones you can also increase database availability.

The following diagram illustrates the relationship between the cluster volume, the primary instance, and Aurora Replicas in an Aurora DB cluster.



Availability

Availability in AWS Regions for Amazon Aurora varies by database engine compatibility.

Database Engine	Availability
Amazon Aurora MySQL	See Availability for Amazon Aurora MySQL (p. 484)
Amazon Aurora PostgreSQL	See Availability for Amazon Aurora PostgreSQL (p. 641)

Aurora Endpoints

You can connect to DB instances in an Amazon Aurora DB cluster by using an endpoint. An *endpoint* is a URL that contains a host address and a port, separated by a colon. The following endpoints are available from an Aurora DB cluster.

Cluster endpoint

An endpoint for a Aurora DB cluster that connects to the current primary instance for that DB cluster. Each Aurora DB cluster has a cluster endpoint.

The cluster endpoint provides failover support for read/write connections to the DB cluster. If the current primary instance of a DB cluster fails, Aurora automatically fails over to a new primary instance. During a failover, the DB cluster continues to serve connection requests to the cluster endpoint from the new primary instance, with minimal interruption of service.

The following example illustrates a cluster endpoint for an Aurora MySQL DB cluster.

```
mydbcluster.cluster-123456789012.us-east-1.rds.amazonaws.com:3306
```

Reader endpoint

An endpoint for an Aurora DB cluster that connects to one of the available Aurora Replicas for that DB cluster. Each Aurora DB cluster has a reader endpoint.

The reader endpoint provides load balancing support for read-only connections to the DB cluster. The DB cluster distributes connection requests to the reader endpoint among available Aurora Replicas. If the DB cluster contains only a primary instance, the reader endpoint serves connection requests from the primary instance. If an Aurora Replica is created for that DB cluster, the reader endpoint continues to serve connection requests to the reader endpoint from the new Aurora Replica, with minimal interruption in service.

The following example illustrates a reader endpoint for an Aurora MySQL DB cluster.

```
mydbcluster.cluster-ro-123456789012.us-east-1.rds.amazonaws.com:3306
```

Instance endpoint

An endpoint for a DB instance in an Aurora DB cluster that connects to that specific DB instance. Each DB instance in a DB cluster, regardless of instance type, has its own unique instance endpoint.

The instance endpoint provides direct control over connections to the DB cluster, for scenarios where using the cluster endpoint or reader endpoint may not be appropriate. For example, your client application may require load balancing by read workload, instead of by connections, in which case you can configure multiple clients to connect to different Aurora Replicas in a DB cluster to distribute read workloads. For an example that uses instance endpoints to improve connection speed after a failover, see Fast Failover with Amazon Aurora PostgreSQL (p. 647).

The following example illustrates an instance endpoint for a DB instance in an Aurora MySQL DB cluster.

```
mydbinstance.123456789012.us-east-1.rds.amazonaws.com:3306
```

Endpoint Considerations

Some considerations for working with Aurora endpoints are as follows:

- Before using an instance endpoint to connect to a specific DB instance in a DB cluster, consider using the cluster endpoint or reader endpoint for the DB cluster instead.

  The cluster endpoint and reader endpoint provide support for high-availability scenarios. If the primary instance of a DB cluster fails, Aurora automatically fails over to a new primary instance. It does so by either promoting an existing Aurora Replica to a new primary instance or creating a new primary instance. If a failover occurs, you can use the cluster endpoint to reconnect to the newly promoted or created primary instance, or use the reader endpoint to reconnect to one of the other Aurora Replicas in the DB cluster.

  If you don't take this approach, you can still make sure that you're connecting to the right DB instance in the DB cluster for the intended operation. To do so, you can manually or programmatically discover the resulting set of available DB instances in the DB cluster and confirm their instance types after failover, before using the instance endpoint of a specific DB instance.

  For more information about failovers, see Fault Tolerance for an Aurora DB Cluster (p. 468).
- The reader endpoint only load-balances connections to available Aurora Replicas in an Aurora DB cluster. It does not load-balance specific queries. If you want to load-balance queries to distribute the read workload for a DB cluster, you need to manage that in your application and use instance endpoints to connect directly to Aurora Replicas to balance the load.
- During a failover, the reader endpoint might direct connections to the new primary instance of a DB cluster for a short time, when an Aurora Replica is promoted to the new primary instance.

Amazon Aurora Storage

Aurora data is stored in the cluster volume, which is a single, virtual volume that utilizes solid state disk (SSD) drives. A cluster volume consists of copies of the data across multiple Availability Zones in a single region. Because the data is automatically replicated across Availability Zones, your data is highly durable with less possibility of data loss. This replication also ensures that your database is more available during a failover because the data copies already exist in the other Availability Zones and continue to serve data requests to the instances in your DB cluster.

Aurora cluster volumes automatically grow as the amount of data in your database increases. An Aurora cluster volume can grow to a maximum size of 64 terabytes (TB). Table size is limited to the size of the cluster volume. That is, the maximum table size for a table in an Aurora DB cluster is 64 TB. Even though an Aurora cluster volume can grow to up to 64 TB, you are only charged for the space that you use in an Aurora cluster volume. For pricing information, see Amazon RDS for Aurora Pricing.

Amazon Aurora Replication

Aurora Replicas are independent endpoints in an Aurora DB cluster. They provide read-only access to the data in the DB cluster volume. They enable you to scale the read workload for your data over multiple replicated instances, to both improve the performance of data reads and also increase the availability of the data in your Aurora DB cluster. Aurora Replicas are also failover targets and are quickly promoted if the primary instance for your Aurora DB cluster fails.

For more information on Aurora Replicas and other options for replicating data in an Aurora DB cluster, see Replication with Amazon Aurora (p. 478).

Amazon Aurora Reliability

Aurora is designed to be reliable, durable, and fault tolerant. You can architect your Aurora DB cluster to improve availability by doing things such as adding Aurora Replicas and placing them in different Availability Zones, and also Aurora includes several automatic features that make it a reliable database solution.

Storage Auto-Repair

Because Aurora maintains multiple copies of your data in three Availability Zones, the chance of losing data as a result of a disk failure is greatly minimized. Aurora automatically detects failures in the disk volumes that make up the cluster volume. When a segment of a disk volume fails, Aurora immediately repairs the segment. When Aurora repairs the disk segment, it uses the data in the other volumes that make up the cluster volume to ensure that the data in the repaired segment is current. As a result, Aurora avoids data loss and reduces the need to perform a point-in-time restore to recover from a disk failure.

Survivable Cache Warming

Aurora "warms" the buffer pool cache when a database starts up after it has been shut down or restarted after a failure. That is, Aurora preloads the buffer pool with the pages for known common queries that are stored in an in-memory page cache. This provides a performance gain by bypassing the need for the buffer pool to "warm up" from normal database use.

The Aurora page cache is managed in a separate process from the database, which allows the page cache to survive independently of the database. In the unlikely event of a database failure, the page cache remains in memory, which ensures that the buffer pool is warmed with the most current state when the database restarts.

Crash Recovery

Aurora is designed to recover from a crash almost instantaneously and continue to serve your application data. Aurora performs crash recovery asynchronously on parallel threads, so that your database is open and available immediately after a crash. For more information, see Fault Tolerance for an Aurora DB Cluster (p. 468).

Amazon Aurora Performance Enhancements

Amazon Aurora includes performance enhancements to support the diverse needs of high-end commercial databases.

Amazon Aurora MySQL Performance Enhancements

Aurora MySQL includes the following performance enhancements:

Fast insert

Fast insert accelerates parallel inserts sorted by primary key and applies specifically to `LOAD DATA` and `INSERT INTO ... SELECT ...` statements.

For more information about performance enhancements for Aurora MySQL, see Amazon Aurora MySQL Performance Enhancements (p. 485).

Amazon Aurora Security

Security for Amazon Aurora is managed at three levels:

- To control who can perform Amazon RDS management actions on Aurora DB clusters and DB instances, you use AWS Identity and Access Management (IAM). When you connect to AWS using IAM credentials, your IAM account must have IAM policies that grant the permissions required to perform Amazon RDS management operations. For more information, see Authentication and Access Control for Amazon RDS (p. 327).

  If you are using an IAM account to access the Amazon RDS console, you must first log on to the AWS Management Console with your IAM account, and then go to the Amazon RDS console at https://console.aws.amazon.com/rds.
- Aurora DB clusters must be created in an Amazon Virtual Private Cloud (VPC). To control which devices and Amazon EC2 instances can open connections to the endpoint and port of the DB instance for Aurora DB clusters in a VPC, you use a VPC security group. These endpoint and port connections can be made using Secure Sockets Layer (SSL). In addition, firewall rules at your company can control whether devices running at your company can open connections to a DB instance. For more information on VPCs, see Amazon Virtual Private Cloud (VPCs) and Amazon RDS (p. 390).
- To authenticate logins and permissions for an Amazon Aurora DB cluster, you can take either of the following approaches, or a combination of them.
  - You can take the same approach as with a stand-alone instance of MySQL or PostgreSQL.

    Techniques for authenticating logins and permissions for stand-alone instances of MySQL or PostgreSQL, such as using SQL commands or modifying database schema tables, also work with Aurora. For more information, see Security with Amazon Aurora MySQL (p. 548) or Security with Amazon Aurora PostgreSQL (p. 646).
  - You can also use IAM database authentication for Aurora MySQL.

    With IAM database authentication, you authenticate to your Aurora MySQL DB cluster by using an IAM user or IAM role and an authentication token. An *authentication token* is a unique value that is generated using the Signature Version 4 signing process. By using IAM database authentication, you can use the same credentials to control access to your AWS resources and your databases. For more information, see IAM Database Authentication for MySQL and Amazon Aurora (p. 360).

Securing Aurora Data with SSL

Amazon Aurora DB clusters support Secure Sockets Layer (SSL) connections from applications using the same process and public key as Amazon RDS DB instances. For more information, see Security with Amazon Aurora MySQL (p. 548) or Security with Amazon Aurora PostgreSQL (p. 646).

Local Time Zone for Amazon Aurora DB Clusters

By default, the time zone for an Amazon Aurora DB cluster is Universal Time Coordinated (UTC). You can set the time zone for instances in your DB cluster to the local time zone for your application instead.

To set the local time zone for a DB cluster, set the `time_zone` parameter in the cluster parameter group for your DB cluster to one of the supported values listed later in this section. When you set the `time_zone` parameter for a DB cluster, all instances in the DB cluster change to use the new local time zone. If other Aurora DB clusters are using the same cluster parameter group, then all instances in those DB clusters change to use the new local time zone also. For information on cluster-level parameters, see Amazon Aurora DB Cluster and DB Instance Parameters (p. 469).

After you set the local time zone, all new connections to the database reflect the change. If you have any open connections to your database when you change the local time zone, you won't see the local time zone update until after you close the connection and open a new connection.

If you are replicating across regions, then the replication master DB cluster and the replica use different parameter groups (parameter groups are unique to a region). To use the same local time zone for each

instance, you must set the `time_zone` parameter in the parameter groups for both the replication master and the replica.

When you restore a DB cluster from a DB cluster snapshot, the local time zone is set to UTC. You can update the time zone to your local time zone after the restore is complete. If you restore a DB cluster to a point in time, then the local time zone for the restored DB cluster is the time zone setting from the parameter group of the restored DB cluster.

You can set your local time zone to one of the values listed in the following table. For some time zones, time values for certain date ranges can be reported incorrectly as noted in the table. For Australia time zones, the time zone abbreviation returned is an outdated value as noted in the table.

Time Zone	Notes
`Africa/Harare`	This time zone setting can return incorrect values from 28 Feb 1903 21:49:40 GMT to 28 Feb 1903 21:55:48 GMT.
`Africa/Monrovia`	
`Africa/Nairobi`	This time zone setting can return incorrect values from 31 Dec 1939 21:30:00 GMT to 31 Dec 1959 21:15:15 GMT.
`Africa/Windhoek`	
`America/Bogota`	This time zone setting can return incorrect values from 23 Nov 1914 04:56:16 GMT to 23 Nov 1914 04:56:20 GMT.
`America/Caracas`	
`America/Chihuahua`	
`America/Cuiaba`	
`America/Denver`	
`America/Fortaleza`	
`America/Guatemala`	
`America/Halifax`	This time zone setting can return incorrect values from 27 Oct 1918 05:00:00 GMT to 31 Oct 1918 05:00:00 GMT.
`America/Manaus`	
`America/Matamoros`	
`America/Monterrey`	
`America/Montevideo`	
`America/Phoenix`	
`America/Tijuana`	
`Asia/Ashgabat`	
`Asia/Baghdad`	
`Asia/Baku`	
`Asia/Bangkok`	

Time Zone	Notes
Asia/Beirut	
Asia/Calcutta	
Asia/Kabul	
Asia/Karachi	
Asia/Kathmandu	
Asia/Muscat	This time zone setting can return incorrect values from 31 Dec 1919 20:05:36 GMT to 31 Dec 1919 20:05:40 GMT.
Asia/Riyadh	This time zone setting can return incorrect values from 13 Mar 1947 20:53:08 GMT to 31 Dec 1949 20:53:08 GMT.
Asia/Seoul	This time zone setting can return incorrect values from 30 Nov 1904 15:30:00 GMT to 07 Sep 1945 15:00:00 GMT.
Asia/Shanghai	This time zone setting can return incorrect values from 31 Dec 1927 15:54:08 GMT to 02 Jun 1940 16:00:00 GMT.
Asia/Singapore	
Asia/Taipei	This time zone setting can return incorrect values from 30 Sep 1937 16:00:00 GMT to 29 Sep 1979 15:00:00 GMT.
Asia/Tehran	
Asia/Tokyo	This time zone setting can return incorrect values from 30 Sep 1937 15:00:00 GMT to 31 Dec 1937 15:00:00 GMT.
Asia/Ulaanbaatar	
Atlantic/Azores	This time zone setting can return incorrect values from 24 May 1911 01:54:32 GMT to 01 Jan 1912 01:54:32 GMT.
Australia/Adelaide	The abbreviation for this time zone is returned as CST instead of ACDT/ACST.
Australia/Brisbane	The abbreviation for this time zone is returned as EST instead of AEDT/AEST.
Australia/Darwin	The abbreviation for this time zone is returned as CST instead of ACDT/ACST.
Australia/Hobart	The abbreviation for this time zone is returned as EST instead of AEDT/AEST.
Australia/Perth	The abbreviation for this time zone is returned as WST instead of AWDT/ AWST.
Australia/Sydney	The abbreviation for this time zone is returned as EST instead of AEDT/AEST.
Brazil/East	
Canada/ Saskatchewan	This time zone setting can return incorrect values from 27 Oct 1918 08:00:00 GMT to 31 Oct 1918 08:00:00 GMT.
Europe/Amsterdam	
Europe/Athens	
Europe/Dublin	

Time Zone	Notes
Europe/Helsinki	This time zone setting can return incorrect values from 30 Apr 1921 22:20:08 GMT to 30 Apr 1921 22:20:11 GMT.
Europe/Paris	
Europe/Prague	
Europe/Sarajevo	
Pacific/Auckland	
Pacific/Guam	
Pacific/Honolulu	This time zone setting can return incorrect values from 21 May 1933 11:30:00 GMT to 30 Sep 1945 11:30:00 GMT.
Pacific/Samoa	This time zone setting can return incorrect values from 01 Jan 1911 11:22:48 GMT to 01 Jan 1950 11:30:00 GMT.
US/Alaska	
US/Central	
US/Eastern	
US/East-Indiana	
US/Pacific	
UTC	

Creating an Amazon Aurora DB Cluster

An Amazon Aurora DB cluster consists of one DB instances, compatible with either MySQL or PostgreSQL, and a cluster volume that represents the data for the DB cluster, copied across three Availability Zones as a single, virtual volume. The DB cluster contains a primary instance and, optionally, up to 15 Aurora Replicas. For more information about Aurora DB clusters, see Overview of Amazon Aurora (p. 430).

The following topic shows how to create an Aurora DB cluster and then add an Aurora Replica for that DB cluster.

> **Important**
> You must complete the tasks in the Setting Up for Amazon RDS (p. 5) section before you can create an Aurora DB cluster.

This topic describes how you can create an Aurora DB cluster using either the AWS Management Console or the AWS CLI. For simple instructions on connecting to your Aurora DB cluster, see Connecting to an Amazon Aurora DB Cluster (p. 457). For a detailed guide on connecting to an Amazon Aurora DB cluster, see RDS Aurora Connectivity.

DB Cluster Prerequisites

The following are prerequisites to create a DB cluster.

VPC

An Amazon Aurora DB cluster can only be created in an Amazon Virtual Private Cloud (VPC) with at least one subnet in each of at least two of the Availability Zones in the AWS Region where you want to deploy your DB cluster. By distributing your cluster instances across at least two Availability Zones, you ensure that there will be instances available in your DB cluster in the unlikely case of an Availability Zone failure. Note that the cluster volume for your Aurora DB cluster will always span three Availability Zones to provide durable storage with less possibility of data loss.

If you are using the AWS Management Console to create your Aurora DB cluster, then you can have Amazon RDS automatically create a VPC for you. Alternatively, you can use an existing VPC or create a new VPC for your Aurora DB cluster. Your VPC must have at least one subnet in each of at least two Availability Zones in order for you to use it with an Amazon Aurora DB cluster. For more information, see How to Create a VPC for Use with Amazon Aurora (p. 452). For information on VPCs, see Amazon Virtual Private Cloud (VPCs) and Amazon RDS (p. 390).

> **Note**
> You can communicate with an EC2 instance that is not in a VPC and an Amazon Aurora DB cluster using ClassicLink. For more information, see A DB Instance in a VPC Accessed by an EC2 Instance Not in a VPC (p. 395).

If you don't have a default VPC or you have not created a VPC, you can have Amazon RDS automatically create a VPC for you when you create an Aurora DB cluster using the AWS Management Console. Otherwise, you must do the following:

- Create a VPC with at least one subnet in each of at least two of the Availability Zones in the region where you want to deploy your DB cluster. For more information, see How to Create a VPC for Use with Amazon Aurora (p. 452).
- Specify a VPC security group that authorizes connections to your Aurora DB cluster. For more information, see Working with a DB Instance in a VPC (p. 400).
- Specify an RDS DB subnet group that defines at least two subnets in the VPC that can be used by the Aurora DB cluster. For more information, see Working with DB Subnet Groups (p. 400).

Additional Prerequisites

- If you are connecting to AWS using IAM credentials, your IAM account must have IAM policies that grant the permissions required to perform Amazon RDS operations. For more information, see Authentication and Access Control for Amazon RDS (p. 327).

  If you are using an IAM account to access the Amazon RDS console, you must first log on to the AWS Management Console with your IAM account, and then go to the Amazon RDS console at https://console.aws.amazon.com/rds/.
- If you want to tailor the configuration parameters for your DB cluster, you must specify a DB cluster parameter group and DB parameter group with the required parameter settings. For information about creating or modifying a DB cluster parameter group or DB parameter group, see Working with DB Parameter Groups (p. 170).
- You must determine the TCP/IP port number you will specify for your DB cluster. The firewalls at some companies block connections to the default ports (3306 for MySQL, 5432 for PostgreSQL) for Aurora. If your company firewall blocks the default port, choose another port for your DB cluster. All instances in a DB cluster use the same port.

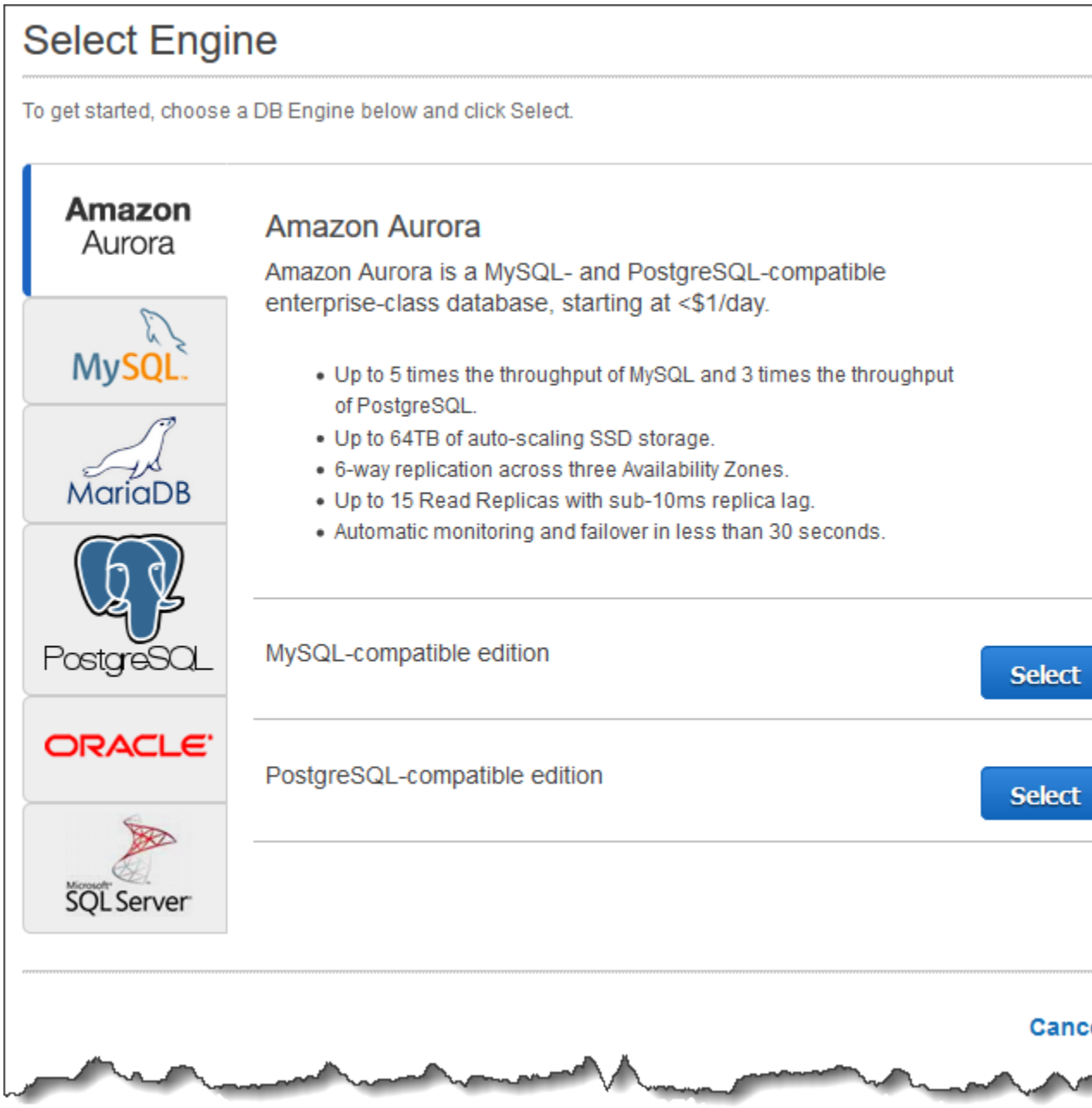
AWS Management Console

Launching an Aurora DB Cluster

The following procedures describe how to use the AWS Management Console to launch an Aurora DB cluster and create an Aurora Replica.

To launch an Aurora DB cluster using the AWS Management Console

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the top-right corner of the AWS Management Console, select the AWS Region in which you want to create the Aurora DB cluster.
3. In the navigation pane, choose **Instances**.
4. Choose **Launch DB Instance** to start the Launch DB Instance wizard. The wizard opens on the **Select Engine** page.
5. On the **Select Engine** page, choose the **Select** button for either the MySQL-compatible or PostgreSQL-compatible edition of Aurora.

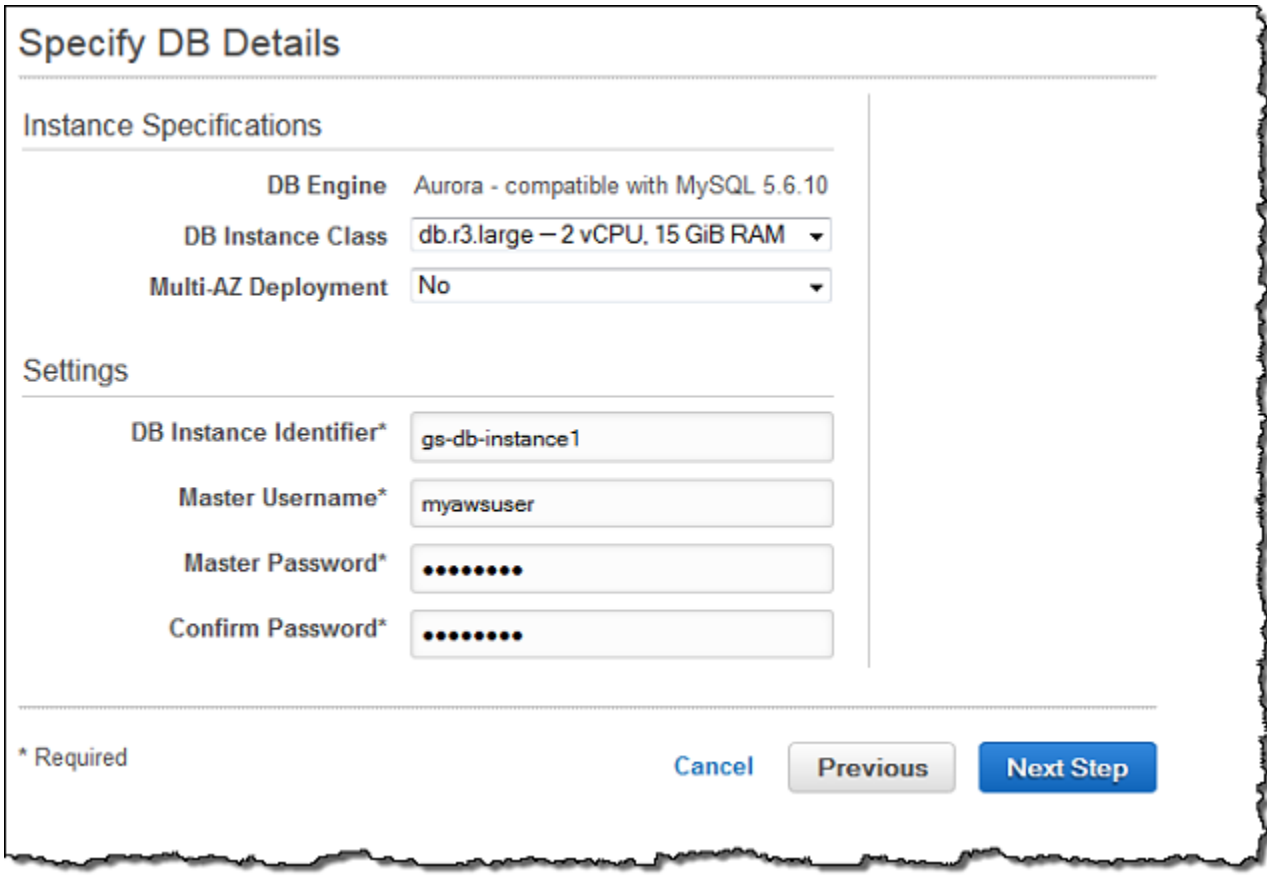


6. On the **Specify DB Details** page, specify your DB cluster information. The following table shows settings for a DB instance.

For This Option...	Do this
DB Instance Class	Select a DB instance class that defines the processing and memory requirements for each instance in the DB cluster. For more information about DB instance classes, see DB Instance Class (p. 92).

For This Option...	Do this
Multi-AZ Deployment	Determine if you want to create Aurora Replicas in other Availability Zones for failover support. If you select **Create Replica in Different Zone**, then Amazon RDS will create an Aurora Replica for you in your DB cluster in a different Availability Zone than the primary instance for your DB cluster. For more information about multiple Availability Zones, see Regions and Availability Zones (p. 97).
DB Instance Identifier	Type a name for the primary instance in your DB cluster. This identifier will be used in the endpoint address for the primary instance of your DB cluster. The DB instance identifier has the following constraints: • It must contain from 1 to 63 alphanumeric characters or hyphens. • Its first character must be a letter. • It cannot end with a hyphen or contain two consecutive hyphens. • It must be unique for all DB instances per AWS account, per AWS Region.
Master Username	Type a name using alphanumeric characters that you will use as the master user name to log on to your DB cluster.
Master Password	Type a password that contains from 8 to 41 printable ASCII characters (excluding /,", and @) for your master user password.

A typical **Specify DB Details** page looks like the following.



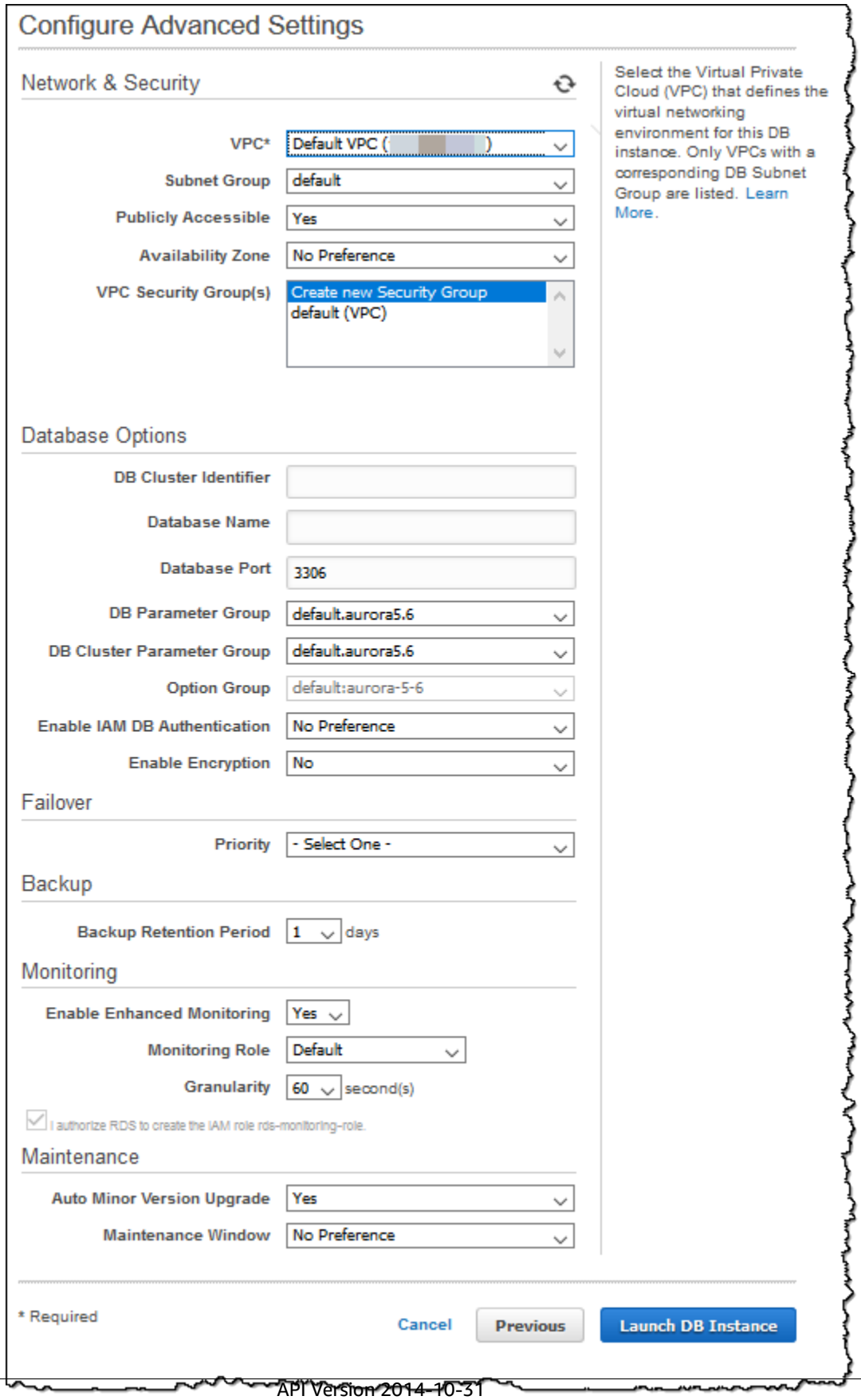
7. Confirm your master password and choose **Next**.
8. On the **Configure Advanced Settings** page, you can customize additional settings for your Aurora DB cluster. The following table shows the advanced settings for a DB cluster.

For This Option...	Do This
VPC	Select the VPC that will host the DB cluster. Select **Create a New VPC** to have Amazon RDS create a VPC for you. For more information, see DB Cluster Prerequisites (p. 437) earlier in this topic.
Subnet Group	Select the DB subnet group to use for the DB cluster. For more information, see DB Cluster Prerequisites (p. 437) earlier in this topic.
Publicly Accessible	Select `Yes` to give the DB cluster a public IP address; otherwise, select `No`. The instances in your DB cluster can be a mix of both public and private DB instances. For more information about hiding instances from public access, see Hiding a DB Instance in a VPC from the Internet (p. 401).
Availability Zone	Determine if you want to specify a particular Availability Zone. For more information about Availability Zones, see Regions and Availability Zones (p. 97).

For This Option...	Do This
VPC Security Group(s)	Select one or more VPC security groups to secure network access to the DB cluster. Select **Create a New VPC Security Group** to have Amazon RDS create a VPC security group for you. For more information, see DB Cluster Prerequisites (p. 437) earlier in this topic.
DB Cluster Identifier	Type a name for your DB cluster that is unique for your account in the region you selected. This identifier will be used in the cluster endpoint address for your DB cluster. For information on the cluster endpoint, see Aurora Endpoints (p. 431). The DB cluster identifier has the following constraints: • It must contain from 1 to 63 alphanumeric characters or hyphens. • Its first character must be a letter. • It cannot end with a hyphen or contain two consecutive hyphens. • It must be unique for all DB clusters per AWS account, per region.
Database Name	Type a name for your default database of up to 64 alpha-numeric characters. If you don't provide a name, Amazon RDS will not create a database on the DB cluster you are creating. To create additional databases, connect to the DB cluster and use the SQL command CREATE DATABASE. For more information about connecting to the DB cluster, see Connecting to an Amazon Aurora DB Cluster (p. 457).
Database Port	Specify the port that applications and utilities will use to access the database. Aurora MySQL DB clusters default to the default MySQL port, 3306, and Aurora PostgreSQL DB clusters default to the default PostgreSQL port, 5432. The firewalls at some companies block connections to these default ports. If your company firewall blocks the default port, choose another port for the new DB cluster.
DB Parameter Group	Select a parameter group. Aurora has a default parameter group you can use, or you can create your own parameter group. For more information about parameter groups, see Working with DB Parameter Groups (p. 170).
DB Cluster Parameter Group	Select a cluster parameter group. Aurora has a default cluster parameter group you can use, or you can create your own cluster parameter group. For more information about cluster parameter groups, see Working with DB Parameter Groups (p. 170).
Option Group	Select an option group. Aurora has a default option group you can use, or you can create your own option group. For more information about option groups, see Working with Option Groups (p. 153).

For This Option...	Do This
Copy Tags to Snapshots	Applies only to Aurora PostgreSQL. Select to specify that tags defined for this DB instance are copied to DB snapshots created from this DB instance. For more information, see Tagging Amazon RDS Resources (p. 129).
Enable IAM DB Authentication	Applies only to Aurora MySQL. Select **`Yes`** to enable IAM database authentication. For more information, see IAM Database Authentication for MySQL and Amazon Aurora (p. 360).
Enable Encryption	Select `Yes` to enable encryption at rest for this DB cluster. For more information, see Encrypting Amazon RDS Resources (p. 355).
Master Key	Only available if **Enable Encryption** is set to **`Yes`**. Select the master key to use for encrypting this DB cluster. For more information, see Encrypting Amazon RDS Resources (p. 355).
Priority	Choose a failover priority for the instance. If you don't select a value, the default is **tier-1**. This priority determines the order in which Aurora Replicas are promoted when recovering from a primary instance failure. For more information, see Fault Tolerance for an Aurora DB Cluster (p. 468).
Backup Retention Period	Select the length of time, from 1 to 35 days, that Aurora will retain backup copies of the database. Backup copies can be used for point-in-time restores (PITR) of your database down to the second.
Enable Enhanced Monitoring	Choose **`Yes`** to enable gathering metrics in real time for the operating system that your DB cluster runs on. For more information, see Enhanced Monitoring (p. 258).
Monitoring Role	Only available if **Enable Enhanced Monitoring** is set to **`Yes`**. Choose the IAM role that you created to permit Amazon RDS to communicate with Amazon CloudWatch Logs for you, or choose **`Default`** to have RDS create a role for you named `rds-monitoring-role`. For more information, see Enhanced Monitoring (p. 258).
Granularity	Only available if **Enable Enhanced Monitoring** is set to **`Yes`**. Set the interval, in seconds, between when metrics are collected for your DB cluster.
Auto Minor Version Upgrade	Select **`Yes`** if you want to enable your Aurora DB cluster to receive minor MySQL DB Engine version upgrades automatically when they become available. The **Auto Minor Version Upgrade** option only applies to upgrades to MySQL minor engine versions for your Amazon Aurora DB cluster. It doesn't apply to regular patches applied to maintain system stability.
Maintenance Window	Select the weekly time range during which system maintenance can occur.

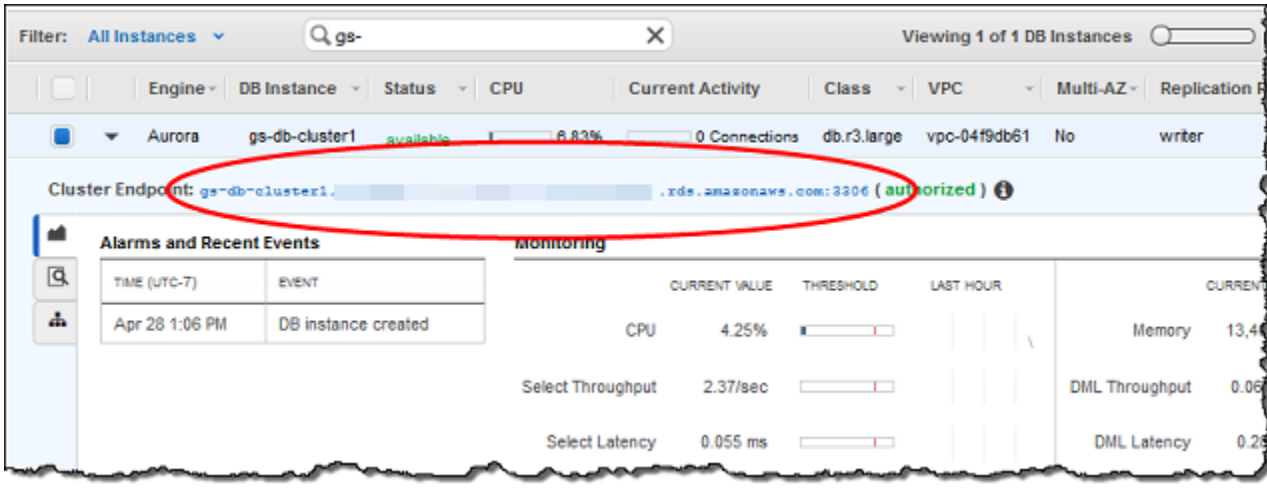
A typical **Configure Advanced Settings** page looks like the following.


Configure Advanced Settings
Network & Security
VPC*
Default VPC (
)
Subnet Group
default
Publicly Accessible
Yes
Availability Zone
No Preference
VPC Security Group(s)
Create new Security Group
default (VPC)
Select the Virtual Private Cloud (VPC) that defines the virtual networking environment for this DB instance. Only VPCs with a corresponding DB Subnet Group are listed. Learn More.
Database Options
DB Cluster Identifier
Database Name
Database Port
3306
DB Parameter Group
default.aurora5.6
DB Cluster Parameter Group
default.aurora5.6
Option Group
default:aurora-5-6
Enable IAM DB Authentication
No Preference
Enable Encryption
No
Failover
Priority
- Select One -
Backup
Backup Retention Period
1
days
Monitoring
Enable Enhanced Monitoring
Yes
Monitoring Role
Default
Granularity
60
second(s)
I authorize RDS to create the IAM role rds-monitoring-role.
Maintenance
Auto Minor Version Upgrade
Yes
Maintenance Window
No Preference
* Required
Cancel
Previous
Launch DB Instance

9. Click **Launch DB Instance** to launch your Aurora DB instance, and then click **Close** to close the wizard.

   On the Amazon RDS console, the new DB instance appears in the list of DB instances. The DB instance will have a status of **creating** until the DB instance is created and ready for use. When the state changes to available, you can connect to the primary instance for your DB cluster. Depending on the DB instance class and store allocated, it can take several minutes for the new instance to be available.

   To view the newly created cluster, choose the **Clusters** view in the Amazon RDS console. For more information, see Viewing an Amazon Aurora DB Cluster (p. 461).



   Note the port and the endpoint of the cluster. Use the endpoint and port of the cluster in your JDBC and ODBC connection strings for any application that performs write or read operations.

Creating an Aurora Replica Using the Console

After creating the primary instance for your Aurora DB cluster, you can add up to 15 Aurora Replicas by using the Create Aurora Replica wizard.

> **Note**
> Amazon Aurora also supports replication with an external database, or an RDS DB instance. When using Amazon Aurora, your RDS DB instance must be in the same region. For more information, see Replication with Amazon Aurora (p. 478).

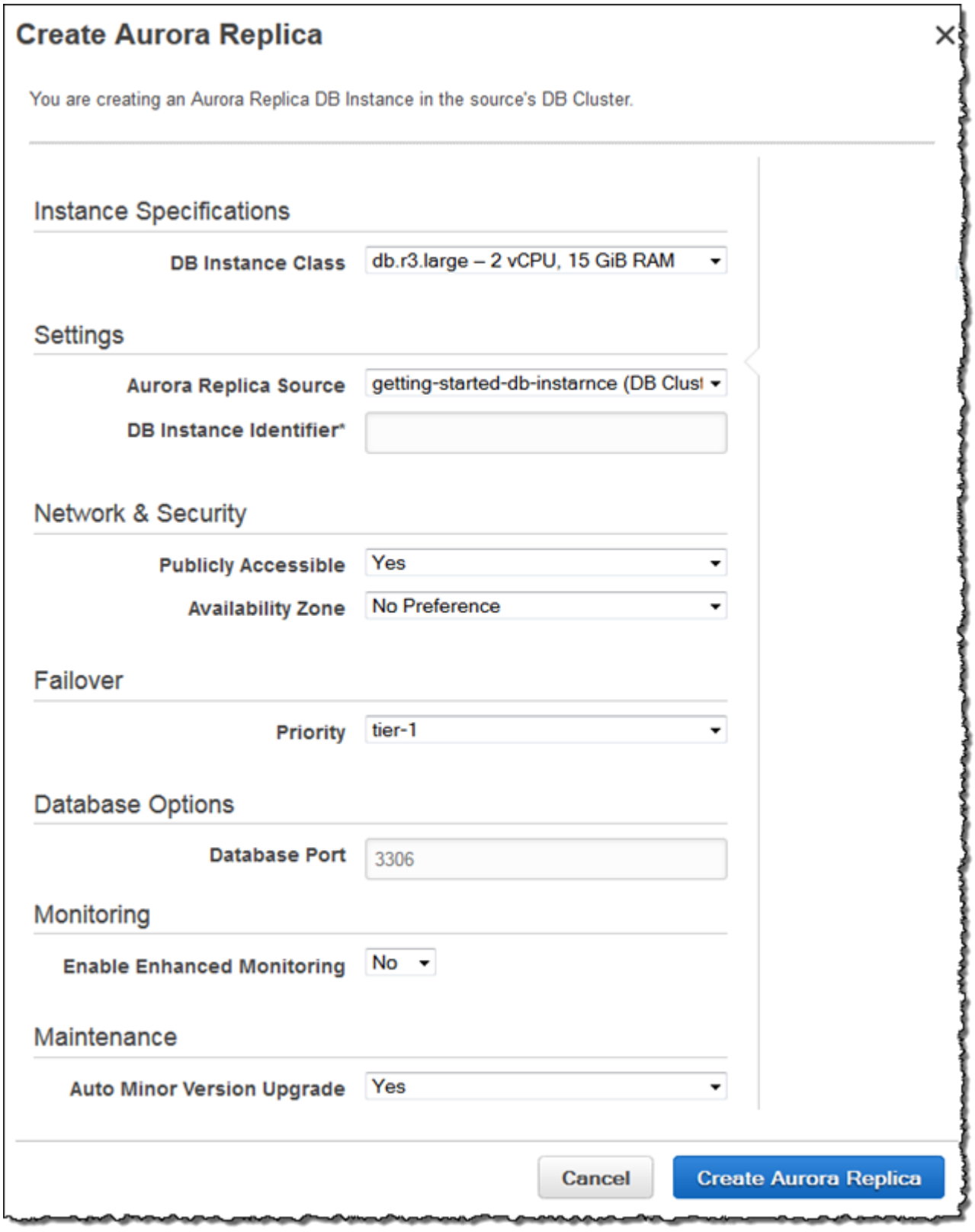
To create an Aurora Replica by using the AWS Management Console

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Instances**.
3. Select the check box to the left of the primary instance for your Aurora DB cluster.
4. Choose **Instance Actions**, and then choose **Create Aurora Replica**.
5. On the Create Aurora Replica page, specify options for your Aurora Replica. The following table shows settings for an Aurora Replica.

For This Option...	Do This
DB Instance Class	Select a DB instance class that defines the processing and memory requirements for the Aurora Replica. For

For This Option...	Do This
	more information about DB instance class options, see DB Instance Class (p. 92).
Aurora Replica Source	Select the identifier of the primary instance to create an Aurora Replica for.
DB Instance Identifier	Type a name for the instance that is unique for your account in the region you selected. You might choose to add some intelligence to the name such as including the region and DB engine you selected, for example **`aurora-read-instance1`**.
Publicly Accessible	Select `Yes` to give the Aurora Replica a public IP address; otherwise, select `No`. For more information about hiding Aurora Replicas from public access, see Hiding a DB Instance in a VPC from the Internet (p. 401).
Availability Zone	Determine if you want to specify a particular Availability Zone. The list includes only those Availability Zones that are mapped by the DB subnet group you specified earlier. For more information about Availability Zones, see Regions and Availability Zones (p. 97).
Priority	Choose a failover priority for the instance. If you don't select a value, the default is **tier-1**. This priority determines the order in which Aurora Replicas are promoted when recovering from a primary instance failure. For more information, see Fault Tolerance for an Aurora DB Cluster (p. 468).
Database Port	The port for an Aurora Replica is the same as the port for the DB cluster.
Auto Minor Version Upgrade	Select `Yes` if you want to enable your Aurora Replica to receive minor Aurora DB engine version upgrades automatically when they become available. The **Auto Minor Version Upgrade** option only applies to upgrades to MySQL minor engine versions for your Amazon Aurora DB cluster. It doesn't apply to regular patches applied to maintain system stability.

A typical **Create Aurora Replica** page looks like the following.



6. Click **Create Aurora Replica** to create the Aurora Replica.

Note the endpoint of the Aurora Replica. Use the endpoint of the Aurora Replica in your JDBC and ODBC connection strings for any application that performs only read operations.

CLI

Note
Before you can create an Aurora DB cluster using the AWS CLI, you must fulfill the required prerequisites, such as creating a VPC and an RDS DB subnet group. For more information, see DB Cluster Prerequisites (p. 437).

To launch an Aurora MySQL DB cluster using the AWS CLI

1. Identify the DB subnet group and VPC security group ID for your new DB cluster, and then call the create-db-cluster AWS CLI command to create the Aurora MySQL DB cluster.

   For example, the following command creates a new DB cluster named `sample-cluster`.

   For Linux, OS X, or Unix:

   ```
   aws rds create-db-cluster --db-cluster-identifier sample-cluster --engine aurora \
        --master-username user-name --master-user-password password \
        --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2
   ```

   For Windows:

   ```
   aws rds create-db-cluster --db-cluster-identifier sample-cluster --engine aurora ^
       --master-username user-name --master-user-password password ^
       --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2
   ```

2. If you use the console to create a DB cluster, then Amazon RDS automatically creates the primary instance (writer) for your DB cluster. If you use the AWS CLI to create a DB cluster, you must explicitly create the primary instance for your DB cluster. The primary instance is the first instance that is created in a DB cluster.

   Call the create-db-instance AWS CLI command to create the primary instance for your DB cluster. Include the name of the DB cluster as the `--db-cluster-identifier` parameter value.

   For Linux, OS X, or Unix:

   ```
   aws rds create-db-instance --db-instance-identifier sample-instance \
        --db-cluster-identifier sample-cluster --engine aurora --db-instance-class
    db.r3.large
   ```

   For Windows:

   ```
   aws rds create-db-instance --db-instance-identifier sample-instance ^
        --db-cluster-identifier sample-cluster --engine aurora --db-instance-class
    db.r3.large
   ```

To launch an Aurora PostgreSQL DB cluster using the AWS CLI

1. Identify the DB subnet group and VPC security group ID for your new DB cluster, and then call the create-db-cluster AWS CLI command to create the Aurora PostgreSQL DB cluster.

   For example, the following command creates a new DB cluster named `sample-cluster`.

   For Linux, OS X, or Unix:

```
aws rds create-db-cluster --db-cluster-identifier sample-cluster --engine aurora-
postgresql \
     --master-username user-name --master-user-password password \
     --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2
```

For Windows:

```
aws rds create-db-cluster --db-cluster-identifier sample-cluster --engine aurora-
postgresql ^
    --master-username user-name --master-user-password password ^
    --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2
```

2. If you use the console to create a DB cluster, then Amazon RDS automatically creates the primary instance (writer) for your DB cluster. If you use the AWS CLI to create a DB cluster, you must explicitly create the primary instance for your DB cluster. The primary instance is the first instance that is created in a DB cluster.

   Call the create-db-instance AWS CLI command to create the primary instance for your DB cluster. Include the name of the DB cluster as the `--db-cluster-identifier` parameter value.

   For Linux, OS X, or Unix:

```
aws rds create-db-instance --db-instance-identifier sample-instance \
     --db-cluster-identifier sample-cluster --engine aurora-postgresql --db-instance-
class db.r4.large
```

   For Windows:

```
aws rds create-db-instance --db-instance-identifier sample-instance ^
     --db-cluster-identifier sample-cluster --engine aurora-postgresql --db-instance-
class db.r4.large
```

To create an Aurora Replica in a DB cluster using the AWS CLI

After you create the primary instance for a DB cluster, you can create up to 15 Aurora Replicas in your DB cluster to support read-only queries.

We recommend that you distribute the primary instance and Aurora Replicas in your DB cluster over multiple Availabilty Zones to improve the availability of your DB cluster. For more information, see Availability (p. 431).

Call the create-db-instance AWS CLI command to create an Aurora Replica in your DB cluster. Include the name of the DB cluster as the `--db-cluster-identifier` parameter value. You can optionally specify an Availability Zone for the Aurora Replica using the `--availability-zone` parameter, as shown in the following example.

For Linux, OS X, or Unix:

```
aws rds create-db-instance --db-instance-identifier sample-instance-us-west-2a \
    --db-cluster-identifier sample-cluster --engine aurora --db-instance-class db.r3.large
 \
    --availability-zone us-west-2a
```

For Windows:

```
aws rds create-db-instance --db-instance-identifier sample-instance-us-west-2a ^
```

```
    --db-cluster-identifier sample-cluster --engine aurora --db-instance-class db.r3.large
^
    --availability-zone us-west-2a
```

How to Create a VPC for Use with Amazon Aurora

The following sections discuss how to create a VPC for use with Amazon Aurora.

> **Note**
> For a helpful and detailed guide on connecting to an Amazon Aurora DB cluster, you can see RDS Aurora Connectivity.

Create a VPC and Subnets

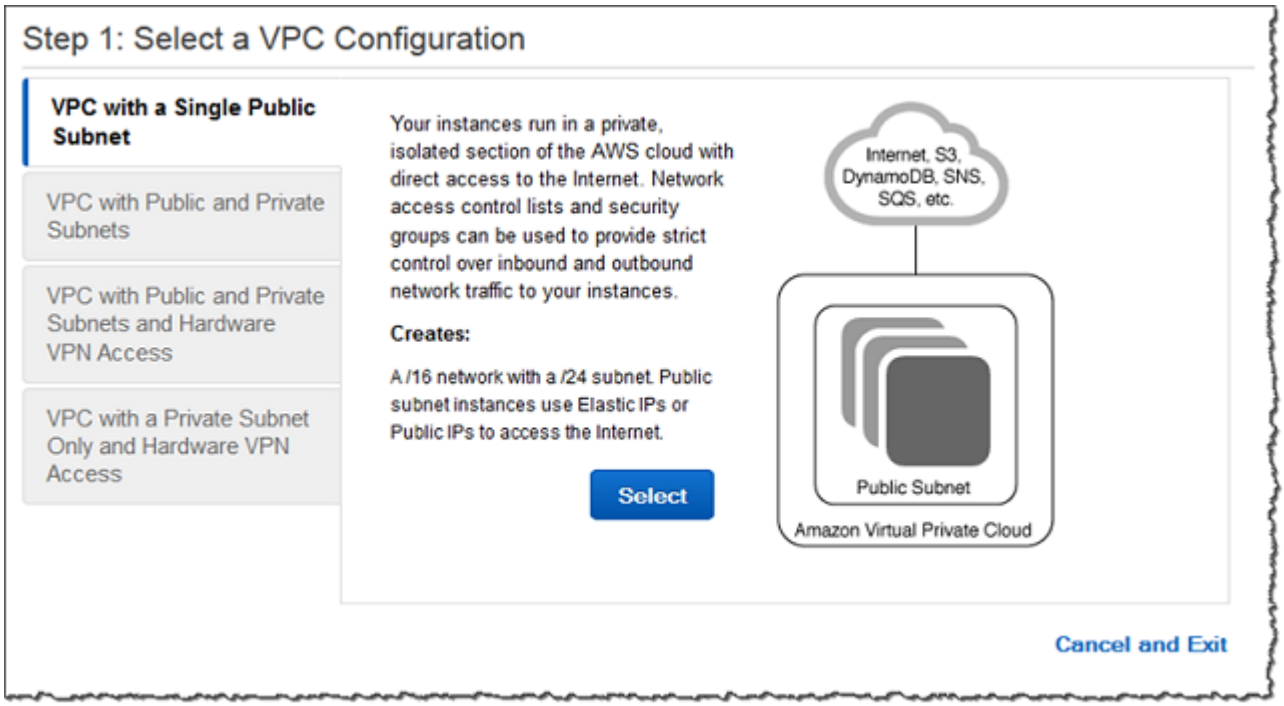
You can only create an Amazon Aurora DB cluster in an Amazon Virtual Private Cloud (VPC) that has at least one subnet in at least two of the Availability Zones in the region where you want to deploy your DB cluster. You can create an Aurora DB cluster in the default VPC for your AWS account, or you can create a user-defined VPC. For information, see Amazon Virtual Private Cloud (VPCs) and Amazon RDS (p. 390).

Amazon RDS will, optionally, create a VPC and subnet group for you to use with your Amazon Aurora DB cluster. This can be helpful if you have never created a VPC, or if you would like to create a new VPC that is separate from your other VPCs. If you want Amazon RDS to create a VPC and subnet group for you, then skip this procedure and see Create a DB Cluster (p. 10).

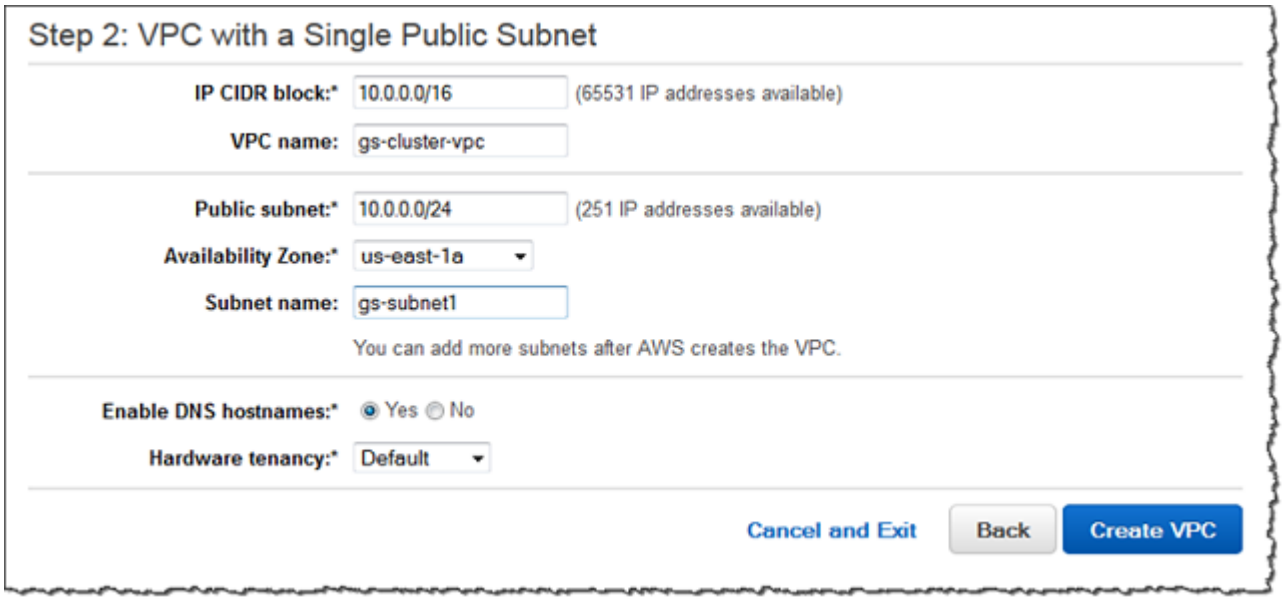
> **Note**
> All VPC and EC2 resources that you use with your Aurora DB cluster must be in one of the following regions: US East (N. Virginia), US East (Ohio), US West (Oregon), Asia Pacific (Mumbai), Asia Pacific (Seoul), Asia Pacific (Sydney), Asia Pacific (Tokyo), Canada (Central), EU (Ireland), EU (London).

To create a VPC for use with an Aurora DB cluster

1. Sign in to the AWS Management Console and open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the top-right corner of the AWS Management Console, select the region to create your VPC in. This example uses the US East (Ohio) region.
3. In the upper-left corner, click **VPC Dashboard**. Click **Start VPC Wizard** to begin creating a VPC.
4. In the Create VPC wizard, click **VPC with a Single Public Subnet**. Click **Select**.



5. Set the following values in the **Create VPC** panel:
   - **IP CIDR block:** `10.0.0.0/16`
   - **VPC name:** `gs-cluster-vpc`
   - **Public subnet:** `10.0.0.0/24`
   - **Availability Zone:** `us-east-2a`
   - **Subnet name:** `gs-subnet1`
   - **Enable DNS hostnames:** `Yes`
   - **Hardware tenancy:** `Default`

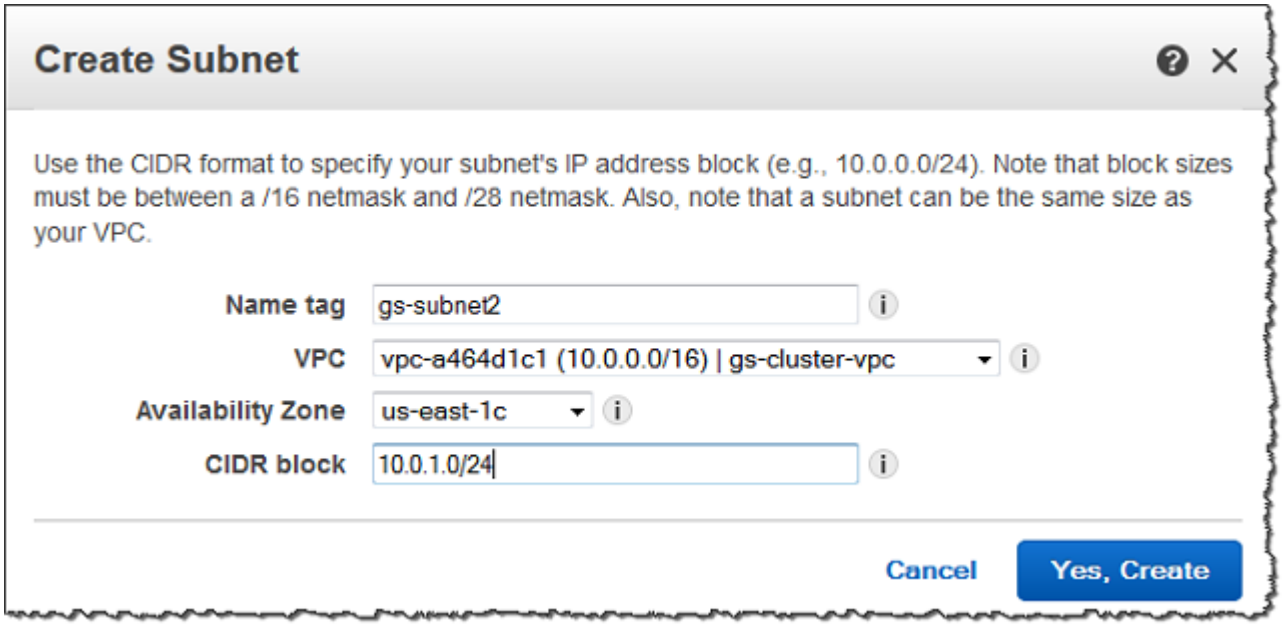


6. Click **Create VPC**.

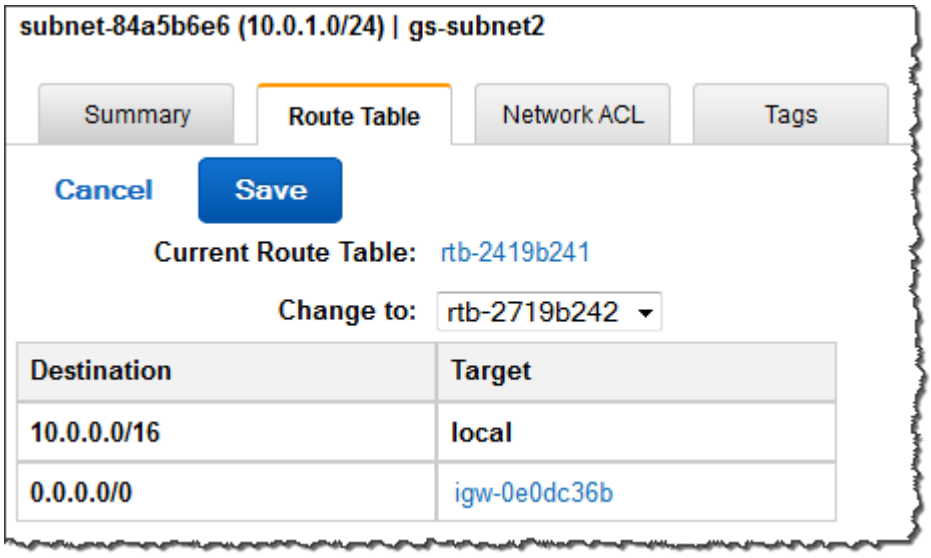
7. When your VPC has been created, click **Close** on the notification page.

To create additional subnets

1. To add the second to your VPC, in the VPC Dashboard click **Subnets**, and then click **Create Subnet**. An Amazon Aurora DB cluster requires at least two VPC subnets.
2. Set the following values in the **Create Subnet** panel:
   - **Name tag:** `gs-subnet2`
   - **VPC:** Select the VPC that you created in the previous step, for example: `vpc-a464d1c1 (10.0.0.0/16) | gs-cluster-vpc`.
   - **Availability Zone:** `us-east-2c`
   - **CIDR block:** `10.0.1.0/24`



3. Click **Yes Create**.
4. To ensure that the second subnet that you created uses the same route table as the first subnet, in the VPC Dashboard, click **Subnets**, and then select the first subnet that was created for the VPC, `gs-subnet1`. Click the **Route Table** tab, and note the **Current Route Table**, for example: `rtb-2719b242`.
5. In the list of subnets, select the second subnet, `gs-subnet2`. Select the **Route Table** tab, and then click **Edit**. In the **Change to** list, select the route table from the previous step, for example: `rtb-2719b242`. Click **Save** to save your selection.



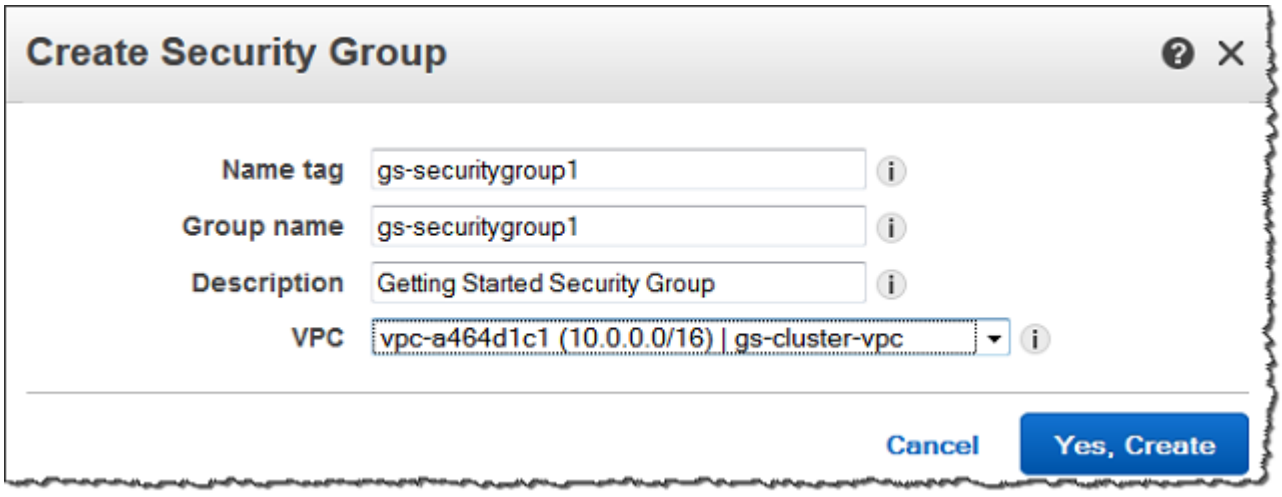
Create a Security Group and Add Inbound Rules

After you've created your VPC and subnets, the next step is to create a security group and add inbound rules.

To create a security group

The last step in creating a VPC for use with your Amazon Aurora DB cluster is to create a VPC security group, which will identify which network addresses and protocols are allowed to access instances in your VPC.

1. In the VPC Dashboard, click **Security Groups**, and then click **Create Security Group**.
2. Set the following values in the **Create Security Group** panel:
   - **Name tag:** `gs-securitygroup1`
   - **Group name:** `gs-securitygroup1`
   - **Description:** `Getting Started Security Group`
   - **VPC:** Select the VPC that you created earlier, for example: `vpc-a464d1c1 (10.0.0.0/16) | gs-cluster-vpc`.



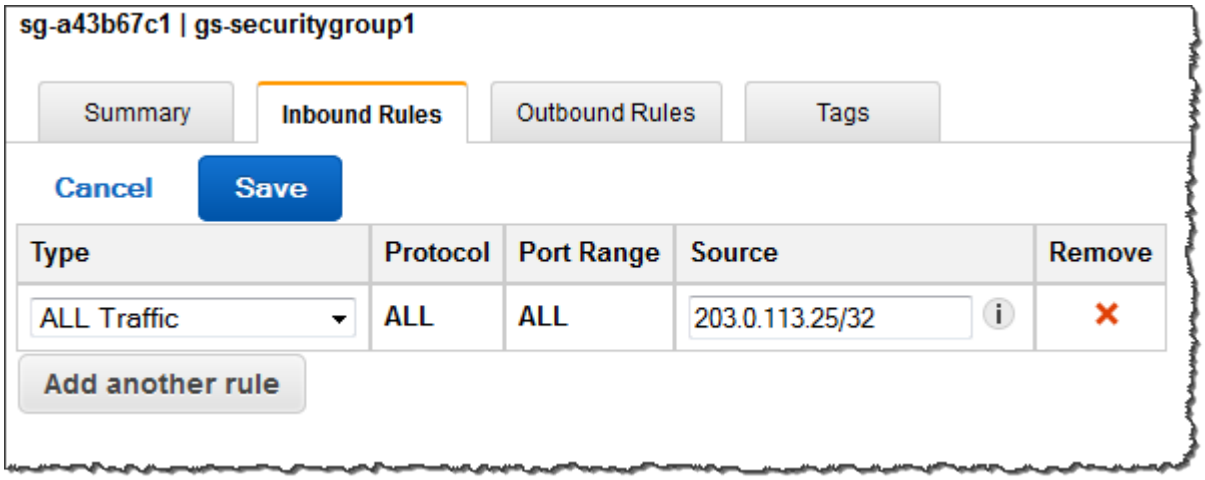
3. Click **Yes, Create** to create the security group.

To add inbound rules to the security group

To connect to your Aurora DB instance, you will need to add an inbound rule to your VPC security group that allows inbound traffic to connect.

1. Determine the IP address that you will be using to connect to the Aurora cluster. You can use the service at http://checkip.amazonaws.com to determine your public IP address. If you are connecting through an ISP or from behind your firewall without a static IP address, you need to find out the range of IP addresses used by client computers.

   Warning
   If you use `0.0.0.0/0`, you enable all IP addresses to access your DB cluster. This is acceptable for a short time in a test environment, but it's unsafe for production environments. In production, you'll authorize only a specific IP address or range of addresses to access your DB cluster.
2. In the VPC Dashboard, click **Security Groups**, and then select the `gs-securitygroup1` security group that you created in the previous procedure.
3. Select the **Inbound Rules** tab, and then click the **Edit** button.
4. Set the following values for your new inbound rule:
   - **Type:** `All Traffic`
   - **Source:** The IP address or range from the previous step, for example `203.0.113.25/32`.



5. Click **Save** to save your settings.

Create an RDS Subnet Group

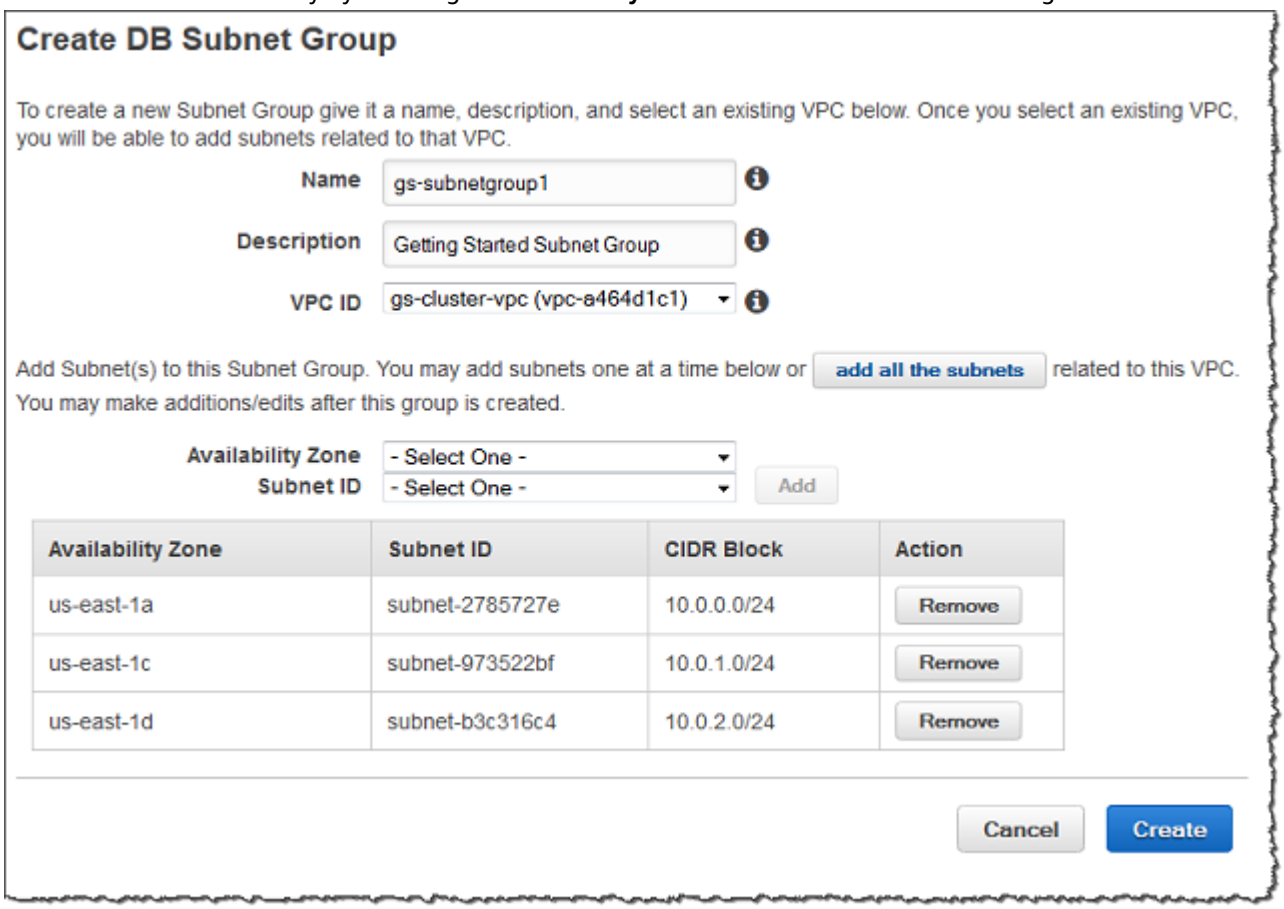
The last thing that you need before you can create an Aurora DB cluster is a DB subnet group. Your RDS DB subnet group identifies the subnets that your DB cluster will use from the VPC that you created in the previous steps. Your DB subnet group must include at least one subnet in at least two of the Availability Zones in the region where you want to deploy your DB cluster.

To create a DB subnet group for use with your Aurora DB cluster

1. Open the Amazon Aurora console at https://console.aws.amazon.com/rds.
2. Select **Subnet Groups**, and then click **Create DB Subnet Group**.
3. Set the following values for your new DB subnet group:
   - **Name:** `gs-subnetgroup1`

- **Description:** `Getting Started Subnet Group`
- **VPC ID:** Select the VPC that you created in the previous procedure, for example, `vpc-a464d1c1`.

4. Click **add all the subnets** to add the subnets for the VPC that you created in earlier steps. You can also add each subnet individually by selecting the **Availability Zone** and the **Subnet ID** and clicking **Add**.



5. Click **Yes, Create** to create the subnet group.

Related Topics

- Amazon Aurora on Amazon RDS (p. 428)

Connecting to an Amazon Aurora DB Cluster

You can connect to a DB instance in an Amazon Aurora DB cluster using the same tools that you use to connect to a MySQL or PostgreSQL database, including using the same public key for Secure Sockets Layer (SSL) connections. You can use the endpoint and port information from the primary instance or Aurora Replicas in your Aurora DB cluster in the connection string of any script, utility, or application that connects to a MySQL or PostgreSQL DB instance. In the connection string, specify the DNS address from the primary instance or Aurora Replica endpoint as the host parameter, and specify the port number from the endpoint as the port parameter.

Connecting to an Amazon Aurora MySQL DB Cluster

To authenticate to your Aurora MySQL DB cluster, you can use either MySQL username and password authentication or IAM database authentication.

- To learn how to authenticate to MySQL using username and password authentication, see User Account Management in the MySQL documentation.
- To learn how to authenticate to MySQL using IAM database authentication, see IAM Database Authentication for MySQL and Amazon Aurora (p. 360).

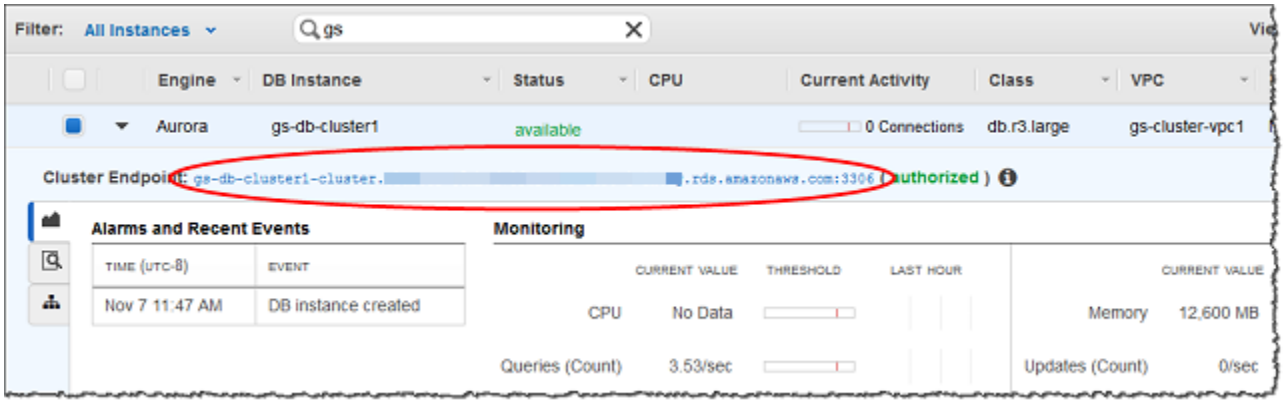
Once you have a connection to your Amazon Aurora DB cluster, you can execute any SQL command that is compatible with MySQL version 5.6. For more information about MySQL 5.6 SQL syntax, see the MySQL 5.6 Reference Manual.

> **Note**
> For a helpful and detailed guide on connecting to an Amazon Aurora MySQL DB cluster, you can see RDS Aurora Connectivity.

In the details view for your DB cluster you will find the cluster endpoint, which you can use in your MySQL connection string. The endpoint is made up of the domain name and port for your DB cluster. For example, if an endpoint value is `mycluster.cluster-123456789012.us-east-1.rds.amazonaws.com:3306`, then you specify the following values in a MySQL connection string:

- For host or host name, specify `mycluster.cluster-123456789012.us-east-1.rds.amazonaws.com`
- For port, specify `3306` or the port value you used when you created the DB cluster

The cluster endpoint connects you to the primary instance for the DB cluster. You can perform both read and write operations using the cluster endpoint. Your DB cluster can also have up to 15 Aurora Replicas that support read-only access to the data in your DB cluster. The primary instance and each Aurora Replica each have a unique endpoint that is independent of the cluster endpoint and allows you to connect to a specific DB instance in the cluster directly. The cluster endpoint will always point to the primary instance. If the primary instance fails and is replaced, then the cluster endpoint will point to the new primary instance.



Connection Utilities

- **Command line** – You can connect to an Amazon Aurora DB cluster by using tools like the MySQL command line utility. For more information on using the MySQL utility, see mysql - The MySQL Command Line Tool in the MySQL documentation.

- **GUI** – You can use the MySQL Workbench utility to connect by using a UI interface. For more information, see the Download MySQL Workbench page.
- **Applications** – You can use the MariaDB Connector/J utility to connect your applications to your Aurora DB cluster. For more information, see the MariaDB Connector/J download page.

A GUI-based application you can use to connect is MySQL Workbench. For more information, see the Download MySQL Workbench page.

You can use SSL encryption on connections to an Amazon Aurora DB instance. For information, see SSL Support for MySQL DB Instances (p. 826).

> **Note**
> Because an Amazon Aurora DB cluster can only be created in an Amazon Virtual Private Cloud (VPC), connections to an Amazon Aurora DB cluster from AWS instances that are not in a VPC have been required to use the public endpoint address of the Amazon Aurora DB cluster. However, you can now communicate with an EC2 instance that is not in a VPC and an Amazon Aurora DB cluster using ClassicLink. For more information, see A DB Instance in a VPC Accessed by an EC2 Instance Not in a VPC (p. 395).

Connecting with SSL

To connect using SSL, use the MySQL utility as described in the following procedure. If you are using IAM database authentication, you must use an SSL connection. For information, see IAM Database Authentication for MySQL and Amazon Aurora (p. 360).

> **Note**
> In order to connect to the cluster endpoint using SSL, your client connection utility must support Subject Alternative Names (SAN). If your client connection utility doesn't support SAN, you can connect directly to the instances in your Aurora DB cluster. For more information on Aurora endpoints, see Aurora Endpoints (p. 431).

To connect to a DB cluster with SSL using the MySQL utility

1. Download the public key for the Amazon RDS signing certificate from https://s3.amazonaws.com/rds-downloads/rds-combined-ca-bundle.pem. Note that this will download a file named `rds-combined-ca-bundle.pem`.
2. Type the following command at a command prompt to connect to the primary instance of a DB cluster with SSL using the MySQL utility. For the -h parameter, substitute the endpoint DNS name for your primary instance. For the --ssl_ca parameter, substitute the SSL certificate file name as appropriate. Type the master user password when prompted.

   ```
   mysql -h mycluster-primary.123456789012.us-east-1.rds.amazonaws.com --ssl-ca=[full path]rds-combined-ca-bundle.pem --ssl-verify-server-cert
   ```

You will see output similar to the following:

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 350
Server version: 5.6.10-log MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

For general instructions on constructing Amazon RDS MySQL connection strings and finding the public key for SSL connections, see Connecting to a DB Instance Running the MySQL Database Engine (p. 840).

Connecting to an Amazon Aurora PostgreSQL DB Cluster

You can connect to a DB instance in your Amazon Aurora PostgreSQL DB cluster using the same tools that you use to connect to a PostgreSQL database. As part of this, you use the same public key for Secure Sockets Layer (SSL) connections. You can use the endpoint and port information from the primary instance or Aurora Replicas in your Aurora PostgreSQL DB cluster in the connection string of any script, utility, or application that connects to a PostgreSQL DB instance. In the connection string, specify the DNS address from the primary instance or Aurora Replica endpoint as the host parameter, and specify the port number from the endpoint as the port parameter.

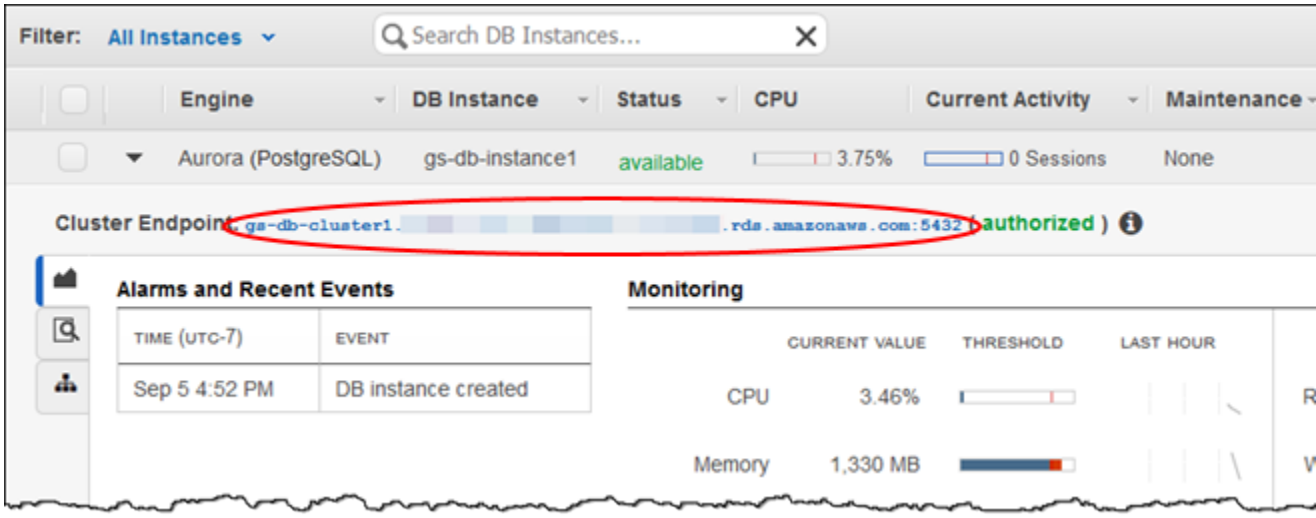
Once you have a connection to a DB instance in your Amazon Aurora PostgreSQL DB cluster, you can run any SQL command that is compatible with PostgreSQL version 9.6.3.

For a helpful and detailed guide on connecting to an Amazon Aurora DB cluster, see RDS Aurora Connectivity.

In the details view for your Aurora PostgreSQL DB cluster you can find the cluster endpoint. You use this endpoint in your PostgreSQL connection string. The endpoint is made up of the domain name and port for your DB cluster. For example, if an endpoint value is `mycluster.cluster-123456789012.us-east-1.rds.amazonaws.com:5432`, then you specify the following values in a PostgreSQL connection string:

- For host or host name, specify `mycluster.cluster-123456789012.us-east-1.rds.amazonaws.com`
- For port, specify `5432` or the port value you used when you created the DB cluster

The cluster endpoint connects you to the primary instance for the DB cluster. You can perform both read and write operations using the cluster endpoint. Your DB cluster can also have up to 15 Aurora Replicas that support read-only access to the data in your DB cluster. The primary instance and each Aurora Replica each has a unique endpoint that is independent of the cluster endpoint. This unique endpoint allows you to connect to a specific DB instance in the cluster directly. The cluster endpoint always points to the primary instance. If the primary instance fails and is replaced, the cluster endpoint points to the new primary instance.



Connection Utilities

- **Command line** – You can connect to an Amazon Aurora PostgreSQL DB instance by using tools like `psql`, the PostgreSQL interactive terminal. For more information on using the PostgreSQL interactive terminal, see psql in the PostgreSQL documentation.
- **GUI** – You can use the pgAdmin utility to connect to an PostgreSQL DB instance by using a UI interface. For more information, see the Download page from the pgAdmin website.
- **Applications** – You can use the PostgreSQL JDBC driver to connect your applications to your PostgreSQL DB instance. For more information, see the Download page from the PostgreSQL JDBC driver website.

> **Note**
> Because an Amazon Aurora PostgreSQL DB cluster can only be created in an Amazon Virtual Private Cloud (VPC), connections to an Aurora PostgreSQL DB cluster from AWS instances that are not in a VPC have been required to use the public endpoint address of the Aurora PostgreSQL DB cluster. However, you can now communicate with an Amazon EC2 instance that is not in a VPC and an Aurora PostgreSQL DB cluster using ClassicLink. For more information, see A DB Instance in a VPC Accessed by an EC2 Instance Not in a VPC (p. 395).

Troubleshooting Aurora Connection Failures

> **Note**
> For a helpful and detailed guide on connecting to an Amazon Aurora DB cluster, you can see RDS Aurora Connectivity.

Common causes of connection failures to a new Aurora DB cluster are as follows:

- The DB cluster was created using a VPC that does not allow connections from your device. To fix this failure, modify the VPC to allow connections from your device, or create a new VPC for your DB cluster that allows connections from your device. For an example, see Create a VPC and Subnets (p. 452).
- The DB cluster was created using the default port, and your company has firewall rules blocking connections to that port from devices in your company network. To fix this failure, recreate the instance with a different port.
- If you are using IAM database authentication, you might need to configure IAM database authentication. For information, see IAM Database Authentication for MySQL and Amazon Aurora (p. 360).

Related Topics

- Amazon Aurora on Amazon RDS (p. 428)

Viewing an Amazon Aurora DB Cluster

You have several options for viewing information about your Amazon Aurora DB clusters and the DB instances in your DB clusters.

- You can view DB clusters and DB instances in the Amazon RDS console by using the **Clusters** view.
- You can get DB cluster and DB instance information using the AWS Command Line Interface (AWS CLI).
- You can get DB cluster and DB instance information using the Amazon RDS API.

AWS Management Console

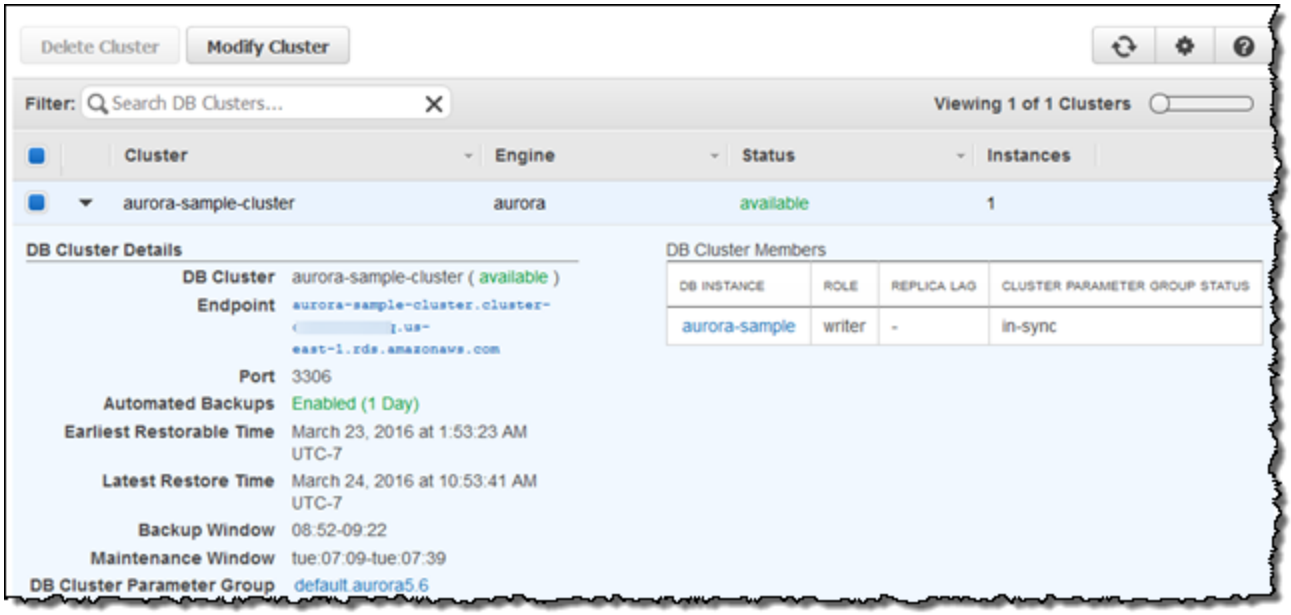
The Amazon RDS console has two sections where you can see information about a DB cluster. You can see details about a DB cluster by using the **Clusters** view and you can see details about DB instances that are members of an Amazon Aurora DB cluster by using the **Instances** view.

The **Clusters** view lists all of the DB clusters for your AWS account. When you select a DB cluster, you see both information about the DB cluster and also a list of the DB instances that are members of that DB cluster. You can choose the identifier for a DB instance in the list to go directly to the details page for that DB instance in the RDS console.

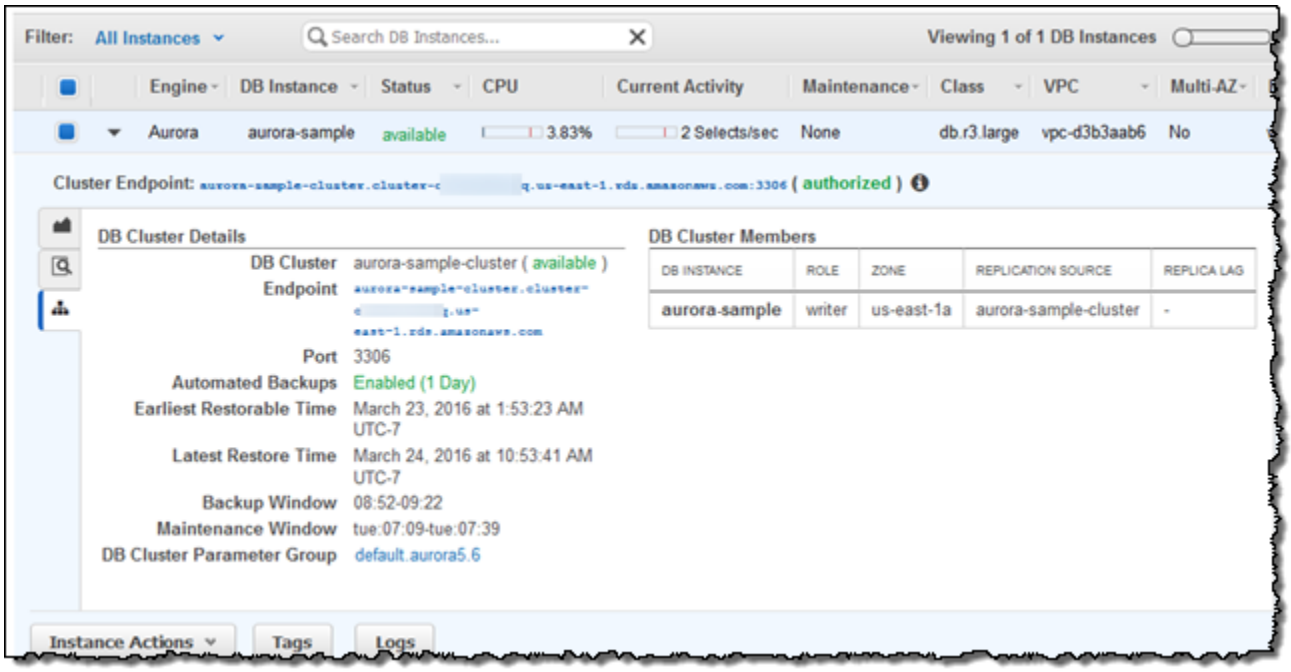
You can modify your DB cluster by using the **Clusters** view of the RDS console. To modify a DB cluster, choose the DB cluster from the **Clusters** list and choose **Modify Cluster**.

To modify a DB instance that is a member of a DB cluster, use the **Instances** view.

For example, the following screenshot shows the details page for the DB cluster named `aurora-sample-cluster`. The DB cluster has one DB instance shown in the **DB Cluster Members** list, named `aurora-sample`. This instance is the primary instance for the DB cluster.



If you click the link for the `aurora-sample` DB instance identifier, the Amazon RDS console takes you to the **Instances** view for the `aurora-sample` DB instance as shown in the following screenshot.



CLI

To view DB cluster information by using the AWS CLI, use the describe-db-clusters command. For example, the following AWS CLI command lists the DB cluster information for all of the DB clusters in the `us-east-1` region for the configured AWS account.

```
aws rds describe-db-clusters --region us-east-1
```

The command returns the following output if your AWS CLI is configured for JSON output.

```
{
    "DBClusters": [
        {
            "Status": "available",
            "Engine": "aurora",
            "Endpoint": "sample-cluster1.cluster-123456789012.us-east-1.rds.amazonaws.com"
            "AllocatedStorage": 1,
            "DBClusterIdentifier": "sample-cluster1",
            "MasterUsername": "mymasteruser",
            "EarliestRestorableTime": "2016-03-30T03:35:42.563Z",
            "DBClusterMembers": [
                {
                    "IsClusterWriter": false,
                    "DBClusterParameterGroupStatus": "in-sync",
                    "DBInstanceIdentifier": "sample-replica"
                },
                {
                    "IsClusterWriter": true,
                    "DBClusterParameterGroupStatus": "in-sync",
                    "DBInstanceIdentifier": "sample-primary"
                }
            ],
            "Port": 3306,
            "PreferredBackupWindow": "03:34-04:04",
            "VpcSecurityGroups": [
```

```
                {
                    "Status": "active",
                    "VpcSecurityGroupId": "sg-ddb65fec"
                }
            ],
            "DBSubnetGroup": "default",
            "StorageEncrypted": false,
            "DatabaseName": "sample",
            "EngineVersion": "5.6.10a",
            "DBClusterParameterGroup": "default.aurora5.6",
            "BackupRetentionPeriod": 1,
            "AvailabilityZones": [
                "us-east-1b",
                "us-east-1c",
                "us-east-1d"
            ],
            "LatestRestorableTime": "2016-03-31T20:06:08.903Z",
            "PreferredMaintenanceWindow": "wed:08:15-wed:08:45"
        },
        {
            "Status": "available",
            "Engine": "aurora",
            "Endpoint": "aurora-sample.cluster-123456789012.us-east-1.rds.amazonaws.com",
            "AllocatedStorage": 1,
            "DBClusterIdentifier": "aurora-sample-cluster",
            "MasterUsername": "mymasteruser",
            "EarliestRestorableTime": "2016-03-30T10:21:34.826Z",
            "DBClusterMembers": [
                {
                    "IsClusterWriter": false,
                    "DBClusterParameterGroupStatus": "in-sync",
                    "DBInstanceIdentifier": "aurora-replica-sample"
                },
                {
                    "IsClusterWriter": true,
                    "DBClusterParameterGroupStatus": "in-sync",
                    "DBInstanceIdentifier": "aurora-sample"
                }
            ],
            "Port": 3306,
            "PreferredBackupWindow": "10:20-10:50",
            "VpcSecurityGroups": [
                {
                    "Status": "active",
                    "VpcSecurityGroupId": "sg-55da224b"
                }
            ],
            "DBSubnetGroup": "default",
            "StorageEncrypted": false,
            "DatabaseName": "sample",
            "EngineVersion": "5.6.10a",
            "DBClusterParameterGroup": "default.aurora5.6",
            "BackupRetentionPeriod": 1,
            "AvailabilityZones": [
                "us-east-1b",
                "us-east-1c",
                "us-east-1d"
            ],
            "LatestRestorableTime": "2016-03-31T20:00:11.491Z",
            "PreferredMaintenanceWindow": "sun:03:53-sun:04:23"
        }
    ]
}
```

API

To view DB cluster information using the Amazon RDS API, use the DescribeDBClusters action. For example, the following Amazon RDS API command lists the DB cluster information for all of the DB clusters in the `us-east-1` region.

```
https://rds.us-east-1.amazonaws.com/
    ?Action=DescribeDBClusters
    &MaxRecords=100
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-10-31
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20140722/us-east-1/rds/aws4_request
    &X-Amz-Date=20140722T200807Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=2d4f2b9e8abc31122b5546f94c0499bba47de813cb875f9b9c78e8e19c9afe1b
```

The action returns the following output:

```
<DescribeDBClustersResponse xmlns="http://rds.amazonaws.com/doc/2014-10-31/">
  <DescribeDBClustersResult>
    <DBClusters>
      <DBCluster>
        <Engine>aurora5.6</Engine>
        <Status>available</Status>
        <BackupRetentionPeriod>0</BackupRetentionPeriod>
        <DBSubnetGroup>my-subgroup</DBSubnetGroup>
        <EngineVersion>5.6.10a</EngineVersion>
        <Endpoint>sample-cluster2.cluster-cbfvmgb0y5fy.us-east-1.rds.amazonaws.com</
Endpoint>
        <DBClusterIdentifier>sample-cluster2</DBClusterIdentifier>
        <PreferredBackupWindow>04:45-05:15</PreferredBackupWindow>
        <PreferredMaintenanceWindow>sat:05:56-sat:06:26</PreferredMaintenanceWindow>
        <DBClusterMembers/>
        <AllocatedStorage>15</AllocatedStorage>
        <MasterUsername>awsuser</MasterUsername>
      </DBCluster>
      <DBCluster>
        <Engine>aurora5.6</Engine>
        <Status>available</Status>
        <BackupRetentionPeriod>0</BackupRetentionPeriod>
        <DBSubnetGroup>my-subgroup</DBSubnetGroup>
        <EngineVersion>5.6.10a</EngineVersion>
        <Endpoint>sample-cluster3.cluster-cefgqfx9y5fy.us-east-1.rds.amazonaws.com</
Endpoint>
        <DBClusterIdentifier>sample-cluster3</DBClusterIdentifier>
        <PreferredBackupWindow>07:06-07:36</PreferredBackupWindow>
        <PreferredMaintenanceWindow>tue:10:18-tue:10:48</PreferredMaintenanceWindow>
        <DBClusterMembers>
          <DBClusterMember>
            <IsClusterWriter>true</IsClusterWriter>
            <DBInstanceIdentifier>sample-cluster3-master</DBInstanceIdentifier>
          </DBClusterMember>
          <DBClusterMember>
            <IsClusterWriter>false</IsClusterWriter>
            <DBInstanceIdentifier>sample-cluster3-read1</DBInstanceIdentifier>
          </DBClusterMember>
        </DBClusterMembers>
        <AllocatedStorage>15</AllocatedStorage>
        <MasterUsername>awsuser</MasterUsername>
      </DBCluster>
```

```
    </DBClusters>
  </DescribeDBClustersResult>
  <ResponseMetadata>
    <RequestId>d682b02c-1383-11b4-a6bb-172dfac7f170</RequestId>
  </ResponseMetadata>
</DescribeDBClustersResponse>
```

Related Topics

- Amazon Aurora on Amazon RDS (p. 428)

Migrating Data to an Amazon Aurora DB Cluster

You have several options for migrating data from your existing database to an Amazon Aurora DB cluster, depending on database engine compatibility. Your migration options also depend on the database that you are migrating from and the size of the data that you are migrating.

Migrating Data to an Amazon Aurora MySQL DB Cluster

You can migrate data from one of the following sources to an Amazon Aurora MySQL DB cluster.

- An Amazon RDS MySQL DB instance
- A MySQL database external to Amazon RDS
- A database that is not MySQL-compatible

For more information, see Migrating Data to an Amazon Aurora MySQL DB Cluster (p. 487).

Migrating Data to an Amazon Aurora PostgreSQL DB Cluster

You can migrate data from one of the following sources to an Amazon Aurora PostgreSQL DB cluster.

- An Amazon RDS PostgreSQL DB instance
- A database that is not PostgreSQL-compatible

For more information, see Migrating Data to Amazon Aurora PostgreSQL (p. 641).

Related Topics

- Amazon Aurora on Amazon RDS (p. 428)

Managing an Amazon Aurora DB Cluster

In the following sections, you can find information about managing performance, scaling, fault tolerance, backup, and restoring for an Amazon Aurora DB cluster.

Managing Performance and Scaling for Aurora DB Clusters

You can use the following options to manage performance and scaling for Aurora DB clusters and DB instances:

- Storage Scaling (p. 467)
- Instance Scaling (p. 467)
- Read Scaling (p. 467)
- Managing Connections (p. 467)

Storage Scaling

Aurora storage automatically scales with the data in your cluster volume. As your data grows, your cluster volume storage grows in 10 gigabyte (GB) increments up to 64 TB.

The size of your cluster volume is checked on an hourly basis to determine your storage costs. For pricing information, see the Amazon RDS product page.

Instance Scaling

You can scale your Aurora DB cluster as needed by modifying the DB instance class for each DB instance in the DB cluster. Aurora supports several DB instance classes optimized for Aurora, depending on database engine compatibility.

Database Engine	Instance Scaling
Amazon Aurora MySQL	See Scaling Aurora MySQL DB Instances (p. 518)
Amazon Aurora PostgreSQL	See Scaling Aurora PostgreSQL DB Instances (p. 644)

Read Scaling

You can achieve read scaling for your Aurora DB cluster by creating up to 15 Aurora Replicas in the DB cluster. Each Aurora Replica returns the same data from the cluster volume with minimal replica lag—usually considerably less than 100 milliseconds after the primary instance has written an update. As your read traffic increases, you can create additional Aurora Replicas and connect to them directly to distribute the read load for your DB cluster. Aurora Replicas don't have to be of the same DB instance class as the primary instance.

Managing Connections

The maximum number of connections allowed to an Aurora DB instance is determined by the `max_connections` parameter in the instance-level parameter group for the DB instance. The default value of that parameter varies depends on the DB instance class used for the DB instance and database engine compatibility.

Database engine	max_connections default value
Amazon Aurora MySQL	See Maximum Connections to an Aurora MySQL DB Instance (p. 519)

Database engine	max_connections default value
Amazon Aurora PostgreSQL	See Maximum Connections to an Aurora PostgreSQL DB Instance (p. 645)

Fault Tolerance for an Aurora DB Cluster

An Aurora DB cluster is fault tolerant by design. The cluster volume spans multiple Availability Zones in a single AWS Region, and each Availability Zone contains a copy of the cluster volume data. This functionality means that your DB cluster can tolerate a failure of an Availability Zone without any loss of data and only a brief interruption of service.

If the primary instance in a DB cluster fails, Aurora automatically fails over to a new primary instance in one of two ways:

- By promoting an existing Aurora Replica to the new primary instance
- By creating a new primary instance

If the DB cluster has one or more Aurora Replicas, then an Aurora Replica is promoted to the primary instance during a failure event. A failure event results in a brief interruption, during which read and write operations fail with an exception. However, service is typically restored in less than 120 seconds, and often less than 60 seconds. To increase the availability of your DB cluster, we recommend that you create at least one or more Aurora Replicas in two or more different Availability Zones.

You can customize the order in which your Aurora Replicas are promoted to the primary instance after a failure by assigning each replica a priority. Priorities range from 0 for the highest priority to 15 for the lowest priority. If the primary instance fails, Amazon RDS promotes the Aurora Replica with the highest priority to the new primary instance. You can modify the priority of an Aurora Replica at any time. Modifying the priority doesn't trigger a failover.

More than one Aurora Replica can share the same priority, resulting in promotion tiers. If two or more Aurora Replicas share the same priority, then Amazon RDS promotes the replica that is largest in size. If two or more Aurora Replicas share the same priority and size, then Amazon RDS promotes an arbitrary replica in the same promotion tier.

If the DB cluster doesn't contain any Aurora Replicas, then the primary instance is recreated during a failure event. A failure event results in an interruption during which read and write operations fail with an exception. Service is restored when the new primary instance is created, which typically takes less than 10 minutes. Promoting an Aurora Replica to the primary instance is much faster than creating a new primary instance.

> **Note**
> Amazon Aurora also supports replication with an external MySQL database, or an RDS MySQL DB instance. For more information, see Replication Between Aurora and MySQL or Between Aurora and Another Aurora DB Cluster (p. 537).

Backing Up and Restoring an Aurora DB Cluster

In the following sections, you can find information about Aurora backups and how to restore your Aurora DB cluster using the AWS Management Console.

Backups

Aurora backs up your cluster volume automatically and retains restore data for the length of the *backup retention period*. Aurora backups are continuous and incremental so you can quickly restore to any point within the backup retention period. No performance impact or interruption of database service occurs

as backup data is being written. You can specify a backup retention period, from 1 to 35 days, when you create or modify a DB cluster.

If you want to retain a backup beyond the backup retention period, you can also take a snapshot of the data in your cluster volume. Storing snapshots incurs the standard storage charges for Amazon RDS. For more information about RDS storage pricing, see Amazon Relational Database Service Pricing.

Because Aurora retains incremental restore data for the entire backup retention period, you only need to create a snapshot for data that you want to retain beyond the backup retention period. You can create a new DB cluster from the snapshot.

Restoring Data

You can recover your data by creating a new Aurora DB cluster from the backup data that Aurora retains, or from a DB cluster snapshot that you have saved. You can quickly restore a new copy of a DB cluster created from backup data to any point in time during your backup retention period. The continuous and incremental nature of Aurora backups during the backup retention period means you don't need to take frequent snapshots of your data to improve restore times.

To determine the latest or earliest restorable time for a DB instance, look for the `Latest Restorable Time` or `Earliest Restorable Time` values on the RDS console. The latest restorable time for a DB cluster is the most recent point at which you can restore your DB cluster, typically within 5 minutes of the current time. The earliest restorable time specifies how far back within the backup retention period that you can restore your cluster volume.

You can determine when the restore of a DB cluster is complete by checking the `Latest Restorable Time` and `Earliest Restorable Time` values. The `Latest Restorable Time` and `Earliest Restorable Time` values return NULL until the restore operation is complete. You can't request a backup or restore operation if `Latest Restorable Time` or `Earliest Restorable Time` returns NULL.

To restore a DB cluster to a specified time using the AWS Management Console

1. Open the Amazon Aurora console at https://console.aws.amazon.com/rds.
2. In the navigation pane, choose **Instances**. Choose the primary instance for the DB cluster that you want to restore.
3. Choose **Instance Actions**, and then choose **Restore To Point In Time**.

   In the **Restore DB Cluster** window, choose **Use Custom Restore Time**.
4. Type the date and time that you want to restore to for **Use Custom Restore Time**.
5. Type a name for the new, restored DB instance for **DB Instance Identifier**.
6. Choose **Launch DB Cluster** to launch the restored DB cluster.

Database Cloning for Aurora

You can also use database cloning to clone the databases of your Aurora DB cluster to a new DB cluster, instead of restoring a DB cluster snapshot. The clone databases use only minimal additional space when first created. Data is copied only as data changes, either on the source databases or the clone databases. You can make multiple clones from the same DB cluster, or create additional clones even from other clones. For more information, see Cloning Databases in an Aurora DB Cluster (p. 479).

Amazon Aurora DB Cluster and DB Instance Parameters

You manage your Amazon Aurora DB cluster in the same way that you manage other Amazon RDS DB instances, by using parameters in a DB parameter group. Amazon Aurora differs from other DB engines

in that you have a cluster of DB instances. As a result, some of the parameters that you use to manage your Amazon Aurora DB cluster apply to the entire cluster. Other parameters apply only to a particular DB instance in the DB cluster.

Cluster-level parameters are managed in DB cluster parameter groups. Instance-level parameters are managed in DB parameter groups.

Although each DB instance in an Aurora DB cluster is compatible with a specific database engine, some of the database engine parameters must be applied at the cluster level. You manage these using DB cluster parameter groups. Cluster-level parameters are not found in the DB parameter group for an instance in an Aurora DB cluster and are listed later in this topic.

The DB cluster and DB instance parameters available to you in Aurora vary depending on database engine compatibility.

Database Engine	Parameters
Amazon Aurora MySQL	See Amazon Aurora MySQL Parameters (p. 600)
Amazon Aurora PostgreSQL	See Amazon Aurora PostgreSQL Parameters (p. 654)

Related Topics

- Amazon Aurora on Amazon RDS (p. 428)

Monitoring an Amazon Aurora DB Cluster

Amazon Aurora provides a variety of Amazon CloudWatch metrics that you can monitor to determine the health and performance of your Aurora DB cluster. You can use various tools, such as the Amazon RDS Management Console, AWS CLI, and CloudWatch API, to view Aurora metrics. For more information, see Monitoring Amazon RDS (p. 245).

Amazon Aurora MySQL Metrics

The following metrics are available from Amazon Aurora MySQL.

Metric	Description
`ActiveTransactions`	The average number of current transactions executing on an Aurora MySQL DB instance per second.
`AuroraBinlogReplicaLag`	The amount of time an Aurora MySQL DB cluster lags behind the source DB cluster. This metric reports the value of the `Seconds_Behind_Master` field of the MySQL `SHOW SLAVE STATUS` command and is useful for monitoring replica lag between Aurora DB clusters that are replicating across different AWS regions. For more information, see Replicating Amazon Aurora MySQL DB Clusters Across AWS Regions (p. 528).
`AuroraReplicaLag`	For an Aurora MySQL Replica, the amount of lag when replicating updates from the primary instance, in milliseconds.
`AuroraReplicaLagMaximum`	The maximum amount of lag between the primary instance and each Aurora MySQL DB instance in the DB cluster, in milliseconds.

Metric	Description
AuroraReplicaLagMinimum	The minimum amount of lag between the primary instance and each Aurora MySQL DB instance in the DB cluster, in milliseconds.
BlockedTransactions	The average number of transactions in the database that are blocked per second.
BufferCacheHitRatio	The percentage of requests that are served by the buffer cache.
CommitLatency	The amount of latency for commit operations, in milliseconds.
CommitThroughput	The average number of commit operations per second.
CPUCreditBalance	The number of CPU credits that an instance has accumulated. This metric applies only to `db.t2.small` and `db.t2.medium` instances. It's used to determine how long an Aurora MySQL DB instance can burst beyond its baseline performance level at a given rate. **Note** CPU credit metrics are reported at 5-minute intervals.
CPUCreditUsage	The number of CPU credits consumed during the specified period. This metric applies only to `db.t2.small` and `db.t2.medium` instances. It identifies the amount of time during which physical CPUs have been used for processing instructions by virtual CPUs allocated to the Aurora MySQL DB instance. **Note** CPU credit metrics are reported at 5-minute intervals.
CPUUtilization	The percentage of CPU used by an Aurora MySQL DB instance.
DatabaseConnections	The number of connections to an Aurora MySQL DB instance.
DDLLatency	The amount of latency for data definition language (DDL) requests, in milliseconds — for example, create, alter, and drop requests.
DDLThroughput	The average number of DDL requests per second.
Deadlocks	The average number of deadlocks in the database per second.
DeleteLatency	The amount of latency for delete queries, in milliseconds.
DeleteThroughput	The average number of delete queries per second.
DMLLatency	The amount of latency for inserts, updates, and deletes, in milliseconds.
DMLThroughput	The average number of inserts, updates, and deletes per second.
EngineUptime	The amount of time that the instance has been running, in seconds.
FreeableMemory	The amount of available random access memory, in bytes.
FreeLocalStorage	The amount of storage available for temporary tables and logs, in bytes. Unlike for other DB engines, for Aurora MySQL this metric reports the amount of storage available to each DB instance for temporary tables and logs. This value depends on the DB instance class (for pricing information, see the Amazon RDS product page). You can increase the amount of free storage space for an instance by choosing a larger DB instance class for your instance.

Metric	Description
InsertLatency	The amount of latency for insert queries, in milliseconds.
InsertThroughput	The average number of insert queries per second.
LoginFailures	The average number of failed login attempts per second.
NetworkReceiveThroughput	The amount of network throughput received from clients by each instance in the Aurora MySQL DB cluster, in bytes per second. This throughput doesn't include network traffic between instances in the Aurora MySQL DB cluster and the cluster volume.
NetworkThroughput	The amount of network throughput both received from and transmitted to clients by each instance in the Aurora MySQL DB cluster, in bytes per second. This throughput doesn't include network traffic between instances in the DB cluster and the cluster volume.
NetworkTransmitThroughput	The amount of network throughput sent to clients by each instance in the Aurora MySQL DB cluster, in bytes per second. This throughput doesn't include network traffic between instances in the DB cluster and the cluster volume.
Queries	The average number of queries executed per second.
ResultSetCacheHitRatio	The percentage of requests that are served by the Resultset cache.
SelectLatency	The amount of latency for select queries, in milliseconds.
SelectThroughput	The average number of select queries per second.
UpdateLatency	The amount of latency for update queries, in milliseconds.
UpdateThroughput	The average number of update queries per second.
VolumeBytesUsed	The amount of storage used by your Aurora MySQL database, in bytes. This value affects the cost of the Aurora MySQL DB cluster (for pricing information, see the Amazon RDS product page).
VolumeReadIOPs	The average number of billed read I/O operations from a cluster volume, reported at 5-minute intervals. Billed read operations are calculated at the cluster volume level, aggregated from all instances in the Aurora MySQL DB cluster, and then reported at 5-minute intervals. The value is calculated by taking the value of the **Read operations** metric over a 5-minute period. You can determine the amount of billed read operations per second by taking the value of the **Billed read operations** metric and dividing by 300 seconds. For example, if the **Billed read operations** returns 13,686, then the billed read operations per second is 45 (13,686 / 300 = 45.62). You accrue billed read operations for queries that request database pages that are not present in the buffer cache and therefore must be loaded from storage. You might see spikes in billed read operations as query results are read from storage and then loaded into the buffer cache.
VolumeWriteIOPs	The average number of write disk I/O operations to the cluster volume, reported at 5-minute intervals.

Amazon Aurora PostgreSQL Metrics

The following metrics are available from Amazon Aurora PostgreSQL.

Metric	Description
AuroraReplicaLag	For an Aurora PostgreSQL Replica, the amount of lag when replicating updates from the primary instance, in milliseconds.
AuroraReplicaLagMaximum	The maximum amount of lag between the primary instance and each Aurora PostgreSQL DB instance in the DB cluster, in milliseconds.
AuroraReplicaLagMinimum	The minimum amount of lag between the primary instance and each Aurora PostgreSQL DB instance in the DB cluster, in milliseconds.
BufferCacheHitRatio	The percentage of requests that are served by the buffer cache.
CommitLatency	The amount of latency for commit operations, in milliseconds.
CommitThroughput	The average number of commit operations per second.
CPUUtilization	The percentage of CPU used by an Aurora PostgreSQL DB instance.
DatabaseConnections	The number of connections to an Aurora PostgreSQL DB instance.
Deadlocks	The average number of deadlocks in the database per second.
DiskQueueDepth	The number of outstanding read/write requests waiting to access the disk.
EngineUptime	The amount of time that the instance has been running, in seconds.
FreeableMemory	The amount of available random access memory, in bytes.
FreeLocalStorage	Unlike for other DB engines, for Aurora PostgreSQL this metric reports the amount of storage available to each DB instance for temporary tables and logs. This value depends on the DB instance class (for pricing information, see the Amazon RDS product page). You can increase the amount of free storage space for an instance by choosing a larger DB instance class for your instance.
MaximumUsedTransactionIDs	The age of the oldest unvacuumed transaction ID, in transactions. If this value reaches 2,146,483,648 (2^31 - 1,000,000), the database is forced into read-only mode, to avoid transaction ID wraparound. For more information, see Preventing Transaction ID Wraparound Failures in the PostgreSQL documentation.
NetworkReceiveThroughput	The amount of network throughput received from clients by each instance in the Aurora PostgreSQL DB cluster, in bytes per second. This throughput doesn't include network traffic between instances in the DB cluster and the cluster volume.
NetworkThroughput	The amount of network throughput both received from and transmitted to clients by each instance in the Aurora PostgreSQL DB cluster, in bytes per second. This throughput doesn't include network traffic between instances in the DB cluster and the cluster volume.
NetworkTransmitThroughput	The amount of network throughput sent to clients by each instance in the Aurora PostgreSQL DB cluster, in bytes per second. This throughput doesn't

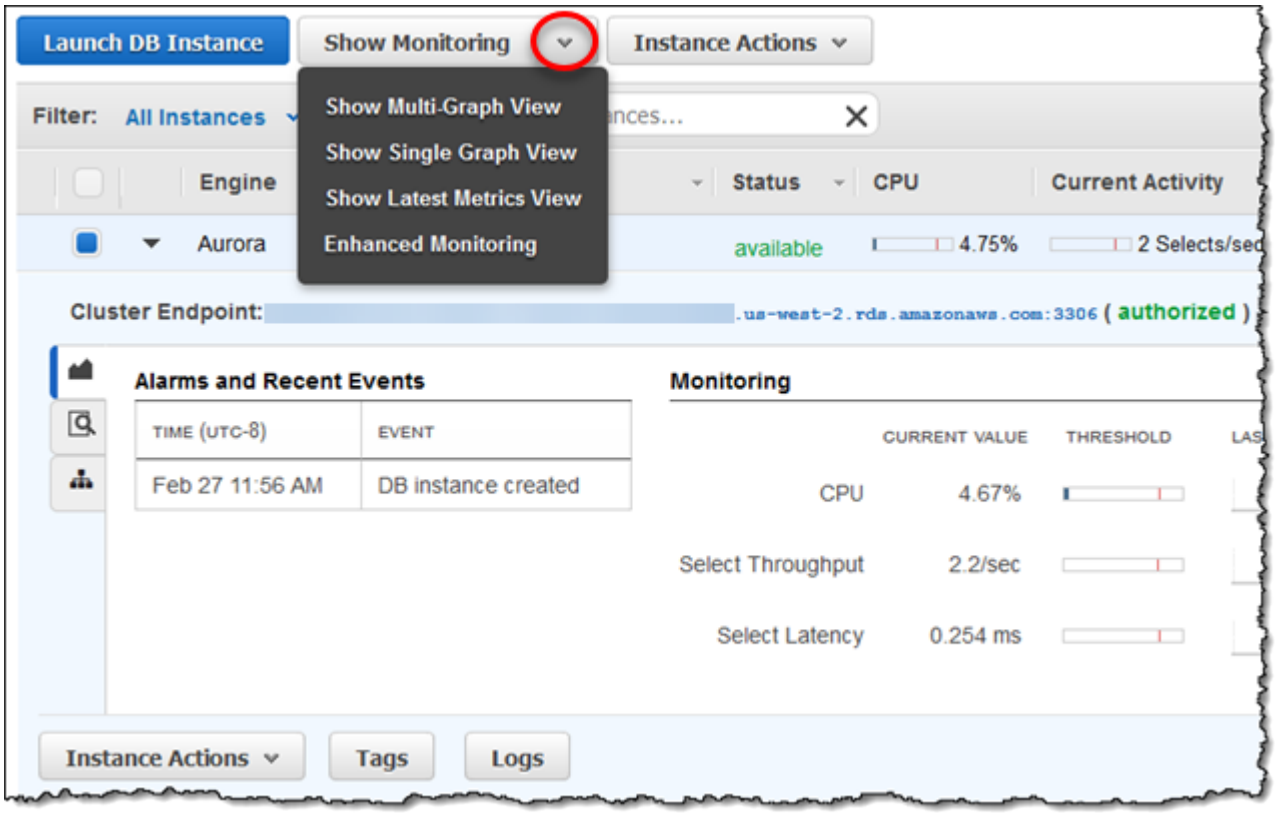
Metric	Description
	include network traffic between instances in the DB cluster and the cluster volume.
`ReadIOPS`	The average number of disk I/O operations per second. Aurora PostgreSQL reports read and write IOPS separately, on one minute intervals.
`ReadLatency`	The average amount of time taken per disk I/O operation.
`ReadThroughput`	The average number of bytes read from disk per second.
`SwapUsage`	The amount of swap space used on the Aurora PostgreSQL DB instance.
`TransactionLogsDiskUsage`	The amount of disk space occupied by transaction logs on the Aurora PostgreSQL DB instance.
`VolumeBytesUsed`	The amount of storage used by your Aurora PostgreSQL database, in bytes. This value affects the cost of the Aurora PostgreSQL DB cluster (for pricing information, see the Amazon RDS product page).
`VolumeReadIOPs`	The average number of billed read I/O operations from a cluster volume, reported at 5-minute intervals. Billed read operations are calculated at the cluster volume level, aggregated from all instances in the Aurora PostgreSQL DB cluster, and then reported at 5-minute intervals. The value is calculated by taking the value of the **Read operations** metric over a 5-minute period. You can determine the amount of billed read operations per second by taking the value of the **Billed read operations** metric and dividing by 300 seconds. For example, if the **Billed read operations** returns 13,686, then the billed read operations per second is 45 (13,686 / 300 = 45.62). You accrue billed read operations for queries that request database pages that are not present in the buffer cache and therefore must be loaded from storage. You might see spikes in billed read operations as query results are read from storage and then loaded into the buffer cache.
`VolumeWriteIOPs`	The average number of write disk I/O operations to the cluster volume, reported at 5-minute intervals.
`WriteIOPS`	The average number of disk I/O operations per second. Aurora PostgreSQL reports read and write IOPS separately, on one minute intervals.
`WriteLatency`	The average amount of time taken per disk I/O operation.
`WriteThroughput`	The average number of bytes written to disk per second.

Viewing Aurora Metrics in the Amazon RDS Console

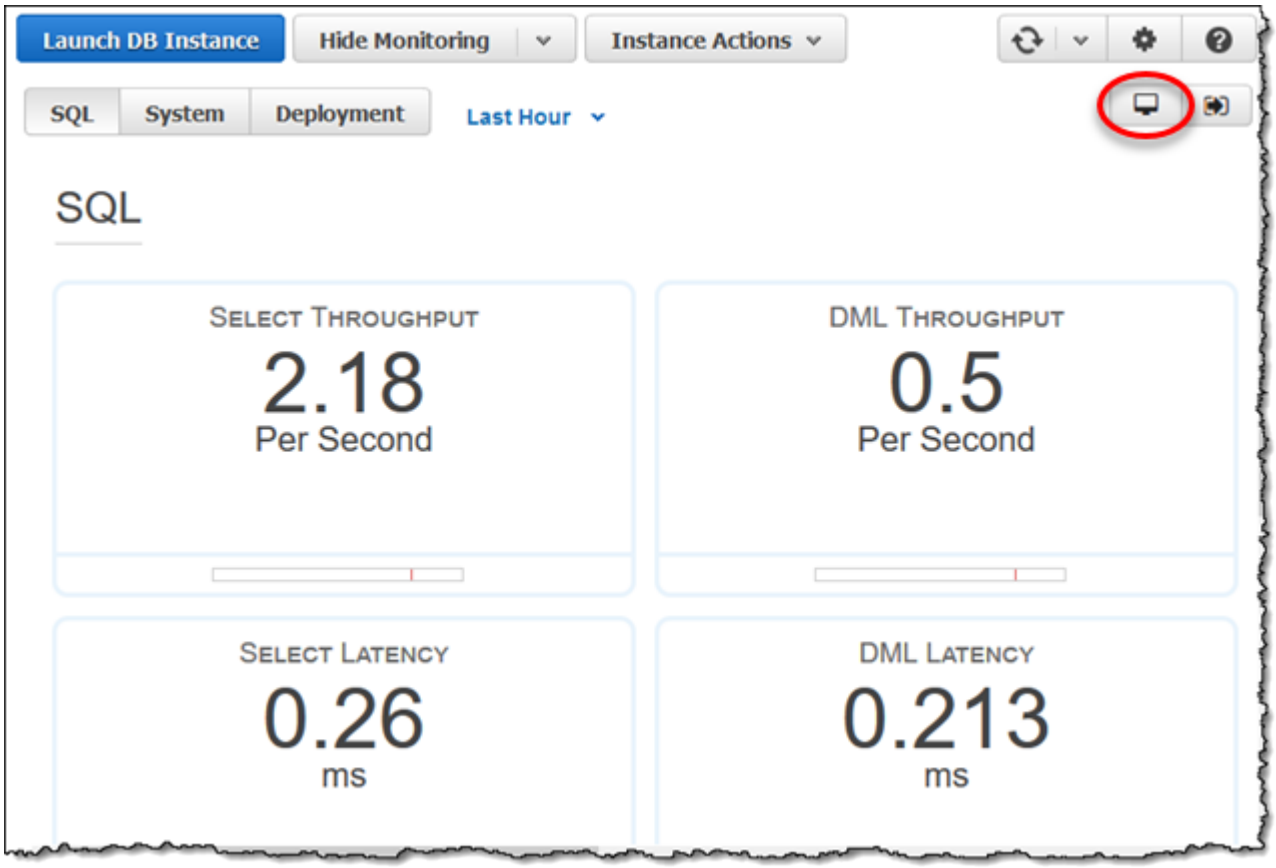
To monitor the health and performance of your Aurora DB cluster, you can view some, but not all, of the metrics provided by Amazon Aurora in the Amazon RDS console. For a detailed list of Aurora metrics available to the Amazon RDS console, see Aurora Metrics Available in the Amazon RDS Console (p. 476).

To view Aurora metrics in the Amazon RDS console

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Instances**.
3. Choose the check box to the left of the DB instance you need information about. For **Show Monitoring**, choose one of the following options for how you want to view your metrics:
   - **Show Multi-Graph View** – Shows a summary of Aurora metrics. Each metric includes a graph showing the metric monitored over a specific time span.
   - **Show Single Graph View** – Shows a single Aurora metric at a time with more detail. Each metric includes a graph showing the metric monitored over a specific time span.
   - **Show Latest Metrics View** – Shows a summary of Aurora metrics without graphs. You can choose an option to display this summary full-screen.
   - **Enhanced Monitoring** – Shows a summary of OS metrics available to an Aurora DB instance with Enhanced Monitoring enabled. Each metric includes a graph showing the metric monitored over a specific time span. For more information, see Enhanced Monitoring (p. 258).



4. If you chose **Show Latest Metrics View**, choose the full screen button to view only your metrics in full-screen mode.



Aurora Metrics Available in the Amazon RDS Console

Not all of the metrics provided by Amazon Aurora are available to you in the Amazon RDS console. You can view them using other tools, however, such as the AWS CLI and CloudWatch API. In addition, some of the metrics that are available in the Amazon RDS console are either shown only for specific instance classes, or with different names and different units of measurement.

The following Aurora metrics are not available in the Amazon RDS console:

- `AuroraBinlogReplicaLag`
- `DeleteLatency`
- `DeleteThroughput`
- `EngineUptime`
- `InsertLatency`
- `InsertThroughput`
- `NetworkThroughput`
- `Queries`
- `UpdateLatency`
- `UpdateThroughput`

In addition, some Aurora metrics are either shown only for specific instance classes, or only for DB instances, or with different names and different units of measurement:

- The `CPUCreditBalance` and `CPUCreditUsage` metrics are displayed only for `db.t2.small` and `db.t2.medium` instances
- The following metrics that are displayed with different names, as listed:

Metric	Display Name
`AuroraReplicaLagMaximum`	Replica lag maximum
`AuroraReplicaLagMinimum`	Replica lag minimum
`DDLThroughput`	DDL
`NetworkReceiveThroughput`	Network throughput
`VolumeBytesUsed`	[Billed] Volume bytes used
`VolumeReadIOPs`	[Billed] Volume read IOPs
`VolumeWriteIOPs`	[Billed] Volume write IOPs

- The following metrics apply to an entire Aurora DB cluster, but are displayed only when viewing DB instances for an Aurora DB cluster in the Amazon RDS console:
  - `VolumeBytesUsed`
  - `VolumeReadIOPs`
  - `VolumeWriteIOPs`
- The following metrics are displayed in megabytes, instead of bytes, in the Amazon RDS console:
  - `FreeableMemory`
  - `FreeLocalStorage`
  - `NetworkReceiveThroughput`
  - `NetworkTransmitThroughput`

Latest Metrics View

You can view a subset of categorized Aurora metrics in the Latest Metrics view of the Amazon RDS console. The following table lists the categories and associated metrics displayed in the Amazon RDS console for an Aurora instance.

Category	Metrics

SQL	`ActiveTransactions`
	`BlockedTransactions`
	`BufferCacheHitRatio`
	`CommitLatency`
	`CommitThroughput`
	`DatabaseConnections`
	`DDLLatency`
	`DDLThroughput`
	`Deadlocks`

Category	Metrics
	DMLLatency DMLThroughput LoginFailures ResultSetCacheHitRatio SelectLatency SelectThroughput
System	AuroraReplicaLag AuroraReplicaLagMaximum AuroraReplicaLagMinimum CPUCreditBalance CPUCreditUsage CPUUtilization FreeableMemory FreeLocalStorage NetworkReceiveThroughput
Deployment	AuroraReplicaLag BufferCacheHitRatio ResultSetCacheHitRatio SelectThroughput

Note
The **Failed SQL Statements** metric, displayed under the **SQL** category of the Latest Metrics view in the Amazon RDS console, does not apply to Amazon Aurora.

Related Topics

- Amazon Aurora on Amazon RDS (p. 428)

Replication with Amazon Aurora

Aurora Replicas

Aurora Replicas are independent endpoints in an Aurora DB cluster, best used for scaling read operations and increasing availability. Up to 15 Aurora Replicas can be distributed across the Availability Zones that a DB cluster spans within an AWS Region. The DB cluster volume is made up of multiple copies of the data for the DB cluster. However, the data in the cluster volume is represented as a single, logical volume to the primary instance and to Aurora Replicas in the DB cluster.

As a result, all Aurora Replicas return the same data for query results with minimal replica lag—usually much less than 100 milliseconds after the primary instance has written an update. Replica lag varies depending on the rate of database change. That is, during periods where a large amount of write operations occur for the database, you might see an increase in replica lag.

Aurora Replicas work well for read scaling because they are fully dedicated to read operations on your cluster volume. Write operations are managed by the primary instance. Because the cluster volume is shared among all DB instances in your DB cluster, no additional work is required to replicate a copy of the data for each Aurora Replica.

To increase availability, you can use Aurora Replicas as failover targets. That is, if the primary instance fails, an Aurora Replica is promoted to the primary instance. There is a brief interruption during which read and write requests made to the primary instance fail with an exception. If your Aurora DB cluster doesn't include any Aurora Replicas, then the primary instance is recreated during a failure event. However, promoting an Aurora Replica is much faster than recreating the primary instance. For high-availability scenarios, we recommend that you create one or more Aurora Replicas. These should be of the same DB instance class as the primary instance and in different Availability Zones for your Aurora DB cluster. For more information on Aurora Replicas as failover targets, see Fault Tolerance for an Aurora DB Cluster (p. 468).

For details on how to create an Aurora Replica, see Creating an Aurora Replica Using the Console (p. 447).

Replication with Aurora MySQL

In addition to Aurora Replicas, you have the following options for replication with Aurora MySQL:

- Two Aurora MySQL DB clusters in different AWS Regions, by creating an Aurora Read Replica of an Aurora MySQL DB cluster in a different AWS Region.
- Two Aurora MySQL DB clusters in the same region, by using MySQL binary log (binlog) replication.
- An Amazon RDS MySQL DB instance as the master and an Aurora MySQL DB cluster, by creating an Aurora Read Replica of an Amazon RDS MySQL DB instance. Typically, this approach is used for migration to Aurora MySQL, rather than for ongoing replication.

For more information about replication with Aurora MySQL, see Replication with Amazon Aurora MySQL (p. 527).

Cloning Databases in an Aurora DB Cluster

Using database cloning, you can quickly and cost-effectively create clones of all your databases. The clone databases require only minimal additional space when first created. Database cloning uses a copy-on-write protocol, in which data is copied at the time that data changes, either on the source databases or the clone databases. You can make multiple clones from the same DB cluster. You can also create additional clones from other clones. For more information on how the copy-on-write protocol works in the context of Aurora storage, see Copy-on-Write Protocol for Database Cloning (p. 480).

You can use database cloning in a variety of use cases, especially where you don't want to have an impact on your production environment, such as the following:

- Experiment with and assess the impact of changes, such as schema changes or parameter group changes
- Perform workload-intensive operations, such as exporting data or running analytical queries
- Create a copy of a production DB cluster in a non-production environment for development or testing

Limitations

There are some limitations involved with database cloning, described following:

- Cloning is only supported for Aurora MySQL instances.
- You cannot create clone databases across AWS regions. The clone databases must be created in the same region as the source databases.
- Currently, you are limited to up to 15 clones based on a copy, including clones based on other clones. After that, only copies can be created. However, each copy can also have up to 15 clones.
- Cross-account database cloning is not currently supported.
- You can provide a different virtual private cloud (VPC) for your clone. However, the subnets in those VPCs must map to the same set of Availability Zones.

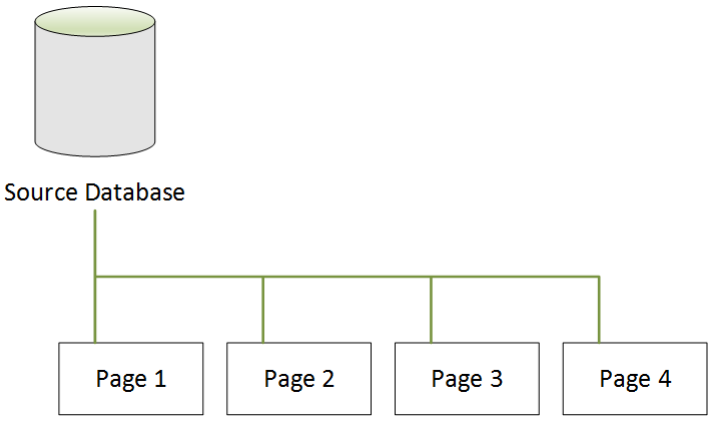
Copy-on-Write Protocol for Database Cloning

The following scenarios illustrate how the copy-on-write protocol works.

- Before Database Cloning (p. 480)
- After Database Cloning (p. 480)
- When a Change Occurs on the Source Database (p. 481)
- When a Change Occurs on the Clone Database (p. 481)

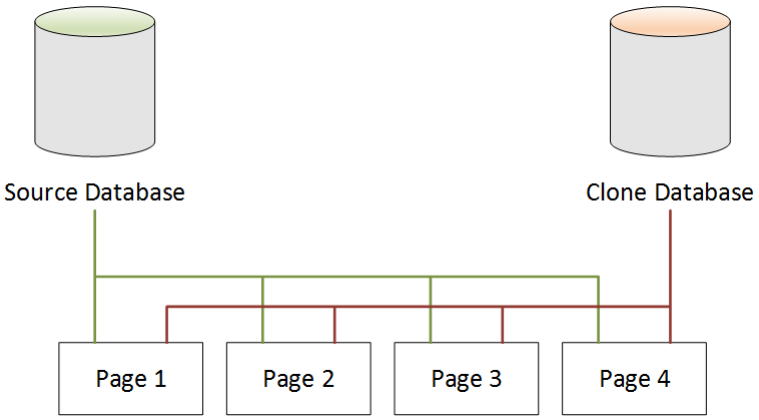
Before Database Cloning

Data in a source database is stored in pages. In the following diagram, the source database has four pages.



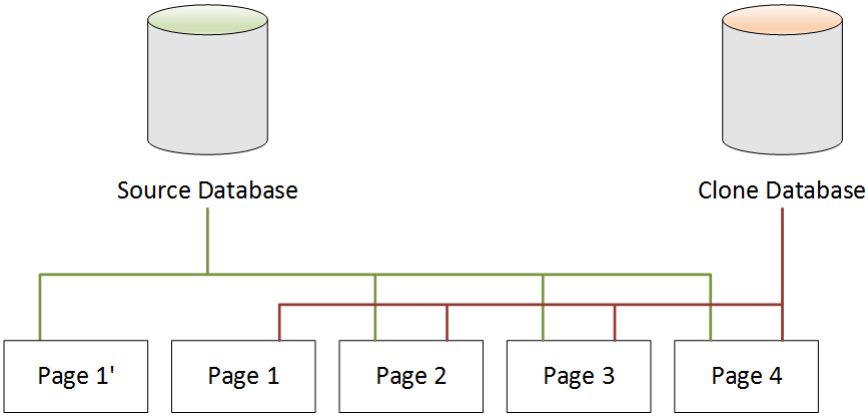
After Database Cloning

As shown in the following diagram, there are no changes in the source database after database cloning. Both the source database and the clone database point to the same four pages. None of the pages has been physically copied, so no additional storage is required.



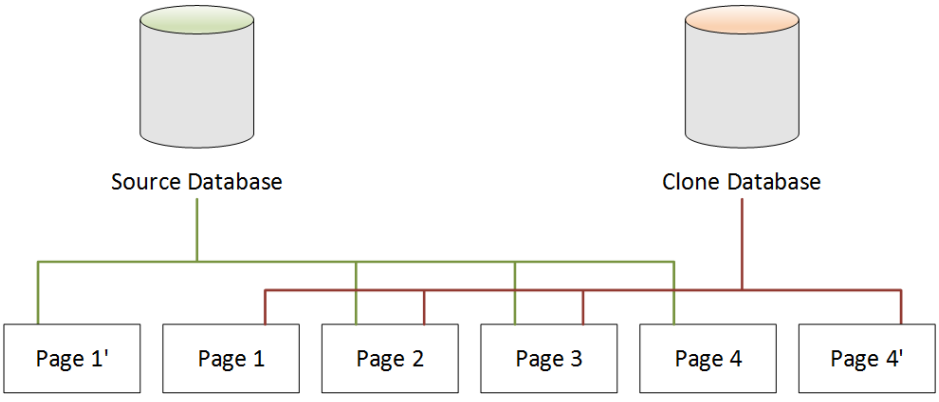
When a Change Occurs on the Source Database

In the following example, the source database makes a change to the data in `Page 1`. Instead of writing to the original `Page 1`, additional storage is used to create a new page, called `Page 1'`. The source database now points to the new `Page 1'`, and also to `Page 2`, `Page 3`, and `Page 4`. The clone database continues to point to `Page 1` through `Page 4`.



When a Change Occurs on the Clone Database

In the following diagram, the clone database has also made a change, this time in `Page 4`. Instead of writing to the original `Page 4`, additional storage is used to create a new page, called `Page 4'`. The source database continues to point to `Page 1'`, and also `Page 2` through Page 4, but the clone database now points to `Page 1` through `Page 3`, and also `Page 4'`.



As shown in the second scenario, after database cloning there is no additional storage required at the point of clone creation. However, as changes occur in the source database and clone database, only the changed pages are created, as shown in the third and fourth scenarios. As more changes occur over time in both the source database and clone database, you need incrementally more storage to capture and store the changes.

Deleting Source Databases

When deleting a source database that has one or more clone databases associated with it, the clone databases are not affected. The clone databases continue to point to the pages that were previously owned by the source database.

AWS Management Console

The following procedure describes how to clone an Aurora DB cluster using the AWS Management Console.

To create a clone of a DB cluster using the AWS Management Console

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Instances**. Choose the primary instance for the DB cluster that you want to create a clone of.
3. Choose **Instance Actions**, and then choose **Create Clone**.
4. In the **Create Clone** pane, type a name for the primary instance of the clone DB cluster as the **DB Instance Identifier**.

   If you want to, set any other settings for the clone DB cluster. For information about the different DB cluster settings, see AWS Management Console (p. 439).
5. Choose **Create Clone** to launch the clone DB cluster.

CLI

The following procedure describes how to clone an Aurora DB cluster using the AWS CLI.

To create a clone of a DB cluster using the AWS CLI

- Call the restore-db-cluster-to-point-in-time AWS CLI command and supply the following values:
  - `--source-db-cluster-identifier` – the name of the source DB cluster to create a clone of.
  - `--db-cluster-identifier` – the name of the clone DB cluster.
  - `--restore-type copy-on-write` – values that indicate to create a clone DB cluster.
  - `--use-latest-restorable-time` – specifies that the latest restorable backup time is used.

  The following example creates a clone of the DB cluster named `sample-source-cluster`. The name of the clone DB cluster is `sample-cluster-clone`.

  For Linux, OS X, or Unix:

```
aws rds restore-db-cluster-to-point-in-time \
    --source-db-cluster-identifier sample-source-cluster \
    --db-cluster-identifier sample-cluster-clone \
    --restore-type copy-on-write \
```

```
    --use-latest-restorable-time
```

For Windows:

```
aws rds restore-db-cluster-to-point-in-time ^
    --source-db-cluster-identifier sample-source-cluster ^
    --db-cluster-identifier sample-cluster-clone ^
    --restore-type copy-on-write ^
    --use-latest-restorable-time
```

Note
The restore-db-cluster-to-point-in-time AWS CLI command only restores the DB cluster, not the DB instances for that DB cluster. You must invoke the create-db-instance command to create DB instances for the restored DB cluster, specifying the identifier of the restored DB cluster in `--db-instance-identifier`. You can create DB instances only after the `restore-db-cluster-to-point-in-time` command has completed and the DB cluster is available.

Integrating Aurora with Other AWS Services

Amazon Aurora integrates with other AWS services so that you can extend your Aurora DB cluster to use additional capabilities in the AWS Cloud.

Integrating with Amazon Aurora MySQL

Amazon Aurora MySQL integrates with other AWS services so that you can extend your Aurora MySQL DB cluster to use additional capabilities in the AWS Cloud. Your Aurora MySQL DB cluster can use AWS services to do the following:

- Asynchronously invoke an AWS Lambda function using the `mysql.lambda_async` procedure.
- Load data from text or XML files stored in an Amazon Simple Storage Service (Amazon S3) bucket into your DB cluster using the `LOAD DATA FROM S3` or `LOAD XML FROM S3` command.
- Save data to text files stored in an Amazon S3 bucket from your DB cluster using the `SELECT INTO OUTFILE S3` command.
- Automatically add or remove Aurora Replicas with Application Auto Scaling. For more information, see Using Amazon Aurora Auto Scaling with Aurora Replicas (p. 577).

For more information about integrating Aurora MySQL with other AWS services, see Integrating Amazon Aurora MySQL with Other AWS Services (p. 550).

Integrating with Amazon Aurora PostgreSQL

Amazon Aurora PostgreSQL integrates with other AWS services so that you can extend your Aurora PostgreSQL DB cluster to use additional capabilities in the AWS Cloud. Your Aurora PostgreSQL DB cluster can use AWS services to do the following:

- Quickly collect, view, and assess performance on your relational database workloads with Performance Insights.

For more information about integrating Aurora PostgreSQL with other AWS services, see Integrating Amazon Aurora PostgreSQL with Other AWS Services (p. 647).

Related Topics

- Amazon Aurora on Amazon RDS (p. 428)

Working with Amazon Aurora MySQL

Amazon Aurora MySQL is a fully managed, MySQL-compatible, relational database engine that combines the speed and reliability of high-end commercial databases with the simplicity and cost-effectiveness of open-source databases. Aurora MySQL is a drop-in replacement for MySQL and makes it simple and cost-effective to set up, operate, and scale your new and existing MySQL deployments, thus freeing you to focus on your business and applications. Amazon RDS provides administration for Aurora by handling routine database tasks such as provisioning, patching, backup, recovery, failure detection, and repair. Amazon RDS also provides push-button migration tools to convert your existing Amazon RDS for MySQL applications to Aurora MySQL.

Availability for Amazon Aurora MySQL

The following table shows the regions where Aurora MySQL is currently available.

Region	Console Link
US East (N. Virginia)	https://console.aws.amazon.com/rds/home?region=us-east-1
US East (Ohio)	https://console.aws.amazon.com/rds/home?region=us-east-2
US West (N. California)	https://console.aws.amazon.com/rds/home?region=us-west-1
US West (Oregon)	https://console.aws.amazon.com/rds/home?region=us-west-2
Canada (Central)	https://console.aws.amazon.com/rds/home?region=ca-central-1
Asia Pacific (Mumbai)	https://console.aws.amazon.com/rds/home?region=ap-south-1
Asia Pacific (Tokyo)	https://console.aws.amazon.com/rds/home?region=ap-northeast-1
Asia Pacific (Seoul)	https://console.aws.amazon.com/rds/home?region=ap-northeast-2
Asia Pacific (Sydney)	https://console.aws.amazon.com/rds/home?region=ap-southeast-2
EU (Frankfurt)	https://console.aws.amazon.com/rds/home?region=eu-central-1
EU (Ireland)	https://console.aws.amazon.com/rds/home?region=eu-west-1
EU (London)	https://console.aws.amazon.com/rds/home?region=eu-west-2

Amazon Aurora MySQL Performance Enhancements

Amazon Aurora includes performance enhancements to support the diverse needs of high-end commercial databases.

Fast Insert

Fast insert accelerates parallel inserts sorted by primary key and applies specifically to `LOAD DATA` and `INSERT INTO ... SELECT ...` statements. Fast insert caches the position of a cursor in an index traversal while executing the statement. This avoids unnecessarily traversing the index again.

You can monitor the following metrics to determine the effectiveness of fast insert for your DB cluster:

- `aurora_fast_insert_cache_hits`: A counter that is incremented when the cached cursor is successfully retrieved and verified.
- `aurora_fast_insert_cache_misses`: A counter that is incremented when the cached cursor is no longer valid and Aurora performs a normal index traversal.

You can retrieve the current value of the fast insert metrics using the following command:

```
mysql> show global status like 'Aurora_fast_insert%';
```

You will get output similar to the following:

```
+---------------------------------+-----------+
| Variable_name                   | Value     |
+---------------------------------+-----------+
| Aurora_fast_insert_cache_hits   | 3598300   |
| Aurora_fast_insert_cache_misses | 436401336 |
+---------------------------------+-----------+
```

Amazon Aurora MySQL and Spatial Data

Amazon Aurora MySQL supports the same Spatial Data Types and Spatial Relation Functions as MySQL 5.6. Aurora MySQL also supports spatial indexing on InnoDB tables, similar to that offered by MySQL 5.7, which improves query performance on large datasets for queries that use spatial data. Note that Aurora MySQL uses a different indexing strategy than MySQL, using a space-filling curve on a B-tree instead of an R-tree.

The following data definition language (DDL) statements are supported for creating indexes on columns that use spatial data types.

CREATE TABLE

You can use the SPATIAL INDEX keywords in a CREATE TABLE statement to add a spatial index to a column in a new table. For example:

```
CREATE TABLE test (shape POLYGON NOT NULL, SPATIAL INDEX(shape));
```

ALTER TABLE

You can use the SPATIAL INDEX keywords in an ALTER TABLE statement to add a spatial index to a column in an existing table. For example:

```
ALTER TABLE test ADD SPATIAL INDEX(shape);
```

CREATE INDEX

You can also use the SPATIAL keyword in a CREATE INDEX statement to add a spatial index to a column in an existing table. For example:

```
CREATE SPATIAL INDEX shape_index ON test (shape);
```

Comparison of Amazon Aurora MySQL and Amazon RDS for MySQL

Although Aurora instances are compatible with MySQL client applications, Aurora has advantages over MySQL as well as limitations to the MySQL features that Aurora supports. This functionality can influence your decision about whether Amazon Aurora or MySQL on Amazon RDS are the best cloud database for your solution. The following table shows the differences between Amazon Aurora and Amazon RDS for MySQL.

Feature	Amazon Aurora	Amazon RDS for MySQL
Read scaling	Supports up to 15 Aurora Replicas with minimal impact on the performance of write operations.	Supports up to 5 Read Replicas with some impact on the performance of write operations.
Failover target	Aurora Replicas are automatic failover targets with no data loss.	Read Replicas can be manually promoted to the master DB instance with potential data loss.
MySQL version	Supports only MySQL version 5.6.	Supports MySQL versions 5.5, 5.6, and 5.7.
AWS Region	Aurora DB clusters can only be created in the following regions: US East (N. Virginia) (us-east-1), US East (Ohio) (us-east-2), US West (N. California) (us-west-1), US West (Oregon) (us-west-2), Canada (Central) (ca-central-1), Asia Pacific (Mumbai) (ap-south-1), Asia Pacific (Tokyo) (ap-northeast-1), Asia Pacific (Seoul) (ap-northeast-2), Asia Pacific (Sydney) (ap-southeast-2), EU (Frankfurt) (eu-central-1), EU (Ireland) (eu-west-1), EU (London) (eu-west-2).	Available in all AWS regions.
MySQL storage engine	Supports only InnoDB. Tables from other storage engines are automatically converted to InnoDB.	Supports both MyISAM and InnoDB.

Feature	Amazon Aurora	Amazon RDS for MySQL
	For information on converting existing MySQL tables to InnoDB and importing into an Aurora cluster, see Migrating Data to an Amazon Aurora MySQL DB Cluster (p. 487). Because Amazon Aurora only supports the InnoDB engine, the `NO_ENGINE_SUBSTITUTION` option of the `SQL_MODE` database parameter is enabled. This disables the ability to create an in-memory table, unless that table is specified as `TEMPORARY`.	
Read Replicas with a different storage engine than the master instance	MySQL (non-RDS) Read Replicas that replicate with an Aurora DB cluster can only use InnoDB.	Read Replicas can use both MyISAM and InnoDB.
Database engine parameters	Some parameters apply to the entire Aurora DB cluster and are managed by DB cluster parameter groups. Other parameters apply to each individual DB instance in a DB cluster and are managed by DB parameter groups. For more information, see Amazon Aurora DB Cluster and DB Instance Parameters (p. 469).	Parameters apply to each individual DB instance or Read Replica and are managed by DB parameter groups.

Migrating Data to an Amazon Aurora MySQL DB Cluster

You have several options for migrating data from your existing database to an Amazon Aurora MySQL DB cluster. Your migration options also depend on the database that you are migrating from and the size of the data that you are migrating. The following table describes your options.

Migrating From	Solution
An RDS MySQL DB instance	• You can migrate from an RDS MySQL DB instance by first by creating an Aurora MySQL Read Replica of a MySQL DB instance. When the replica lag between the MySQL DB instance and the Aurora MySQL Read Replica is 0, you can direct your client applications to read from the Aurora Read Replica and then stop replication to make the Aurora MySQL Read Replica a standalone Aurora MySQL DB cluster for reading and writing. For details, see Migrating Data from a MySQL DB Instance to an Amazon Aurora MySQL DB Cluster by Using an Aurora Read Replica (p. 507). • You can migrate data directly from an Amazon RDS MySQL DB snapshot to an Amazon Aurora MySQL DB cluster. For details, see Migrating Data from a MySQL DB Instance to an Amazon Aurora MySQL DB Cluster by Using a DB Snapshot (p. 501).

Migrating From	Solution
A MySQL database external to Amazon RDS	If your database supports the InnoDB or MyISAM tablespaces, you have these options for migrating your data to an Amazon Aurora MySQL DB cluster: • You can create a dump of your data using the `mysqldump` utility, and then import that data into an existing Amazon Aurora MySQL DB cluster. For details, see Migrating from MySQL to Amazon Aurora by Using mysqldump (p. 501). • You can copy the backup files from your database to an Amazon Simple Storage Service (Amazon S3) bucket, and then restore an Amazon Aurora MySQL DB cluster from those files. This option can be considerably faster than migrating data using `mysqldump`. For details, see Migrating Data from MySQL by Using an Amazon S3 Bucket (p. 488). • You can save data from your database as text files and copy those files to an Amazon S3 bucket. You can then load that data into an existing Aurora MySQL DB cluster using the `LOAD DATA FROM S3` MySQL command. For more information, see Loading Data into an Amazon Aurora MySQL DB Cluster from Text Files in an Amazon S3 Bucket (p. 560).
A database that is not MySQL-compatible	You can use AWS Database Migration Service (AWS DMS) to migrate data from a database that is not MySQL-compatible. For more information on AWS DMS, see What Is AWS Database Migration Service?

Migrating Data from an External MySQL Database to an Amazon Aurora MySQL DB Cluster

If your database supports the InnoDB or MyISAM tablespaces, you have these options for migrating your data to an Amazon Aurora MySQL DB cluster:

- You can create a dump of your data using the `mysqldump` utility, and then import that data into an existing Amazon Aurora MySQL DB cluster. For more information, see Migrating from MySQL to Amazon Aurora by Using mysqldump (p. 501).
- You can copy the full and incremental backup files from your database to an Amazon S3 bucket, and then restore an Amazon Aurora MySQL DB cluster from those files. This option can be considerably faster than migrating data using `mysqldump`. For more information, see Migrating Data from MySQL by Using an Amazon S3 Bucket (p. 488).

Migrating Data from MySQL by Using an Amazon S3 Bucket

You can copy the full and incremental backup files from your source MySQL version 5.5 or 5.6 database to an Amazon S3 bucket, and then restore an Amazon Aurora MySQL DB cluster from those files.

This option can be considerably faster than migrating data using `mysqldump`, because using `mysqldump` replays all of the commands to recreate the schema and data from your source database in your new Aurora MySQL DB cluster. By copying your source MySQL data files, Aurora MySQL can immediately use those files as the data for an Aurora MySQL DB cluster.

Note
The Amazon S3 bucket and the Amazon Aurora MySQL DB cluster must be in the same region.

Aurora MySQL doesn't restore everything from your database. You should save the database schema and values for the following items from your source MySQL or MariaDB database and add them to your restored Aurora MySQL DB cluster after it has been created.

- User accounts
- Functions
- Stored procedures
- Time zone information. Time zone information is loaded from the local operating system of your Amazon Aurora MySQL DB cluster. For more information, see Local Time Zone for Amazon Aurora DB Clusters (p. 434).

Before You Begin

Before you can copy your data to an Amazon S3 bucket and restore a DB cluster from those files, you must do the following:

- Install Percona XtraBackup on your local server.
- Permit Aurora MySQL to access your Amazon S3 bucket on your behalf.

Installing Percona XtraBackup

Amazon Aurora can restore a DB cluster from files that were created using Percona XtraBackup. You can install Percona XtraBackup from the Percona website.

> **Note**
> You must use Percona XtraBackup version 2.3 or later. Aurora MySQL is not compatible with earlier versions of Percona XtraBackup.

Required Permissions

To migrate your MySQL data to an Amazon Aurora MySQL DB cluster, several permissions are required:

- The user that is requesting that Amazon RDS create a new cluster from an Amazon S3 bucket must have permission to list the buckets for your AWS account. You grant the user this permission using an AWS Identity and Access Management (IAM) policy.
- Amazon RDS requires permission to act on your behalf to access the Amazon S3 bucket where you store the files used to create your Amazon Aurora MySQL DB cluster. You grant Amazon RDS the required permissions using an IAM service role.
- The user making the request must also have permission to list the IAM roles for your AWS account.
- If the user making the request will create the IAM service role, or will request that Amazon RDS create the IAM service role (by using the console), then the user must have permission to create an IAM role for your AWS account.

For example, the following IAM policy grants a user the minimum required permissions to use the console to list IAM roles, create an IAM role, and list the Amazon S3 buckets for your account.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:ListRoles",
                "iam:CreateRole",
                "iam:CreatePolicy",
                "iam:AttachRolePolicy",
```

```
                "s3:ListBucket",
                "s3:ListObjects"
            ],
            "Resource": "*"
        }
    ]
}
```

Additionally, for a user to associate an IAM role with an Amazon S3 bucket, the IAM user must have the `iam:PassRole` permission for that IAM role. This permission allows an administrator to restrict which IAM roles a user can associate with Amazon S3 buckets.

For example, the following IAM policy allows a user to associate the role named `S3Access` with an Amazon S3 bucket.

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"AllowS3AccessRole",
            "Effect":"Allow",
            "Action":"iam:PassRole",
            "Resource":"arn:aws:iam::123456789012:role/S3Access"
        }
    ]
}
```

For more information on IAM user permissions, see Using Identity-Based Policies (IAM Policies) for Amazon RDS (p. 332).

Creating the IAM Service Role

You can have the AWS Management Console create a role for you by choosing the **Create a New Role** option (shown later in this topic). If you select this option and specify a name for the new role, then Amazon RDS creates the IAM service role required for Amazon RDS to access your Amazon S3 bucket with the name that you supply.

As an alternative, you can manually create the role using the following procedure.

To create an IAM role for Amazon RDS to access Amazon S3

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Roles**.
3. Choose **Create New Role**, specify a value for **Role Name** for the new role, and then choose **Next Step**.
4. Under **AWS Service Roles**, find **Amazon RDS** and choose **Select**.
5. Don't select a policy to attach in the **Attach Policy** step. Instead, choose **Next Step**.
6. Review your role information, and then choose **Create Role**.
7. In the list of roles, choose the name of your newly created role. Choose the **Permissions** tab.
8. Choose **Inline Policies**. Because your new role has no policy attached, you are prompted to create one. Click the link to create a new policy.
9. On the **Set Permissions** page, choose **Custom Policy** and then choose **Select**.
10. Type a **Policy Name** such as `S3-bucket-policy`. Add the following code for **Policy Document**, replacing *<bucket name>* with the name of the Amazon S3 bucket that you are allowing access to.

    As part of the policy document, you can also include a file name prefix. If you specify a prefix, then Aurora creates the DB cluster using the files in the Amazon S3 bucket that begin with the specified

prefix. If you don't specify a prefix, then Aurora creates the DB cluster using all of the files in the Amazon S3 bucket.

To specify a prefix, replace *<prefix>* following with the prefix of your file names. Include the asterisk (*) after the prefix. If you don't want to specify a prefix, specify only an asterisk.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::<bucket name>"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::<bucket name>/<prefix>*"
            ]
        }
    ]
}
```

11. Choose **Apply Policy**.

Backing Up Files to be Restored as an Amazon Aurora MySQL DB Cluster

You can create a full backup of your MySQL database files using Percona XtraBackup and upload the backup files to an Amazon S3 bucket. Alternatively, if you already use Percona XtraBackup to back up your MySQL database files, you can upload your existing full and incremental backup directories and files to an Amazon S3 bucket.

Creating a Full Backup With Percona XtraBackup

To create a full backup of your MySQL database files that can be restored from Amazon S3 to create an Amazon Aurora MySQL DB cluster, use the Percona XtraBackup utility (`innobackupex`) to back up your database.

For example, the following command creates a backup of a MySQL database and stores the files in the `/s3-restore/backup` folder.

```
innobackupex --user=myuser --password=<password> --no-timestamp /s3-restore/backup
```

If you want to compress your backup into a single file (which can be split, if needed), you can use the `--stream` option to save your backup in one of the following formats:

- Gzip (.gz)
- tar (.tar)
- Percona xbstream (.xbstream)

The following command creates a backup of your MySQL database split into multiple Gzip files.

```
innobackupex --user=myuser --password=<password> --stream=tar \
   /mydata/s3-restore/backup | gzip - | split -d --bytes=500MB \
   - /mydata/s3-restore/backup/backup.tar.gz
```

The following command creates a backup of your MySQL database split into multiple tar files.

```
innobackupex --user=myuser --password=<password> --stream=tar \
   /mydata/s3-restore/backup | split -d --bytes=500MB \
   - /mydata/s3-restore/backup/backup.tar
```

The following command creates a backup of your MySQL database split into multiple xbstream files.

```
innobackupex --stream=xbstream  \
   /mydata/s3-restore/backup | split -d --bytes=500MB \
   - /mydata/s3-restore/backup/backup.xbstream
```

Once you have backed up your MySQL database using the Percona XtraBackup utility, then you can copy your backup directories and files to an Amazon S3 bucket.

For information on creating and uploading a file to an Amazon S3 bucket, see Getting Started with Amazon Simple Storage Service in the *Amazon S3 Getting Started Guide*.

Using Incremental Backups With Percona XtraBackup

Amazon Aurora MySQL supports both full and incremental backups created using Percona XtraBackup. If you already use Percona XtraBackup to perform full and incremental backups of your MySQL database files, you don't need to create a full backup and upload the backup files to Amazon S3. Instead, you can save a significant amount of time by copying your existing backup directories and files for your full and incremental backups to an Amazon S3 bucket. For more information about creating incremental backups using Percona XtraBackup, see Incremental Backups with innobackupex

When copying your existing full and incremental backup files to an Amazon S3 bucket, you must recursively copy the contents of the base directory. Those contents include the full backup and also all incremental backup directories and files. This copy must preserve the directory structure in the Amazon S3 bucket. Aurora iterates through all files and directories. Aurora uses the `xtrabackup-checkpoints` file included with each incremental backup to identify the base directory and to order incremental backups by log sequence number (LSN) range.

For information on creating and uploading a file to an Amazon S3 bucket, see Getting Started with Amazon Simple Storage Service in the *Amazon S3 Getting Started Guide*.

Backup Considerations

Amazon S3 limits the size of a file uploaded to an Amazon S3 bucket to 5 terabytes (TB). If the backup data for your database exceeds 5 TB, then you must use the `split` command to split the backup files into multiple files that are each less than 5 TB.

Amazon RDS limits the number of source files uploaded to an Amazon S3 bucket to 1 million files. If the backup data for your database, including all full and incremental backups, include a large number of files, use a tarball (.tar.gz) file to store full and incremental backup files in the Amazon S3 bucket.

Aurora consumes your backup files based on the file name. Be sure to name your backup files with the appropriate file extension based on the file format—for example, `.xbstream` for files stored using the Percona xbstream format.

Aurora consumes your backup files in alphabetical order and also in natural number order. Always use the `split` option when you issue the `innobackupex` command to ensure that your backup files are written and named in the proper order.

Aurora doesn't support partial backups created using Percona XtraBackup. You cannot use the `--include`, `--tables-file`, or `--databases` options to create a partial backup when you backup the source files for your database.

Aurora supports incremental backups created using Percona XtraBackup, with or without the `--no-timestamp` option. We recommend that you use the `--no-timestamp` option, to reduce the depth of the directory structure for your incremental backup.

For more information, see The innobackupex Script.

Restoring an Amazon Aurora MySQL DB Cluster from an Amazon S3 Bucket

You can restore your backup files from your Amazon S3 bucket to create a new Amazon Aurora MySQL DB cluster by using the Amazon RDS console.

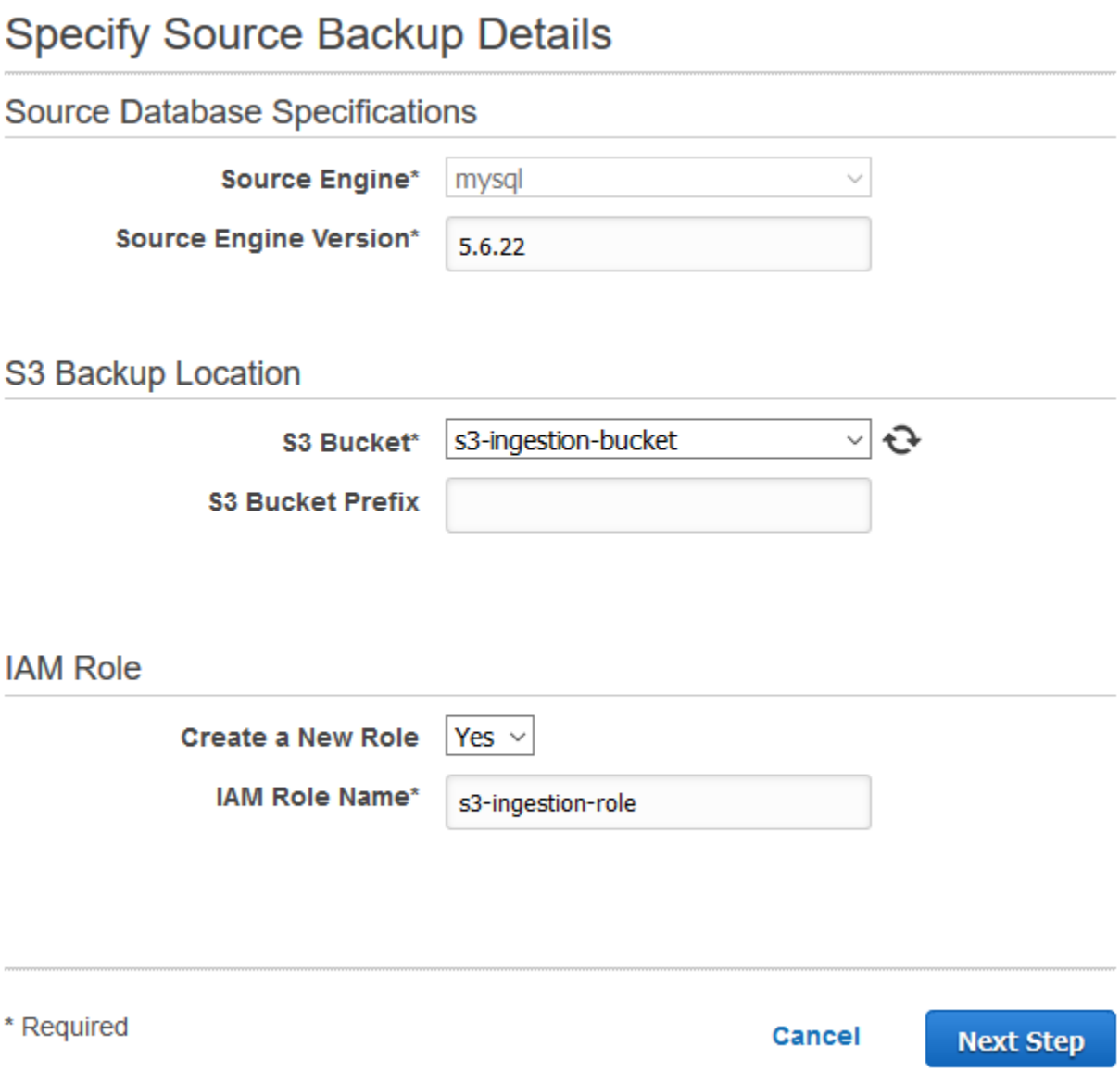
To restore an Amazon Aurora MySQL DB cluster from files on an Amazon S3 bucket

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the RDS dashboard, choose **Restore Aurora DB Cluster from Amazon S3**.
3. In the **Specify Source Backup Details**, specify the following:

For This Option	Do This
Source Engine	Aurora MySQL currently supports only restoring from backup files for the `mysql` database engine.
Source Engine Version	Specify the version of the MySQL database that the backup files were created from. MySQL version 5.5 and 5.6 are supported.
S3 Bucket	Select the Amazon S3 bucket where your backup files are stored.
S3 Bucket Prefix	(Optional) Specify a file path prefix for the files stored in your Amazon S3 bucket. The **S3 Bucket Prefix** is optional. If you don't specify a prefix, then Aurora MySQL creates the DB cluster using all of the files and folders in the root folder of the Amazon S3 bucket. If you specify a prefix, then Aurora MySQL creates the DB cluster using the files and folders in the Amazon S3 bucket where the full path for the file begins with the specified prefix. Aurora doesn't traverse other subfolders in your Amazon S3 bucket looking for backup files. Only the files from the folder identified by the **S3 Bucket Prefix** are used. If you store your backup files in a subfolder in your Amazon S3 bucket, then you must specify a prefix that identifies the full path to the folder where the files are stored. For example, suppose that you store your backup files in a subfolder of your Amazon S3 bucket named `backups`, and you have multiple sets of backup files, each in its own directory (`gzip_backup1`, `gzip_backup2`, and so on.) In this case, you specify a prefix of `backups/gzip_backup1` to restore from the files in the `gzip_backup1` folder.

For This Option	Do This
Create a New Role	Choose **Yes** to create a new IAM role, or **No** to select an existing IAM role, to authorize Aurora to access Amazon S3 on your behalf. For more information, see Required Permissions (p. 489).
IAM Role Name	This option is available only if **Create a New Role** is set to **Yes**. Specify the name of the new IAM role to be created. The new role is used to authorize Amazon Aurora to access Amazon S3 on your behalf. For more information, see Required Permissions (p. 489).
IAM Role	This option is available only if **Create a New Role** is set to **No**. Select the IAM role that you created to authorize Aurora to access Amazon S3 on your behalf. If you have not created an IAM role, you can instead set **Create a New Role** to **Yes** to create one. For more information, see Required Permissions (p. 489).

A typical **Specify Source Backup Details** page looks like the following.

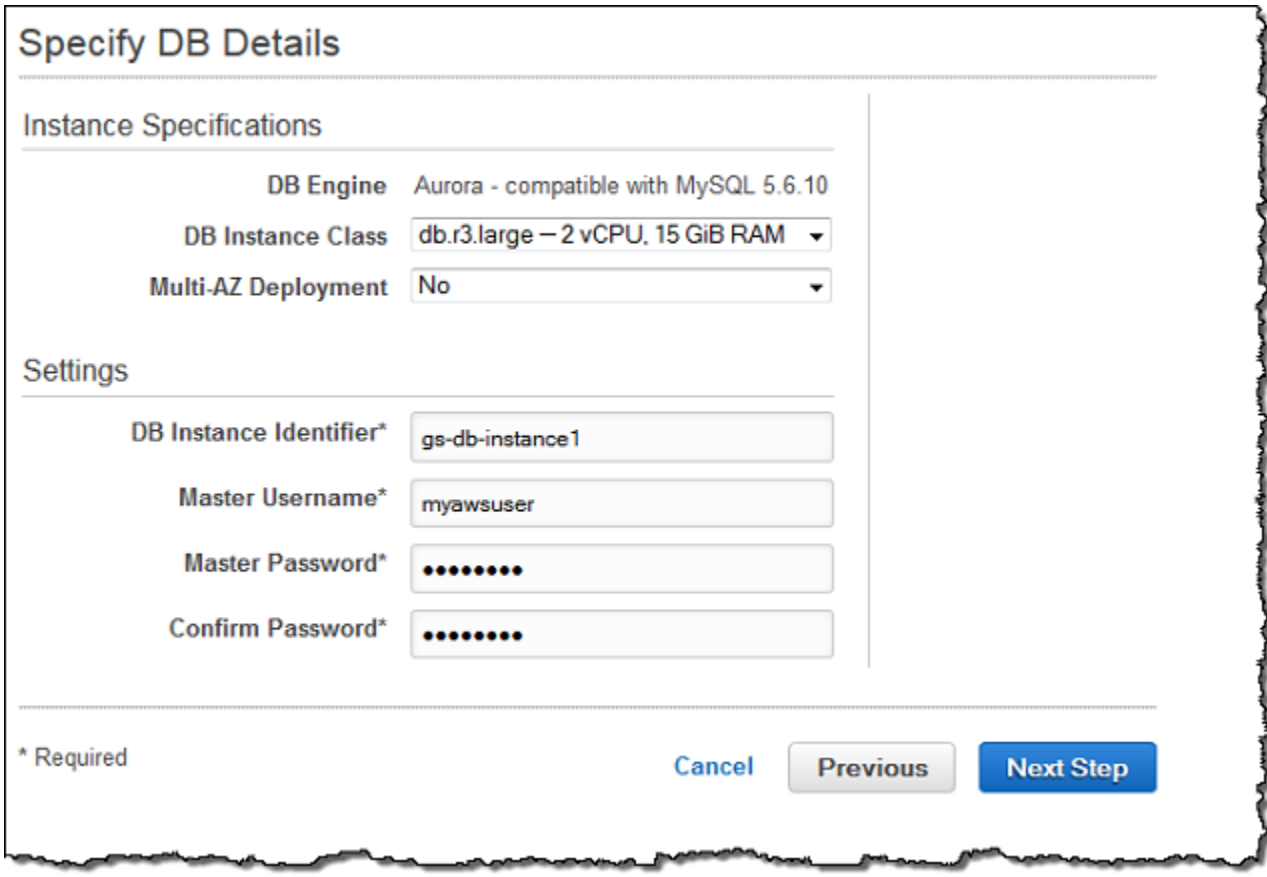


4. Choose **Next Step**.
5. On the **Specify DB Details** page, specify your DB cluster information. The following table shows settings for a DB instance.

For This Option	Do This
DB Instance Class	Select a DB instance class that defines the processing and memory requirements for each instance in the DB cluster. Aurora supports the `db.t2.small`, `db.t2.medium`, `db.r3.large`, `db.r3.xlarge`, `db.r3.2xlarge`, `db.r3.4xlarge`, and `db.r3.8xlarge` DB instance classes. For more information about DB instance class options, see DB Instance Class (p. 92).

For This Option	Do This
Multi-AZ Deployment	Determine if you want to create Aurora Replicas in other Availability Zones for failover support. For more information about multiple Availability Zones, see Regions and Availability Zones (p. 97).
DB Instance Identifier	Type a name for the primary instance in your DB cluster. This identifier is used in the endpoint address for the primary instance of your DB cluster. The DB instance identifier has the following constraints: • It must contain from 1 to 63 alphanumeric characters or hyphens. • Its first character must be a letter. • It cannot end with a hyphen or contain two consecutive hyphens. • It must be unique for all DB instances per AWS account, per region.
Master Username	Type a name using alphanumeric characters that you will use as the master user name to log on to your DB cluster.
Master Password	Type a password that contains from 8 to 41 printable ASCII characters (excluding /,", and @) for your master user password.

A typical **Specify DB Details** page looks like the following.



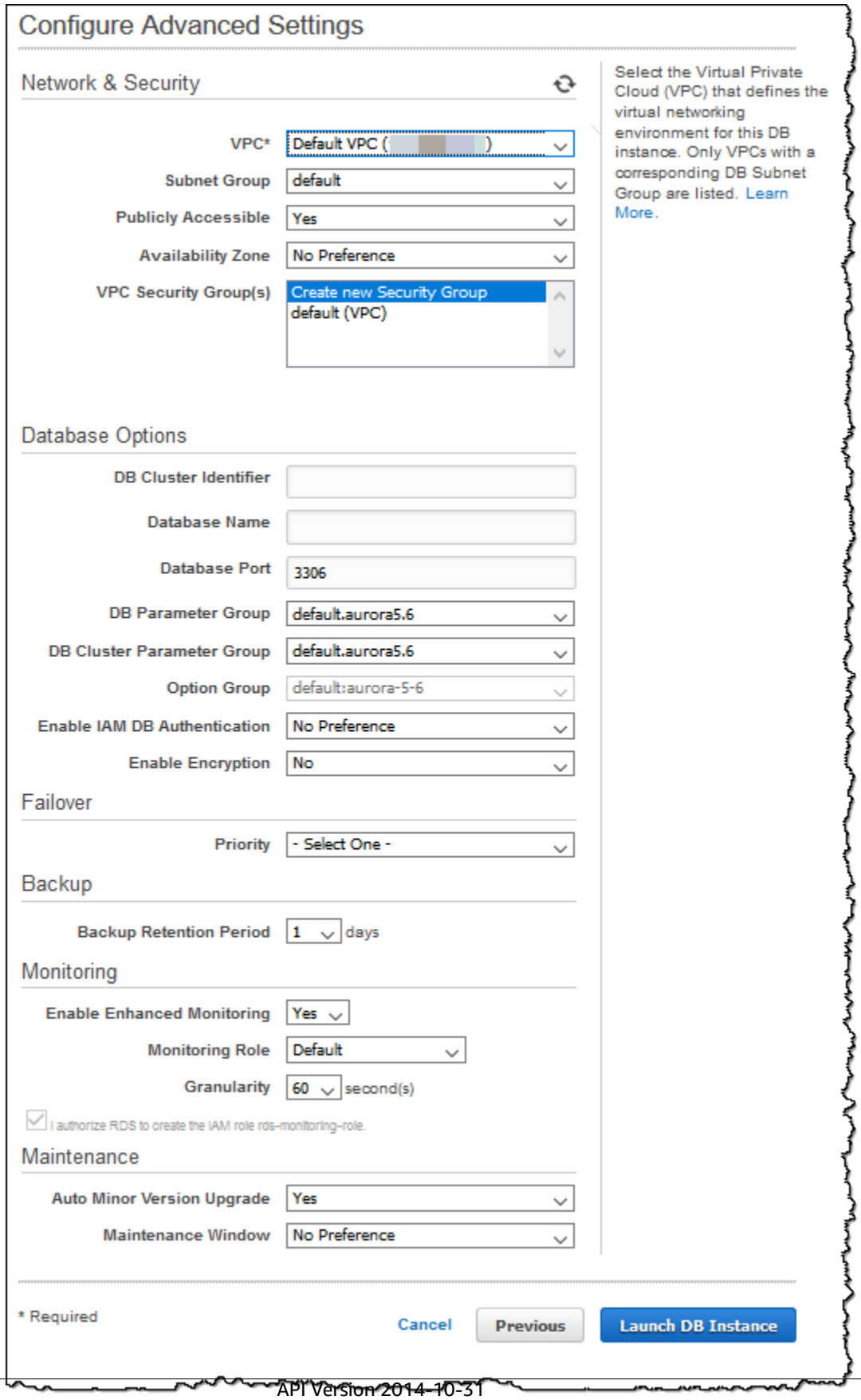
6. Confirm your master password, and then choose **Next**.
7. On the **Configure Advanced Settings** page, you can customize additional settings for your Aurora MySQL DB cluster. The following table shows the advanced settings for a DB cluster.

For This Option	Do This
VPC	Select the VPC that will host the DB cluster. Select **Create a New VPC** to have Amazon RDS create a VPC for you. For more information, see DB Cluster Prerequisites (p. 437) earlier in this topic.
Subnet Group	Select the DB subnet group to use for the DB cluster. Select **Create a New DB Subnet Group** to have Amazon RDS create a DB subnet group for you. For more information, see DB Cluster Prerequisites (p. 437) earlier in this topic.
Publicly Accessible	Select **Yes** to give the DB cluster a public IP address; otherwise, select **No**. The instances in your DB cluster can be a mix of both public and private DB instances. For more information about hiding instances from public access, see Hiding a DB Instance in a VPC from the Internet (p. 401).

For This Option	Do This
Availability Zone	Determine if you want to specify a particular Availability Zone. For more information about Availability Zones, see Regions and Availability Zones (p. 97).
VPC Security Group(s)	Select one or more VPC security groups to secure network access to the DB cluster. Select **Create a New VPC Security Group** to have Amazon RDS create a VPC security group for you. For more information, see DB Cluster Prerequisites (p. 437) earlier in this topic.
DB Cluster Identifier	Type a name for your DB cluster that is unique for your account in the region you selected. This identifier is used in the cluster endpoint address for your DB cluster. For information on the cluster endpoint, see Aurora Endpoints (p. 431). The DB cluster identifier has the following constraints: • It must contain from 1 to 63 alphanumeric characters or hyphens. • Its first character must be a letter. • It cannot end with a hyphen or contain two consecutive hyphens. • It must be unique for all DB clusters per AWS account, per region.
Database Name	Type a name for your database of up to 64 alphanumeric characters. If you don't provide a name, Aurora won't create a database on the DB cluster you are creating.
Database Port	Specify the port that applications and utilities will use to access the database. Aurora MySQL DB clusters default to the default MySQL port, 3306. The firewalls at some companies block connections to the default MySQL port. If your company firewall blocks the default port, choose another port for the new DB cluster.
DB Parameter Group	Select a DB parameter group. Aurora has a default DB parameter group you can use, or you can create your own DB parameter group. For more information about DB parameter groups, see Working with DB Parameter Groups (p. 170).
DB Cluster Parameter Group	Select a DB cluster parameter group. Aurora has a default DB cluster parameter group you can use, or you can create your own DB cluster parameter group. For more information about DB cluster parameter groups, see Working with DB Parameter Groups (p. 170).
Option Group	Select an option group. Aurora has a default option group you can use, or you can create your own option group. For more information about option groups, see Working with Option Groups (p. 153).

For This Option	Do This
Enable Encryption	Select **Yes** to enable encryption at rest for this DB cluster. For more information, see Encrypting Amazon RDS Resources (p. 355).
Priority	Choose a failover priority for the instance. If you don't select a value, the default is **tier-1**. This priority determines the order in which Aurora Replicas are promoted when recovering from a primary instance failure. For more information, see Fault Tolerance for an Aurora DB Cluster (p. 468).
Backup Retention Period	Select the length of time, from 1 to 35 days, that Aurora will retain backup copies of the database. Backup copies can be used for point-in-time restores (PITR) of your database, timed down to the second.
Enable Enhanced Monitoring	Choose **Yes** to enable gathering metrics in real time for the operating system that your DB cluster runs on. For more information, see Enhanced Monitoring (p. 258).
Monitoring Role	This option is only available if **Enable Enhanced Monitoring** is set to **Yes**. Select the IAM role that you created to permit Amazon RDS to communicate with Amazon CloudWatch Logs for you.
Granularity	This option is only available if **Enable Enhanced Monitoring** is set to **Yes**. Set the interval, in seconds, between times at which metrics are collected for your DB cluster.
Auto Minor Version Upgrade	This option is not applicable to Amazon Aurora. You can ignore it.
Maintenance Window	Select the weekly time range during which system maintenance can occur.

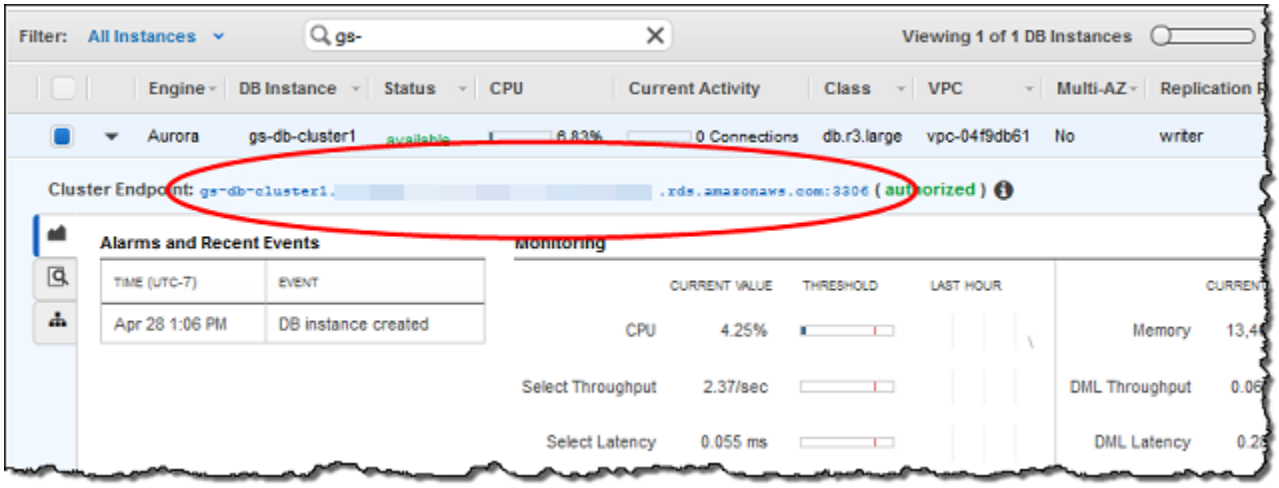
A typical **Configure Advanced Settings** page looks like the following.


Configure Advanced Settings
Network & Security
VPC*
Default VPC (
)
Subnet Group
default
Publicly Accessible
Yes
Availability Zone
No Preference
VPC Security Group(s)
Create new Security Group
default (VPC)
Select the Virtual Private Cloud (VPC) that defines the virtual networking environment for this DB instance. Only VPCs with a corresponding DB Subnet Group are listed. Learn More.
Database Options
DB Cluster Identifier
Database Name
Database Port
3306
DB Parameter Group
default.aurora5.6
DB Cluster Parameter Group
default.aurora5.6
Option Group
default:aurora-5-6
Enable IAM DB Authentication
No Preference
Enable Encryption
No
Failover
Priority
- Select One -
Backup
Backup Retention Period
1
days
Monitoring
Enable Enhanced Monitoring
Yes
Monitoring Role
Default
Granularity
60
second(s)
I authorize RDS to create the IAM role rds-monitoring-role.
Maintenance
Auto Minor Version Upgrade
Yes
Maintenance Window
No Preference
* Required
Cancel
Previous
Launch DB Instance

8. Choose **Launch DB Instance** to launch your Aurora DB instance, and then choose **Close** to close the wizard.

On the Amazon RDS console, the new DB instance appears in the list of DB instances. The DB instance has a status of **creating** until the DB instance is created and ready for use. When the state changes to **available**, you can connect to the primary instance for your DB cluster. Depending on the DB instance class and store allocated, it can take several minutes for the new instance to be available.

To view the newly created cluster, choose the **Clusters** view in the Amazon RDS console. For more information, see Viewing an Amazon Aurora DB Cluster (p. 461).



Note the port and the endpoint of the cluster. Use the endpoint and port of the cluster in your JDBC and ODBC connection strings for any application that performs write or read operations.

Migrating from MySQL to Amazon Aurora by Using mysqldump

Because Amazon Aurora MySQL is a MySQL-compatible database, you can use the `mysqldump` utility to copy data from your MySQL or MariaDB database to an existing Aurora MySQL DB cluster. For a discussion of how to do so with MySQL databases that are very large, see Importing Data to an Amazon RDS MySQL or MariaDB DB Instance with Reduced Downtime (p. 873). For MySQL databases that have smaller amounts of data, see Importing Data from a MySQL or MariaDB DB to an Amazon RDS MySQL or MariaDB DB Instance (p. 872).

Related Topics

- Amazon Aurora on Amazon RDS (p. 428)
- Migrating Data to an Amazon Aurora DB Cluster (p. 466)

Migrating Data from a MySQL DB Instance to an Amazon Aurora MySQL DB Cluster by Using a DB Snapshot

You can migrate (copy) data to an Amazon Aurora MySQL DB cluster from an Amazon RDS MySQL DB snapshot, as described following.

Topics

- Migrating an RDS MySQL Snapshot to Aurora (p. 502)
- Migrating Data from a MySQL DB Instance to an Amazon Aurora MySQL DB Cluster by Using an Aurora Read Replica (p. 507)

Note
Because Amazon Aurora MySQL is compatible with MySQL, you can migrate data from your MySQL database by setting up replication between your MySQL database and an Amazon Aurora MySQL DB cluster. If you want to use replication to migrate data from your MySQL database, we recommend that your MySQL database run MySQL version 5.5 or later. For more information, see Amazon Aurora Replication (p. 432).

Migrating an RDS MySQL Snapshot to Aurora

You can migrate a DB snapshot of an Amazon RDS MySQL DB instance to create an Aurora MySQL DB cluster. The new Aurora MySQL DB cluster is populated with the data from the original Amazon RDS MySQL DB instance. The DB snapshot must have been made from an Amazon RDS DB instance running MySQL version 5.6.

You can migrate either a manual or automated DB snapshot. After the DB cluster is created, you can then create optional Aurora Replicas.

The general steps you must take are as follows:

1. Determine the amount of space to provision for your Aurora MySQL DB cluster. For more information, see How Much Space Do I Need? (p. 502)
2. Use the console to create the snapshot in the region where the Amazon RDS MySQL 5.6 instance is located. For information about creating a DB snapshot, see Creating a DB Snapshot.
3. If the DB snapshot is not in the same region as your DB cluster, use the Amazon RDS console to copy the DB snapshot to that region. For information about copying a DB snapshot, see Copying a DB Snapshot.
4. Use the console to migrate the DB snapshot and create an Aurora MySQL DB cluster with the same databases as the original MySQL DB instance.

Warning
Amazon RDS limits each AWS account to one snapshot copy into each AWS Region at a time.

How Much Space Do I Need?

When you migrate a snapshot of a MySQL DB instance into an Aurora MySQL DB cluster, Aurora uses an Amazon Elastic Block Store (Amazon EBS) volume to format the data from the snapshot before migrating it. In some cases, additional space is needed to format the data for migration. When migrating data into your DB cluster, observe the following guidelines and limitations:

- Although Aurora supports storage up to 64 TB in size, the process of migrating a snapshot into an Aurora MySQL DB cluster is limited by the size of the Amazon EBS volume of the snapshot. Thus, the maximum size for a snapshot that you can migrate is 6 TB.
- Tables that are not MyISAM tables and are not compressed can be up to 6 TB in size. If you have MyISAM tables, then Aurora must use additional space in the volume to convert the tables to be compatible with Aurora MySQL. If you have compressed tables, then Aurora must use additional space in the volume to expand these tables before storing them on the Aurora cluster volume. Because of this additional space requirement, you should ensure that none of the MyISAM and compressed tables being migrated from your MySQL DB instance exceeds 3 TB in size.

Reducing the Amount of Space Required to Migrate Data into Amazon Aurora MySQL

You might want to modify your database schema prior to migrating it into Amazon Aurora. Such modification can be helpful in the following cases:

- You want to speed up the migration process.
- You are unsure of how much space you need to provision.

- You have attempted to migrate your data and the migration has failed due to a lack of provisioned space.

You can make the following changes to improve the process of migrating a database into Amazon Aurora.

Important
Be sure to perform these updates on a new DB instance restored from a snapshot of a production database, rather than on a production instance. You can then migrate the data from the snapshot of your new DB instance into your Aurora DB cluster to avoid any service interruptions on your production database.

Table Type	Limitation or Guideline
MyISAM tables	Aurora MySQL supports InnoDB tables only. If you have MyISAM tables in your database, then those tables must be converted before being migrated into Aurora MySQL. The conversion process requires additional space for the MyISAM to InnoDB conversion during the migration procedure. To reduce your chances of running out of space or to speed up the migration process, convert all of your MyISAM tables to InnoDB tables before migrating them. The size of the resulting InnoDB table is equivalent to the size required by Aurora MySQL for that table. To convert a MyISAM table to InnoDB, run the following command: `alter table <schema>.<table_name> engine=innodb, algorithm=copy;`
Compressed tables	Aurora MySQL doesn't support compressed tables (that is, tables created with `ROW_FORMAT=COMPRESSED`). To reduce your chances of running out of space or to speed up the migration process, expand your compressed tables by setting `ROW_FORMAT` to `DEFAULT`, `COMPACT`, `DYNAMIC`, or `REDUNDANT`. For more information, see https://dev.mysql.com/doc/refman/5.6/en/innodb-row-format.html.

You can use the following SQL script on your existing MySQL DB instance to list the tables in your database that are MyISAM tables or compressed tables.

```
-- This script examines a MySQL database for conditions that will block
-- migrating the database into Amazon's Aurora DB.
-- It needs to be run from an account that has read permission for the
-- INFORMATION_SCHEMA database.

-- Verify that this is a supported version of MySQL.

select msg as `==> Checking current version of MySQL.`
from
  (
  select
    'This script should be run on MySQL version 5.6. ' +
    'Earlier versions are not supported.' as msg,
    cast(substring_index(version(), '.', 1) as unsigned) * 100 +
      cast(substring_index(substring_index(version(), '.', 2), '.', -1)
      as unsigned)
    as major_minor
```

```
  ) as T
where major_minor <> 506;


-- List MyISAM and compressed tables. Include the table size.

select concat(TABLE_SCHEMA, '.', TABLE_NAME) as `==> MyISAM or Compressed Tables`,
round(((data_length + index_length) / 1024 / 1024), 2) "Approx size (MB)"
from INFORMATION_SCHEMA.TABLES
where
  ENGINE <> 'InnoDB'
  and
  (
    -- User tables
    TABLE_SCHEMA not in ('mysql', 'performance_schema',
                         'information_schema')
    or
    -- Non-standard system tables
    (
      TABLE_SCHEMA = 'mysql' and TABLE_NAME not in
        (
          'columns_priv', 'db', 'event', 'func', 'general_log',
          'help_category', 'help_keyword', 'help_relation',
          'help_topic', 'host', 'ndb_binlog_index', 'plugin',
          'proc', 'procs_priv', 'proxies_priv', 'servers', 'slow_log',
          'tables_priv', 'time_zone', 'time_zone_leap_second',
          'time_zone_name', 'time_zone_transition',
          'time_zone_transition_type', 'user'
        )
    )
  )
  or
  (
    -- Compressed tables
       ROW_FORMAT = 'Compressed'
  );
```

The script produces output similar to the output in the following example. The example shows two tables that must be converted from MyISAM to InnoDB. The output also includes the approximate size of each table in megabytes (MB).

```
+---------------------------------+------------------+
| ==> MyISAM or Compressed Tables | Approx size (MB) |
+---------------------------------+------------------+
| test.name_table                 |          2102.25 |
| test.my_table                   |            65.25 |
+---------------------------------+------------------+
2 rows in set (0.01 sec)
```

AWS Management Console

You can migrate a DB snapshot of an Amazon RDS MySQL DB instance to create an Aurora MySQL DB cluster. The new Aurora MySQL DB cluster will be populated with the data from the original Amazon RDS MySQL DB instance. The DB snapshot must have been made from an Amazon RDS DB instance running MySQL 5.6. For information about creating a DB snapshot, see Creating a DB Snapshot.

If the DB snapshot is not in the AWS Region where you want to locate your data, use the Amazon RDS console to copy the DB snapshot to that AWS Region. For information about copying a DB snapshot, see Copying a DB Snapshot.

When you migrate the DB snapshot by using the AWS Management Console, the console takes the actions necessary to create both the DB cluster and the primary instance.

You can also choose for your new Aurora MySQL DB cluster to be encrypted at rest using an AWS Key Management Service (AWS KMS) encryption key.

To migrate a MySQL 5.6 DB snapshot by using the AWS Management Console

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. Either start the migration from the MySQL DB instance or from the snapshot:

   To start the migration from the DB instance:

   1. In the navigation pane, choose **DB Instances**, and then select the MySQL DB instance.
   2. Choose **Instance Actions**, and then choose **Migrate latest snapshot**.

   To start the migration from the snapshot:

   1. Choose **Snapshots**.
   2. On the **Snapshots** page, choose the snapshot that you want to migrate into an Aurora MySQL DB cluster.
   3. Choose **Migrate Database**.

   The **Migrate Database** page appears.
3. Set the following values on the **Migrate Database** page:

   - **Migrate to DB Engine**: Select `aurora`.
   - **DB Engine Version**: Select the DB engine version for the Aurora MySQL DB cluster.
   - **DB Instance Class**: Select a DB instance class that has the required storage and capacity for your database, for example `db.r3.large`. Aurora cluster volumes automatically grow as the amount of data in your database increases, up to a maximum size of 64 terabytes (TB). So you only need to select a DB instance class that meets your current storage requirements. For more information, see Amazon Aurora Storage (p. 432).
   - **DB Instance Identifier**: Type a name for the DB cluster that is unique for your account in the region you selected. This identifier is used in the endpoint addresses for the instances in your DB cluster. You might choose to add some intelligence to the name, such as including the region and DB engine you selected, for example **`aurora-cluster1`**.

     The DB instance identifier has the following constraints:
     - It must contain from 1 to 63 alphanumeric characters or hyphens.
     - Its first character must be a letter.
     - It cannot end with a hyphen or contain two consecutive hyphens.
     - It must be unique for all DB instances per AWS account, per AWS Region.
   - **Virtual Private Cloud (VPC)**: If you have an existing VPC, then you can use that VPC with your Aurora MySQL DB cluster by selecting your VPC identifier, for example `vpc-a464d1c1`. For information on using an existing VPC, see How to Create a VPC for Use with Amazon Aurora (p. 452).

     Otherwise, you can choose to have Amazon RDS create a VPC for you by selecting **Create a new VPC**.
   - **Subnet group**: If you have an existing subnet group, then you can use that subnet group with your Aurora MySQL DB cluster by selecting your subnet group identifier, for example `gs-subnet-group1`.

     Otherwise, you can choose to have Amazon RDS create a subnet group for you by selecting **Create a new subnet group**.

- **Public accessibility**: Select **No** to specify that instances in your DB cluster can only be accessed by resources inside of your VPC. Select **Yes** to specify that instances in your DB cluster can be accessed by resources on the public network. The default is **Yes**.

  Note
  Your production DB cluster might not need to be in a public subnet, because only your application servers will require access to your DB cluster. If your DB cluster doesn't need to be in a public subnet, set **Publicly Accessible** to **No**.
- **Availability Zone**: Select the Availability Zone to host the primary instance for your Aurora MySQL DB cluster. To have Amazon RDS select an Availability Zone for you, select **No Preference**.
- **Database Port**: Type the default port to be used when connecting to instances in the Aurora MySQL DB cluster. The default is `3306`.

  Note
  You might be behind a corporate firewall that doesn't allow access to default ports such as the MySQL default port, 3306. In this case, provide a port value that your corporate firewall allows. Remember that port value later when you connect to the Aurora MySQL DB cluster.
- **Encryption**: Choose **Enable Encryption** for your new Aurora MySQL DB cluster to be encrypted at rest. If you choose **Enable Encryption**, you will be required to choose an AWS KMS encryption key as the **Master Key** value.

  If your DB snapshot isn't encrypted, specify an encryption key to have your DB cluster encrypted at rest.

  If your DB snapshot is encrypted, specify an encryption key to have your DB cluster encrypted at rest using the specified encryption key. You can specify the encryption key used by the DB snapshot or a different key. You can't create an unencrypted DB cluster from an encrypted DB snapshot.
- **Auto Minor Version Upgrade**: Select **Yes** if you want to enable your Aurora MySQL DB cluster to receive minor MySQL DB engine version upgrades automatically when they become available.

  The **Auto Minor Version Upgrade** option only applies to upgrades to MySQL minor engine versions for your Amazon Aurora MySQL DB cluster. It doesn't apply to regular patches applied to maintain system stability.

4. Choose **Migrate** to migrate your DB snapshot.
5. Choose **Instances**, and then choose the arrow icon to show the DB cluster details and monitor the progress of the migration. On the details page, you will find the cluster endpoint used to connect to the primary instance of the DB cluster. For more information on connecting to an Aurora MySQL DB cluster, see Connecting to an Amazon Aurora DB Cluster (p. 457).

CLI

You can migrate a DB snapshot of an Amazon RDS MySQL DB instance to create an Aurora DB cluster. The new DB cluster is then populated with the data from the DB snapshot. The DB snapshot must come from an Amazon RDS DB instance running MySQL 5.6. For more information, see Creating a DB Snapshot.

If the DB snapshot is not in the AWS Region where you want to locate your data, copy the DB snapshot to that region. For more information, see Copying a DB Snapshot.

You can create an Aurora DB cluster from a DB snapshot of an Amazon RDS MySQL DB instance by using the `restore-db-cluster-from-snapshot` command with the following parameters:

- `--db-cluster-identifier`

  The name of the DB cluster to create.

- `--engine aurora`
- `--kms-key-id`

  The AWS Key Management Service (AWS KMS) encryption key to optionally encrypt the DB cluster with, depending on whether your DB snapshot is encrypted.

  - If your DB snapshot isn't encrypted, specify an encryption key to have your DB cluster encrypted at rest. Otherwise, your DB cluster isn't encrypted.
  - If your DB snapshot is encrypted, specify an encryption key to have your DB cluster encrypted at rest using the specified encryption key. Otherwise, your DB cluster is encrypted at rest using the encryption key for the DB snapshot.

    Note
    You can't create an unencrypted DB cluster from an encrypted DB snapshot.
- `--snapshot-identifier`

  The Amazon Resource Name (ARN) of the DB snapshot to migrate. For more information about Amazon RDS ARNs, see Amazon Relational Database Service (Amazon RDS).

When you migrate the DB snapshot by using the `RestoreDBClusterFromSnapshot` command, the command creates both the DB cluster and the primary instance.

In this example, you create a DB cluster named *`mydbcluster`* from a DB snapshot with an ARN set to *`mydbsnapshotARN`*.

For Linux, OS X, or Unix:

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier mydbcluster \
    --snapshot-identifier mydbsnapshotARN \
    --engine aurora
```

For Windows:

```
aws rds restore-db-cluster-from-snapshot ^
    --db-cluster-identifier mydbcluster ^
    --snapshot-identifier mydbsnapshotARN ^
    --engine aurora
```

Related Topics

- Amazon Aurora on Amazon RDS (p. 428)
- Migrating Data to an Amazon Aurora DB Cluster (p. 466)

Migrating Data from a MySQL DB Instance to an Amazon Aurora MySQL DB Cluster by Using an Aurora Read Replica

Amazon RDS uses the MySQL DB engines' binary log replication functionality to create a special type of DB cluster called an Aurora Read Replica for a source MySQL DB instance. Updates made to the source MySQL DB instance are asynchronously replicated to the Aurora Read Replica.

We recommend using this functionality to migrate from a MySQL DB instance to an Aurora MySQL DB cluster by creating an Aurora Read Replica of your source MySQL DB instance. When the replica lag between the MySQL DB instance and the Aurora Read Replica is 0, you can direct your client applications to the Aurora Read Replica and then stop replication to make the Aurora Read Replica a standalone

Aurora MySQL DB cluster. Be prepared for migration to take a while, roughly several hours per terabyte (TB) of data.

For a list of regions where Aurora is available, see Availability for Amazon Aurora MySQL (p. 484).

When you create an Aurora Read Replica of a MySQL DB instance, Amazon RDS creates a DB snapshot of your source MySQL DB instance (private to Amazon RDS, and incurring no charges). Amazon RDS then migrates the data from the DB snapshot to the Aurora Read Replica. After the data from the DB snapshot has been migrated to the new Aurora MySQL DB cluster, Amazon RDS starts replication between your MySQL DB instance and the Aurora MySQL DB cluster. If your MySQL DB instance contains tables that use storage engines other than InnoDB, or that use compressed row format, you can speed up the process of creating an Aurora Read Replica by altering those tables to use the InnoDB storage engine and dynamic row format before you create your Aurora Read Replica. For more information about the process of copying a MySQL DB snapshot to an Aurora MySQL DB cluster, see Migrating Data from a MySQL DB Instance to an Amazon Aurora MySQL DB Cluster by Using a DB Snapshot (p. 501).

You can only have one Aurora Read Replica for a MySQL DB instance.

> **Note**
> Replication issues can arise due to feature differences between Amazon Aurora MySQL and the MySQL database engine version of your RDS MySQL DB instance that is the replication master. If you encounter an error, you can find help in the Amazon RDS Community Forum or by contacting AWS Support.

For more information on MySQL Read Replicas, see Working with PostgreSQL, MySQL, and MariaDB Read Replicas (p. 134).

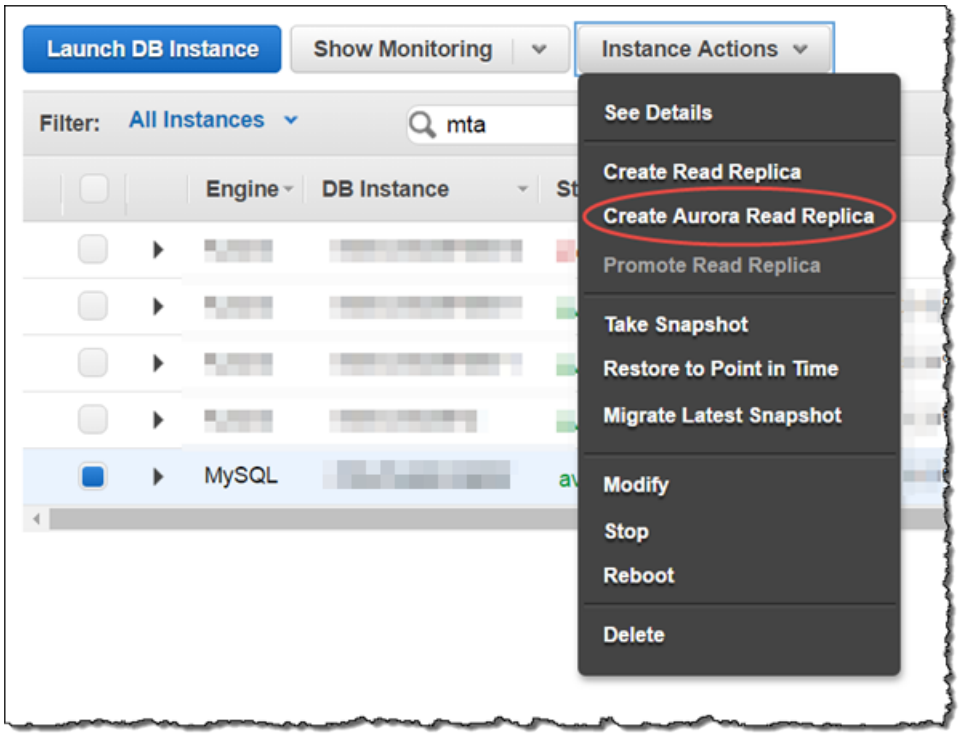
Creating an Aurora Read Replica

You can create an Aurora Read Replica for a MySQL DB instance by using the console or the AWS CLI.

AWS Management Console

To create an Aurora Read Replica from a source MySQL DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Instances**.
3. Choose the MySQL DB instance that you want to use as the source for your Aurora Read Replica and choose **Create Aurora Read Replica** from **Instance Actions**.



4. Choose the DB cluster specifications you want to use for the Aurora Read Replica, as described in the following table.

Option	Description
DB Instance Class	Choose a DB instance class that defines the processing and memory requirements for the primary instance in the DB cluster. For more information about DB instance class options, see DB Instance Class (p. 92).
Multi-AZ Deployment	Choose **Create Replica in Different Zone** to create a standby replica of the new DB cluster in another Availability Zone in the target region for failover support. For more information about multiple Availability Zones, see Regions and Availability Zones (p. 97).
DB Instance Identifier	Type a name for the primary instance in your Aurora Read Replica DB cluster. This identifier is used in the endpoint address for the primary instance of the new DB cluster. The DB instance identifier has the following constraints: • It must contain from 1 to 63 alphanumeric characters or hyphens. • Its first character must be a letter. • It cannot end with a hyphen or contain two consecutive hyphens. • It must be unique for all DB instances for each AWS account, for each region.

Option	Description
	Because the Aurora Read Replica DB cluster is created from a snapshot of the source DB instance, the master user name and master password for the Aurora Read Replica are the same as the master user name and master password for the source DB instance.
VPC	Select the VPC that will host the DB cluster. Select **Create a New VPC** to have Amazon RDS create a VPC for you. For more information, see DB Cluster Prerequisites (p. 437).
Subnet Group	Select the DB subnet group to use for the DB cluster. Select **Create a New DB Subnet Group** to have Amazon RDS create a DB subnet group for you. For more information, see DB Cluster Prerequisites (p. 437).
Publicly Accessible	Select `Yes` to give the DB cluster a public IP address; otherwise, select `No`. The instances in your DB cluster can be a mix of both public and private DB instances. For more information about hiding instances from public access, see Hiding a DB Instance in a VPC from the Internet (p. 401).
Availability Zone	Determine if you want to specify a particular Availability Zone. For more information about Availability Zones, see Regions and Availability Zones (p. 97).
VPC Security Group(s)	Select one or more VPC security groups to secure network access to the DB cluster. Select **Create a New VPC Security Group** to have Amazon RDS create a VPC security group for you. For more information, see DB Cluster Prerequisites (p. 437).
DB Cluster Identifier	Type a name for your Aurora Read Replica DB cluster that is unique for your account in the target AWS Region for your replica. This identifier will be used in the cluster endpoint address for your DB cluster. For information on the cluster endpoint, see Aurora Endpoints (p. 431). The DB cluster identifier has the following constraints: • It must contain from 1 to 63 alphanumeric characters or hyphens. • Its first character must be a letter. • It cannot end with a hyphen or contain two consecutive hyphens. • It must be unique for all DB clusters for each AWS account, for each region.
Database Port	Specify the port that applications and utilities will use to access the database. Aurora MySQL DB clusters default to the default MySQL port, 3306. Firewalls at some companies block connections to this port. If your company firewall blocks the default port, choose another port for the new DB cluster.

Option	Description
DB Parameter Group	Select a DB parameter group for the Aurora MySQL DB cluster. Aurora has a default DB parameter group you can use, or you can create your own DB parameter group. For more information about DB parameter groups, see Working with DB Parameter Groups (p. 170).
DB Cluster Parameter Group	Select a DB cluster parameter group for the Aurora MySQL DB cluster. Aurora has a default DB cluster parameter group you can use, or you can create your own DB cluster parameter group. For more information about DB cluster parameter groups, see Working with DB Parameter Groups (p. 170).
Enable Encryption	Choose **Yes** for your new Aurora DB cluster to be encrypted at rest. If you choose **Yes**, you will be required to choose an AWS KMS encryption key as the **Master Key** value. If your MySQL DB instance isn't encrypted, specify an encryption key to have your DB cluster encrypted at rest. If your MySQL DB instance is encrypted, specify an encryption key to have your DB cluster encrypted at rest using the specified encryption key. You can specify the encryption key used by the MySQL DB instance or a different key. You can't create an unencrypted DB cluster from an encrypted MySQL DB instance.
Priority	Choose a failover priority for the DB cluster. If you don't select a value, the default is **tier-1**. This priority determines the order in which Aurora Replicas are promoted when recovering from a primary instance failure. For more information, see Fault Tolerance for an Aurora DB Cluster (p. 468).
Backup Retention Period	Select the length of time, from 1 to 35 days, that Aurora will retain backup copies of the database. Backup copies can be used for point-in-time restores (PITR) of your database down to the second.
Enable Enhanced Monitoring	Choose **`Yes`** to enable gathering metrics in real time for the operating system that your DB cluster runs on. For more information, see Enhanced Monitoring (p. 258).
Monitoring Role	The IAM role to use for Enhanced Monitoring. For more information, see Setting Up for and Enabling Enhanced Monitoring (p. 258).
Granularity	Only available if **Enable Enhanced Monitoring** is set to **`Yes`**. Set the interval, in seconds, between when metrics are collected for your DB cluster.

Option	Description
Auto Minor Version Upgrade	Select `Yes` if you want to enable your Aurora MySQL DB cluster to receive minor MySQL DB Engine version upgrades automatically when they become available. The **Auto Minor Version Upgrade** option only applies to upgrades to MySQL minor engine versions for your Aurora MySQL DB cluster. It doesn't apply to regular patches applied to maintain system stability.
Maintenance Window	Select the weekly time range during which system maintenance can occur.

5. Choose **Create Read Replica**.

CLI

To create an Aurora Read Replica from a source MySQL DB instance, use the `create-db-cluster` and **create-db-instance** AWS CLI commands to create a new Aurora MySQL DB cluster. When you call the **create-db-cluster** command, include the `--replication-source-identifier` parameter to identify the Amazon Resource Name (ARN) for the source MySQL DB instance. For more information about Amazon RDS ARNs, see Amazon Relational Database Service (Amazon RDS).

Don't specify the master username, master password, or database name as the Aurora Read Replica uses the same master username, master password, and database name as the source MySQL DB instance.

For Linux, OS X, or Unix:

```
aws rds create-db-cluster --db-cluster-identifier sample-replica-cluster --engine aurora \
    --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2 \
    --replication-source-identifier arn:aws:rds:us-west-2:123456789012:db:master-mysql-
instance
```

For Windows:

```
aws rds create-db-cluster --db-cluster-identifier sample-replica-cluster --engine aurora ^
    --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2 ^
    --replication-source-identifier arn:aws:rds:us-west-2:123456789012:db:master-mysql-
instance
```

If you use the console to create an Aurora Read Replica, then Amazon RDS automatically creates the primary instance for your DB cluster Aurora Read Replica. If you use the AWS CLI to create an Aurora Read Replica, you must explicitly create the primary instance for your DB cluster. The primary instance is the first instance that is created in a DB cluster.

You can create a primary instance for your DB cluster by using the `create-db-instance` AWS CLI command with the following parameters.

- `--db-cluster-identifier`

  The name of your DB cluster.
- `--db-instance-class`

  The name of the DB instance class to use for your primary instance.
- `--db-instance-identifier`

The name of your primary instance.

- `--engine aurora`

In this example, you create a primary instance named *myreadreplicainstance* for the DB cluster named *myreadreplicacluster*, using the DB instance class specified in *myinstanceclass*.

Example

For Linux, OS X, or Unix:

```
aws rds create-db-instance \
    --db-cluster-identifier myreadreplicacluster \
    --db-instance-class myinstanceclass
    --db-instance-identifier myreadreplicainstance \
    --engine aurora
```

For Windows:

```
aws rds create-db-instance \
    --db-cluster-identifier myreadreplicacluster \
    --db-instance-class myinstanceclass
    --db-instance-identifier myreadreplicainstance \
    --engine aurora
```

API

To create an Aurora Read Replica from a source MySQL DB instance, use the `CreateDBCluster` and `CreateDBInstance` Amazon RDS API commands to create a new Aurora DB cluster and primary instance. Do not specify the master username, master password, or database name as the Aurora Read Replica uses the same master username, master password, and database name as the source MySQL DB instance.

You can create a new Aurora DB cluster for an Aurora Read Replica from a source MySQL DB instance by using the `CreateDBCluster` Amazon RDS API command with the following parameters:

- `DBClusterIdentifier`

  The name of the DB cluster to create.

- `DBSubnetGroupName`

  The name of the DB subnet group to associate with this DB cluster.

- `Engine=aurora`
- `KmsKeyId`

  The AWS Key Management Service (AWS KMS) encryption key to optionally encrypt the DB cluster with, depending on whether your MySQL DB instance is encrypted.

  - If your MySQL DB instance isn't encrypted, specify an encryption key to have your DB cluster encrypted at rest. Otherwise, your DB cluster is encrypted at rest using the default encryption key for your account.
  - If your MySQL DB instance is encrypted, specify an encryption key to have your DB cluster encrypted at rest using the specified encryption key. Otherwise, your DB cluster is encrypted at rest using the encryption key for the MySQL DB instance.

    Note
    You can't create an unencrypted DB cluster from an encrypted MySQL DB instance.

- `ReplicationSourceIdentifier`

  The Amazon Resource Name (ARN) for the source MySQL DB instance. For more information about Amazon RDS ARNs, see Amazon Relational Database Service (Amazon RDS).

- `VpcSecurityGroupIds`

  The list of EC2 VPC security groups to associate with this DB cluster.

In this example, you create a DB cluster named *myreadreplicacluster* from a source MySQL DB instance with an ARN set to *mysqlmasterARN*, associated with a DB subnet group named *mysubnetgroup* and a VPC security group named *mysecuritygroup*.

Example

```
https://rds.us-east-1.amazonaws.com/
    ?Action=CreateDBCluster
    &DBClusterIdentifier=myreadreplicacluster
    &DBSubnetGroupName=mysubnetgroup
    &Engine=aurora
    &ReplicationSourceIdentifier=mysqlmasterARN
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-10-31
    &VpcSecurityGroupIds=mysecuritygroup
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20150927/us-east-1/rds/aws4_request
    &X-Amz-Date=20150927T164851Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=6a8f4bd6a98f649c75ea04a6b3929ecc75ac09739588391cd7250f5280e716db
```

If you use the console to create an Aurora Read Replica, then Amazon RDS automatically creates the primary instance for your DB cluster Aurora Read Replica. If you use the AWS CLI to create an Aurora Read Replica, you must explicitly create the primary instance for your DB cluster. The primary instance is the first instance that is created in a DB cluster.

You can create a primary instance for your DB cluster by using the `CreateDBInstance` Amazon RDS API command with the following parameters:

- `DBClusterIdentifier`

  The name of your DB cluster.

- `DBInstanceClass`

  The name of the DB instance class to use for your primary instance.

- `DBInstanceIdentifier`

  The name of your primary instance.

- `Engine=aurora`

In this example, you create a primary instance named *myreadreplicainstance* for the DB cluster named *myreadreplicacluster*, using the DB instance class specified in *myinstanceclass*.

Example

```
https://rds.us-east-1.amazonaws.com/
    ?Action=CreateDBInstance
```

```
&DBClusterIdentifier=myreadreplicacluster
&DBInstanceClass=myinstanceclass
&DBInstanceIdentifier=myreadreplicainstance
&Engine=aurora
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&Version=2014-09-01
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20140424/us-east-1/rds/aws4_request
&X-Amz-Date=20140424T194844Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=bee4aabc750bf7dad0cd9e22b952bd6089d91e2a16592c2293e532eeaab8bc77
```

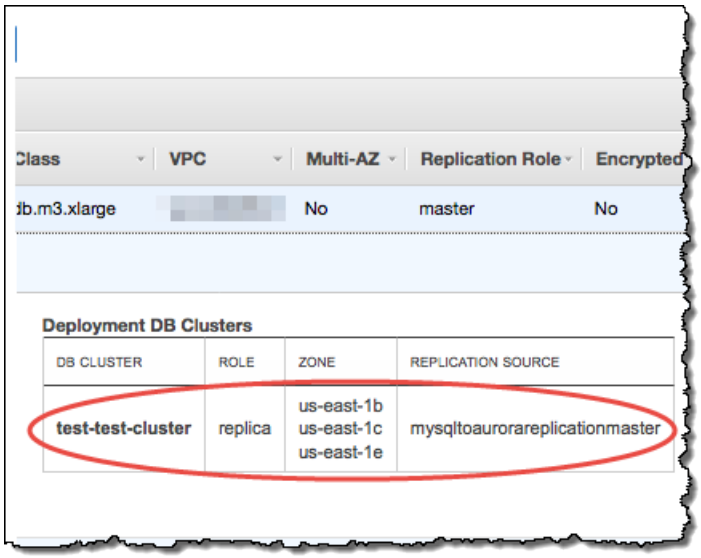
Viewing an Aurora Read Replica

You can view the MySQL to Aurora MySQL replication relationships for your Aurora MySQL DB clusters by using the AWS Management Console or the AWS CLI.

AWS Management Console

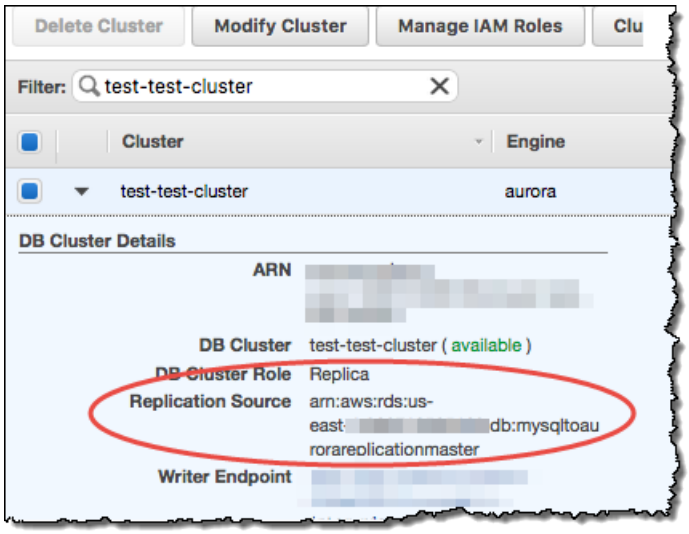
To view the Aurora Read Replica for a MySQL DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Instances**.
3. Expand the MySQL DB instance and choose the **Replication** tab. The Aurora Read Replica information will be in the **Deployment DB Clusters** table in the row with a role of **replica**.



To view the master MySQL DB instance for an Aurora Read Replica

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Clusters**.
3. Expand the DB cluster for the Aurora Read Replica and choose the **Replication** tab. The master MySQL DB instance information will be in the **Replication Source** field.



CLI

To view the MySQL to Aurora MySQL replication relationships for your Aurora MySQL DB clusters by using the AWS CLI, use the **describe-db-clusters** and **describe-db-instances** commands.

To determine which MySQL DB instance is the master, use the **describe-db-clusters** and specify the cluster identifier of the Aurora Read Replica for the `--db-cluster-identifier` option. Refer to the `ReplicationSourceIdentifier` element in the output for the ARN of the DB instance that is the replication master.

To determine which DB cluster is the Aurora Read Replica, use the `describe-db-instances` and specify the instance identifier of the MySQL DB instance for the `--db-instance-identifier` option. Refer to the `ReadReplicaDBClusterIdentifiers` element in the output for the DB cluster identifier of the Aurora Read Replica.

Example

For Linux, OS X, or Unix:

```
aws rds describe-db-clusters \
    --db-cluster-identifier myreadreplicacluster
```

```
aws rds describe-db-instances \
    --db-instance-identifier mysqlmaster
```

For Windows:

```
aws rds describe-db-clusters ^
    --db-cluster-identifier myreadreplicacluster
```

```
aws rds describe-db-instances ^
    --db-instance-identifier mysqlmaster
```

Promoting an Aurora Read Replica

After migration completes, you can promote the Aurora Read Replica to a stand-alone DB cluster and direct your client applications to the endpoint for the Aurora Read Replica. For more information on the

Aurora endpoints, see Aurora Endpoints (p. 431). Promotion should complete fairly quickly, and you can read from and write to the Aurora Read Replica during promotion. However, you can't delete the master MySQL DB instance or unlink the DB Instance and the Aurora Read Replica during this time.

Before you promote your Aurora Read Replica, stop any transactions from being written to the source MySQL DB instance, and then wait for the replica lag on the Aurora Read Replica to reach 0. You can view the replica lag for an Aurora Read Replica by calling the `SHOW SLAVE STATUS` command on your Aurora Read Replica and reading the **Seconds behind master** value.

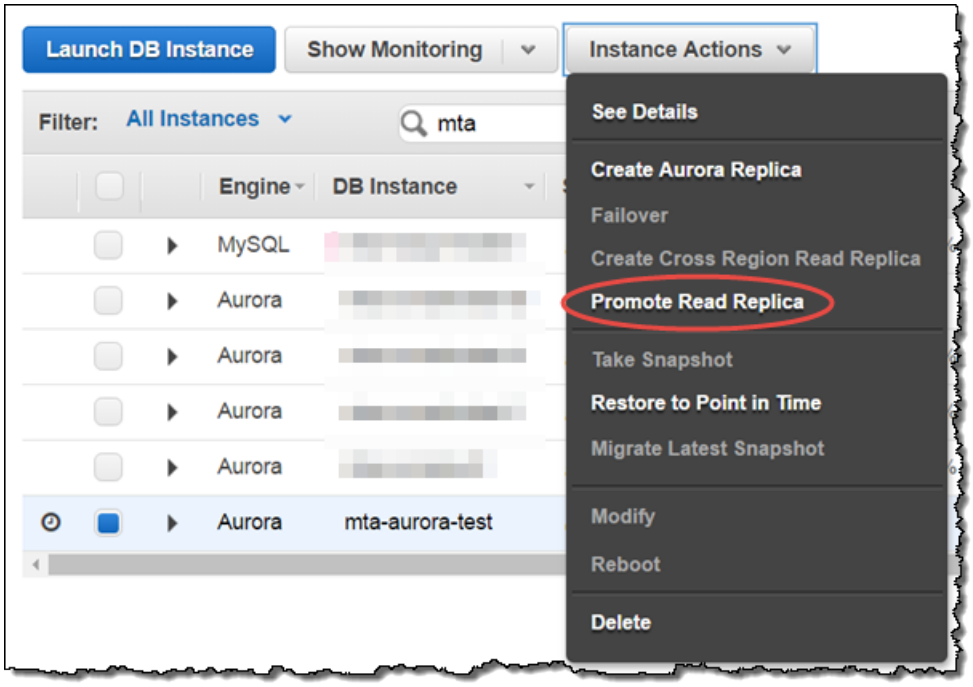
You can start writing to the Aurora Read Replica after write transactions to the master have stopped and replica lag is 0. If you write to the Aurora Read Replica before this and you modify tables that are also being modified on the MySQL master, you risk breaking replication to Aurora. If this happens, you must delete and recreate your Aurora Read Replica.

After you promote, confirm that the promotion has completed by choosing **Instances** in the navigation pane and confirming that there is a **Promoted Read Replica cluster to stand-alone database cluster** event for the Aurora Read Replica. After promotion is complete, the master MySQL DB Instance and the Aurora Read Replica are unlinked, and you can safely delete the DB instance if you want to.

AWS Management Console

To promote an Aurora Read Replica to an Aurora DB cluster

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Instances**.
3. Choose the DB instance for the Aurora Read Replica and choose **Promote Read Replica** from **Instance Actions**.



4. Choose **Promote Read Replica**.

CLI

To promote an Aurora Read Replica to a stand-alone DB cluster, use the **promote-read-replica-db-cluster** AWS CLI command.

Example

For Linux, OS X, or Unix:

```
aws rds promote-read-replica-db-cluster \
    --db-cluster-identifier myreadreplicacluster
```

For Windows:

```
aws rds promote-read-replica-db-cluster ^
    --db-cluster-identifier myreadreplicacluster
```

Related Topics

- Migrating Data to an Amazon Aurora DB Cluster (p. 466)
- Replication with Amazon Aurora (p. 478)
- Amazon Aurora on Amazon RDS (p. 428)

Related Topics

- Amazon Aurora on Amazon RDS (p. 428)

Managing Amazon Aurora MySQL

The following sections discuss managing performance, scaling, fault tolerance, backup, and restoring for an Amazon Aurora MySQL DB cluster.

Managing Performance and Scaling for Amazon Aurora MySQL

Scaling Aurora MySQL DB Instances

You can scale Aurora MySQL DB instances in two ways, instance scaling and read scaling. For more information about read scaling, see Read Scaling (p. 467).

You can scale your Aurora MySQL DB cluster by modifying the DB instance class for each DB instance in the DB cluster. Aurora MySQL supports several DB instance classes optimized for Aurora. The following table describes the specifications of the DB instance classes supported by Aurora MySQL.

Instance Class	vCPU	ECU	Memory (GiB)
db.t2.small	1	1	2
db.t2.medium	2	2	4
db.r3.large	2	6.5	15.25
db.r3.xlarge	4	13	30.5
db.r3.2xlarge	8	26	61
db.r3.4xlarge	16	52	122
db.r3.8xlarge	32	104	244

Maximum Connections to an Aurora MySQL DB Instance

The maximum number of connections allowed to an Aurora MySQL DB instance is determined by the `max_connections` parameter in the instance-level parameter group for the DB instance. By default, this value is set to the following equation (the log function represents log base 2):

```
GREATEST({log(DBInstanceClassMemory/805306368)*45},
{log(DBInstanceClassMemory/8187281408)*1000}).
```

Setting the `max_connections` parameter to this equation makes sure that the number of allowed connection scales well with the size of the instance. For example, suppose your DB instance class is db.r3.xlarge, which has 30.5 gigabytes (GB) of memory. Then the maximum connections allowed is 2000, as shown in the following equation:

```
log( (30.5 * 1073741824) / 8187281408 ) * 1000 = 2000
```

The following table lists the resulting default value of `max_connections` for each DB instance class available to Aurora MySQL. You can increase the maximum number of connections to your Aurora MySQL DB instance by scaling the instance up to a DB instance class with more memory, or by setting a larger value for the `max_connections` parameter, up to 16,000.

Instance Class	max_connec Default Value		
db.t2.small	45		
db.t2.medium	90		
db.r3.large	1000		
db.r3.xlarge	2000		
db.r3.2xlarge	3000		
db.r3.4xlarge	4000		
db.r3.8xlarge	5000		

Testing Amazon Aurora Using Fault Injection Queries

You can test the fault tolerance of your Amazon Aurora DB cluster by using fault injection queries. Fault injection queries are issued as SQL commands to an Amazon Aurora instance and they enable you to schedule a simulated occurrence of one of the following events:

- A crash of the master instance or an Aurora Replica
- A failure of an Aurora Replica
- A disk failure
- Disk congestion

Fault injection queries that specify a crash force a crash of the Amazon Aurora instance. The other fault injection queries result in simulations of failure events, but don't cause the event to occur. When you submit a fault injection query, you also specify an amount of time for the failure event simulation to occur for.

You can submit a fault injection query to one of your Aurora Replica instances by connecting to the endpoint for the Aurora Replica. For more information, see Aurora Endpoints (p. 431).

Testing an Instance Crash

You can force a crash of an Amazon Aurora instance using the `ALTER SYSTEM CRASH` fault injection query.

For this fault injection query, a failover will not occur. If you want to test a failover, then you can choose the **Failover** instance action for your DB cluster in the RDS console, or use the failover-db-cluster AWS CLI command or the FailoverDBCluster RDS API action.

Syntax

```
ALTER SYSTEM CRASH [ INSTANCE | DISPATCHER | NODE ];
```

Options

This fault injection query takes one of the following crash types:

- **`INSTANCE`**—A crash of the MySQL-compatible database for the Amazon Aurora instance is simulated.
- **`DISPATCHER`**—A crash of the dispatcher on the master instance for the Aurora DB cluster is simulated. The *dispatcher* writes updates to the cluster volume for an Amazon Aurora DB cluster.
- **`NODE`**—A crash of both the MySQL-compatible database and the dispatcher for the Amazon Aurora instance is simulated. For this fault injection simulation, the cache is also deleted.

The default crash type is `INSTANCE`.

Testing an Aurora Replica Failure

You can simulate the failure of an Aurora Replica using the `ALTER SYSTEM SIMULATE READ REPLICA FAILURE` fault injection query.

An Aurora Replica failure will block all requests to an Aurora Replica or all Aurora Replicas in the DB cluster for a specified time interval. When the time interval completes, the affected Aurora Replicas will be automatically synced up with master instance.

Syntax

```
ALTER SYSTEM SIMULATE percentage_of_failure PERCENT READ REPLICA FAILURE
    [ TO ALL | TO "replica name" ]
    FOR INTERVAL quantity { YEAR | QUARTER | MONTH | WEEK | DAY | HOUR | MINUTE | SECOND };
```

Options

This fault injection query takes the following parameters:

- **`percentage_of_failure`**—The percentage of requests to block during the failure event. This value can be a double between 0 and 100. If you specify 0, then no requests are blocked. If you specify 100, then all requests are blocked.
- **Failure type**—The type of failure to simulate. Specify `TO ALL` to simulate failures for all Aurora Replicas in the DB cluster. Specify `TO` and the name of the Aurora Replica to simulate a failure of a single Aurora Replica. The default failure type is `TO ALL`.
- **`quantity`**—The amount of time for which to simulate the Aurora Replica failure. The interval is an amount followed by a time unit. The simulation will occur for that amount of the specified unit. For example, `20 MINUTE` will result in the simulation running for 20 minutes.

Note
Take care when specifying the time interval for your Aurora Replica failure event. If you specify too long of a time interval, and your master instance writes a large amount of data during the failure event, then your Aurora DB cluster might assume that your Aurora Replica has crashed and replace it.

Testing a Disk Failure

You can simulate a disk failure for an Aurora DB cluster using the `ALTER SYSTEM SIMULATE DISK FAILURE` fault injection query.

During a disk failure simulation, the Aurora DB cluster randomly marks disk segments as faulting. Requests to those segments will be blocked for the duration of the simulation.

Syntax

```
ALTER SYSTEM SIMULATE percentage_of_failure PERCENT DISK FAILURE
    [ IN DISK index | NODE index ]
    FOR INTERVAL quantity { YEAR | QUARTER | MONTH | WEEK | DAY | HOUR | MINUTE | SECOND };
```

Options

This fault injection query takes the following parameters:

- **`percentage_of_failure`** — The percentage of the disk to mark as faulting during the failure event. This value can be a double between 0 and 100. If you specify 0, then none of the disk is marked as faulting. If you specify 100, then the entire disk is marked as faulting.
- **`DISK index`** — A specific logical block of data to simulate the failure event for. If you exceed the range of available logical blocks of data, you will receive an error that tells you the maximum index value that you can specify. For more information, see Displaying Volume Status for an Aurora DB Cluster (p. 523).
- **`NODE index`** — A specific storage node to simulate the failure event for. If you exceed the range of available storage nodes, you will receive an error that tells you the maximum index value that you can specify. For more information, see Displaying Volume Status for an Aurora DB Cluster (p. 523).
- **`quantity`** — The amount of time for which to simulate the disk failure. The interval is an amount followed by a time unit. The simulation will occur for that amount of the specified unit. For example, `20 MINUTE` will result in the simulation running for 20 minutes.

Testing Disk Congestion

You can simulate a disk failure for an Aurora DB cluster using the `ALTER SYSTEM SIMULATE DISK CONGESTION` fault injection query.

During a disk congestion simulation, the Aurora DB cluster randomly marks disk segments as congested. Requests to those segments will be delayed between the specified minimum and maximum delay time for the duration of the simulation.

Syntax

```
ALTER SYSTEM SIMULATE percentage_of_failure PERCENT DISK CONGESTION
    BETWEEN minimum AND maximum MILLISECONDS
    [ IN DISK index | NODE index ]
    FOR INTERVAL quantity { YEAR | QUARTER | MONTH | WEEK | DAY | HOUR | MINUTE | SECOND };
```

Options

This fault injection query takes the following parameters:

- **`percentage_of_failure`—**The percentage of the disk to mark as congested during the failure event. This value can be a double between 0 and 100. If you specify 0, then none of the disk is marked as congested. If you specify 100, then the entire disk is marked as congested.
- **`DISK index` or `NODE index`—**A specific disk or node to simulate the failure event for. If you exceed the range of indexes for the disk or node, you will receive an error that tells you the maximum index value that you can specify.
- **`minimum` and `maximum`—**The minimum and maximum amount of congestion delay, in milliseconds. Disk segments marked as congested will be delayed for a random amount of time within the range of the minimum and maximum amount of milliseconds for the duration of the simulation.
- **`quantity`—**The amount of time for which to simulate the disk congestion. The interval is an amount followed by a time unit. The simulation will occur for that amount of the specified time unit. For example, `20 MINUTE` will result in the simulation running for 20 minutes.

Altering Tables in Amazon Aurora Using Fast DDL

In MySQL, many data definition language (DDL) operations have a significant performance impact. Performance impacts occur even with recent online DDL improvements.

For example, suppose that you use an ALTER TABLE operation to add a column to a table. Depending on the algorithm specified for the operation, this operation can involve the following:

- Creating a full copy of the table
- Creating a temporary table to process concurrent data manipulation language (DML) operations
- Rebuilding all indexes for the table
- Applying table locks while applying concurrent DML changes
- Slowing concurrent DML throughput

In Amazon Aurora, you can use fast DDL to execute an ALTER TABLE operation in place, nearly instantaneously. The operation completes without requiring the table to be copied and without having a material impact on other DML statements. Because the operation doesn't consume temporary storage for a table copy, it makes DDL statements practical even for large tables on small instance types.

> **Note**
> Fast DDL is available for Aurora version 1.12 and later. For more information about Aurora versions, see Amazon Aurora MySQL Database Engine Updates (p. 610)

Limitations

Currently, fast DDL has the following limitations:

- Fast DDL only supports adding nullable columns, without default values, to the end of an existing table.
- Fast DDL does not support partitioned tables.
- Fast DDL does not support InnoDB tables that use the REDUNDANT row format.

Syntax

```
ALTER TABLE tbl_name ADD COLUMN col_name column_definition
```

Options

This statement takes the following options:

- **`tbl_name`** — The name of the table to be modified.
- **`col_name`** — The name of the column to be added.
- **`col_definition`** — The definition of the column to be added.

  Note
  You must specify a nullable column definition without a default value. Otherwise, fast DDL isn't used.

Displaying Volume Status for an Aurora DB Cluster

In Amazon Aurora, a DB cluster volume consists of a collection of logical blocks. Each of these represents 10 gigabytes of allocated storage. These blocks are called *protection groups*.

The data in each protection group is replicated across six physical storage devices, called *storage nodes*. These storage nodes are allocated across three Availability Nodes (AZs) in the region where the DB cluster resides. In turn, each storage node contains one or more logical blocks of data for the DB cluster volume. For more information about protection groups and storage nodes, see Introducing the Aurora Storage Engine on the AWS Database Blog.

You can simulate the failure of an entire storage node, or a single logical block of data within a storage node. To do so, you use the `ALTER SYSTEM SIMULATE DISK FAILURE` fault injection query. For the query, you specify the index value of a specific logical block of data or storage node. However, if you specify an index value greater than the number of logical blocks of data or storage nodes used by the DB cluster volume, the query returns an error. For more information about fault injection queries, see Testing Amazon Aurora Using Fault Injection Queries (p. 519).

You can avoid that error by using the `SHOW VOLUME STATUS` query. The query returns two server status variables, `Disks` and `Nodes`. These variables represent the total number of logical blocks of data and storage nodes, respectively, for the DB cluster volume.

Note
The SHOW VOLUME STATUS query is available for Aurora version 1.12 and later. For more information about Aurora versions, see Amazon Aurora MySQL Database Engine Updates (p. 610).

Syntax

```
SHOW VOLUME STATUS
```

Example

The following example illustrates a typical SHOW VOLUME STATUS result.

```
mysql> SHOW VOLUME STATUS;
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| Disks         | 96    |
| Nodes         | 74    |
+---------------+-------+
```

Using Advanced Auditing with an Amazon Aurora MySQL DB Cluster

You can use the high-performance Advanced Auditing feature in Amazon Aurora MySQL to audit database activity. To do so, you enable the collection of audit logs by setting several DB cluster parameters. When Advanced Auditing is enabled, you can use it to log any combination of supported events. You can view or download the audit logs to review them.

You must be using Aurora MySQL 1.10.1 or greater to use Advanced Auditing. For more information about Aurora MySQL versions, see Amazon Aurora MySQL Database Engine Updates (p. 610).

Enabling Advanced Auditing

Use the parameters described in this section to enable and configure Advanced Auditing for your DB cluster.

Use the `server_audit_logging` parameter to enable or disable Advanced Auditing, and the `server_audit_events` parameter to specify what events to log.

Use the `server_audit_excl_users` and `server_audit_incl_users` parameters to specify who gets audited. If `server_audit_excl_users` and `server_audit_incl_users` are empty (the default), all users are audited. If you add users to `server_audit_incl_users` and leave `server_audit_excl_users` empty, then only those users are audited. If you add users to `server_audit_excl_users` and leave `server_audit_incl_users` empty, then only those users are not audited, and all other users are. If you add the same users to both `server_audit_excl_users` and `server_audit_incl_users`, then those users are audited because `server_audit_incl_users` is given higher priority.

Configure Advanced Auditing by setting these parameters in the parameter group used by your DB cluster. You can use the procedure shown in Modifying Parameters in a DB Parameter Group (p. 172) to modify DB cluster parameters using the AWS Management Console. You can use the modify-db-cluster-parameter-group AWS CLI command or the ModifyDBClusterParameterGroup Amazon RDS API command to modify DB cluster parameters programmatically.

Modifying these parameters doesn't require a DB cluster restart.

`server_audit_logging`

Enables or disables Advanced Auditing. This parameter defaults to OFF; set it to ON to enable Advanced Auditing.

`server_audit_events`

Contains the comma-delimited list of events to log. Events must be specified in all caps, and there should be no white space between the list elements, for example: `CONNECT,QUERY_DDL`. This parameter defaults to an empty string.

You can log any combination of the following events:

- CONNECT – Logs both successful and failed connections and also disconnections. This event includes user information.
- QUERY – Logs all queries in plain text, including queries that fail due to syntax or permission errors.
- QUERY_DCL – Similar to the QUERY event, but returns only data control language (DCL) queries (GRANT, REVOKE, and so on).

- QUERY_DDL – Similar to the QUERY event, but returns only data definition language (DDL) queries (CREATE, ALTER, and so on).
- QUERY_DML – Similar to the QUERY event, but returns only data manipulation language (DML) queries (INSERT, UPDATE, and so on).
- TABLE – Logs the tables that were affected by query execution.

server_audit_excl_users

Contains the comma-delimited list of user names for users whose activity isn't logged. There should be no white space between the list elements, for example: `rdsadmin,user_1,user_2`. This parameter defaults to an empty string. Specified user names must match corresponding values in the `User` column of the `mysql.user` table. For more information about user names, see the MySQL documentation.

Connect and disconnect events aren't affected by this variable; they are always logged if specified. A user is logged if that user is also specified in the `server_audit_incl_users` parameter, because that setting has higher priority than `server_audit_excl_users`.

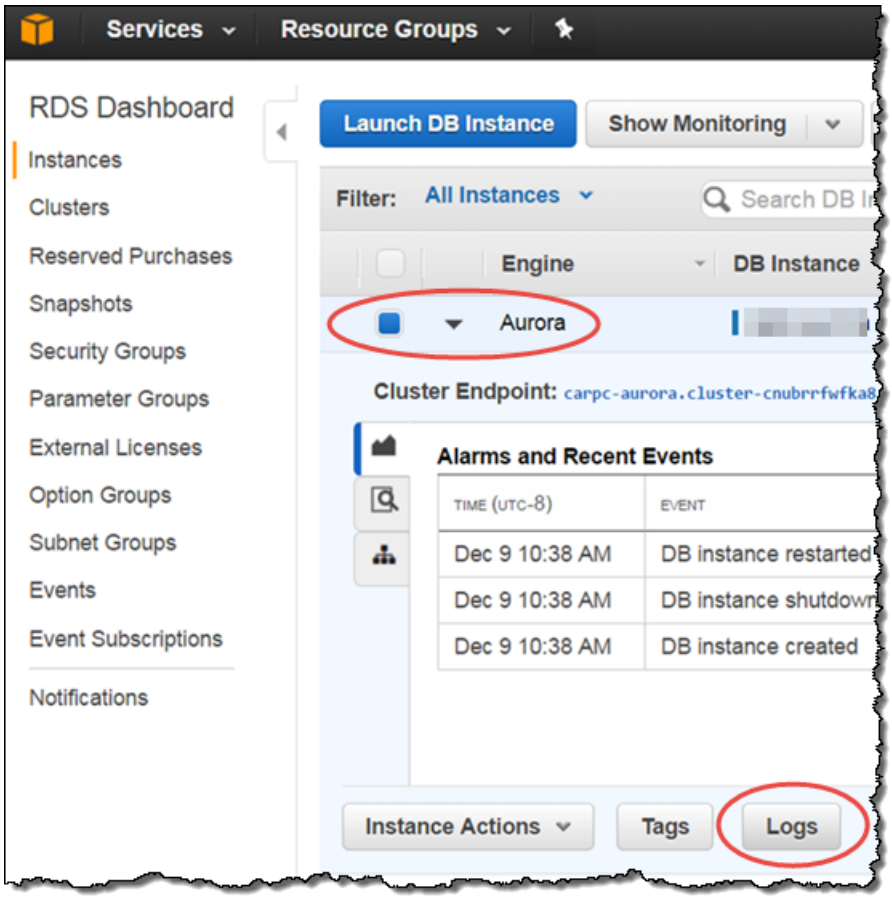
server_audit_incl_users

Contains the comma-delimited list of user names for users whose activity is logged. There should be no white space between the list elements, for example: `user_3,user_4`. This parameter defaults to an empty string. Specified user names must match corresponding values in the `User` column of the `mysql.user` table. For more information about user names, see the MySQL documentation.

Connect and disconnect events aren't affected by this variable; they are always logged if specified. A user is logged even if that user is also specified in the `server_audit_excl_users` parameter, because `server_audit_incl_users` has higher priority.

Viewing Audit Logs

You can view and download the audit logs by using the AWS console. On the **Instances** page, select and expand the DB cluster, then choose **Logs**.



To download a log file, locate that file in the **Logs** section and then choose **download**.

You can also get a list of the log files by using the describe-db-log-files AWS CLI command. You can view the content of a log file by using the download-db-log-file-portion AWS CLI command, and download a log file by using the DownloadCompleteDBLogFile (p. 1243) REST API.

Audit Log Details

Log files are in UTF-8 format. Logs are written in multiple files, the number of which varies based on instance size. To see the latest events, you might have to review all of the audit log files.

Log entries are not in sequential order. You can use the timestamp value for ordering.

Log files are rotated when they reach 100 MB in aggregate. This limit is not configurable.

The audit log files include the following comma-delimited information in rows, in the specified order:

Field	Description
timestamp	The Unix time stamp for the logged event with microsecond precision.
serverhost	The name of the instance that the event is logged for.
username	The connected user name of the user.
host	The host that the user connected from.

Field	Description
connectionid	The connection ID number for the logged operation.
queryid	The query ID number, which can be used for finding the relational table events and related queries. For `TABLE` events, multiple lines are added.
operation	The recorded action type. Possible values are: `CONNECT`, `QUERY`, `READ`, `WRITE`, `CREATE`, `ALTER`, `RENAME`, and `DROP`.
database	The active database, as set by the `USE` command.
object	For QUERY events, this value indicates the executed query. For TABLE events, it indicates the table name.
retcode	The return code of the logged operation.

Replication with Amazon Aurora MySQL

Using Aurora Replicas

Aurora Replicas are independent endpoints in an Aurora DB cluster, best used for scaling read operations and increasing availability. Up to 15 Aurora Replicas can be distributed across the Availability Zones that a DB cluster spans within an AWS Region. Although the DB cluster volume is made up of multiple copies of the data for the DB cluster, the data in the cluster volume is represented as a single, logical volume to the primary instance and to Aurora Replicas in the DB cluster. For more information about Aurora Replicas, see Aurora Replicas (p. 478).

Aurora Replicas work well for read scaling because they are fully dedicated to read operations on your cluster volume. Write operations are managed by the primary instance. Because the cluster volume is shared among all instances in your Aurora MySQL DB cluster, no additional work is required to replicate a copy of the data for each Aurora Replica. In contrast, MySQL Read Replicas must replay, on a single thread, all write operations from the master DB instance to their local data store. This limitation can affect the ability of MySQL Read Replicas to support large volumes of read traffic.

> **Important**
> Aurora Replicas for Aurora MySQL always use the `REPEATABLE READ` default transaction isolation level for operations on InnoDB tables. You can use the `SET TRANSACTION ISOLATION LEVEL` command to change the transaction level only for the primary instance of an Aurora MySQL DB cluster. This restriction avoids user-level locks on Aurora Replicas, and allows Aurora Replicas to scale to support thousands of active user connections while still keeping replica lag to a minimum.

Replication Options for Amazon Aurora MySQL

You can set up replication between any of the following options:

- Two Aurora MySQL DB clusters in different AWS regions, by creating an Aurora Read Replica of an Aurora MySQL DB cluster in a different AWS Region.

  For more information, see Replicating Amazon Aurora MySQL DB Clusters Across AWS Regions (p. 528).
- Two Aurora MySQL DB clusters in the same region, by using MySQL binary log (binlog) replication.

  For more information, see Replication Between Aurora and MySQL or Between Aurora and Another Aurora DB Cluster (p. 537).

- An Amazon RDS MySQL DB instance as the master and an Aurora MySQL DB cluster, by creating an Aurora Read Replica of an Amazon RDS MySQL DB instance.

  Typically, this approach is used for migration to Aurora MySQL, rather than for ongoing replication. For more information, see Migrating Data from a MySQL DB Instance to an Amazon Aurora MySQL DB Cluster by Using a DB Snapshot (p. 501).

  Note
  Rebooting the primary instance of an Amazon Aurora DB cluster also automatically reboots the Aurora Replicas for that DB cluster, in order to re-establish an entry point that guarantees read/write consistency across the DB cluster.

Monitoring Amazon Aurora MySQL Replication

Read scaling and high availability depend on minimal lag time. You can monitor how far an Aurora Replica is lagging behind the primary instance of your Aurora MySQL DB cluster by monitoring the Amazon CloudWatch `ReplicaLag` metric. Because Aurora Replicas read from the same cluster volume as the primary instance, the `ReplicaLag` metric has a different meaning for an Aurora MySQL DB cluster. The `ReplicaLag` metric for an Aurora Replica indicates the lag for the page cache of the Aurora Replica compared to that of the primary instance.

If you need the most current value for Aurora Replica lag, you can query the `mysql.ro_replica_status` table in your Aurora MySQL DB cluster and check the value in the `Replica_lag_in_msec` column. This column value is provided to Amazon CloudWatch as the value for the `ReplicaLag` metric. The values in the `mysql.ro_replica_status` are also provided in the `INFORMATION_SCHEMA.REPLICA_HOST_STATUS` table in your Aurora MySQL DB cluster.

For more information on monitoring RDS instances and CloudWatch metrics, see Monitoring Amazon RDS (p. 245).

Replicating Amazon Aurora MySQL DB Clusters Across AWS Regions

You can create an Amazon Aurora MySQL DB cluster as a Read Replica in a different AWS Region than the source DB cluster. Taking this approach can improve your disaster recovery capabilities, let you scale read operations into a region that is closer to your users, and make it easier to migrate from one region to another.

You can create Read Replicas of both encrypted and unencrypted DB clusters. The Read Replica must be encrypted if the source DB cluster is encrypted.

When you create an Aurora MySQL DB cluster Read Replica in another region, you should be aware of the following:

- In a cross-region scenario, there is more lag time between the source DB cluster and the Read Replica due to the longer network channels between regions.
- Data transferred for cross-region replication incurs Amazon RDS data transfer charges. The following cross-region replication actions generate charges for the data transferred out of the source region:
  - When you create the Read Replica, Amazon RDS takes a snapshot of the source cluster and transfers the snapshot to the Read Replica region.
  - For each data modification made in the source databases, Amazon RDS transfers data from the source region to the Read Replica region.

  For more information about Amazon RDS data transfer pricing, see Amazon Aurora Pricing.

For each source DB cluster, you can only have one cross-region Read Replica DB cluster. Both your source DB cluster and your cross-region Read Replica DB cluster can have up to 15 Aurora Replicas along with the primary instance for the DB cluster. This functionality lets you scale read operations for both your source region and your replication target region.

Before You Begin

Before you can create an Aurora MySQL DB cluster that is a cross-region Read Replica, you must enable binary logging on your source Aurora MySQL DB cluster. Cross-region replication for Aurora MySQL uses MySQL binary replication to replay changes on the cross-region Read Replica DB cluster.

To enable binary logging on an Aurora MySQL DB cluster, update the `binlog_format` parameter for your source DB cluster. The `binlog_format` parameter is a cluster-level parameter that is in the `default.aurora5.6` cluster parameter group by default. If your DB cluster uses the default DB cluster parameter group, you will need to create a new DB cluster parameter group to modify `binlog_format` settings. We recommend that you set the `binlog_format` to `MIXED`. However, you can also set `binlog_format` to `ROW` or `STATEMENT` if you need a specific binlog format. Reboot your Aurora DB cluster for the change to take effect.

For more information, see Amazon Aurora DB Cluster and DB Instance Parameters (p. 469) and Working with DB Parameter Groups (p. 170).

Creating an Amazon Aurora MySQL DB Cluster That Is a Cross-Region Read Replica

You can create an Aurora DB cluster that is a cross-region Read Replica by using the AWS Management Console, the AWS Command Line Interface (AWS CLI), or the Amazon RDS API. You can create cross-region Read Replicas from both encrypted and unencrypted DB clusters.

When you create a cross-region Read Replica for Aurora MySQL by using the AWS Management Console, Amazon RDS creates a DB cluster in the target AWS Region, and then automatically creates a DB instance that is the primary instance for that DB cluster.

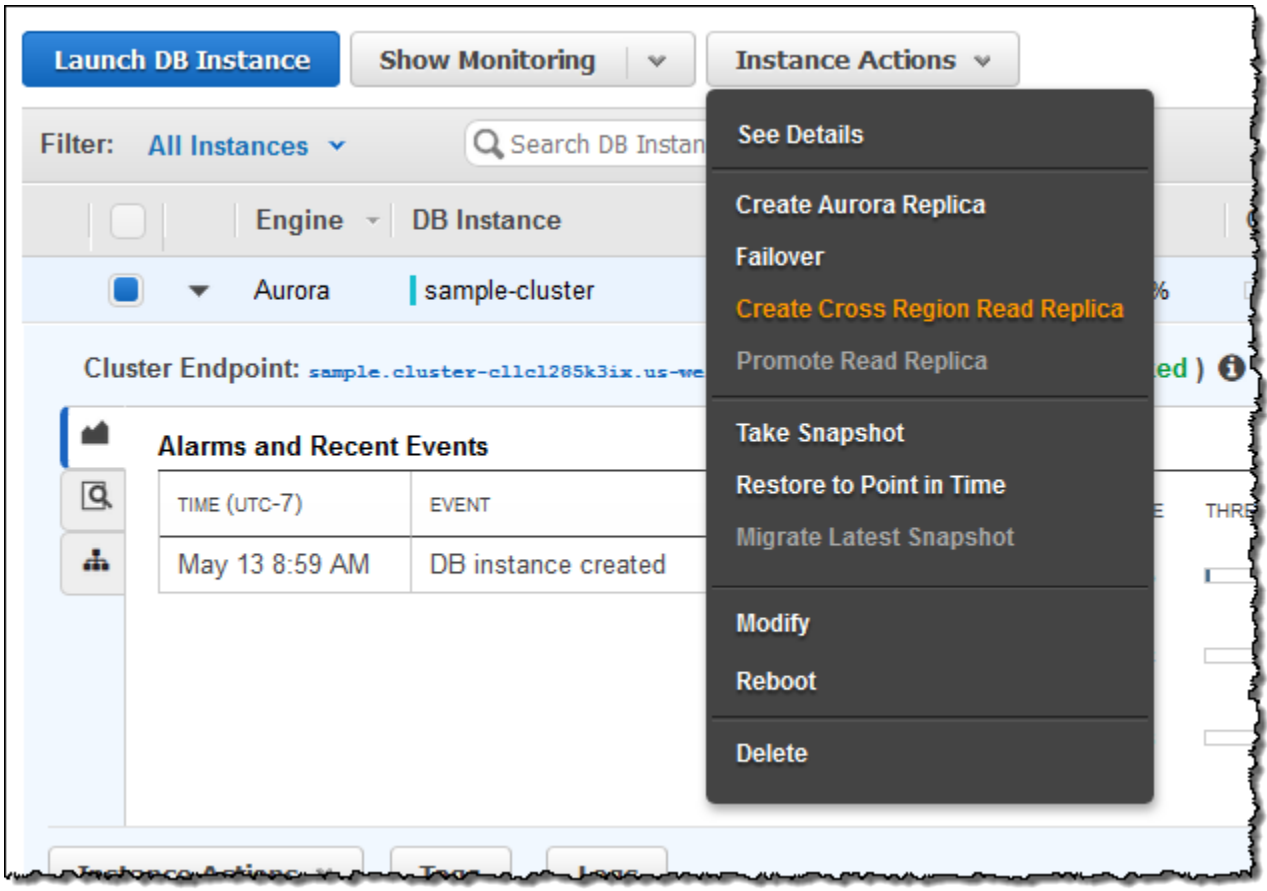
When you create a cross-region Read Replica using the AWS CLI or RDS API, you first create the DB cluster in the target AWS Region and wait for it to become active. Once it is active, you then create a DB instance that is the primary instance for that DB cluster.

Replication begins when the primary instance of the Read Replica DB cluster becomes available.

Use the following procedures to create a cross-region Read Replica from an Aurora MySQL DB cluster. These procedures work for creating Read Replicas from either encrypted or unencrypted DB clusters.

AWS Management Console

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the top-right corner of the AWS Management Console, select the AWS Region that hosts your source DB cluster.
3. In the navigation pane, choose **Instances**.
4. Select the check box for the DB cluster that you want to create a cross-region Read Replica for. Choose **Instance Actions**, and then choose **Create Cross Region Read Replica**.



5. In the **Cross Region Read Replica** panel, select the option settings for your cross-region Read Replica DB cluster, as described in the following table.

Option	Description
DB Instance Class	Choose a DB instance class that defines the processing and memory requirements for the primary instance in the DB cluster. For more information about DB instance class options, see DB Instance Class (p. 92).
Multi-AZ Deployment	Choose **Create Replica in Different Zone** to create a standby replica of the new DB cluster in another Availability Zone in the target region for failover support. For more information about multiple Availability Zones, see Regions and Availability Zones (p. 97).
Read Replica Source	Choose the source DB cluster to create a cross-region Read Replica for.
DB Instance Identifier	Type a name for the primary instance in your cross-region Read Replica DB cluster. This identifier is used in the endpoint address for the primary instance of the new DB cluster. The DB instance identifier has the following constraints:

Option	Description
	• It must contain from 1 to 63 alphanumeric characters or hyphens. • Its first character must be a letter. • It cannot end with a hyphen or contain two consecutive hyphens. • It must be unique for all DB instances for each AWS account, for each region. Because the cross-region Read Replica DB cluster is created from a snapshot of the source DB cluster, the master user name and master password for the Read Replica are the same as the master user name and master password for the source DB cluster.
DB Cluster Identifier	Type a name for your cross-region Read Replica DB cluster that is unique for your account in the target AWS Region for your replica. This identifier will be used in the cluster endpoint address for your DB cluster. For information on the cluster endpoint, see Aurora Endpoints (p. 431). The DB cluster identifier has the following constraints: • It must contain from 1 to 63 alphanumeric characters or hyphens. • Its first character must be a letter. • It cannot end with a hyphen or contain two consecutive hyphens. • It must be unique for all DB clusters for each AWS account, for each region.
Destination Region	Choose the AWS Region that will host the new cross-region Read Replica DB cluster.
Destination DB Subnet Group	Choose the DB subnet group to use for the cross-region Read Replica DB cluster.
Enable Encryption	The **Enable Encryption** value is `Yes` if the source DB cluster is encrypted, and `No` if it isn't. You can't change this value.
Master Key	This field only appears if the source DB cluster is encrypted and the **Enable Encryption** value is `Yes`. Specify the AWS KMS key identifier to use to encrypt the Read Replica. Because you are replicating the encrypted DB cluster across regions, and KMS keys are region-specific, you must specify a different KMS encryption key than that used on the source DB cluster. The KMS encryption key you enter must be valid for the destination region. For more information about using KMS keys, see What is AWS Key Management Service?.

Option	Description
Publicly Accessible	Choose **Yes** to give the cross-region Read Replica DB cluster a public IP address; otherwise, select **No**.
Availability Zone	Determine if you want to specify a particular Availability Zone, and then either choose that Availability Zone, or choose **No preference** to have Amazon RDS choose the Availability Zone for your new DB cluster.
Priority	Choose a failover priority for the primary instance of the new DB cluster. This priority determines the order in which Aurora Replicas are promoted when recovering from a primary instance failure. If you don't select a value, the default is **tier-1**. For more information, see Fault Tolerance for an Aurora DB Cluster (p. 468).
Database Port	Specify the port that applications and utilities will use to access the database. Aurora DB clusters default to the default MySQL port, 3306. Firewalls at some companies block connections to this port. If your company firewall blocks the default port, choose another port for the new DB cluster.
Enable Enhanced Monitoring	This option will be **`Yes`** to enable gathering metrics in real time for the operating system that your DB cluster runs on. For more information, see Enhanced Monitoring (p. 258).
Monitoring Role	This option is only available if **Enable Enhanced Monitoring** is set to **`Yes`**. Set the **Monitoring Role** property to the AWS Identity and Access Management (IAM) role that you created to permit Amazon RDS to communicate with Amazon CloudWatch Logs for you, or choose **Default** to have RDS create a role for you named `rds-monitoring-role`.
Granularity	This option is only available if **Enable Enhanced Monitoring** is set to **`Yes`**. Set the interval, in seconds, at which metrics are collected for your new DB cluster.
Auto Minor Version Upgrade	Choose **Yes** if you want your cross-region Read Replica DB cluster to receive minor MySQL DB engine version upgrades automatically when they become available. The **Auto Minor Version Upgrade** option only applies to upgrades to MySQL minor engine versions for your DB cluster. It doesn't apply to regular patches applied to maintain system stability.

6. Choose **Create** to create your cross-region Read Replica for Aurora.

CLI

1. Call the AWS CLI **`create-db-cluster`** command in the region where you want to create the Read Replica DB cluster. Include the `--replication-source-identifier` option and specify the Amazon Resource Name (ARN) of the source DB cluster to create a Read Replica for.

For cross-region replication where the DB cluster identified by `--replication-source-identifier` is encrypted, you must also specify either the `--source-region` or `--pre-signed-url` option, and the `--kms-key-id` option. Using `--source-region` autogenerates a pre-signed URL that is a valid request for the `CreateDBCluster` API action that can be executed in the source region that contains the encrypted DB cluster to be replicated. Using `--pre-signed-url` requires you to construct a pre-signed URL manually instead. The KMS key ID is used to encrypt the Read Replica, and must be a KMS encryption key valid for the destination region. To learn more about these options, see **`create-db-cluster`**.

> **Note**
> You can set up cross-region replication from an unencrypted DB cluster to an encrypted Read Replica by specifying `--storage-encrypted` and providing a value for `--kms-key-id`. In this case, you don't need to specify `--source-region` or `--pre-signed-url`.

You don't need to include the `--master-username` and `--master-user-password` parameters, because those values are taken from the source DB cluster.

The following code example creates a Read Replica in the us-east-1 region from an unencrypted DB cluster snapshot in the us-west-2 region. The command is called in the us-east-1 region.

For Linux, OS X, or Unix:

```
aws rds create-db-cluster \
  --db-cluster-identifier sample-replica-cluster \
  --engine aurora \
  --replication-source-identifier arn:aws:rds:us-west-2:123456789012:cluster:sample-
master-cluster
```

For Windows:

```
aws rds create-db-cluster ^
  --db-cluster-identifier sample-replica-cluster ^
  --engine aurora ^
  --replication-source-identifier arn:aws:rds:us-west-2:123456789012:cluster:sample-
master-cluster
```

The following code example creates a Read Replica in the us-east-1 region from an encrypted DB cluster snapshot in the us-west-2 region. The command is called in the us-east-1 region.

For Linux, OS X, or Unix:

```
aws rds create-db-cluster \
  --db-cluster-identifier sample-replica-cluster \
  --engine aurora \
  --replication-source-identifier arn:aws:rds:us-west-2:123456789012:cluster:sample-
master-cluster \
  --kms-key-id my-us-east-1-key \
  --source-region us-west-2
```

For Windows:

```
aws rds create-db-cluster ^
  --db-cluster-identifier sample-replica-cluster ^
  --engine aurora ^
```

```
  --replication-source-identifier arn:aws:rds:us-west-2:123456789012:cluster:sample-
master-cluster ^
  --kms-key-id my-us-east-1-key ^
  --source-region us-west-2
```

2. Check that the DB cluster has become available to use by using the AWS CLI `describe-db-clusters` command, as shown in the following example.

```
aws rds describe-db-clusters --db-cluster-identifier sample-replica-cluster
```

When the **`describe-db-clusters`** results show a status of `available`, create the primary instance for the DB cluster so that replication can begin. To do so, use the AWS CLI **`create-db-instance`** command as shown in the following example.

For Linux, OS X, or Unix:

```
aws rds create-db-instance \
  --db-cluster-identifier sample-replica-cluster \
  --db-instance-class db.r3.large \
  --db-instance-identifier sample-replica-instance \
  --engine aurora
```

For Windows:

```
aws rds create-db-instance ^
  --db-cluster-identifier sample-replica-cluster ^
  --db-instance-class db.r3.large ^
  --db-instance-identifier sample-replica-instance ^
  --engine aurora
```

When the DB instance is created and available, replication begins. You can determine if the DB instance is available by calling the AWS CLI `describe-db-instances` command.

API

1. Call the RDS API `CreateDBCluster` action in the region where you want to create the Read Replica DB cluster. Include the `ReplicationSourceIdentifier` parameter and specify the Amazon Resource Name (ARN) of the source DB cluster to create a Read Replica for.

   For cross-region replication where the DB cluster identified by `ReplicationSourceIdentifier` is encrypted, you must also specify the `PreSignedUrl` and `KmsKeyId` parameters. The pre-signed URL must be a valid request for the `CreateDBCluster` API action that can be executed in the source region that contains the encrypted DB cluster to be replicated. The KMS key ID is used to encrypt the Read Replica, and must be a KMS encryption key valid for the destination region. To automatically rather than manually generate a presigned URL, use the AWS CLI **`create-db-cluster`** command with the `--source-region` option instead.

   Note
   You can set up cross-region replication from an unencrypted DB cluster to an encrypted Read Replica by specifying `StorageEncrypted` as **`true`** and providing a value for `KmsKeyId`. In this case, you don't need to specify `PreSignedUrl`.

You don't need to include the `MasterUsername` and `MasterUserPassword` parameters, because those values are taken from the source DB cluster.

The following code example creates a Read Replica in the us-east-1 region from an unencrypted DB cluster snapshot in the us-west-2 region. The action is called in the us-east-1 region.

```
https://rds.us-east-1.amazonaws.com/
  ?Action=CreateDBCluster
  &ReplicationSourceIdentifier=arn:aws:rds:us-west-2:123456789012:cluster:sample-
master-cluster
  &DBClusterIdentifier=sample-replica-cluster
  &Engine=aurora
  &SignatureMethod=HmacSHA256
  &SignatureVersion=4
  &Version=2014-10-31
  &X-Amz-Algorithm=AWS4-HMAC-SHA256
  &X-Amz-Credential=AKIADQKE4SARGYLE/20161117/us-east-1/rds/aws4_request
  &X-Amz-Date=20160201T001547Z
  &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
  &X-Amz-Signature=a04c831a0b54b5e4cd236a90dcb9f5fab7185eb3b72b5ebe9a70a4e95790c8b7
```

The following code example creates a Read Replica in the us-east-1 region from an encrypted DB cluster snapshot in the us-west-2 region. The action is called in the us-east-1 region.

```
https://rds.us-east-1.amazonaws.com/
  ?Action=CreateDBCluster
  &KmsKeyId=my-us-east-1-key
  &PreSignedUrl=https%253A%252F%252Frds.us-west-2.amazonaws.com%252F
         %253FAction%253DCreateDBCluster
         %2526DestinationRegion%253Dus-east-1
         %2526KmsKeyId%253Dmy-us-east-1-key
         %2526ReplicationSourceIdentifier%253Darn%25253Aaws%25253Ards%25253Aus-
west-2%25253A123456789012%25253Acluster%25253Asample-master-cluster
         %2526SignatureMethod%253DHmacSHA256
         %2526SignatureVersion%253D4
         %2526Version%253D2014-10-31
         %2526X-Amz-Algorithm%253DAWS4-HMAC-SHA256
         %2526X-Amz-Credential%253DAKIADQKE4SARGYLE%252F20161117%252Fus-west-2%252Frds
%252Faws4_request
         %2526X-Amz-Date%253D20161117T215409Z
         %2526X-Amz-Expires%253D3600
         %2526X-Amz-SignedHeaders%253Dcontent-type%253Bhost%253Buser-agent%253Bx-amz-
content-sha256%253Bx-amz-date
         %2526X-Amz-Signature
%253D255a0f17b4e717d3b67fad163c3ec26573b882c03a65523522cf890a67fca613
  &ReplicationSourceIdentifier=arn:aws:rds:us-west-2:123456789012:cluster:sample-
master-cluster
  &DBClusterIdentifier=sample-replica-cluster
  &Engine=aurora
  &SignatureMethod=HmacSHA256
  &SignatureVersion=4
  &Version=2014-10-31
  &X-Amz-Algorithm=AWS4-HMAC-SHA256
  &X-Amz-Credential=AKIADQKE4SARGYLE/20161117/us-east-1/rds/aws4_request
  &X-Amz-Date=20160201T001547Z
  &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
  &X-Amz-Signature=a04c831a0b54b5e4cd236a90dcb9f5fab7185eb3b72b5ebe9a70a4e95790c8b7
```

2. Check that the DB cluster has become available to use by using the RDS API `DescribeDBClusters` action, as shown in the following example.

```
https://rds.us-east-1.amazonaws.com/
  ?Action=DescribeDBClusters
  &DBClusterIdentifier=sample-replica-cluster
  &SignatureMethod=HmacSHA256
  &SignatureVersion=4
  &Version=2014-10-31
  &X-Amz-Algorithm=AWS4-HMAC-SHA256
  &X-Amz-Credential=AKIADQKE4SARGYLE/20161117/us-east-1/rds/aws4_request
  &X-Amz-Date=20160201T002223Z
  &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
  &X-Amz-Signature=84c2e4f8fba7c577ac5d820711e34c6e45ffcd35be8a6b7c50f329a74f35f426
```

When the **`DescribeDBClusters`** results show a status of `available`, create the primary instance for the DB cluster so that replication can begin. To do so, use the RDS API `CreateDBInstance` action as shown in the following example.

```
https://rds.us-east-1.amazonaws.com/
  ?Action=CreateDBInstance
  &DBClusterIdentifier=sample-replica-cluster
  &DBInstanceClass=db.r3.large
  &DBInstanceIdentifier=sample-replica-instance
  &Engine=aurora
  &SignatureMethod=HmacSHA256
  &SignatureVersion=4
  &Version=2014-10-31
  &X-Amz-Algorithm=AWS4-HMAC-SHA256
  &X-Amz-Credential=AKIADQKE4SARGYLE/20161117/us-east-1/rds/aws4_request
  &X-Amz-Date=20160201T003808Z
  &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
  &X-Amz-Signature=125fe575959f5bbcebd53f2365f907179757a08b5d7a16a378dfa59387f58cdb
```

When the DB instance is created and available, replication begins. You can determine if the DB instance is available by calling the AWS CLI `DescribeDBInstances` command.

Viewing Amazon Aurora MySQL Cross-Region Replicas

You can view the cross-region replication relationships for your Amazon Aurora MySQL DB clusters by calling the `describe-db-clusters` AWS CLI command or the `DescribeDBClusters` RDS API action. In the response, refer to the `ReadReplicaIdentifiers` field for the DB cluster identifiers of any cross-region Read Replica DB clusters, and refer to the `ReplicationSourceIdentifier` element for the ARN of the source DB cluster that is the replication master.

Troubleshooting Amazon Aurora MySQL Cross Region Replicas

Following you can find a list of common error messages that you might encounter when creating an Amazon Aurora cross-region Read Replica, and the resolutions for the specified errors.

Source cluster [DB cluster ARN] doesn't have binlogs enabled

To resolve this issue, enable binary logging on the source DB cluster. For more information, see Before You Begin (p. 529).

Source cluster [DB cluster ARN] doesn't have cluster parameter group in sync on writer

You receive this error if you have updated the `binlog_format` DB cluster parameter, but have not rebooted the primary instance for the DB cluster. Reboot the primary instance (that is, the writer) for the DB cluster and try again.

Source cluster [DB cluster ARN] already has a read replica in this region

You can only have one cross-region Read Replica DB cluster for each source DB cluster. You must delete the existing cross-region DB cluster that is a Read Replica in order to create a new one.

DB cluster [DB cluster ARN] requires a database engine upgrade for cross-region replication support

To resolve this issue, upgrade the database engine version for all of the instances in the source DB cluster to the most recent database engine version, and then try creating a cross-region Read Replica DB again.

Replication Between Aurora and MySQL or Between Aurora and Another Aurora DB Cluster

Because Amazon Aurora MySQL is compatible with MySQL, you can set up replication between a MySQL database and an Amazon Aurora MySQL DB cluster. We recommend that your MySQL database run MySQL version 5.5 or later. You can set up replication where your Aurora MySQL DB cluster is the replication master or the replica, and you can replicate with an Amazon RDS MySQL DB instance, a MySQL database external to Amazon RDS, or another Aurora MySQL DB cluster.

You can also replicate with an Amazon RDS MySQL DB instance or Aurora MySQL DB cluster in another AWS Region. When you're performing replication across AWS regions, ensure that your DB clusters and DB instances are publicly accessible. Aurora MySQL DB clusters must be part of a public subnet in your VPC.

Warning
When you replicate between Aurora MySQL and MySQL, you must ensure that you use only InnoDB tables. If you have MyISAM tables that you want to replicate, then you can convert them to InnoDB prior to setting up replication, with the following command:

```
alter table <schema>.<table_name> engine=innodb, algorithm=copy;
```

Setting up MySQL replication with Aurora MySQL involves the following steps, which are discussed in detail following in this topic.

1. Enable Binary Logging on the Replication Master (p. 537)

2. Retain Binary Logs on the Replication Master Until No Longer Needed (p. 539)

3. Create a Snapshot of Your Replication Master (p. 540)

4. Load the Snapshot into Your Replica (p. 542)

5. Enable Replication (p. 544)

6. Monitor Your Replica (p. 546)

Setting Up Replication with MySQL or Another Aurora DB Cluster

To set up Aurora replication with MySQL, take the following steps.

1. Enable Binary Logging on the Replication Master

Find instructions on how to enable binary logging on the replication master for your database engine following.

Database Engine	Instructions
Aurora	**To enable binary logging on an Aurora MySQL DB cluster** Set the `binlog_format` parameter to `ROW`, `STATEMENT`, or `MIXED`. `MIXED` is recommended unless you have a need for a specific binlog format. The `binlog_format` parameter is a cluster-level parameter that is in the `default.aurora5.6` cluster parameter group by default. If you are changing the `binlog_format` parameter from `OFF` to another value, then you need to reboot your Aurora DB cluster for the change to take effect. For more information, see Amazon Aurora DB Cluster and DB Instance Parameters (p. 469) and Working with DB Parameter Groups (p. 170).
RDS MySQL	**To enable binary logging on an Amazon RDS DB instance** You cannot enable binary logging directly for an Amazon RDS DB instance, but you can enable it by doing one of the following: • Enable automated backups for the DB instance. You can enable automated backups when you create a DB instance, or you can enable backups by modifying an existing DB instance. For more information, see Creating a DB Instance Running the MySQL Database Engine (p. 830) and Working With Backups (p. 201). • Create a Read Replica for the DB instance. For more information, see Working with PostgreSQL, MySQL, and MariaDB Read Replicas (p. 134).

MySQL (external)	**To enable binary logging on an external MySQL database**
	
	1. From a command shell, stop the mysql service:
	
	`sudo service mysqld stop`
	
	2. Edit the my.cnf file (this file is usually under /etc):
	
	`sudo vi /etc/my.cnf`
	
	Add the `log_bin` and `server_id` options to the `[mysqld]` section. The `log_bin` option provides a file name identifier for binary log files. The `server_id` option provides a unique identifier for the server in master-replica relationships.
	
	The following example shows the updated `[mysqld]` section of a my.cnf file:
	
	`[mysqld]`
	`log-bin=mysql-bin`
	`server-id=1`
	
	Additionally, the `sql_mode` option for your MySQL DB instance must be set to 0, or must not be included in your my.cnf file.
	
	For more information, see Setting the Replication Master Configuration in the MySQL documentation.
	3. Start the mysql service:
	
	`sudo service mysqld start`

2. Retain Binary Logs on the Replication Master Until No Longer Needed

When you use MySQL binlog replication, Amazon RDS doesn't manage the replication process. As a result, you need to ensure that the binlog files on your replication master are retained until after the changes have been applied to the replica. This maintenance helps ensure that you can restore your master database in the event of a failure.

Find instructions on how to retain binary logs for your database engine following.

Database Engine	Instructions
Aurora	**To retain binary logs on an Aurora MySQL DB cluster** You do not have access to the binlog files for an Aurora MySQL DB cluster. As a result, you must choose a time frame to retain the binlog files on your replication master long enough to ensure that the changes have been applied to your replica before the binlog file is deleted by Amazon RDS. You can retain binlog files on an Aurora MySQL DB cluster for up to 90 days. If you are setting up replication with a MySQL database or RDS MySQL DB instance as the replica, and the database that you are creating a replica for is very large, choose a large time frame to retain binlog files until the initial copy of the database to the replica is complete and the replica lag has reached 0. To set the binlog retention time frame, use the mysql.rds_set_configuration (p. 923) procedure and specify a configuration parameter of `'binlog retention hours'` along with the number of hours to retain binlog files on the DB cluster, up to 2160 (90 days). The following example that sets the retention period for binlog files to 6 days: `CALL mysql.rds_set_configuration('binlog retention hours', 144);` After replication has been started, you can verify that changes have been applied to your replica by running the `SHOW SLAVE STATUS` command on your replica and checking the **Seconds behind master** field. If the **Seconds behind master** field is 0, then there is no replica lag. When there is no replica lag, reduce the length of time that binlog files are retained by setting the `binlog retention hours` configuration parameter to a smaller time frame.
RDS MySQL	**To retain binary logs on an Amazon RDS DB instance** You can retain binlog files on an Amazon RDS DB instance by setting the binlog retention hours just as with an Aurora MySQL DB cluster, described in the previous section. You can also retain binlog files on an Amazon RDS DB instance by creating a Read Replica for the DB instance. This Read Replica is temporary and solely for the purpose of retaining binlog files. After the Read Replica has been created, call the mysql.rds_stop_replication (p. 918) procedure on the Read Replica (the `mysql.rds_stop_replication` procedure is only available for MySQL versions 5.5, 5.6 and later, and 5.7 and later). While replication is stopped, Amazon RDS doesn't delete any of the binlog files on the replication master. After you have set up replication with your permanent replica, you can delete the Read Replica when the replica lag (**Seconds behind master** field) between your replication master and your permanent replica reaches 0.
MySQL (external)	**To retain binary logs on an external MySQL database** Because binlog files on an external MySQL database are not managed by Amazon RDS, they are retained until you delete them.

Database Engine	Instructions
	After replication has been started, you can verify that changes have been applied to your replica by running the `SHOW SLAVE STATUS` command on your replica and checking the **Seconds behind master** field. If the **Seconds behind master** field is 0, then there is no replica lag. When there is no replica lag, you can delete old binlog files.

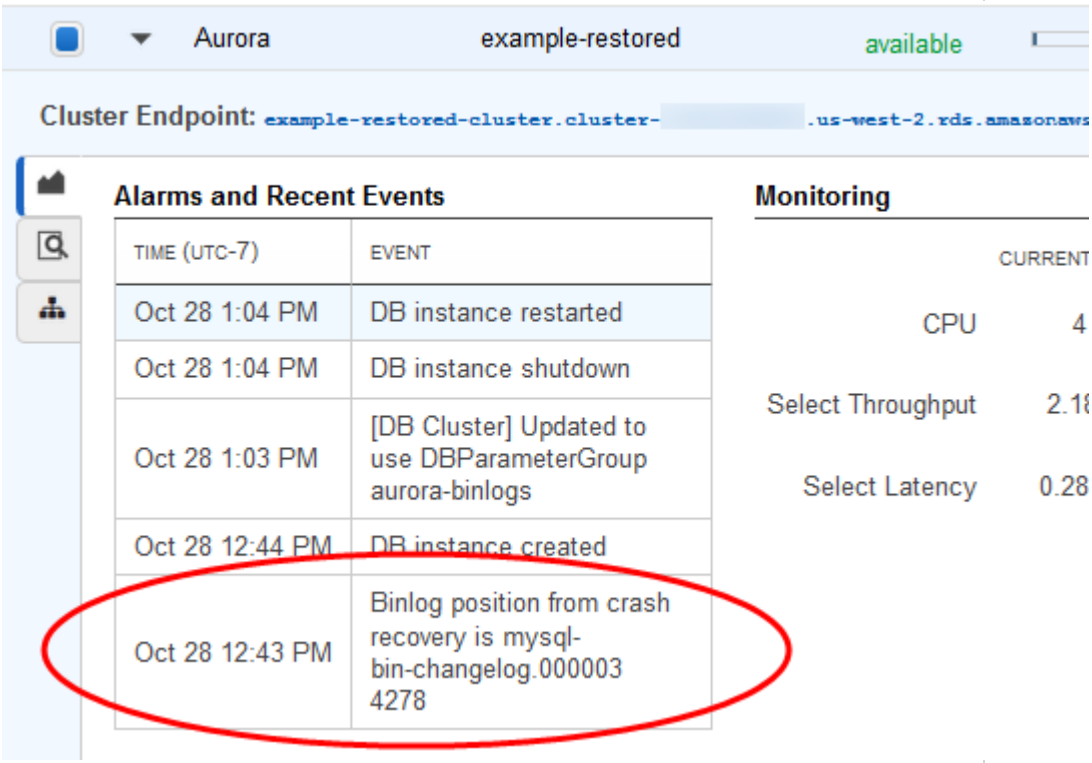
3. Create a Snapshot of Your Replication Master

You use a snapshot of your replication master to load a baseline copy of your data onto your replica and then start replicating from that point on.

Find instructions on how to create a snapshot of your replication master for your database engine following.

Database Engine	Instructions
Aurora	**To create a snapshot of an Aurora MySQL DB cluster** 1. Create a DB cluster snapshot of your Amazon Aurora DB cluster. For more information, see Creating a DB Snapshot (p. 207). 2. Create a new Aurora DB cluster by restoring from the DB cluster snapshot that you just created. Be sure to retain the same DB parameter group for your restored DB cluster as your original DB cluster. This will ensure that the copy of your DB cluster has binary logging enabled. For more information, see Restoring from a DB Snapshot (p. 209). 3. In the console, choose **Instances** and select the primary instance (writer) for your restored Aurora DB cluster. View the **Alarms and Recent Events**. An event message will show that includes the binlog file name and position. The event message is in the following format: `Binlog position from crash recovery is binlog-file-name binlog-position` For example, the following shows an event message where the binlog file name is `mysql-bin-changelog.000003` and the binlog position is `4278`.

Database Engine	Instructions



Save the binlog file name and position values for when you start replication.

You can also get the binlog file name and position by calling the describe-events command from the AWS CLI. The following shows an example `describe-events` command with example output.

```
PROMPT> aws rds describe-events
```

```
{
    "Events": [
        {
            "EventCategories": [],
            "SourceType": "db-instance",
            "SourceArn": "arn:aws:rds:us-west-2:123456789012:db:sample-
restored-instance",
            "Date": "2016-10-28T19:43:46.862Z",
            "Message": "Binlog position from crash recovery is mysql-bin-
changelog.000003 4278",
            "SourceIdentifier": "sample-restored-instance"
        }
    ]
}
```

4. If your replica will be an Aurora DB cluster in another region, an Aurora DB cluster owned by another AWS account, an external MySQL database, or an RDS MySQL DB instance, then you cannot load the data from an Amazon Aurora DB cluster snapshot. Instead, you can create a dump of your Amazon Aurora DB cluster by connecting to your DB cluster using a MySQL client and issuing the `mysqldump` command. Be sure to

Database Engine	Instructions
	run the `mysqldump` command against the copy of your Amazon Aurora DB cluster that you created. The following is an example: `PROMPT> mysqldump --databases <database_name> --single-transaction --order-by-primary -r backup.sql -u <local_user> -p` 5. When you have finished creating the dump of your data from the newly created Aurora DB cluster, delete that DB cluster as it is no longer needed.
RDS MySQL	**To create a snapshot of an Amazon RDS DB instance** 1. Create a Read Replica of your Amazon RDS DB instance. For more information on creating a Read Replica, see Creating a Read Replica (p. 139). 2. Connect to your Read Replica and stop replication by running the mysql.rds_stop_replication (p. 918) command. 3. While the Read Replica is **Stopped**, Connect to the Read Replica and run the `SHOW SLAVE STATUS` command. Retrieve the current binary log file name from the `Relay_Master_Log_File` field and the log file position from the `Exec_Master_Log_Pos` field. Save these values for when you start replication. 4. While the Read Replica remains **Stopped**, create a DB snapshot of the Read Replica. For more information on creating a DB snapshot, see Creating a DB Snapshot (p. 207). 5. Delete the Read Replica.

MySQL (external)	**To create a snapshot of an external MySQL database**
	
	1. Before you create a snapshot, you need to ensure that the binlog location for the snapshot is current with the data in your master instance. To do this, you must first stop any write operations to the instance with the following command:
	
	`mysql> FLUSH TABLES WITH READ LOCK;`
	
	2. Create a dump of your MySQL database using the `mysqldump` command as shown following:
	
	`PROMPT> sudo mysqldump --databases <database_name> --master-data=2 --single-transaction --order-by-primary -r backup.sql -u <local_user> -p`
	
	3. After you have created the snapshot, unlock the tables in your MySQL database with the following command:
	
	`mysql> UNLOCK TABLES;`

4. Load the Snapshot into Your Replica

Before loading the snapshot of your replication master into your replica, make sure that you consider the following:

- If you will be replicating across AWS regions, you cannot use an Aurora MySQL DB cluster snapshot to load your replica. DB cluster snapshots cannot be copied across regions. To work across regions, you can create an Aurora MySQL DB instance in another region from a DB snapshot of an RDS MySQL DB instance. Copy the DB snapshot to the region where your replication slave will be hosted and then

create an Aurora MySQL DB cluster or MySQL DB instance from that snapshot. For information on copying snapshots to other regions, see Copying a DB Snapshot or DB Cluster Snapshot (p. 213).

- If you will be loading data from a dump of a MySQL database that is external to Amazon RDS, then you might want to create an EC2 instance to copy the dump files to, and then load the data into your DB cluster or DB instance from that EC2 instance. Using this approach, you can compress the dump file(s) before copying them to the EC2 instance in order to reduce the network costs associated with copying data to Amazon RDS. You can also encrypt the dump file or files to secure the data as it is being transferred across the network.

Find instructions on how to load the snapshot of your replication master into your replica for your database engine following.

Database Engine	Instructions

Aurora	**To load a snapshot into an Aurora MySQL DB cluster**
	• If the snapshot of your replica master is a DB cluster snapshot, then you can restore from the DB cluster snapshot to create a new Aurora MySQL DB cluster as your replica. For more information, see Restoring from a DB Snapshot (p. 209).
	• If the snapshot of your replica master is a DB snapshot, then you can migrate the data from your DB snapshot into a new Aurora MySQL DB cluster. For more information, see Migrating Data to an Amazon Aurora DB Cluster (p. 466).
	• If the snapshot of your replica master is the output from the `mysqldump` command, then follow these steps:
	1. Copy the output of the `mysqldump` command from your replica master to a location that can also connect to your Aurora MySQL DB cluster.
	2. Connect to your Aurora MySQL DB cluster using the `mysql` command. The following is an example:
	`PROMPT> mysql -h <host_name> -port=3306 -u <db_master_user> -p`
	3. At the `mysql` prompt, run the `source` command and pass it the name of your database dump file to load the data into the Aurora MySQL DB cluster, for example:
	`mysql> source backup.sql;`

RDS MySQL	**To load a snapshot into an Amazon RDS DB instance**
	1. Copy the output of the `mysqldump` command from your replica master to a location that can also connect to your MySQL DB instance.
	2. Connect to your MySQL DB instance using the `mysql` command. The following is an example:
	`PROMPT> mysql -h <host_name> -port=3306 -u <db_master_user> -p`
	3. At the `mysql` prompt, run the `source` command and pass it the name of your database dump file to load the data into the MySQL DB instance, for example:
	`mysql> source backup.sql;`

MySQL (external)	**To load a snapshot into an external MySQL database**

Database Engine	Instructions

	You cannot load a DB snapshot or a DB cluster snapshot into an external MySQL database. Instead, you must use the output from the `mysqldump` command.
	1. Copy the output of the `mysqldump` command from your replica master to a location that can also connect to your MySQL database.
	2. Connect to your MySQL database using the `mysql` command. The following is an example:
	`PROMPT> mysql -h <host_name> -port=3306 -u <db_master_user> -p`
	3. At the `mysql` prompt, run the `source` command and pass it the name of your database dump file to load the data into your MySQL database, for example:
	`mysql> source backup.sql;`

5. Enable Replication

Before you enable replication, we recommend that you take a manual snapshot of the Aurora MySQL DB cluster or RDS MySQL DB instance replica prior to starting replication. If a problem arises and you need to reestablish replication with the DB cluster or DB instance replica, you can restore the DB cluster or DB instance from this snapshot instead of having to import the data into your replica again.

You also might want to create a user ID that is used solely for replication. That user ID will require the `REPLICATION CLIENT` and `REPLICATION SLAVE` privileges. The following is an example:

```
REPLICATION CLIENT and REPLICATION SLAVE privileges. For example:
CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY '<password>';

GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com' IDENTIFIED
 BY '<password>';
```

Find instructions on how to enable replication for your database engine following.

Database Engine	Instructions

Aurora	**To enable replication from an Aurora MySQL DB cluster**
	1. If your DB cluster was created from a DB cluster snapshot, then connect to the DB cluster and issue the `SHOW MASTER STATUS` command. Retrieve the current binary log file name from the `File` field and the log file position from the `Position` field.
	If your DB cluster was created from a DB snapshot, then you need the binlog file and binlog position that are the starting place for replication. You retrieved these values from the `SHOW SLAVE STATUS` command when you created the snapshot of your replication master.
	If your DB cluster was populated from the output of the `mysqldump` command with the `--master-data=2` option, then the binlog file and binlog position are included in the output. The following is an example:
	`--`
	`-- Position to start replication or point-in-time recovery from`

<table>
<tr><th>Database Engine</th><th>Instructions</th></tr>
<tr><td></td><td>

```
--

-- CHANGE MASTER TO MASTER_LOG_FILE='mysql-bin-changelog.000031',
 MASTER_LOG_POS=107;
```

2. Connect to the DB cluster and issue the mysql.rds_set_external_master (p. 914) and mysql.rds_start_replication (p. 917) commands to start replication with your replication master using the binary log file name and location from the previous step. The following is an example:

```
CALL mysql.rds_set_external_master ('mydbinstance.123456789012.us-
east-1.rds.amazonaws.com', 3306,
    'repl_user', '<password>', 'mysql-bin-changelog.000031', 107, 0);
CALL mysql.rds_start_replication;
```

</td></tr>
<tr><td>RDS MySQL</td><td>

To enable replication from an Amazon RDS DB instance

1. If your DB instance was created from a DB snapshot, then you need the binlog file and binlog position that are the starting place for replication. You retrieved these values from the `SHOW SLAVE STATUS` command when you created the snapshot of your replication master.

   If your DB instance was populated from the output of the `mysqldump` command with the `--master-data=2` option, then the binlog file and binlog position are included in the output. The following is an example:

```
--
-- Position to start replication or point-in-time recovery from
--

-- CHANGE MASTER TO MASTER_LOG_FILE='mysql-bin-changelog.000031',
 MASTER_LOG_POS=107;
```

2. Connect to the DB instance and issue the mysql.rds_set_external_master (p. 914) and mysql.rds_start_replication (p. 917) commands to start replication with your replication master using the binary log file name and location from the previous step. The following is an example:

```
CALL mysql.rds_set_external_master ('mydbcluster.cluster-123456789012.us-
east-1.rds.amazonaws.com', 3306,
    'repl_user', '<password>', 'mysql-bin-changelog.000031', 107, 0);
CALL mysql.rds_start_replication;
```

</td></tr>
</table>

Database Engine	Instructions

MySQL (external)	**To enable replication from an external MySQL database**
	1. Retrieve the binlog file and binlog position that are the starting place for replication. You retrieved these values from the `SHOW SLAVE STATUS` command when you created the snapshot of your replication master. If your database was populated from the output of the `mysqldump` command with the `--master-data=2` option, then the binlog file and binlog position are included in the output. The following is an example:
	`-- Position to start replication or point-in-time recovery from --`
	`-- CHANGE MASTER TO MASTER_LOG_FILE='mysql-bin-changelog.000031', MASTER_LOG_POS=107;`
	2. Connect to the database and issue `CHANGE MASTER TO` and `START SLAVE` to start replication with your replication master using the binary log file name and location from the previous step, for example:
	`CHANGE MASTER TO MASTER_HOST = 'mydbcluster.cluster-123456789012.us-east-1.rds.amazonaws.com' MASTER_PORT = 3306 MASTER_USER = 'repl_user' MASTER_PASSWORD = '<password>' MASTER_LOG_FILE = 'mysql-bin-changelog.000031' MASTER_LOG_POS = 107; START SLAVE;`

6. Monitor Your Replica

When you set up MySQL replication with an Aurora MySQL DB cluster, you must monitor failover events for the Aurora MySQL DB cluster when it is the replica. If a failover occurs, then the DB cluster that is your replica might be recreated on a new host with a different network address. For information on how to monitor failover events, see Using Amazon RDS Event Notification (p. 279).

You can also monitor how far the replica is behind the replication master by connecting to the replica and running the `SHOW SLAVE STATUS` command. In the command output, the `Seconds Behind Master` field will tell you how far the replica is behind the master.

Stopping Replication Between Aurora and MySQL or Between Aurora and Another Aurora DB Cluster

To stop binlog replication with a MySQL DB instance, external MySQL database, or another Aurora DB cluster, follow these steps, discussed in detail following in this topic.

1. Stop Binlog Replication (p. 546)

2. Disable Binary Logging on the Replication Master (p. 547)

1. Stop Binlog Replication

Find instructions on how to stop binlog replication for your database engine following.

Database Engine	Instructions
Aurora	**To stop binlog replication on an Aurora MySQL DB cluster** 1. Connect to the Aurora DB cluster that is the replica and call the mysql.rds_stop_replication (p. 918) procedure (the `mysql.rds_stop_replication` procedure is only available for MySQL versions 5.5 and later, 5.6 and later, and 5.7 and later). 2. Set the binlog retention time frame to 0. To set the binlog retention time frame, use the mysql.rds_set_configuration (p. 923) procedure and specify a configuration parameter of '`binlog retention hours`' along with the number of hours to retain binlog files on the DB cluster, in this case 0, as shown in the following example: `CALL mysql.rds_set_configuration('binlog retention hours', 0);`
RDS MySQL	**To stop binlog replication on an Amazon RDS DB instance** Connect to the RDS DB instance that is the replica and call the mysql.rds_stop_replication (p. 918) procedure (the `mysql.rds_stop_replication` procedure is only available for MySQL versions 5.5 and later, 5.6 and later, and 5.7 and later).
MySQL (external)	**To stop binlog replication on an external MySQL database** Connect to the MySQL database and call the `STOP REPLICATION` command.

2. Disable Binary Logging on the Replication Master

Find instructions on how to disable binary logging on the replication master for your database engine following.

Database Engine	Instructions
Aurora	**To disable binary logging on an Amazon Aurora DB cluster** Set the `binlog_format` parameter to `OFF`. The `binlog_format` parameter is a cluster-level parameter that is in the `default.aurora5.6` cluster parameter group by default. After you have changed the `binlog_format` parameter value, reboot your DB cluster for the change to take effect. For more information, see Amazon Aurora DB Cluster and DB Instance Parameters (p. 469) and Modifying Parameters in a DB Parameter Group (p. 172).
RDS MySQL	**To disable binary logging on an Amazon RDS DB instance** You cannot disable binary logging directly for an Amazon RDS DB instance, but you can disable it by doing the following: 1. Disable automated backups for the DB instance. You can disable automated backups by modifying an existing DB instance and setting the **Backup Retention Period** to 0. For more information, see Modifying a DB Instance Running the MySQL Database Engine (p. 843) and Working With Backups (p. 201).

Database Engine	Instructions
	2. Delete all Read Replicas for the DB instance. For more information, see Working with PostgreSQL, MySQL, and MariaDB Read Replicas (p. 134).

MySQL (external)	**To disable binary logging on an external MySQL database**
	
	Connect to the MySQL database and call the `STOP REPLICATION` command.
	
	1. From a command shell, stop the mysql service:
	
	`sudo service mysqld stop`
	
	2. Edit the my.cnf file (this file is usually under /etc):
	
	`sudo vi /etc/my.cnf`
	
	Delete the `log_bin` and `server_id` options from the `[mysqld]` section.
	
	For more information, see Setting the Replication Master Configuration in the MySQL documentation.
	
	3. Start the mysql service:
	
	`sudo service mysqld start`

Related Topics

- Migrating Data to an Amazon Aurora DB Cluster (p. 466)
- Amazon Aurora on Amazon RDS (p. 428)

Security with Amazon Aurora MySQL

Security for Amazon Aurora MySQL is managed at three levels:

- To control who can perform Amazon RDS management actions on Aurora MySQL DB clusters and DB instances, you use AWS Identity and Access Management (IAM). When you connect to AWS using IAM credentials, your IAM account must have IAM policies that grant the permissions required to perform Amazon RDS management operations. For more information, see Authentication and Access Control for Amazon RDS (p. 327).

  If you are using an IAM account to access the Amazon RDS console, you must first log on to the AWS Management Console with your IAM account. You then go to the Amazon RDS console at https://console.aws.amazon.com/rds.

- Aurora MySQL DB clusters must be created in an Amazon Virtual Private Cloud (VPC). To control which devices and Amazon EC2 instances can open connections to the endpoint and port of the DB instance for Aurora MySQL DB clusters in a VPC, you use a VPC security group. These endpoint and port connections can be made using Secure Sockets Layer (SSL). In addition, firewall rules at your company can control whether devices running at your company can open connections to a DB instance. For more information on VPCs, see Amazon Virtual Private Cloud (VPCs) and Amazon RDS (p. 390).

  The supported VPC tenancy depends on the instance class used by your Aurora MySQL DB clusters. With `default` VPC tenancy, the VPC runs on shared hardware. With `dedicated` VPC tenancy, the VPC

runs on a dedicated hardware instance. Aurora MySQL supports the following VPC tenancy based on the instance class:

- The db.r3 instance classes support both `default` and `dedicated` VPC tenancy.
- The db.r4 instance classes support `default` VPC tenancy only.
- The db.t2 instance classes support `default` VPC tenancy only.

For more information about instance classes, see DB Instance Class (p. 92). For more information about `default` and `dedicated` VPC tenancy, see Dedicated Instances in the *Amazon Elastic Compute Cloud User Guide*.

- To authenticate login and permissions for an Amazon Aurora MySQL DB cluster, you can take either of the following approaches, or a combination of them:
  - You can take the same approach as with a standalone instance of MySQL.

    Commands such as `CREATE USER`, `RENAME USER`, `GRANT`, `REVOKE`, and `SET PASSWORD` work just as they do in on-premises databases, as does directly modifying database schema tables. For more information, see .For information, see MySQL User Account Management in the MySQL documentation.
  - You can also use IAM database authentication.

    With IAM database authentication, you authenticate to your DB cluster by using an IAM user or IAM role and an authentication token. An *authentication token* is a unique value that is generated using the Signature Version 4 signing process. By using IAM database authentication, you can use the same credentials to control access to your AWS resources and your databases. For more information, see IAM Database Authentication for MySQL and Amazon Aurora (p. 360).

When you create an Amazon Aurora MySQL DB instance, the master user has the following default privileges:

- `ALTER`
- `ALTER ROUTINE`
- `CREATE`
- `CREATE ROUTINE`
- `CREATE TEMPORARY TABLES`
- `CREATE USER`
- `CREATE VIEW`
- `DELETE`
- `DROP`
- `EVENT`
- `EXECUTE`
- `GRANT OPTION`
- `INDEX`
- `INSERT`
- `LOAD FROM S3`
- `LOCK TABLES`
- `PROCESS`
- `REFERENCES`
- `RELOAD`
- `REPLICATION CLIENT`
- `REPLICATION SLAVE`

- SELECT
- SHOW DATABASES
- SHOW VIEW
- TRIGGER
- UPDATE

To provide management services for each DB cluster, the rdsadmin user is created when the DB cluster is created. Attempting to drop, rename, change the password, or change privileges for the rdsadmin account results in an error.

For management of the Aurora MySQL DB cluster, the standard kill and kill_query commands have been restricted. Instead, use the Amazon RDS commands rds_kill and rds_kill_query to terminate user sessions or queries on Aurora MySQL DB instances.

Securing Aurora MySQL Data with SSL

Amazon Aurora MySQL DB clusters support Secure Sockets Layer (SSL) connections from applications using the same process and public key as Amazon RDS MySQL DB instances.

Amazon RDS creates an SSL certificate and installs the certificate on the DB instance when Amazon RDS provisions the instance. These certificates are signed by a certificate authority. The SSL certificate includes the DB instance endpoint as the Common Name (CN) for the SSL certificate to guard against spoofing attacks. As a result, you can only use the DB cluster endpoint to connect to a DB cluster using SSL if your client supports Subject Alternative Names (SAN). Otherwise, you must use the endpoint of the primary instance.

We recommend the MariaDB Connector/J client as a client that supports SAN with SSL. For more information, see the MariaDB Connector/J download page.

The public key is stored at https://s3.amazonaws.com/rds-downloads/rds-combined-ca-bundle.pem.

To encrypt connections using the default **mysql** client, launch the mysql client using the --ssl-ca parameter to reference the public key, for example:

```
mysql -h mycluster-primary.123456789012.us-east-1.rds.amazonaws.com --ssl-
ca=[full path]rds-combined-ca-bundle.pem --ssl-verify-server-cert
```

You can use the GRANT statement to require SSL connections for specific users accounts. For example, you can use the following statement to require SSL connections on the user account encrypted_user.

```
GRANT USAGE ON *.* TO 'encrypted_user'@'%' REQUIRE SSL
```

> **Note**
> For more information on SSL connections with MySQL, see the MySQL documentation.

Integrating Amazon Aurora MySQL with Other AWS Services

Amazon Aurora MySQL integrates with other AWS services so that you can extend your Aurora MySQL DB cluster to use additional capabilities in the AWS Cloud. Your Aurora MySQL DB cluster can use AWS services to do the following:

- Asynchronously invoke an AWS Lambda function using the mysql.lambda_async procedure. For more information, see Invoking a Lambda Function from an Amazon Aurora MySQL DB Cluster (p. 572).

- Load data from text or XML files stored in an Amazon Simple Storage Service (Amazon S3) bucket into your DB cluster using the `LOAD DATA FROM S3` or `LOAD XML FROM S3` command. For more information, see Loading Data into an Amazon Aurora MySQL DB Cluster from Text Files in an Amazon S3 Bucket (p. 560).
- Save data to text files stored in an Amazon S3 bucket from your DB cluster using the `SELECT INTO OUTFILE S3` command. For more information, see Saving Data from an Amazon Aurora MySQL DB Cluster into Text Files in an Amazon S3 Bucket (p. 567).
- Automatically add or remove Aurora Replicas with Application Auto Scaling. For more information, see Using Amazon Aurora Auto Scaling with Aurora Replicas (p. 577).

Aurora secures the ability to access other AWS services by using AWS Identity and Access Management (IAM). You grant permission to access other AWS services by creating an IAM role with the necessary permissions, and then associating the role with your DB cluster. For details and instructions on how to permit your Aurora MySQL DB cluster to access other AWS services on your behalf, see Authorizing Amazon Aurora MySQL to Access Other AWS Services on Your Behalf (p. 551).

Authorizing Amazon Aurora MySQL to Access Other AWS Services on Your Behalf

Note
Integration with other AWS services is available for Amazon Aurora MySQL version 1.8 and later. Some integration features are only available for later versions of Aurora MySQL. For more information on Aurora versions, see Amazon Aurora MySQL Database Engine Updates (p. 610).

For your Aurora MySQL DB cluster to access other services on your behalf, you must create and configure an AWS Identity and Access Management (IAM) role to authorize database users in your DB cluster to access other AWS services. For more information, see Setting Up IAM Roles to Access AWS Services (p. 551).

You must also configure your Aurora DB cluster to allow outbound connections to the target AWS service. For more information, see Enabling Network Communication from Amazon Aurora MySQL to Other AWS Services (p. 560).

If you do so, your database users can perform these actions using other AWS services:

- Asynchronously invoke an AWS Lambda function using the `mysql.lambda_async` procedure. For more information, see Invoking a Lambda Function from an Amazon Aurora MySQL DB Cluster (p. 572).
- Load data from text or XML files stored in an Amazon S3 bucket into your DB cluster by using the `LOAD DATA FROM S3` or `LOAD XML FROM S3` statement. For more information, see Loading Data into an Amazon Aurora MySQL DB Cluster from Text Files in an Amazon S3 Bucket (p. 560).
- Save data from your DB cluster into text files stored in an Amazon S3 bucket by using the `SELECT INTO OUTFILE S3` statement. For more information, see Saving Data from an Amazon Aurora MySQL DB Cluster into Text Files in an Amazon S3 Bucket (p. 567).
- Export audit log data to Amazon CloudWatch Logs MySQL. For more information, see Exporting Audit Log Data From Amazon Aurora to Amazon CloudWatch Logs (p. 576).
- Automatically add or remove Aurora Replicas with Application Auto Scaling. For more information, see Using Amazon Aurora Auto Scaling with Aurora Replicas (p. 577).

Setting Up IAM Roles to Access AWS Services

To permit your Aurora DB cluster to access another AWS service, do the following:

1. Create an IAM policy that grants permission to the AWS service. For more information, see:

- Creating an IAM Policy to Access Amazon S3 Resources (p. 552)
- Creating an IAM Policy to Access AWS Lambda Resources (p. 553)
- Allowing Amazon Aurora to Access Amazon CloudWatch Logs Resources (p. 554)

2. Create an IAM role and attach the policy that you created. For more information, see Creating an IAM Role to Allow Amazon Aurora to Access AWS Services (p. 555).
3. Associate that IAM role with your Aurora DB cluster. For more information, see Associating an IAM Role with an Amazon Aurora MySQL DB Cluster (p. 556).

Creating an IAM Policy to Access Amazon S3 Resources

Aurora can access Amazon S3 resources to either load data to or save data from an Aurora DB cluster. However, you must first create an IAM policy that provides the bucket and object permissions that allow Aurora to access Amazon S3.

The following table lists the Aurora features that can access an Amazon S3 bucket on your behalf, and the minimum required bucket and object permissions required by each feature.

Feature	Bucket Permissions	Object Permissions
`LOAD DATA FROM S3`	`ListBucket`	`GetObject` `GetObjectVersion`
`LOAD XML FROM S3`	`ListBucket`	`GetObject` `GetObjectVersion`

`SELECT INTO OUTFILE S3`	`ListBucket`	`AbortMultipartUpload`
		`DeleteObject`
		`GetObject`
		`ListMultipartUploadParts`
		`PutObject`

You can use the following steps to create an IAM policy that provides the minimum required permissions for Aurora to access an Amazon S3 bucket on your behalf. To allow Aurora to access all of your Amazon S3 buckets, you can skip these steps and use either the `AmazonS3ReadOnlyAccess` or `AmazonS3FullAccess` predefined IAM policy instead of creating your own.

To create an IAM policy to grant access to your Amazon S3 resources

1. Open the IAM Console.
2. In the navigation pane, choose **Policies**.
3. Choose **Create Policy**.
4. For **Policy Generator**, choose **Select**.
5. In **Edit Permissions**, set the following values to grant bucket permissions:
   - **Effect** – `Allow`
   - **AWS Service** – `Amazon S3`
   - **Actions** – Specify the bucket permissions needed for the IAM policy.

Bucket permissions are permissions for bucket operations in Amazon S3, and need to be granted on either a wildcard (*) or a bucket. For more information about permissions for bucket operations in Amazon S3, see Specifying Permissions in a Policy.

- Set **Amazon Resource Name (ARN)** to the ARN of the Amazon S3 bucket to allow access to. For instance, if you want to allow Aurora to access the Amazon S3 bucket named `example-bucket`, then set the ARN value to `arn:aws:s3:::example-bucket`.

6. Choose **Add Statement**.

   Note
   You can repeat this and the previous step to add corresponding bucket permission statements to your policy for each Amazon S3 bucket that you want Aurora to access. Optionally, you can also grant access to all buckets and objects in Amazon S3.

7. In **Edit Permissions**, set the following values to grant object permissions:

   - **Effect** – `Allow`
   - **AWS Service** – `Amazon S3`
   - **Actions** – Specify the object permissions needed for the IAM policy.

     Object permissions are permissions for object operations in Amazon S3, and need to be granted for objects in a bucket, not the bucket itself. For more information about permissions for object operations in Amazon S3, see Specifying Permissions in a Policy.
   - Set **Amazon Resource Name (ARN)** to the ARN of the Amazon S3 bucket to allow access to. For instance, if you want to allow Aurora to access all of the files in the Amazon S3 bucket named `example-bucket`, then set the ARN value to `arn:aws:s3:::example-bucket/*`.

     Note
     You can set **Amazon Resource Name (ARN)** to a more specific ARN value in order to allow Aurora to access only specific files or folders in an Amazon S3 bucket. For more information about how to define an access policy for Amazon S3, see Managing Access Permissions to Your Amazon S3 Resources.

8. Choose **Add Statement**.

   Note
   You can repeat this and the previous step to add corresponding object permission statements to your policy for each Amazon S3 bucket that you want Aurora to access. Optionally, you can also grant access to all buckets and objects in Amazon S3.

9. Choose **Next Step**.
10. Set **Policy Name** to a name for your IAM policy, for example `AllowAuroraToExampleBucket`. You use this name when you create an IAM role to associate with your Aurora DB cluster. You can also add an optional **Description** value.
11. Choose **Create Policy**.

Creating an IAM Policy to Access AWS Lambda Resources

You can use the following steps to create an IAM policy that provides the minimum required permissions for Aurora to invoke an AWS Lambda function on your behalf. To allow Aurora to invoke all of your AWS Lambda functions, you can skip these steps and use the predefined `AWSLambdaRole` policy instead of creating your own.

To create an IAM policy to grant invoke to your AWS Lambda functions:

1. Open the IAM Console.
2. In the navigation pane, choose **Policies**.

3. Choose **Create Policy**.
4. For the **Policy Generator** option, choose **Select**.
5. In **Edit Permissions**, set the following values:

   - **Effect** – `Allow`
   - **AWS Service** – `AWS Lambda`
   - **Actions** – `InvokeFunction`

   These permissions are the minimum required to enable Amazon Aurora to invoke an AWS Lambda function.
6. Set **Amazon Resource Name (ARN)** to the ARN of the Lambda function to allow access to. For instance, if you want to allow Aurora to access a Lambda function named `example_function`, then set the ARN value to `arn:aws:lambda:::function:example_function`.

   For more information on how to define an access policy for AWS Lambda, see Authentication and Access Control for AWS Lambda.
7. Choose **Add Statement**.

   You can repeat this and the previous step to add multiple ARNs to your policy and allow Aurora to invoke more than one Lambda function.
8. Choose **Next Step**.
9. Set the **Policy Name** to a name for your IAM policy, for example `AllowAuroraToExampleFunction`. You will use this name when you create an IAM role to associate with your Aurora DB cluster. You can also add an optional **Description** value.
10. Choose **Create Policy**.

Allowing Amazon Aurora to Access Amazon CloudWatch Logs Resources

Aurora can access CloudWatch Logs to export audit log data from an Aurora DB cluster. However, you must first create an IAM policy that provides the log group and log stream permissions that allow Aurora to access CloudWatch Logs.

The following policy adds the permissions required by Aurora to access Amazon CloudWatch Logs on your behalf, and the minimum required permissions to create log groups and export data.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EnableCreationAndManagementOfRDSCloudwatchLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:PutRetentionPolicy"
            ],
            "Resource": [
                "arn:aws:logs:*:*:log-group:/aws/rds/*"
            ]
        },
        {
            "Sid": "EnableCreationAndManagementOfRDSCloudwatchLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:DescribeLogStreams",
                "logs:GetLogEvents"
```

```
            ],
            "Resource": [
                "arn:aws:logs:*:*:log-group:/aws/rds/*:log-stream:*"
            ]
        }
    ]
}
```

You can use the following steps to create an IAM policy that provides the minimum required permissions for Aurora to access CloudWatch Logs on your behalf. To allow Aurora full access to CloudWatch Logs, you can skip these steps and use the `CloudWatchLogsFullAccess` predefined IAM policy instead of creating your own. For more information, see Using Identity-Based Policies (IAM Policies) for CloudWatch Logs.

To create an IAM policy to grant access to your CloudWatch Logs resources

1. Open the IAM Console.
2. In the navigation pane, choose **Policies**.
3. Choose **Create Policy**.
4. For **Policy Generator**, choose **Select**.
5. In **Edit Permissions**, set the following values to grant CloudWatch Logs log group permissions:
   - **Effect** – `Allow`
   - **AWS Service** – `Amazon CloudWatch Logs`
   - **Actions** – `logs:CreateLogGroup`, `logs:PutRetentionPolicy`
   - **Amazon Resource Name (ARN)** – `arn:aws:logs:*:*:log-group:/aws/rds/*`
6. Choose **Add Statement**.
7. In **Edit Permissions**, set the following values to grant CloudWatch Logs log stream permissions:
   - **Effect** – `Allow`
   - **AWS Service** – `Amazon CloudWatch Logs`
   - **Actions** – `logs:CreateLogStream`, `logs:PutLogEvents`, `logs:DescribeLogStreams`, `logs:GetLogEvents`
   - **Amazon Resource Name (ARN)** – `arn:aws:logs:*:*:log-group:/aws/rds/*:log-stream:*`
8. Choose **Add Statement**.
9. Choose **Next Step**.
10. Set **Policy Name** to a name for your IAM policy, for example `AmazonRDSCloudWatchLogs`. You use this name when you create an IAM role to associate with your Aurora DB cluster. You can also add an optional **Description** value.
11. Choose **Create Policy**.

Creating an IAM Role to Allow Amazon Aurora to Access AWS Services

After creating an IAM policy to allow Aurora to access AWS resources, you must create an IAM role and attach the IAM policy to the new IAM role.

To create an IAM role to permit your Amazon RDS cluster to communicate with other AWS services on your behalf, take the following steps.

To create an IAM role to allow Amazon RDS to access AWS services

1. Open the IAM Console.

2. In the navigation pane, choose **Roles**.
3. Choose **Create New Role**.
4. For **Role Name**, type a name for your role, for example **`RDSLoadFromS3`**. Choose **Next Step**.
5. Choose **AWS Service Roles**, and then scroll to **Amazon RDS**. Choose **Select**.
6. Choose **Next Step**
7. Review the information, and then choose **Create Role**.
8. In the list of IAM roles, select your newly created role. Choose the **Permissions** tab, and then choose **Attach Policy**.
9. Select the policy that you defined earlier in either Creating an IAM Policy to Access Amazon S3 Resources (p. 552), Creating an IAM Policy to Access AWS Lambda Resources (p. 553), or Allowing Amazon Aurora to Access Amazon CloudWatch Logs Resources (p. 554).
10. Choose **Attach Policy**.

Associating an IAM Role with an Amazon Aurora MySQL DB Cluster

To permit database users in an Amazon Aurora DB cluster to access other AWS services, you associate the role that you created in Creating an IAM Role to Allow Amazon Aurora to Access AWS Services (p. 555) with that DB cluster.

To associate an IAM role with a DB cluster you do two things:

- Add the role to the list of associated roles for a DB cluster by using the RDS console, the add-role-to-db-cluster AWS CLI command, or the AddRoleToDBCluster RDS API action.

  You can add a maximum of five IAM roles for each Aurora DB cluster.
- Set the cluster-level parameter for the related AWS service to the ARN for the associated IAM role.

  The following table describes the cluster-level parameter names for the IAM roles used to access other AWS services.

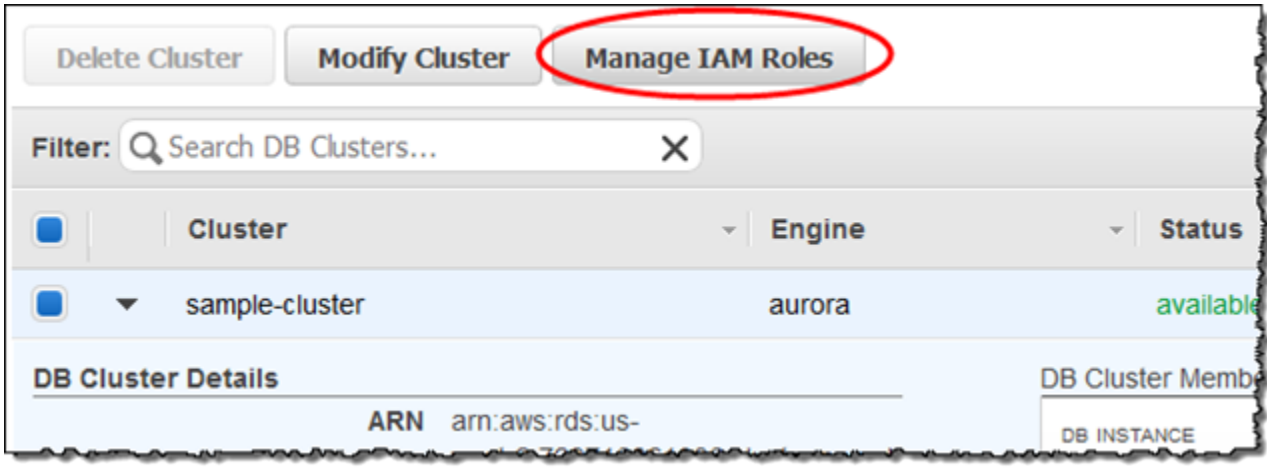
Cluster-level Parameter	Description	
`aws_default_lambda_role`	Used when invoking a Lambda function from your DB cluster.	
`aws_default_logs_role`	Used when exporting audit log data from your DB cluster to Amazon CloudWatch Logs.	
`aws_default_s3_role`	Used when invoking the `LOAD DATA FROM S3`, `LOAD XML FROM S3`, or `SELECT INTO OUTFILE S3` statement from your DB cluster. For Aurora version 1.13 or later, the IAM role specified in this parameter is used only if an IAM role isn't specified for `aurora_load_from_s3_role` or `aurora_select_into_s3_role` for the appropriate statement. For earlier versions of Aurora, the IAM role specified for this parameter is always used.	
`aurora_load_from_s3_role`	For Aurora version 1.13 or later, used when invoking the `LOAD DATA FROM S3` or `LOAD XML`	

Cluster-level Parameter	Description	
	`FROM S3` statement from your DB cluster. If an IAM role is not specified for this parameter, the IAM role specified in `aws_default_s3_role` is used. For earlier versions of Aurora, this parameter is not available.	
`aurora_select_into_s3_role`	For Aurora version 1.13 or later, used when invoking the `SELECT INTO OUTFILE S3` statement from your DB cluster. If an IAM role is not specified for this parameter, the IAM role specified in `aws_default_s3_role` is used. For earlier versions of Aurora, this parameter is not available.	

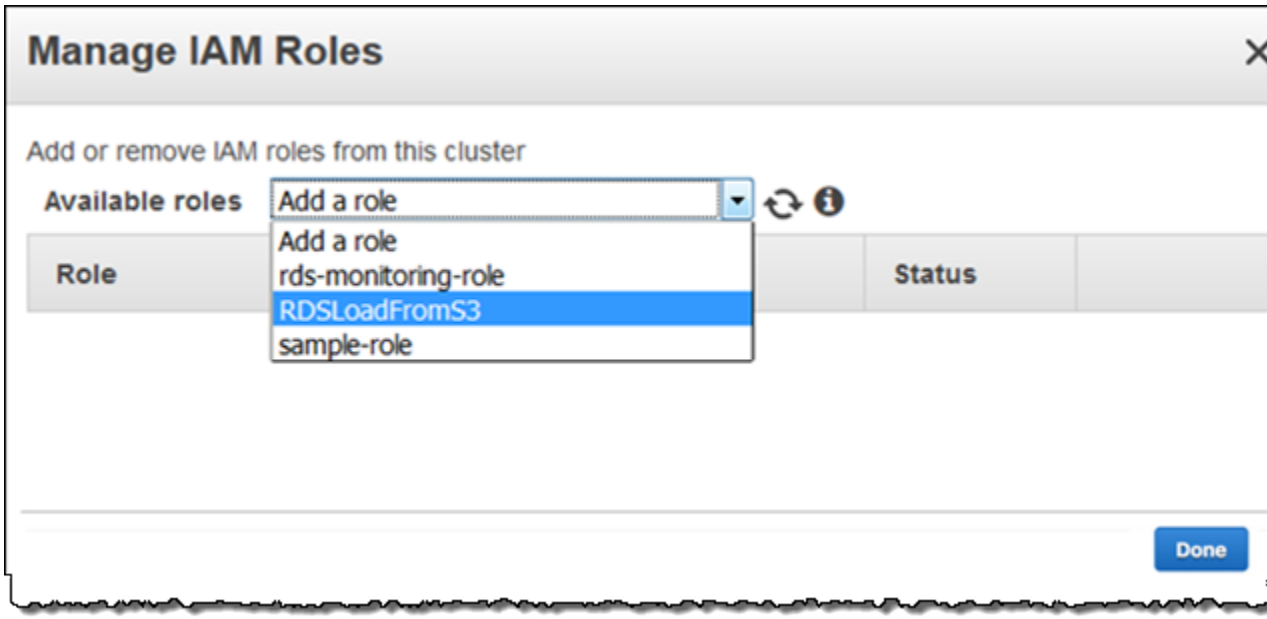
To associate an IAM role to permit your Amazon RDS cluster to communicate with other AWS services on your behalf, take the following steps.

To associate an IAM role with an Aurora DB cluster using the console

1. Open the RDS console at https://console.aws.amazon.com/rds/.
2. Choose **Clusters**.
3. Choose the Aurora DB cluster that you want to associate an IAM role with, and then choose **Manage IAM Roles**.

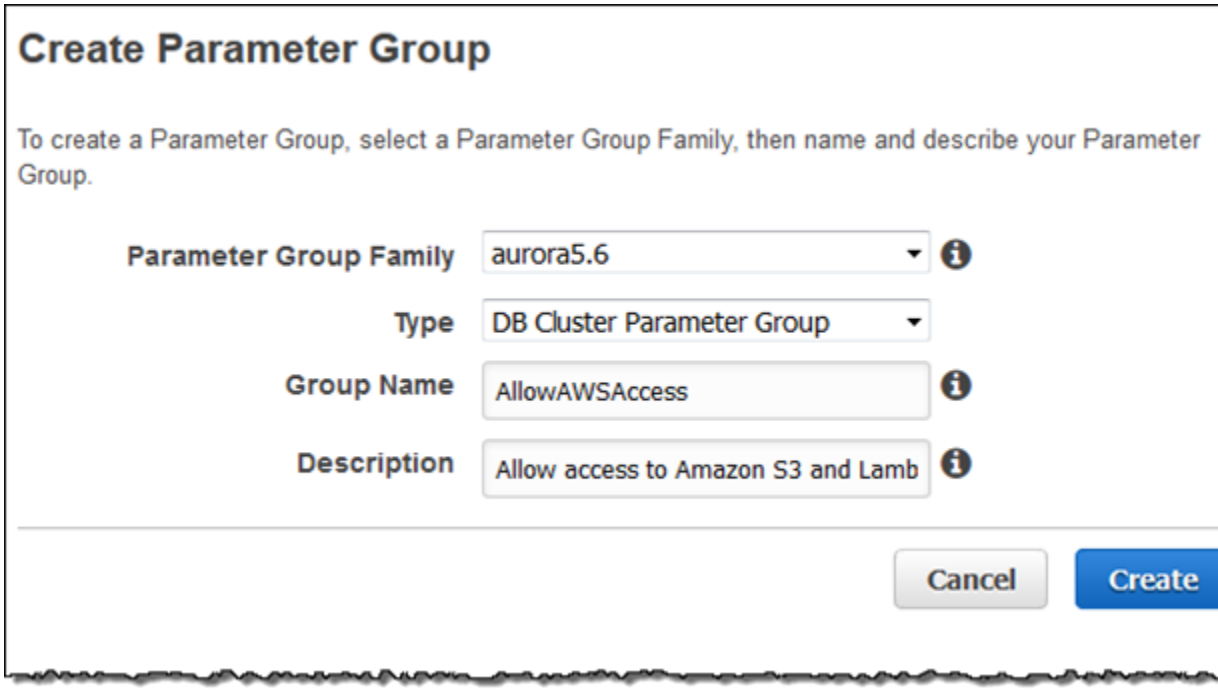


4. In **Manage IAM Roles**, choose the role to associate with your DB cluster from **Available roles**.



5. (Optional) To stop associating an IAM role with a DB cluster and remove the related permission, choose **Delete** for the role.
6. Choose **Done**.
7. In the RDS console, choose **Parameter Groups** in the navigation pane.
8. If you are already using a custom DB parameter group, you can select that group to use instead of creating a new DB cluster parameter group. If you are using the default DB cluster parameter group, you will need to create a new DB cluster parameter group, as described in the following steps:

   a. Choose **Create Parameter Group**.



   For **Parameter Group Family**, choose `aurora5.6`.

b. For **Type**, choose `DB cluster parameter group`.
c. For **Group Name**, type the name of your new DB cluster parameter group.
d. For **Description**, type a description for your new DB cluster parameter group.
e. Choose **Create**.

9. Select your DB cluster parameter group and choose **Edit Parameters**.
10. Set the appropriate cluster-level parameters to the related IAM role ARN values. For example, you can set just the `aws_default_s3_role` parameter to `arn:aws:iam::123456789012:role/AllowAuroraS3Role`.
11. Choose **Save Changes**.
12. Choose **Instances**, and then select the primary instance for your Aurora DB cluster.
13. Choose **Instance Actions** and then choose **Modify**.
14. Set the **DB Cluster Parameter Group** to the new DB cluster parameter group that you created. Select **Apply Immediately**. Choose **Continue**.
15. Verify your changes and then choose **Modify DB Instance**.
16. The primary instance for your DB cluster will still be selected in the list of instances. Choose **Instance Actions**, and then choose **Reboot**.

    When the instance has rebooted, your IAM roles will be associated with your DB cluster.

    For more information about cluster parameter groups, see Amazon Aurora MySQL Parameters (p. 600).

To associate an IAM role with a DB cluster by using the AWS CLI

1. Call the `add-role-to-db-cluster` command from the AWS CLI to add the ARNs for your IAM roles to the DB cluster, as shown following.

```
PROMPT> aws rds add-role-to-db-cluster --db-cluster-identifier my-cluster --role-arn
 arn:aws:iam::123456789012:role/AllowAuroraS3Role
PROMPT> aws rds add-role-to-db-cluster --db-cluster-identifier my-cluster --role-arn
 arn:aws:iam::123456789012:role/AllowAuroraLambdaRole
```

2. If you are using the default DB cluster parameter group, you will need to create a new DB cluster parameter group. If you are already using a custom DB parameter group, you can use that group instead of creating a new DB cluster parameter group.

   To create a new DB cluster parameter group, call the `create-db-cluster-parameter-group` command from the AWS CLI, as shown following.

```
PROMPT> aws rds create-db-cluster-parameter-group  --db-cluster-parameter-group-name
 AllowAWSAccess \
     --db-parameter-group-family aurora5.6 --description "Allow access to Amazon S3 and
 AWS Lambda"
```

3. Set the appropriate cluster-level parameter or parameters and the related IAM role ARN values in your DB cluster parameter group, as shown following.

```
PROMPT> aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-name
 AllowAWSAccess \
    --parameters "name=aws_default_s3_role,value=arn:aws:iam::123456789012:role/
AllowAuroraS3Role,method=pending-reboot" \
    --parameters "name=aws_default_lambda_role,value=arn:aws:iam::123456789012:role/
AllowAuroraLambdaRole,method=pending-reboot"
```

4. Modify the DB cluster to use the new DB cluster parameter group and then reboot the cluster, as shown following.

```
PROMPT> aws rds modify-db-cluster --db-cluster-identifier my-cluster --db-cluster-
parameter-group-name AllowAWSAccess
PROMPT> aws rds reboot-db-instance --db-instance-identifier my-cluster-primary
```

   When the instance has rebooted, your IAM roles will be associated with your DB cluster.

   For more information about cluster parameter groups, see Amazon Aurora MySQL Parameters (p. 600).

Enabling Network Communication from Amazon Aurora MySQL to Other AWS Services

To invoke AWS Lambda functions or access files from Amazon S3, the network configuration of your Aurora DB cluster must allow outbound connections to endpoints for those services. Aurora returns the following error messages if it can't connect to a service endpoint.

```
ERROR 1871 (HY000): S3 API returned error: Network Connection
```

```
ERROR 1873 (HY000): Lambda API returned error: Network Connection. Unable to connect to
 endpoint
```

If you encounter these messages while invoking AWS Lambda functions or accessing files from Amazon S3, check if your Aurora DB cluster is public or private. If your Aurora DB cluster is private, you must configure it to enable connections.

For an Aurora DB cluster to be public, it must be marked as publicly accessible. If you look at the details for the DB cluster in the AWS Management Console, **Publicly Accessible** is **Yes** if this is the case. The DB cluster must also be in an Amazon VPC public subnet. For more information about publicly accessible DB instances, see Working with an Amazon RDS DB Instance in a VPC (p. 399). For more information about public Amazon VPC subnets, see Your VPC and Subnets.

If your Aurora DB cluster isn't publicly accessible and in a VPC public subnet, it is private. If your DB cluster is private and you want to invoke AWS Lambda functions or access Amazon S3 files, configure the cluster so it can connect to Internet addresses through Network Address Translation (NAT). As an alternative for Amazon S3, you can instead configure the VPC to have a VPC endpoint for Amazon S3 associated with the DB cluster's route table. For more information about configuring NAT in your VPC, see NAT Gateways. For more information about configuring VPC endpoints, see VPC Endpoints.

Related Topics

- Integrating Aurora with Other AWS Services (p. 483)
- Amazon Aurora on Amazon RDS (p. 428)

Loading Data into an Amazon Aurora MySQL DB Cluster from Text Files in an Amazon S3 Bucket

You can use the `LOAD DATA FROM S3` or `LOAD XML FROM S3` statement to load data from files stored in an Amazon S3 bucket.

Note
Loading data into a table from text files in an Amazon S3 bucket is available for Amazon Aurora MySQL version 1.8 and later. For more information about Aurora MySQL versions, see Amazon Aurora MySQL Database Engine Updates (p. 610).

Giving Aurora Access to Amazon S3

Before you can load data from an Amazon S3 bucket, you must give your Aurora MySQL DB cluster permission to access Amazon S3. To grant permission, create an AWS Identity and Access Management (IAM) role with the necessary permissions, and then associate the role with your DB cluster. You must also configure your Aurora MySQL DB cluster to allow outbound connections to Amazon S3. For details and instructions on how to permit your Aurora MySQL DB cluster to communicate with Amazon S3 on your behalf, see Setting Up IAM Roles to Access AWS Services (p. 551).

Note
You must set either the `aurora_load_from_s3_role` or `aws_default_s3_role` DB cluster parameter to the Amazon Resource Name (ARN) of the new IAM role. If an IAM role isn't specified for the `aurora_load_from_s3_role`, the IAM role specified in `aws_default_s3_role` is used.
For more information about DB cluster parameters, see Amazon Aurora DB Cluster and DB Instance Parameters (p. 469).

Granting Privileges to Load Data in Amazon Aurora MySQL

The database user that issues the `LOAD DATA FROM S3` or `LOAD XML FROM S3` statement must be granted the `LOAD FROM S3` privilege to issue either statement. The master username for a DB cluster is granted the `LOAD FROM S3` privilege by default. You can grant the privilege to another user by using the following statement.

```
GRANT LOAD FROM S3 ON *.* TO user@domain-or-ip-address
```

The `LOAD FROM S3` privilege is specific to Amazon Aurora and is not available for MySQL databases or RDS MySQL DB instances. If you have set up replication between an Aurora DB cluster as the replication master and a MySQL database as the replication client, then the `GRANT LOAD FROM S3` statement causes replication to stop with an error. You can safely skip the error to resume replication. To skip the error on an RDS MySQL DB instance, use the mysql.rds_skip_repl_error (p. 918) statement. To skip the error on an external MySQL database, use the SET GLOBAL sql_slave_skip_counter statement.

Specifying a Path to an Amazon S3 Bucket

The syntax for specifying a path to files stored on an Amazon S3 bucket is as follows.

```
s3-region://bucket-name/file-name-or-prefix
```

The path includes the following values:

- `region` (optional) – The AWS Region that contains the Amazon S3 bucket to load from. This value is optional. If you don't specify a `region` value, then Aurora loads your file from Amazon S3 in the same region as your DB cluster.
- `bucket-name` – The name of the Amazon S3 bucket that contains the data to load. Object prefixes that identify a virtual folder path are supported.
- `file-name-or-prefix` – The name of the Amazon S3 text file or XML file, or a prefix that identifies one or more text or XML files to load. You can also specify a manifest file that identifies one or more text files to load. For more information about using a manifest file to load text files from Amazon S3, see Using a Manifest to Specify Data Files to Load (p. 563).

LOAD DATA FROM S3

You can use the `LOAD DATA FROM S3` statement to load data from any text file format that is supported by the MySQL LOAD DATA INFILE statement, such as text data that is comma-delimited. Compressed files are not supported.

Syntax

```
LOAD DATA FROM S3 [FILE | PREFIX | MANIFEST] 'S3-URI'
    [REPLACE | IGNORE]
    INTO TABLE tbl_name
    [PARTITION (partition_name,...)]
    [CHARACTER SET charset_name]
    [{FIELDS | COLUMNS}
        [TERMINATED BY 'string']
        [[OPTIONALLY] ENCLOSED BY 'char']
        [ESCAPED BY 'char']
    ]
    [LINES
        [STARTING BY 'string']
        [TERMINATED BY 'string']
    ]
    [IGNORE number {LINES | ROWS}]
    [(col_name_or_user_var,...)]
    [SET col_name = expr,...]
```

Parameters

Following, you can find a list of the required and optional parameters used by the `LOAD DATA FROM S3` statement. You can find more details about some of these parameters in LOAD DATA INFILE Syntax in the MySQL documentation.

- **FILE | PREFIX | MANIFEST** – Identifies whether to load the data from a single file, from all files that match a given prefix, or from all files in a specified manifest. `FILE` is the default.
- **S3-URI** – Specifies the URI for a text or manifest file to load, or an Amazon S3 prefix to use. Specify the URI using the syntax described in Specifying a Path to an Amazon S3 Bucket (p. 561).
- **REPLACE | IGNORE** – Determines what action to take if an input row as the same unique key values as an existing row in the database table.
  - Specify `REPLACE` if you want the input row to replace the existing row in the table.
  - Specify `IGNORE` if you want to discard the input row. `IGNORE` is the default.
- **INTO TABLE** – Identifies the name of the database table to load the input rows into.
- **PARTITION** – Requires that all input rows be inserted into the partitions identified by the specified list of comma-separated partition names. If an input row cannot be inserted into one of the specified partitions, then the statement fails and an error is returned.
- **CHARACTER SET** – Identifies the character set of the data in the input file.
- **FIELDS | COLUMNS** – Identifies how the fields or columns in the input file are delimited. Fields are tab-delimited by default.
- **LINES** – Identifies how the lines in the input file are delimited. Lines are delimited by a carriage return by default.
- **IGNORE *number* LINES | ROWS** – Specifies to ignore a certain number of lines or rows at the start of the input file. For example, you can use `IGNORE 1 LINES` to skip over an initial header line containing column names, or `IGNORE 2 ROWS` to skip over the first two rows of data in the input file.
- **col_name_or_user_var, ...** – Specifies a comma-separated list of one or more column names or user variables that identify which columns to load by name. The name of a user variable used for this purpose must match the name of an element from the text file, prefixed with @. You can employ user variables to store the corresponding field values for subsequent reuse.

For example, the following statement loads the first column from the input file into the first column of `table1`, and sets the value of the `table_column2` column in `table1` to the input value of the second column divided by 100.

```
LOAD DATA FROM S3 's3://mybucket/data.txt'
    INTO TABLE table1
    (column1, @var1)
    SET table_column2 = @var1/100;
```

- **SET** – Specifies a comma-separated list of assignment operations that set the values of columns in the table to values not included in the input file.

  For example, the following statement sets the first two columns of `table1` to the values in the first two columns from the input file, and then sets the value of the `column3` in `table1` to the current time stamp.

```
LOAD DATA FROM S3  's3://mybucket/data.txt'
    INTO TABLE table1
    (column1, column2)
    SET column3 = CURRENT_TIMESTAMP;
```

  You can use subqueries in the right side of `SET` assignments. For a subquery that returns a value to be assigned to a column, you can use only a scalar subquery. Also, you cannot use a subquery to select from the table that is being loaded.

You cannot use the `LOCAL` keyword of the `LOAD DATA FROM S3` statement if you are loading data from an Amazon S3 bucket.

Using a Manifest to Specify Data Files to Load

You can use the `LOAD DATA FROM S3` statement with the `MANIFEST` keyword to specify a manifest file in JSON format that lists the text files to be loaded into a table in your DB cluster. You must be using Aurora 1.11 or greater to use the `MANIFEST` keyword with the `LOAD DATA FROM S3` statement.

The following JSON schema describes the format and content of a manifest file.

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "additionalProperties": false,
    "definitions": {},
    "id": "Aurora_LoadFromS3_Manifest",
    "properties": {
        "entries": {
            "additionalItems": false,
            "id": "/properties/entries",
            "items": {
                "additionalProperties": false,
                "id": "/properties/entries/items",
                "properties": {
                    "mandatory": {
                        "default": "false"
                        "id": "/properties/entries/items/properties/mandatory",
                        "type": "boolean"
                    },
                    "url": {
                        "id": "/properties/entries/items/properties/url",
                        "maxLength": 1024,
                        "minLength": 1,
                        "type": "string"
```

```
                    }
                },
                "required": [
                    "url"
                ],
                "type": "object"
            },
            "type": "array",
            "uniqueItems": true
        }
    },
    "required": [
        "entries"
    ],
    "type": "object"
}
```

Each `url` in the manifest must specify a URL with the bucket name and full object path for the file, not just a prefix. You can use a manifest to load files from different buckets, different regions, or files that do not share the same prefix. If a region is not specified in the URL, the region of the target Aurora DB cluster is used. The following example shows a manifest file that loads four files from different buckets.

```
{
  "entries": [
    {
      "url":"s3://aurora-bucket/2013-10-04-customerdata",
      "mandatory":true
    },
    {
      "url":"s3-us-west-2://aurora-bucket-usw2/2013-10-05-customerdata",
      "mandatory":true
    },
    {
      "url":"s3://aurora-bucket/2013-10-04-customerdata",
      "mandatory":false
    },
    {
      "url":"s3://aurora-bucket/2013-10-05-customerdata"
    }
  ]
}
```

The optional `mandatory` flag specifies whether `LOAD DATA FROM S3` should return an error if the file is not found. The `mandatory` flag defaults to `false`. Regardless of how `mandatory` is set, `LOAD DATA FROM S3` terminates if no files are found.

Manifest files can have any extension. The following example runs the `LOAD DATA FROM S3` statement with the manifest in the previous example, which is named **`customer.manifest`**.

```
LOAD DATA FROM S3 MANIFEST 's3-us-west-2://aurora-bucket/customer.manifest'
    INTO TABLE CUSTOMER
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    (ID, FIRSTNAME, LASTNAME, EMAIL);
```

After the statement completes, an entry for each successfully loaded file is written to the `aurora_s3_load_history` table.

Verifying Loaded Files Using the aurora_s3_load_history Table

Every successful `LOAD DATA FROM S3` statement updates the `aurora_s3_load_history` table in the `mysql` schema with an entry for each file that was loaded.

After you run the `LOAD DATA FROM S3` statement, you can verify which files were loaded by querying the `aurora_s3_load_history` table. To see the files that were loaded from one execution of the statement, use the `WHERE` clause to filter the records on the Amazon S3 URI for the manifest file used in the statement. If you have used the same manifest file before, filter the results using the `timestamp` field.

```
select * from mysql.aurora_s3_load_history where load_prefix = 'S3_URI';
```

The following table describes the fields in the `aurora_s3_load_history` table.

Field	Description
`load_prefix`	The URI that was specified in the load statement. This URI can map to a single data file for a `LOAD DATA FROM S3 FILE` statement, an Amazon S3 prefix that maps to multiple data files for a `LOAD DATA FROM S3 PREFIX` statement, or a single manifest file that contains the names of files to be loaded for a `LOAD DATA FROM S3 MANIFEST` statement.
`file_name`	The name of a file that was loaded into Aurora from Amazon S3 using the URI identified in the `load_prefix` field.
`version_number`	The version number of the file identified by the `file_name` field that was loaded, if the Amazon S3 bucket is versioned.
`bytes_loaded`	The size of the file loaded, in bytes.
`load_timestamp`	The timestamp when the `LOAD DATA FROM S3` statement completed.

Examples

The following statement loads data from an Amazon S3 bucket that is in the same region as the Aurora DB cluster. The statement reads the comma-delimited data in the file `customerdata.txt` that is in the `dbbucket` Amazon S3 bucket, and then loads the data into the table `store-schema.customer-table`.

```
LOAD DATA FROM S3 's3://dbbucket/customerdata.csv'
    INTO TABLE store-schema.customer-table
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    (ID, FIRSTNAME, LASTNAME, ADDRESS, EMAIL, PHONE);
```

The following statement loads data from an Amazon S3 bucket that is in a different region from the Aurora DB cluster. The statement reads the comma-delimited data from all files that match the `employee-data` object prefix in the `my-data` Amazon S3 bucket in the `us-west-2` region, and then loads the data into the `employees` table.

```
LOAD DATA FROM S3 PREFIX 's3-us-west-2://my-data/employee_data'
    INTO TABLE employees
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    (ID, FIRSTNAME, LASTNAME, EMAIL, SALARY);
```

The following statement loads data from the files specified in a JSON manifest file named q1_sales.json into the `sales` table.

```
LOAD DATA FROM S3 MANIFEST 's3-us-west-2://aurora-bucket/q1_sales.json'
    INTO TABLE sales
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    (MONTH, STORE, GROSS, NET);
```

LOAD XML FROM S3

You can use the `LOAD XML FROM S3` statement to load data from XML files stored on an Amazon S3 bucket in one of three different XML formats:

- Column names as attributes of a `<row>` element. The attribute value identifies the contents of the table field.

```
<row column1="value1" column2="value2" .../>
```

- Column names as child elements of a `<row>` element. The value of the child element identifies the contents of the table field.

```
<row>
  <column1>value1</column1>
  <column2>value2</column2>
</row>
```

- Column names in the `name` attribute of `<field>` elements in a `<row>` element. The value of the `<field>` element identifies the contents of the table field.

```
<row>
  <field name='column1'>value1</field>
  <field name='column2'>value2</field>
</row>
```

Syntax

```
LOAD XML FROM S3 'S3-URI'
    [REPLACE | IGNORE]
    INTO TABLE tbl_name
    [PARTITION (partition_name,...)]
    [CHARACTER SET charset_name]
    [ROWS IDENTIFIED BY '<element-name>']
    [IGNORE number {LINES | ROWS}]
    [(field_name_or_user_var,...)]
    [SET col_name = expr,...]
```

Parameters

Following, you can find a list of the required and optional parameters used by the `LOAD DATA FROM S3` statement. You can find more details about some of these parameters in LOAD XML Syntax in the MySQL documentation.

- **FILE | PREFIX** – Identifies whether to load the data from a single file, or from all files that match a given prefix. `FILE` is the default.
- **REPLACE | IGNORE** – Determines what action to take if an input row as the same unique key values as an existing row in the database table.
  - Specify `REPLACE` if you want the input row to replace the existing row in the table.
  - Specify `IGNORE` if you want to discard the input row. `IGNORE` is the default.

- **INTO TABLE** – Identifies the name of the database table to load the input rows into.
- **PARTITION** – Requires that all input rows be inserted into the partitions identified by the specified list of comma-separated partition names. If an input row cannot be inserted into one of the specified partitions, then the statement fails and an error is returned.
- **CHARACTER SET** – Identifies the character set of the data in the input file.
- **ROWS IDENTIFIED BY** – Identifies the element name that identifies a row in the input file. The default is `<row>`.
- **IGNORE *number* LINES | ROWS** – Specifies to ignore a certain number of lines or rows at the start of the input file. For example, you can use `IGNORE 1 LINES` to skip over the first line in the text file, or `IGNORE 2 ROWS` to skip over the first two rows of data in the input XML.
- **field_name_or_user_var, ...** – Specifies a comma-separated list of one or more XML element names or user variables that identify which elements to load by name. The name of a user variable used for this purpose must match the name of an element from the XML file, prefixed with @. You can employ user variables to store the corresponding field values for subsequent reuse.

  For example, the following statement loads the first column from the input file into the first column of `table1`, and sets the value of the `table_column2` column in `table1` to the input value of the second column divided by 100.

```
LOAD XML FROM S3 's3://mybucket/data.xml'
   INTO TABLE table1
   (column1, @var1)
   SET table_column2 = @var1/100;
```

- **SET** – Specifies a comma-separated list of assignment operations that set the values of columns in the table to values not included in the input file.

  For example, the following statement sets the first two columns of `table1` to the values in the first two columns from the input file, and then sets the value of the `column3` in `table1` to the current time stamp.

```
LOAD XML FROM S3 's3://mybucket/data.xml'
   INTO TABLE table1
   (column1, column2)
   SET column3 = CURRENT_TIMESTAMP;
```

  You can use subqueries in the right side of `SET` assignments. For a subquery that returns a value to be assigned to a column, you can use only a scalar subquery. Also, you cannot use a subquery to select from the table that is being loaded.

Related Topics

- Integrating Amazon Aurora MySQL with Other AWS Services (p. 550)
- Saving Data from an Amazon Aurora MySQL DB Cluster into Text Files in an Amazon S3 Bucket (p. 567)
- Amazon Aurora on Amazon RDS (p. 428)
- Migrating Data to an Amazon Aurora DB Cluster (p. 466)

Saving Data from an Amazon Aurora MySQL DB Cluster into Text Files in an Amazon S3 Bucket

You can use the `SELECT INTO OUTFILE S3` statement to query data from an Amazon Aurora MySQL DB cluster and save it directly into text files stored in an Amazon S3 bucket. You can use this

functionality to skip bringing the data down to the client first, and then copying it from the client to Amazon S3. The `LOAD DATA FROM S3` statement can use the files created by this statement to load data into an Aurora DB cluster. For more information, see Loading Data into an Amazon Aurora MySQL DB Cluster from Text Files in an Amazon S3 Bucket (p. 560).

Note
Saving data from a table into text files in an Amazon S3 bucket is available for Amazon Aurora MySQL version 1.13 and later. For more information about Aurora MySQL versions, see Amazon Aurora MySQL Database Engine Updates (p. 610).

Giving Aurora MySQL Access to Amazon S3

Before you can save data into an Amazon S3 bucket, you must first give your Aurora MySQL DB cluster permission to access Amazon S3. To grant permission, create an AWS Identity and Access Management (IAM) role with the necessary permissions, and then associate the role with your DB cluster. You must also configure your Aurora MySQL DB cluster to allow outbound connections to Amazon S3. For details and instructions on how to permit your Aurora MySQL DB cluster to communicate with Amazon S3 on your behalf, see Setting Up IAM Roles to Access AWS Services (p. 551).

Note
You must set either the `aurora_select_into_s3_role` or `aws_default_s3_role` DB cluster parameter to the Amazon Resource Name (ARN) of the new IAM role. If an IAM role isn't specified for the `aurora_select_into_s3_role`, the IAM role specified in `aws_default_s3_role` is used.
For more information about DB cluster parameters, see Amazon Aurora DB Cluster and DB Instance Parameters (p. 469).

Granting Privileges to Save Data in Aurora MySQL

The database user that issues the `SELECT INTO OUTFILE S3` statement must be granted the `SELECT INTO S3` privilege to issue the statement. The master username for a DB cluster is granted the `SELECT INTO S3` privilege by default. You can grant the privilege to another user by using the following statement.

```
GRANT SELECT INTO S3 ON *.* TO user@domain-or-ip-address
```

The `SELECT INTO S3` privilege is specific to Amazon Aurora MySQL and is not available for MySQL databases or RDS MySQL DB instances. If you have set up replication between an Aurora MySQL DB cluster as the replication master and a MySQL database as the replication client, then the `GRANT SELECT INTO S3` statement causes replication to stop with an error. You can safely skip the error to resume replication. To skip the error on an RDS MySQL DB instance, use the mysql.rds_skip_repl_error (p. 918) statement. To skip the error on an external MySQL database, use the SET GLOBAL sql_slave_skip_counter statement.

Specifying a Path to an Amazon S3 Bucket

The syntax for specifying a path to store the data and manifest files on an Amazon S3 bucket is similar to that used in the `LOAD DATA FROM S3 PREFIX` statement, as shown following.

```
s3-region://bucket-name/file-prefix
```

The path includes the following values:

- `region` (optional) – The AWS Region that contains the Amazon S3 bucket to save the data into. This value is optional. If you don't specify a `region` value, then Aurora saves your files into Amazon S3 in the same region as your DB cluster.
- `bucket-name` – The name of the Amazon S3 bucket to save the data into. Object prefixes that identify a virtual folder path are supported.

- `file-prefix` – The Amazon S3 object prefix that identifies the files to be saved in Amazon S3.

The data files created by the `SELECT INTO OUTFILE S3` statement use the following path, in which *`00000`* represents a 5-digit, zero-based integer number.

```
s3-region://bucket-name/file-prefix.part_00000
```

For example, suppose that a `SELECT INTO OUTFILE S3` statement specifies `s3-us-west-2://bucket/prefix` as the path in which to store data files and creates three data files. The specified Amazon S3 bucket contains the following data files.

- s3-us-west-2://bucket/prefix.part_00000
- s3-us-west-2://bucket/prefix.part_00001
- s3-us-west-2://bucket/prefix.part_00002

Creating a Manifest to List Data Files

You can use the `SELECT INTO OUTFILE S3` statement with the `MANIFEST ON` option to create a manifest file in JSON format that lists the text files created by the statement. The `LOAD DATA FROM S3` statement can use the manifest file to load the data files back into an Aurora MySQL DB cluster. For more information about using a manifest to load data files from Amazon S3 into an Aurora MySQL DB cluster, see Using a Manifest to Specify Data Files to Load (p. 563).

The data files included in the manifest created by the `SELECT INTO OUTFILE S3` statement are listed in the order that they're created by the statement. For example, suppose that a `SELECT INTO OUTFILE S3` statement specified `s3-us-west-2://bucket/prefix` as the path in which to store data files and creates three data files and a manifest file. The specified Amazon S3 bucket contains a manifest file named `s3-us-west-2://bucket/prefix.manifest`, that contains the following information.

```
{
  "entries": [
    {
      "url":"s3-us-west-2://bucket/prefix.part_00000"
    },
    {
      "url":"s3-us-west-2://bucket/prefix.part_00001"
    },
    {
      "url":"s3-us-west-2://bucket/prefix.part_00002"
    }
  ]
}
```

SELECT INTO OUTFILE S3

You can use the `SELECT INTO OUTFILE S3` statement to query data from a DB cluster and save it directly into delimited text files stored in an Amazon S3 bucket. Compressed or encrypted files are not supported.

Syntax

```
SELECT
    [ALL | DISTINCT | DISTINCTROW ]
        [HIGH_PRIORITY]
        [STRAIGHT_JOIN]
        [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
        [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
```

```
    select_expr [, select_expr ...]
    [FROM table_references
        [PARTITION partition_list]
    [WHERE where_condition]
    [GROUP BY {col_name | expr | position}
        [ASC | DESC], ... [WITH ROLLUP]]
    [HAVING where_condition]
    [ORDER BY {col_name | expr | position}
         [ASC | DESC], ...]
    [LIMIT {[offset,] row_count | row_count OFFSET offset}]
    [PROCEDURE procedure_name(argument_list)]
INTO OUTFILE S3 's3_uri'
[CHARACTER SET charset_name]
    [export_options]
    [MANIFEST {ON | OFF}]
    [OVERWRITE {ON | OFF}]

export_options:
    [{FIELDS | COLUMNS}
        [TERMINATED BY 'string']
        [[OPTIONALLY] ENCLOSED BY 'char']
        [ESCAPED BY 'char']
    ]
    [LINES
        [STARTING BY 'string']
        [TERMINATED BY 'string']
    ]
```

Parameters

Following, you can find a list of the required and optional parameters used by the `SELECT INTO OUTFILE S3` statement that are specific to Aurora.

- **s3-uri** – Specifies the URI for an Amazon S3 prefix to use. Specify the URI using the syntax described in Specifying a Path to an Amazon S3 Bucket (p. 568).
- **MANIFEST {ON | OFF}** – Indicates whether a manifest file is created in Amazon S3. The manifest file is a JavaScript Object Notation (JSON) file that can be used to load data into an Aurora DB cluster with the `LOAD DATA FROM S3 MANIFEST` statement. For more information about `LOAD DATA FROM S3 MANIFEST`, see Loading Data into an Amazon Aurora MySQL DB Cluster from Text Files in an Amazon S3 Bucket (p. 560).

  If `MANIFEST ON` is specified in the query, the manifest file is created in Amazon S3 after all data files have been created and uploaded. The manifest file is created using the following path:

  ```
  s3-region://bucket-name/file-prefix.manifest
  ```

  For more information about the format of the manifest file's contents, see Creating a Manifest to List Data Files (p. 569).
- **OVERWRITE {ON | OFF}** – Indicates whether existing files in the specified Amazon S3 bucket are overwritten. If `OVERWRITE ON` is specified, existing files that match the file prefix in the URI specified in `s3-uri`are overwritten. Otherwise, an error occurs.

You can find more details about other parameters in SELECT Syntax and LOAD DATA INFILE Syntax, in the MySQL documentation.

Considerations

The number of files written to the Amazon S3 bucket depends on the amount of data selected by the `SELECT INTO OUTFILE S3` statement and the file size threshold for Aurora MySQL. The default

file size threshold is 6 gigabytes (GB). If the data selected by the statement is less than the file size threshold, a single file is created; otherwise, multiple files are created. Other considerations for files created by this statement include the following:

- Aurora MySQL guarantees that rows in data files are not split across file boundaries. For multiple files, the size of every data file except the last is typically close to the file size threshold. However, occasionally staying under the file size threshold results in a row being split across two data files. In this case, Aurora MySQL creates a data file that keeps the row intact, but might be larger than the file size threshold.
- Because each SELECT statement in Aurora MySQL runs as an atomic transaction, a `SELECT INTO OUTFILE S3` statement that selects a large data set might run for some time. If the statement fails for any reason, you might need to start over and execute the statement again. If the statement fails, however, files already uploaded to Amazon S3 remain in the specified Amazon S3 bucket. You can use another statement to upload the remaining data instead of starting over again.
- If the amount of data to be selected is large (more than 25 GB), we recommend that you use multiple `SELECT INTO OUTFILE S3` statements to save the data to Amazon S3. Each statement should select a different portion of the data to be saved, and also specify a different `file_prefix` in the `s3-uri` parameter to use when saving the data files. Partitioning the data to be selected with multiple statements makes it easier to recover from execution error, because only a portion of data needs to be re-selected and uploaded to Amazon S3 if an error occurs during the execution of a particular statement. Using multiple statements also helps to avoid a single long-running transaction, which can improve performance.
- If multiple `SELECT INTO OUTFILE S3` statements that use the same `file_prefix` in the `s3-uri` parameter run in parallel to select data into Amazon S3, the behavior is undefined.
- Metadata, such as table schema or file metadata, is not uploaded by Aurora MySQL to Amazon S3.
- In some cases, you might re-run a `SELECT INTO OUTFILE S3` query, such as to recover from a failure. In these cases, you must either remove any existing data files in the Amazon S3 bucket with the same file prefix specified in `s3-uri`, or include `OVERWRITE ON` in the `SELECT INTO OUTFILE S3` query.

The `SELECT INTO OUTFILE S3` statement returns a typical MySQL error number and response on success or failure. If you don't have access to the MySQL error number and response, the easiest way to determine when it's done is by specifying `MANIFEST ON` in the statement. The manifest file is the last file written by the statement. In other words, if you have a manifest file, the statement has completed execution.

Currently, there's no way to directly monitor the progress of the `SELECT INTO OUTFILE S3` statement during execution. However, suppose that you're writing a large amount of data from Aurora MySQL to Amazon S3 using this statement, and you know the size of the data selected by the statement. In this case, you can estimate progress by monitoring the creation of data files in Amazon S3.

To do so, you can use the fact that a data file is created in the specified Amazon S3 bucket for about every 6GB of data selected by the statement. Divide the size of the data selected by 6GB to get the estimated number of data files to create. You can then estimate the progress of the statement by monitoring the number of files uploaded to Amazon S3 during execution.

Examples

The following statement selects all of the data in the `employees` table and saves the data into an Amazon S3 bucket that is in a different region from the Aurora MySQL DB cluster. The statement creates data files in which each field is terminated by a comma (`,`) character and each row is terminated by a newline (`\n`) character. The statement returns an error if files that match the `sample_employee_data` file prefix exist in the specified Amazon S3 bucket.

```
SELECT * FROM employees INTO OUTFILE S3 's3-us-west-2://aurora-select-into-s3-pdx/
sample_employee_data'
```

```
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n';
```

The following statement selects all of the data in the `employees` table and saves the data into an Amazon S3 bucket that is in the same region as the Aurora MySQL DB cluster. The statement creates data files in which each field is terminated by a comma (`,`) character and each row is terminated by a newline (`\n`) character, and also a manifest file. The statement returns an error if files that match the `sample_employee_data` file prefix exist in the specified Amazon S3 bucket.

```
SELECT * FROM employees INTO OUTFILE S3 's3://aurora-select-into-s3-pdx/
sample_employee_data'
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    MANIFEST ON;
```

The following statement selects all of the data in the `employees` table and saves the data into an Amazon S3 bucket that is in a different region from the Aurora DB cluster. The statement creates data files in which each field is terminated by a comma (`,`) character and each row is terminated by a newline (`\n`) character. The statement overwrites any existing files that match the `sample_employee_data` file prefix in the specified Amazon S3 bucket.

```
SELECT * FROM employees INTO OUTFILE S3 's3-us-west-2://aurora-select-into-s3-pdx/
sample_employee_data'
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    OVERWRITE ON;
```

The following statement selects all of the data in the `employees` table and saves the data into an Amazon S3 bucket that is in the same region as the Aurora MySQL DB cluster. The statement creates data files in which each field is terminated by a comma (`,`) character and each row is terminated by a newline (`\n`) character, and also a manifest file. The statement overwrites any existing files that match the `sample_employee_data` file prefix in the specified Amazon S3 bucket.

```
SELECT * FROM employees INTO OUTFILE S3 's3://aurora-select-into-s3-pdx/
sample_employee_data'
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    MANIFEST ON
    OVERWRITE ON;
```

Related Topics

- Integrating Aurora with Other AWS Services (p. 483)
- Loading Data into an Amazon Aurora MySQL DB Cluster from Text Files in an Amazon S3 Bucket (p. 560)
- Amazon Aurora on Amazon RDS (p. 428)
- Migrating Data to an Amazon Aurora DB Cluster (p. 466)

Invoking a Lambda Function from an Amazon Aurora MySQL DB Cluster

Note
Integration with AWS Lambda is available for Amazon Aurora MySQL version 1.8 and later. For more information about Aurora MySQL versions, see Amazon Aurora MySQL Database Engine Updates (p. 610).

You can invoke an AWS Lambda function from an Amazon Aurora MySQL DB cluster by calling the `mysql.lambda_async` procedure. This approach can be useful when you want to integrate your database running on Amazon Aurora MySQL with other AWS services. For example, you might want to send a notification using Amazon Simple Notification Service (Amazon SNS) whenever a row is inserted into a specific table in your database.

To invoke a Lambda function, you grant your Aurora MySQL DB cluster permission to access AWS Lambda. You grant permission by creating an IAM role with the necessary permissions, and then associating the role with your DB cluster. You must also configure your Aurora MySQL DB cluster to allow outbound connections to AWS Lambda. For details and instructions on how to permit your Aurora DB cluster to communicate with AWS Lambda on your behalf, see Setting Up IAM Roles to Access AWS Services (p. 551).

Working with the mysql.lambda_async Procedure to Invoke a Lambda Function

The `mysql.lambda_async` procedure is a built-in stored procedure that invokes a Lambda function asynchronously. To use this procedure, your database user must have execute privilege on the `mysql.lambda_async` stored procedure.

Syntax

```
CALL mysql.lambda_async (
  lambda_function_ARN,
  lambda_function_input
)
```

Parameters

lambda_function_ARN

The Amazon Resource Name (ARN) of the Lambda function to invoke.

lambda_function_input

The input string, in JSON format, for the invoked Lambda function.

Examples

As a best practice, we recommend that you wrap calls to the `mysql.lambda_async` procedure in a stored procedure that can be called from different sources such as triggers or client code. This approach can help to avoid impedance mismatch issues and make it easier to invoke Lambda functions.

Note
Be careful when invoking an AWS Lambda function from triggers on tables that experience high write traffic. INSERT, UPDATE and DELETE triggers are activated per row. A write-heavy workload on a table with INSERT, UPDATE, or DELETE triggers results in a large number of calls to your AWS Lambda function.
Although calls to the `mysql.lambda_async` procedure are asynchronous, triggers are synchronous. A statement that results in a large number of trigger activations doesn't wait for the call to the AWS Lambda function to complete, but it does wait for the triggers to complete before returning control to the client.

Example Invoke an AWS Lambda Function to Send Email

The following example creates a stored procedure that you can call in your database code to send an email using a Lambda function.

AWS Lambda Function

```
import boto3

ses = boto3.client('ses')

def SES_send_email(event, context):

    return ses.send_email(
        Source=event['email_from'],
        Destination={
            'ToAddresses': [
            event['email_to'],
            ]
        },

        Message={
            'Subject': {
            'Data': event['email_subject']
            },
            'Body': {
                'Text': {
                    'Data': event['email_body']
                }
            }
        }
    )
```

Stored Procedure

```
DROP PROCEDURE IF EXISTS SES_send_email;
DELIMITER ;;
 CREATE PROCEDURE SES_send_email(IN email_from VARCHAR(255),
                                 IN email_to VARCHAR(255),
                                 IN subject VARCHAR(255),
                                 IN body TEXT) LANGUAGE SQL
 BEGIN
  CALL mysql.lambda_async(
        'arn:aws:lambda:us-west-2:123456789012:function:SES_send_email',
        CONCAT('{"email_to" : "', email_to,
            '", "email_from" : "', email_from,
            '", "email_subject" : "', subject,
            '", "email_body" : "', body, '"}')
    );
 END
 ;;
DELIMITER ;
```

Call the Stored Procedure to Invoke the AWS Lambda Function

```
mysql> call SES_send_email('example_from@amazon.com', 'example_to@amazon.com', 'Email
 subject', 'Email content');
```

Example Invoke an AWS Lambda Function to Publish an Event from a Trigger

The following example creates a stored procedure that publishes an event by using Amazon SNS. The code calls the procedure from a trigger when a row is added to a table.

AWS Lambda Function

```
import boto3
```

```
sns = boto3.client('sns')

def SNS_publish_message(event, context):

    return sns.publish(
        TopicArn='arn:aws:sns:us-west-2:123456789012:Sample_Topic',
        Message=event['message'],
        Subject=event['subject'],
        MessageStructure='string'
    )
```

Stored Procedure

```
DROP PROCEDURE IF EXISTS SNS_Publish_Message;
DELIMITER ;;
CREATE PROCEDURE SNS_Publish_Message (IN subject VARCHAR(255),
                                      IN message TEXT) LANGUAGE SQL
BEGIN
  CALL mysql.lambda_async('arn:aws:lambda:us-
west-2:123456789012:function:SNS_publish_message',
     CONCAT('{ "subject" : "', subject,
            '", "message" : "', message, '" }')
     );
END
;;
DELIMITER ;
```

Table

```
CREATE TABLE `Customer_Feedback` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `customer_name` varchar(255) NOT NULL,
  `customer_feedback` varchar(1024) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Trigger

```
DELIMITER ;;
CREATE TRIGGER TR_Customer_Feedback_AI
  AFTER INSERT ON Customer_Feedback
  FOR EACH ROW
BEGIN
  SELECT CONCAT('New customer feedback from ', NEW.customer_name), NEW.customer_feedback
 INTO @subject, @feedback;
  CALL SNS_Publish_Message(@subject, @feedback);
END
;;
DELIMITER ;
```

Insert a Row into the Table to Trigger the Notification

```
mysql> insert into Customer_Feedback (customer_name, customer_feedback) VALUES ('Sample
 Customer', 'Good job guys!');
```

Related Topics

- Integrating Aurora with Other AWS Services (p. 483)

- Amazon Aurora on Amazon RDS (p. 428)

Exporting Audit Log Data From Amazon Aurora to Amazon CloudWatch Logs

You can configure an Amazon Aurora DB cluster to export audit log events to a log group in Amazon CloudWatch Logs. You can use this functionality to perform real-time analysis of the audit events observed on your DB cluster, using CloudWatch to create alarms and view metrics. You can use CloudWatch Logs to store your log data in highly durable storage. You can change the log retention setting so that any log events older than this setting are automatically deleted. The CloudWatch Logs agent makes it easy to quickly send both rotated and non-rotated log data off of a host and into the log service. You can then access the raw log data when you need it.

> **Note**
> Exporting audit log data from Aurora to CloudWatch Logs can incur data ingestion and archived storage charges on CloudWatch Logs. You are only charged for exporting audit log data that exceeds the free tier provided by CloudWatch Logs. For more information, see Amazon CloudWatch Pricing.

Giving Aurora Access to Amazon CloudWatch Logs

Before you can export audit logs to Amazon CloudWatch Logs, you must first give your Aurora DB cluster permission to access CloudWatch Logs. To grant permission, create an AWS AWS Identity and Access Management (IAM) role with the necessary permissions, and then associate the role with your DB cluster. For details and instructions on how to permit your Aurora DB cluster to communicate with CloudWatch Logs on your behalf, see Authorizing Amazon Aurora MySQL to Access Other AWS Services on Your Behalf (p. 551).

> **Note**
> You must set the `aws_default_logs_role` DB cluster parameter to the Amazon Resource Name (ARN) of the new IAM role. If an IAM role isn't specified for the `aws_default_logs_role` the logs will not be streamed to CloudWatch. If the specified role does not have the required permissions, we will report this situation through events. For more information about DB cluster parameters, see Amazon Aurora MySQL Parameters (p. 600).

Enabling Audit Log Exporting to Amazon CloudWatch Logs in an Aurora DB Cluster

To start exporting audit log to Amazon CloudWatch Logs, you need to first enable Advanced Auditing. For more information about enabling Advanced Auditing, see Enabling Advanced Auditing (p. 524).

After you enabled Advanced Auditing and defined the events that will be audited, you can enable audit log exporting to CloudWatch Logs logs by setting the following cluster-level DB parameter:

`server_audit_logs_upload`

Enables or disables audit log exporting. This parameter defaults to 0; set it to 1 to enable audit log exporting.

You can configure audit log exporting to CloudWatch Logs the same way that you configure Advanced Auditing, by setting these parameters in the parameter group used by your DB cluster. You can use the procedure shown in Modifying Parameters in a DB Parameter Group (p. 172) to modify DB cluster parameters using the AWS Management Console. You can use the `modify-db-cluster-parameter-group` AWS CLI command or the `ModifyDBClusterParameterGroup` Amazon RDS API command to modify DB cluster parameters programmatically. Modifying these parameters doesn't require a DB cluster restart.

Note
You can identify the DB instance for each event by using the `serverhost` field included in the audit log details. The full content of the event is exported to CloudWatch Logs. For more information about the `serverhost` field, see Audit Log Details (p. 526).
CloudWatch Logs events will persist after your DB instances are terminated. The log retention period for each log group can be configured using the CloudWatch Logs management console, the AWS CLI, or the CloudWatch Logs API.
For more information about CloudWatch Logs, see What is Amazon CloudWatch Logs?

Encryption Support

Audit log data is automatically encrypted in transit and at rest by CloudWatch Logs, using its default AWS Key Management Service (AWS KMS) encryption key. Currently, you cannot specify a custom AWS KMS key for encrypting audit log data in CloudWatch Logs.

Monitoring Audit Log Events in Amazon CloudWatch

After enabling the feature, you can monitor the audit log events in Amazon CloudWatch Logs. A new log group is automatically created for the Aurora DB cluster under the following prefix, in which *cluster-name* represents the DB cluster name:

```
/aws/rds/cluster/cluster-name
```

If a log group with the specified name exists, Aurora uses that log group to export audit log data for the Aurora DB cluster. You can use automated configuration, such as AWS CloudFormation, to create log groups with predefined log retention periods, metric filters, and customer access. Otherwise, a new log group is automatically created is created using the default log retention period, **Never Expire**, on CloudWatch Logs. You can use the CloudWatch Logs management console, the AWS CLI, or the CloudWatch Logs API to change the log retention period. For more information about changing log retention periods in CloudWatch Logs, see Change Log Data Retention in CloudWatch Logs.

All the audit events from all of the DB instances in a DB cluster are pushed to this log group using different log streams. You can use the CloudWatch Logs console, the AWS CLI, or the CloudWatch Logs API to search information within the log events for a DB cluster. For more information about searching and filtering log data, see Searching and Filtering Log Data.

Note
Aurora does not delete existing log groups or log streams if exporting audit log data is disabled. Existing audit log data remains available in CloudWatch Logs, depending on log retention, and you still incur charges for stored audit log data. You can delete log streams and log groups using the CloudWatch Logs management console, the AWS CLI, or the CloudWatch Logs API.

Using Amazon Aurora Auto Scaling with Aurora Replicas

Amazon Aurora with MySQL compatibility supports Aurora Auto Scaling. To meet your connectivity and workload requirements, Aurora Auto Scaling dynamically adjusts the number of Aurora Replicas provisioned for an Aurora DB cluster. Aurora Auto Scaling enables your Aurora DB cluster to handle sudden increases in connectivity or workload. When the connectivity or workload decreases, Aurora Auto Scaling removes unnecessary Aurora Replicas so that you don't pay for unused provisioned DB instances.

You define and apply a scaling policy to an Aurora DB cluster. The *scaling policy* defines the minimum and maximum number of Aurora Replicas that Aurora Auto Scaling can manage. Based on the policy, Aurora Auto Scaling adjusts the number of Aurora Replicas up or down in response to actual workloads, determined by using Amazon CloudWatch metrics and target values.

You can use the AWS Management Console to apply a scaling policy based on a predefined metric. Alternatively, you can use either the AWS CLI or Aurora Auto Scaling API to apply a scaling policy based on a predefined or custom metric.

Topics

Aurora Auto Scaling Policies

Aurora Auto Scaling uses a scaling policy to adjust the number of Aurora Replicas in an Aurora DB cluster. Aurora Auto Scaling has the following components:

- A service-linked role
- A target metric
- Minimum and maximum capacity
- A cooldown period

Service Linked Role

Aurora Auto Scaling uses the `AWSServiceRoleForApplicationAutoScaling_RDSCluster` service-linked role. For more information, see Service-Linked Roles for Application Auto Scaling in the *Application Auto Scaling API Reference*.

Target Metric

Aurora MySQL supports target-tracking scaling policies. In this type of policy, a predefined or custom metric and a target value for the metric is specified in a target-tracking scaling policy configuration. Aurora Auto Scaling creates and manages CloudWatch alarms that trigger the scaling policy and calculates the scaling adjustment based on the metric and target value. The scaling policy adds or removes Aurora Replicas as required to keep the metric at, or close to, the specified target value. In addition to keeping the metric close to the target value, a target-tracking scaling policy also adjusts to fluctuations in the metric due to a changing workload. Such a policy also minimizes rapid fluctuations in the number of available Aurora Replicas for your DB cluster.

For example, take a scaling policy that uses the predefined average CPU utilization metric. Such a policy can keep CPU utilization at, or close to, a specified percentage of utilization, such as 40 percent.

Aurora Auto Scaling only scales a DB cluster if all Aurora Replicas in a DB cluster are in the available state. If any of the Aurora Replicas are in a state other than available, Aurora Auto Scaling waits until the whole DB cluster becomes available for scaling. Also, Aurora Auto Scaling only removes Aurora Replicas that it created.

> **Note**
> For each Aurora DB cluster, you can create only one Auto Scaling policy for each target metric.

Minimum and Maximum Capacity

You can specify the maximum number of Aurora Replicas to be managed by Application Auto Scaling. This value must be set to 0–15, and must be equal to or greater than the value specified for the minimum number of Aurora Replicas.

You can also specify the minimum number of Aurora Replicas to be managed by Application Auto Scaling. This value must be set to 0–15, and must be equal to or less than the value specified for the maximum number of Aurora Replicas.

Note
The minimum and maximum capacity are set for an Aurora DB cluster. The specified values apply to all of the policies associated with that Aurora DB cluster.

Cooldown Period

You can tune the responsiveness of a target-tracking scaling policy by adding cooldown periods that affect scaling your Aurora DB cluster in and out. A cooldown period blocks subsequent scale-in or scale-out requests until the period expires. These blocks slow the deletions of Aurora Replicas in your Aurora DB cluster for scale-in requests, and the creation of Aurora Replicas for scale-out requests.

You can specify the following cooldown periods:

- A scale-in activity reduces the number of Aurora Replicas in your Aurora DB cluster. A scale-in cooldown period specifies the amount of time, in seconds, after a scale-in activity completes before another scale-in activity can start.
- A scale-out activity increases the number of Aurora Replicas in your Aurora DB cluster. A scale-out cooldown period specifies the amount of time, in seconds, after a scale-out activity completes before another scale-out activity can start.

When a scale-in or a scale-out cooldown period is not specified, the default for each is 300 seconds.

Enable or Disable Scale-In Activities

You can enable or disable scale-in activities for a policy. Enabling scale-in activities allows the scaling policy to delete Aurora Replicas. When scale-in activities are enabled, the scale-in cooldown period in the scaling policy applies to scale-in activities. Disabling scale-in activities prevents the scaling policy from deleting Aurora Replicas.

Note
Scale-out activities are always enabled so that the scaling policy can create Aurora Replicas as needed.

Before You Begin

Before you can use Aurora Auto Scaling with an Aurora DB cluster, you must first create an Aurora DB cluster with a primary instance and at least one Aurora Replica. Although Aurora Auto Scaling manages Aurora Replicas, the Aurora DB cluster must start with at least one Aurora Replica. For more information about creating an Aurora DB cluster, see Creating an Amazon Aurora DB Cluster (p. 437).

When Aurora Auto Scaling adds a new Aurora Replica, the new Aurora Replica is the same DB instance class as the one used by the primary instance. For more information about DB instance classes, see DB Instance Class (p. 92).

To benefit from Aurora Auto Scaling, your applications must support connections to new Aurora Replicas. To do so, we recommend using the Aurora reader endpoint or a driver such as the MariaDB Connector/J utility. For more information, see Connecting to an Amazon Aurora DB Cluster (p. 457).

Adding a Scaling Policy

You can add a scaling policy using the AWS Management Console, the AWS CLI, or the Application Auto Scaling API.

Topics

Adding a Scaling Policy Using the AWS Management Console

You can add a scaling policy to an Aurora DB cluster by using the AWS Management Console.

To add an Auto Scaling policy to an Aurora DB cluster

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Clusters**.
3. Choose the Aurora DB cluster that you want to add a policy for.
4. Choose **Cluster Actions**, and then choose **Add Auto Scaling policy**.

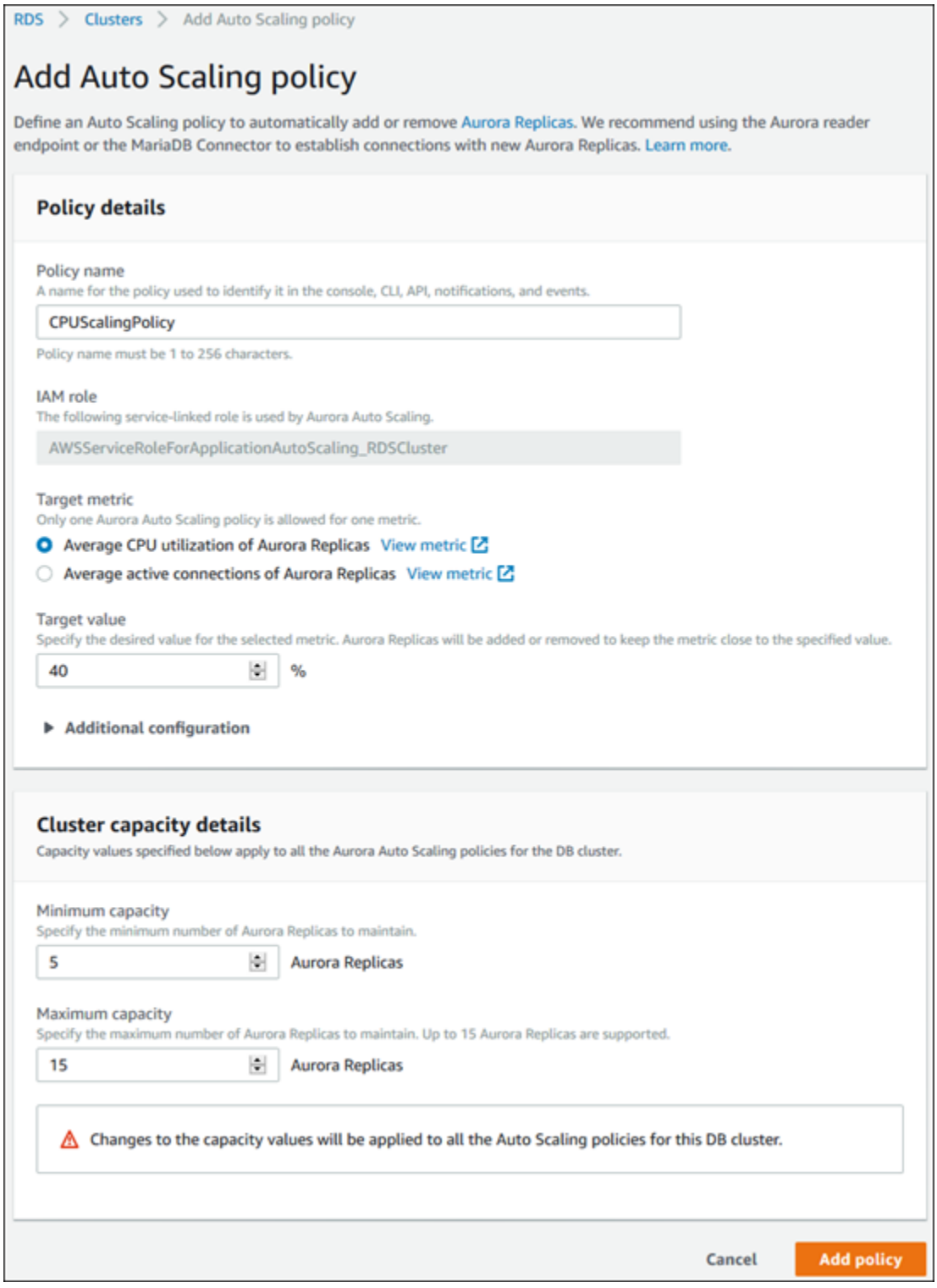
   The **Add Auto Scaling policy** dialog box appears.
5. Type the policy name in the **Policy Name** box.
6. For the target metric, choose one of the following:

   - **Average CPU utilization of Aurora Replicas** to create a policy based on the average CPU utilization.
   - **Average active connections of Aurora Replicas** to create a policy based on the average number of active connections to Aurora Replicas.
7. For the target value, type one of the following:

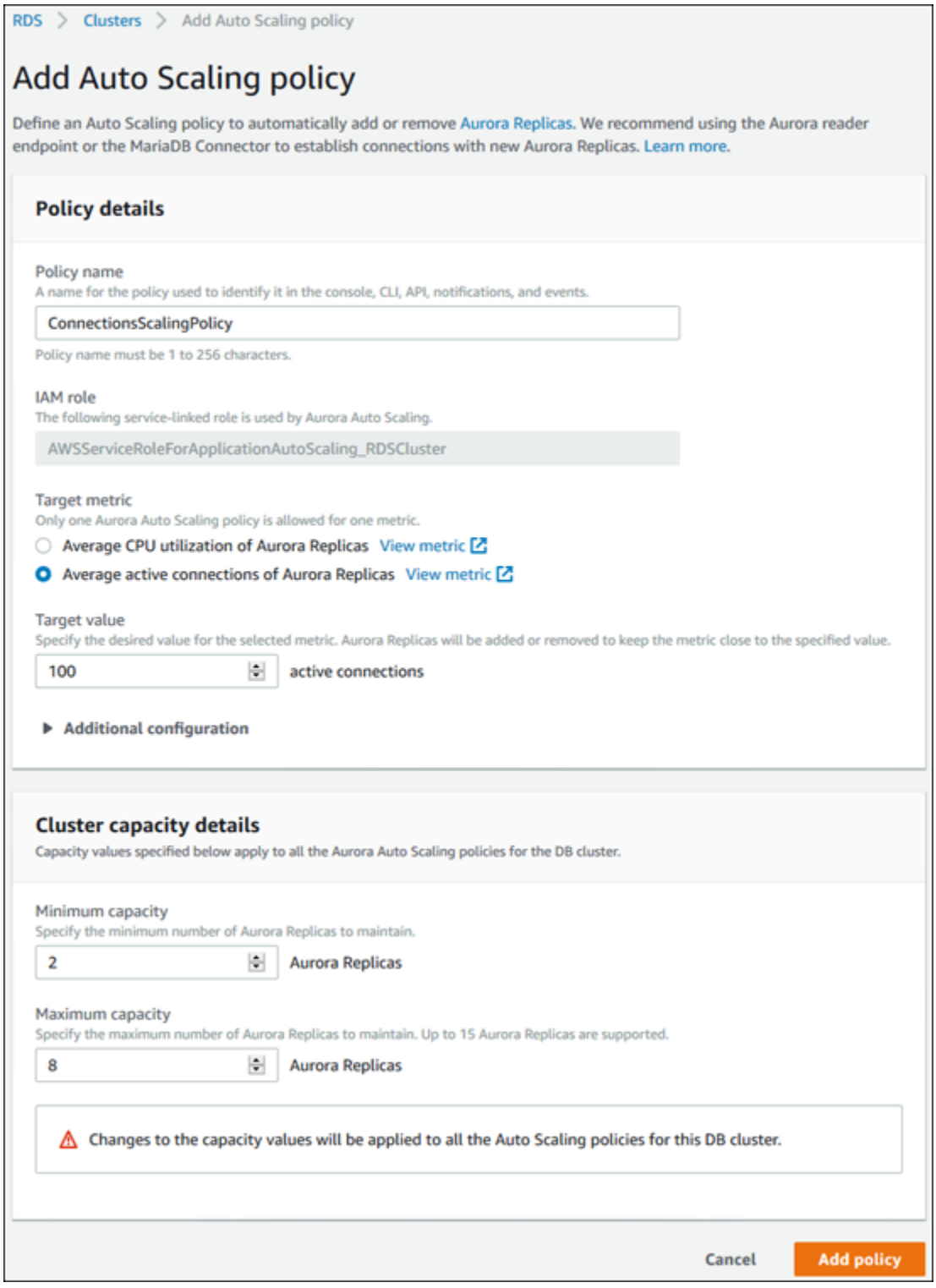
   - If you chose **Average CPU utilization of Aurora Replicas** in the previous step, type the percentage of CPU utilization that you want to maintain on Aurora Replicas.
   - If you chose **Average active connections of Aurora Replicas** in the previous step, type the number of active connections that you want to maintain.

   Aurora Replicas are added or removed to keep the metric close to the specified value.
8. (Optional) Open **Additional Configuration** to create a scale-in or scale-out cooldown period.
9. For **Minimum capacity**, type the minimum number of Aurora Replicas that the Aurora Auto Scaling policy is required to maintain.
10. For **Maximum capacity**, type the maximum number of Aurora Replicas the Aurora Auto Scaling policy is required to maintain.
11. Choose **Add policy**.

The following dialog box creates an Auto Scaling policy based an average CPU utilization of 40 percent. The policy specifies a minimum of 5 Aurora Replicas and a maximum of 15 Aurora Replicas.



The following dialog box creates an Auto Scaling policy based an average number of active connections of 100. The policy specifies a minimum of two Aurora Replicas and a maximum of eight Aurora Replicas.



Adding a Scaling Policy Using the AWS CLI or the Application Auto Scaling API

You can apply a scaling policy based on either a predefined or custom metric. To do so, you can use the AWS CLI or the Application Auto Scaling API. The first step is to register your Aurora DB cluster with Application Auto Scaling.

Registering an Aurora DB Cluster

Before you can use Aurora Auto Scaling with an Aurora DB cluster, you register your Aurora DB cluster with Application Auto Scaling. You do so to define the scaling dimension and limits to be applied to that cluster. Application Auto Scaling dynamically scales the Aurora DB cluster along the `rds:cluster:ReadReplicaCount` scalable dimension, which represents the number of Aurora Replicas.

To register your Aurora DB cluster, you can use either the AWS CLI or the Application Auto Scaling API.

CLI

To register your Aurora DB cluster, use the `register-scalable-target` AWS CLI command with the following parameters:

- `--service-namespace` – Set this value to `rds`.
- `--resource-id` – The resource identifier for the Aurora DB cluster. For this parameter, the resource type is `cluster` and the unique identifier is the name of the Aurora DB cluster, for example `cluster:myscalablecluster`.
- `--scalable-dimension` – Set this value to `rds:cluster:ReadReplicaCount`.
- `--min-capacity` – The minimum number of Aurora Replicas to be managed by Application Auto Scaling. This value must be set to 0–15, and must be equal to or less than the value specified for `max-capacity`.
- `--max-capacity` – The maximum number of Aurora Replicas to be managed by Application Auto Scaling. This value must be set to 0–15, and must be equal to or greater than the value specified for `min-capacity`.

Example

In the following example, you register an Aurora DB cluster named `myscalablecluster`. The registration indicates that the DB cluster should be dynamically scaled to have from one to eight Aurora Replicas.

For Linux, OS X, or Unix:

```
aws application-autoscaling register-scalable-target \
    --service-namespace rds \
    --resource-id cluster:myscalablecluster \
    --scalable-dimension rds:cluster:ReadReplicaCount \
    --min-capacity 1 \
    --max-capacity 8 \
```

For Windows:

```
aws application-autoscaling register-scalable-target ^
    --service-namespace rds ^
    --resource-id cluster:myscalablecluster ^
    --scalable-dimension rds:cluster:ReadReplicaCount ^
    --min-capacity 1 ^
    --max-capacity 8 ^
```

API

To register your Aurora DB cluster with Application Auto Scaling, use the `RegisterScalableTarget` Application Auto Scaling API action with the following parameters:

- `ServiceNamespace` – Set this value to `rds`.
- `ResourceID` – The resource identifier for the Aurora DB cluster. For this parameter, the resource type is `cluster` and the unique identifier is the name of the Aurora DB cluster, for example `cluster:myscalablecluster`.
- `ScalableDimension` – Set this value to `rds:cluster:ReadReplicaCount`.
- `MinCapacity` – The minimum number of Aurora Replicas to be managed by Application Auto Scaling. This value must be set to 0–15, and must be equal to or less than the value specified for `MaxCapacity`.
- `MaxCapacity` – The maximum number of Aurora Replicas to be managed by Application Auto Scaling. This value must be set to 0–15, and must be equal to or greater than the value specified for `MinCapacity`.

Example

In the following example, you register an Aurora DB cluster named `myscalablecluster` with the Application Auto Scaling API. This registration indicates that the DB cluster should be dynamically scaled to have from one to eight Aurora Replicas.

```
POST / HTTP/1.1
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.RegisterScalableTarget
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS

{
    "ServiceNamespace": "rds",
    "ResourceId": "cluster:myscalablecluster",
    "ScalableDimension": "rds:cluster:ReadReplicaCount",
    "MinCapacity": 1,
    "MaxCapacity": 8
}
```

Defining a Scaling Policy for an Aurora DB Cluster

A target-tracking scaling policy configuration is represented by a JSON block that the metrics and target values are defined in. You can save a scaling policy configuration as a JSON block in a text file. You use that text file when invoking the AWS CLI or the Application Auto Scaling API. For more information about policy configuration syntax, see `TargetTrackingScalingPolicyConfiguration` in the *Application Auto Scaling API Reference*.

The following options are available for defining a target-tracking scaling policy configuration.

Topics

Using a Predefined Metric

By using predefined metrics, you can quickly define a target-tracking scaling policy for an Aurora DB cluster that works well with both target tracking and dynamic scaling in Aurora Auto Scaling.

Currently, Aurora MySQL supports the following predefined metrics in Aurora Auto Scaling:

- **RDSReaderAverageCPUUtilization** – The average value of the `CPUUtilization` Aurora MySQL metric in CloudWatch across all Aurora Replicas in the Aurora DB cluster.
- **RDSReaderAverageDatabaseConnections** – The average value of the `DatabaseConnections` Aurora MySQL metric in CloudWatch across all Aurora Replicas in the Aurora DB cluster.

For more information about the `CPUUtilization` and `DatabaseConnections` metrics for Aurora MySQL, see Amazon Aurora MySQL Metrics (p. 470).

To use a predefined metric in your scaling policy, you create a target tracking configuration for your scaling policy. This configuration must include a `PredefinedMetricSpecification` for the predefined metric and a `TargetValue` for the target value of that metric.

Example

The following example describes a typical policy configuration for target-tracking scaling for an Aurora DB cluster. In this configuration, the `RDSReaderAverageCPUUtilization` predefined metric is used to adjust the Aurora DB cluster based on an average CPU utilization of 40 percent across all Aurora Replicas.

```
{
    "TargetValue": 40.0,
    "PredefinedMetricSpecification":
    {
        "PredefinedMetricType": "RDSReaderAverageCPUUtilization"
    }
}
```

Using a Custom Metric

By using custom metrics, you can define a target-tracking scaling policy that meets your custom requirements. You can define a custom metric based on any Aurora MySQL metric that changes in proportion to scaling.

Not all Aurora MySQL metrics work for target tracking. The metric must be a valid utilization metric and describe how busy an instance is. The value of the metric must increase or decrease in proportion to the number of Aurora Replicas in the Aurora DB cluster. This proportional increase or decrease is necessary to use the metric data to proportionally scale out or in the number of Aurora Replicas.

Example

The following example describes a target-tracking configuration for a scaling policy. In this configuration, a custom metric adjusts an Aurora DB cluster based on an average CPU utilization of 50 percent across all Aurora Replicas in an Aurora DB cluster named `my-db-cluster`.

```
{
    "TargetValue": 50,
    "CustomizedMetricSpecification":
    {
        "MetricName": "CPUUtilization",
        "Namespace": "AWS/RDS",
        "Dimensions": [
            {"Name": "DBClusterIdentifier","Value": "my-db-cluster"},
            {"Name": "Role","Value": "READER"}
        ],
        "Statistic": "Average",
        "Unit": "Percent"
    }
```

```
}
```

Using Cooldown Periods

You can specify a value, in seconds, for `ScaleOutCooldown` to add a cooldown period for scaling out your Aurora DB cluster. Similarly, you can add a value, in seconds, for `ScaleInCooldown` to add a cooldown period for scaling in your Aurora DB cluster. For more information about `ScaleInCooldown` and `ScaleOutCooldown`, see `TargetTrackingScalingPolicyConfiguration` in the *Application Auto Scaling API Reference*.

Example

The following example describes a target-tracking configuration for a scaling policy. In this configuration, the `RDSReaderAverageCPUUtilization` predefined metric is used to adjust an Aurora DB cluster based on an average CPU utilization of 40 percent across all Aurora Replicas in that Aurora DB cluster. The configuration provides a scale-in cooldown period of 10 minutes and a scale-out cooldown period of 5 minutes.

```
{
    "TargetValue": 40.0,
    "PredefinedMetricSpecification":
    {
        "PredefinedMetricType": "RDSReaderAverageCPUUtilization"
    },
    "ScaleInCooldown": 600,
    "ScaleOutCooldown": 300
}
```

Disabling Scale-in Activity

You can prevent the target-tracking scaling policy configuration from scaling in your Aurora DB cluster by disabling scale-in activity. Disabling scale-in activity prevents the scaling policy from deleting Aurora Replicas, while still allowing the scaling policy to create them as needed.

You can specify a Boolean value for `DisableScaleIn` to enable or disable scale in activity for your Aurora DB cluster. For more information about `DisableScaleIn`, see `TargetTrackingScalingPolicyConfiguration` in the *Application Auto Scaling API Reference*.

Example

The following example describes a target-tracking configuration for a scaling policy. In this configuration, the `RDSReaderAverageCPUUtilization` predefined metric adjusts an Aurora DB cluster based on an average CPU utilization of 40 percent across all Aurora Replicas in that Aurora DB cluster. The configuration disables scale-in activity for the scaling policy.

```
{
    "TargetValue": 40.0,
    "PredefinedMetricSpecification":
    {
        "PredefinedMetricType": "RDSReaderAverageCPUUtilization"
    },
    "DisableScaleIn": true
}
```

Applying a Scaling Policy to an Aurora DB Cluster

After registering your Aurora DB cluster with Application Auto Scaling and defining a scaling policy, you apply the scaling policy to the registered Aurora DB cluster. To apply a scaling policy to an Aurora DB cluster, you can use the AWS CLI or the Application Auto Scaling API.

CLI

To apply a scaling policy to your Aurora DB cluster, use the `put-scaling-policy` AWS CLI command with the following parameters:

- `--policy-name` – The name of the scaling policy.
- `--policy-type` – Set this value to `TargetTrackingScaling`.
- `--resource-id` – The resource identifier for the Aurora DB cluster. For this parameter, the resource type is `cluster` and the unique identifier is the name of the Aurora DB cluster, for example `cluster:myscalablecluster`.
- `--service-namespace` – Set this value to `rds`.
- `--scalable-dimension` – Set this value to `rds:cluster:ReadReplicaCount`.
- `--target-tracking-scaling-policy-configuration` – The target-tracking scaling policy configuration to use for the Aurora DB cluster.

Example

In the following example, you apply a target-tracking scaling policy named `myscalablepolicy` to an Aurora DB cluster named `myscalablecluster` with Application Auto Scaling. To do so, you use a policy configuration saved in a file named `config.json`.

For Linux, OS X, or Unix:

```
aws application-autoscaling put-scaling-policy \
    --policy-name myscalablepolicy \
    --policy-type TargetTrackingScaling \
    --resource-id cluster:myscalablecluster \
    --service-namespace rds \
    --scalable-dimension rds:cluster:ReadReplicaCount \
    --target-tracking-scaling-policy-configuration file://config.json
```

For Windows:

```
aws application-autoscaling put-scaling-policy ^
    --policy-name myscalablepolicy ^
    --policy-type TargetTrackingScaling ^
    --resource-id cluster:myscalablecluster ^
    --service-namespace rds ^
    --scalable-dimension rds:cluster:ReadReplicaCount ^
    --target-tracking-scaling-policy-configuration file://config.json
```

API

To apply a scaling policy to your Aurora DB cluster with the Application Auto Scaling API, use the `PutScalingPolicy` Application Auto Scaling API action with the following parameters:

- `PolicyName` – The name of the scaling policy.
- `ServiceNamespace` – Set this value to `rds`.
- `ResourceID` – The resource identifier for the Aurora DB cluster. For this parameter, the resource type is `cluster` and the unique identifier is the name of the Aurora DB cluster, for example `cluster:myscalablecluster`.
- `ScalableDimension` – Set this value to `rds:cluster:ReadReplicaCount`.

- `PolicyType` – Set this value to `TargetTrackingScaling`.
- `TargetTrackingScalingPolicyConfiguration` – The target-tracking scaling policy configuration to use for the Aurora DB cluster.

Example

In the following example, you apply a target-tracking scaling policy named `myscalablepolicy` to an Aurora DB cluster named `myscalablecluster` with Application Auto Scaling. You use a policy configuration based on the `RDSReaderAverageCPUUtilization` predefined metric.

```
POST / HTTP/1.1
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.PutScalingPolicy
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS

{
    "PolicyName": "myscalablepolicy",
    "ServiceNamespace": "rds",
    "ResourceId": "cluster:myscalablecluster",
    "ScalableDimension": "rds:cluster:ReadReplicaCount",
    "PolicyType": "TargetTrackingScaling",
    "TargetTrackingScalingPolicyConfiguration": {
        "TargetValue": 40.0,
        "PredefinedMetricSpecification":
        {
            "PredefinedMetricType": "RDSReaderAverageCPUUtilization"
        }
    }
}
```

Editing a Scaling Policy

You can edit a scaling policy using the AWS Management Console, the AWS CLI, or the Application Auto Scaling API.

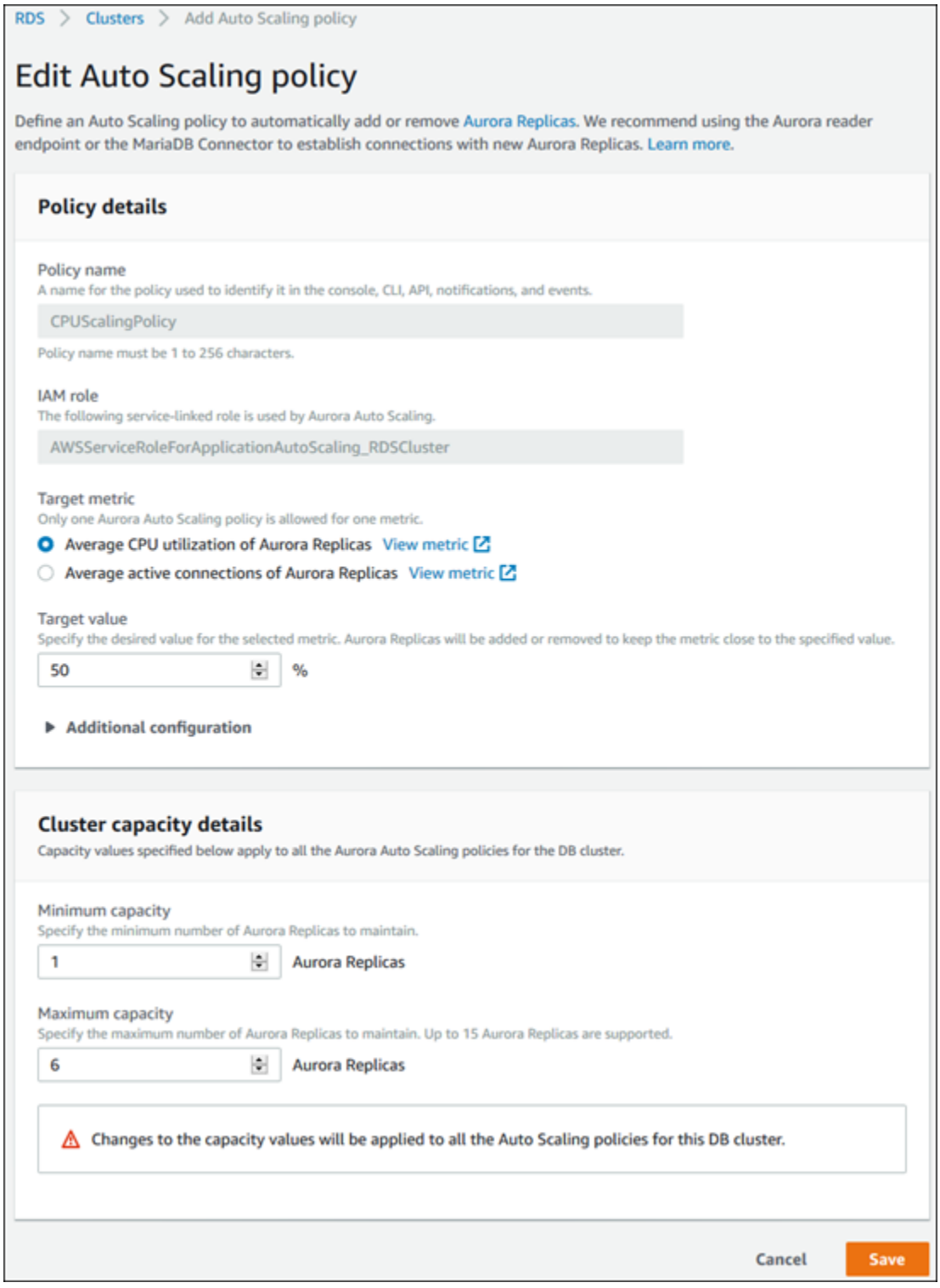
Editing a Scaling Policy Using the AWS Management Console

You can edit a scaling policy by using the AWS Management Console.

To edit an Auto Scaling policy for an Aurora DB cluster

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Clusters**.
3. Choose the Aurora DB cluster whose Auto Scaling policy you want to edit.
4. In the **Auto Scaling Policies** section, choose the Auto Scaling policy, and then choose **Edit**.
5. Make changes to the policy.
6. Choose **Save**.

The following is a sample **Edit Auto Scaling policy** dialog box.



Editing a Scaling Policy Using the AWS CLI or the Application Auto Scaling API

You can use the AWS CLI or the Application Auto Scaling API to edit a scaling policy in the same way that you apply a scaling policy:

- When using the AWS CLI, specify the name of the policy you want to edit in the `--policy-name` parameter. Specify new values for the parameters you want to change.
- When using the Application Auto Scaling API, specify the name of the policy you want to edit in the `PolicyName` parameter. Specify new values for the parameters you want to change.

For more information, see Applying a Scaling Policy to an Aurora DB Cluster (p. 586).

Deleting a Scaling Policy

You can delete a scaling policy using the AWS Management Console, the AWS CLI, or the Application Auto Scaling API.

Deleting a Scaling Policy Using the AWS Management Console

You can delete a scaling policy by using the AWS Management Console.

To delete an Auto Scaling policy for an Aurora DB cluster

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Clusters**.
3. Choose the Aurora DB cluster with the scaling policy that you want to delete.
4. In the **Auto Scaling Policies** section, choose the Auto Scaling policy, and then choose **Delete**.

Deleting a Scaling Policy Using the AWS CLI or the Application Auto Scaling API

You can use the AWS CLI or the Application Auto Scaling API to delete a scaling policy from an Aurora DB cluster.

CLI

To delete a scaling policy from your Aurora DB cluster, use the `delete-scaling-policy` AWS CLI command with the following parameters:

- `--policy-name` – The name of the scaling policy.
- `--resource-id` – The resource identifier for the Aurora DB cluster. For this parameter, the resource type is `cluster` and the unique identifier is the name of the Aurora DB cluster, for example `cluster:myscalablecluster`.
- `--service-namespace` – Set this value to `rds`.
- `--scalable-dimension` – Set this value to `rds:cluster:ReadReplicaCount`.

Example

In the following example, you delete a target-tracking scaling policy named `myscalablepolicy` from an Aurora DB cluster named `myscalablecluster`.

For Linux, OS X, or Unix:

```
aws application-autoscaling delete-scaling-policy \
    --policy-name myscalablepolicy \
    --resource-id cluster:myscalablecluster \
    --service-namespace rds \
    --scalable-dimension rds:cluster:ReadReplicaCount \
```

For Windows:

```
aws application-autoscaling delete-scaling-policy ^
    --policy-name myscalablepolicy ^
    --resource-id cluster:myscalablecluster ^
    --service-namespace rds ^
    --scalable-dimension rds:cluster:ReadReplicaCount ^
```

API

To delete a scaling policy from your Aurora DB cluster, use the `DeleteScalingPolicy` the Application Auto Scaling API action with the following parameters:

- `PolicyName` – The name of the scaling policy.
- `ServiceNamespace` – Set this value to `rds`.
- `ResourceID` – The resource identifier for the Aurora DB cluster. For this parameter, the resource type is `cluster` and the unique identifier is the name of the Aurora DB cluster, for example `cluster:myscalablecluster`.
- `ScalableDimension` – Set this value to `rds:cluster:ReadReplicaCount`.

Example

In the following example, you delete a target-tracking scaling policy named `myscalablepolicy` from an Aurora DB cluster named `myscalablecluster` with the Application Auto Scaling API.

```
POST / HTTP/1.1
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.DeleteScalingPolicy
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS

{
    "PolicyName": "myscalablepolicy",
    "ServiceNamespace": "rds",
    "ResourceId": "cluster:myscalablecluster",
    "ScalableDimension": "rds:cluster:ReadReplicaCount"
}
```

Related Topics

- Integrating Aurora with Other AWS Services (p. 483)
- Amazon Aurora on Amazon RDS (p. 428)

Related Topics

- Amazon Aurora on Amazon RDS (p. 428)

Best Practices with Amazon Aurora MySQL

This topic includes information on best practices and options for using or migrating data to an Amazon Aurora MySQL DB cluster.

Determining Which DB Instance You Are Connected To

You can determine which DB instance in an Aurora MySQL DB cluster that a connection is connected to by checking the `innodb_read_only` global variable, as shown in the following example.

```
SHOW GLOBAL VARIABLES LIKE 'innodb_read_only';
```

The `innodb_read_only` variable will be set to `ON` if you are connected to an Aurora Replica and `OFF` if you are connected to the primary instance.

This can be helpful if you want to add logic to your application code to balance the workload or to ensure that a write operation is using the correct connection.

Using T2 Instances

Amazon Aurora MySQL instances that use the `db.t2.small` or `db.t2.medium` DB instance classes are best suited for applications that do not support a high workload for an extended amount of time. T2 instances are designed to provide moderate baseline performance and the capability to burst to significantly higher performance as required by your workload. They are intended for workloads that don't use the full CPU often or consistently, but occasionally need to burst. The `db.t2.small` and `db.t2.medium` DB instance classes are best used for development and test servers, or other non-production servers. For more details on T2 instances, see T2 Instances.

The Performance Schema should not be enabled on Amazon Aurora MySQL T2 instances. If the Performance Schema is enabled, the T2 instance might run out of memory.

When you use a T2 instance for the primary instance or Aurora Replicas in a Aurora MySQL DB cluster, we recommend the following:

- If you use a T2 instance as a DB instance class in your DB cluster, then we recommend that all instances in your DB cluster use the same DB instance class. For example, if you use `db.t2.medium` for your primary instance, then we recommend that you use `db.t2.medium` for your Aurora Replicas as well.
- Monitor your CPU Credit Balance (`CPUCreditBalance`) to ensure that it is at a sustainable level. That is, CPU credits are being accumulated at the same rate as they are being used.

  When you have exhausted the CPU credits for an instance, you will see an immediate drop in the available CPU and an increase in the read and write latency for the instance. This results in a severe decrease in the overall performance of the instance.

  If your CPU credit balance is not at a sustainable level, then we recommend that you modify your DB instance to use a one of the supported R3 DB instance classes (scale compute).

  For more information on monitoring metrics, see Monitoring an Amazon Aurora DB Cluster (p. 470).
- Monitor the replica lag (`AuroraReplicaLag`) between the primary instance and the Aurora Replicas in your Aurora MySQL DB cluster.

  If an Aurora Replica runs out of CPU credits before the primary instance, the lag behind the primary instance will result in the Aurora Replica frequently restarting. This is common for scenarios where an application maintains a heavy load of read operations distributed between the Aurora Replicas in a Aurora MySQL DB cluster, at the same time that the primary instance maintains a minimal load of write operations.

If you see a sustained increase in replica lag, make sure that your CPU credit balance for the Aurora Replicas in your DB cluster is not being exhausted.

If your CPU credit balance is not at a sustainable level, then we recommend that you modify your DB instance to use a one of the supported R3 DB instance classes (scale compute).

- Keep the number of inserts per transaction below 1 million for DB clusters that have binary logging enabled.

  If the DB cluster parameter group for your DB cluster has the `binlog_format` parameter set to a value other than `OFF`, then your DB cluster may experience out-of-memory conditions if the DB cluster receives large transactions that contain over 1 million rows to insert. You can monitor the freeable memory (`FreeableMemory`) metric to determine if your DB cluster is running out of available memory, and then check the write operations (`VolumeWriteIOPS`) metric to see if your primary instance is receiving a heavy load of writer operations. If this is the case, then we recommend that you update your application to limit the amount of inserts in a transaction to less than 1 million or modify your instance to use one of the supported R3 DB instance classes (scale compute).

Invoking an AWS Lambda Function

We recommend that you wrap calls to the `mysql.lambda_async` procedure in a stored procedure that can be called from different sources such as triggers or client code. This can help to avoid impedance mismatch issues and make it easier for your database programmers to invoke Lambda functions.

For more information on invoking Lambda functions from Amazon Aurora, see Invoking a Lambda Function from an Amazon Aurora MySQL DB Cluster (p. 572).

The following example shows a Lambda function, a stored procedure that invokes the Lambda function, and a call to run the stored procedure and invoke the Lambda function.

Lambda Function

```
import boto3

ses = boto3.client('ses')

def SES_send_email(event, context):

    return ses.send_email(
        Source=event['email_from'],
        Destination={
            'ToAddresses': [
            event['email_to'],
            ]
        },

        Message={
            'Subject': {
            'Data': event['email_subject']
            },
            'Body': {
                'Text': {
                    'Data': event['email_body']
                }
            }
        }
    )
```

Stored Procedure

```
DROP PROCEDURE IF EXISTS SES_send_email;
DELIMITER ;;
 CREATE PROCEDURE SES_send_email(IN email_from VARCHAR(255),
                                 IN email_to VARCHAR(255),
                                 IN subject VARCHAR(255),
                                 IN body TEXT) LANGUAGE SQL
 BEGIN
  CALL mysql.lambda_async(
        'arn:aws:lambda:us-west-2:123456789012:function:SES_send_email',
        CONCAT('{"email_to" : "', email_to,
            '", "email_from" : "', email_from,
            '", "email_subject" : "', subject,
            '", "email_body" : "', body, '"}')
    );
 END
 ;;
DELIMITER ;
```

Call the Stored Procedure to Invoke the Lambda Function

```
mysql> call SES_send_email('example_to@amazon.com', 'example_from@amazon.com', 'Email
 subject', 'Email content');
```

Working with Asynchronous Key Prefetch in Amazon Aurora

> **Note**
> The asynchronous key prefetch (AKP) feature is available for Amazon Aurora MySQL version 1.15 and later. For more information about Aurora MySQL versions, see Amazon Aurora MySQL Database Engine Updates (p. 610).

Amazon Aurora can use AKP to improve the performance of queries that join tables across indexes. This feature improves performance by anticipating the rows needed to run queries in which a JOIN query requires use of the Batched Key Access (BKA) Join algorithm and Multi-Range Read (MRR) optimization features. For more information about BKA and MRR, see Block Nested-Loop and Batched Key Access Joins and Multi-Range Read Optimization in the MySQL documentation.

To take advantage of the AKP feature, a query must use both BKA and MRR. Typically, such a query occurs when the JOIN clause of a query uses a secondary index, but also needs some columns from the primary index. For example, you can use AKP when a JOIN clause represents an equi-join on index values between a small outer and large inner table, and the index is highly selective on the larger table. AKP works in concert with BKA and MRR to perform a secondary to primary index lookup during the evaluation of the JOIN clause. AKP identifies the rows required to run the query during the evaluation of the JOIN clause. It then uses a background thread to asynchronously load the pages containing those rows into memory before running the query.

Enabling Asynchronous Key Prefetch

You can enable the AKP feature by setting the **aurora_use_key_prefetch** MySQL server variable to ON. By default, this value is set to ON. However, AKP cannot be enabled until you also enable the BKA Join algorithm and disable cost-based MRR functionality. To do so, you must set the following values for the **optimizer_switch** MySQL server variable:

- Set **batched_key_access** to ON.

  This value controls the use of the BKA Join algorithm. By default, this value is set to OFF.
- Set **mrr_cost_based** to OFF.

  This value controls the use of cost-based MRR functionality. By default, this value is set to ON.

Currently, you can set these values only at the session level. The following example illustrates how to set these values to enable AKP for the current session by executing SET statements.

```
mysql> set @@session.aurora_use_key_prefetch=on;
mysql> set @@session.optimizer_switch='batched_key_access=on,mrr_cost_based=off';
```

Similarly, you can use SET statements to disable AKP and the BKA Join algorithm and re-enable cost-based MRR functionality for the current session, as shown in the following example.

```
mysql> set @@session.aurora_use_key_prefetch=off;
mysql> set @@session.optimizer_switch='batched_key_access=off,mrr_cost_based=on';
```

For more information about the **batched_key_access** and **mrr_cost_based** optimizer switches, see Switchable Optimizations in the MySQL documentation.

Optimizing Queries for Asynchronous Key Prefetch

You can confirm whether a query can take advantage of the AKP feature. To do so, use the EXPLAIN statement with the EXTENDED keyword to profile the query before running it. The EXPLAIN statement provides information about the execution plan to use for a specified query.

In the output for the EXPLAIN statement, the `Extra` column describes additional information included with the execution plan. If the AKP feature applies to a table used in the query, this column includes one of the following values:

- `Using Key Prefetching`
- `Using join buffer (Batched Key Access with Key Prefetching)`

The following example shows use of EXPLAIN with EXTENDED to view the execution plan for a query that can take advantage of AKP.

```
mysql> explain extended select sql_no_cache
    ->     ps_partkey,
    ->     sum(ps_supplycost * ps_availqty) as value
    -> from
    ->     partsupp,
    ->     supplier,
    ->     nation
    -> where
    ->     ps_suppkey = s_suppkey
    ->     and s_nationkey = n_nationkey
    ->     and n_name = 'ETHIOPIA'
    -> group by
    ->     ps_partkey having
    ->         sum(ps_supplycost * ps_availqty) > (
    ->             select
    ->                 sum(ps_supplycost * ps_availqty) * 0.0000003333
    ->             from
    ->                 partsupp,
    ->                 supplier,
    ->                 nation
    ->             where
    ->                 ps_suppkey = s_suppkey
    ->                 and s_nationkey = n_nationkey
    ->                 and n_name = 'ETHIOPIA'
    ->         )
    -> order by
    ->     value desc;
```

```
+----+-------------+----------+------+----------------------+---------------
+---------+-----------------------------------+------+----------
+-------------------------------------------------------------+
| id | select_type | table    | type | possible_keys        | key           | key_len |
 ref                               | rows | filtered | Extra
                    |
+----+-------------+----------+------+----------------------+---------------
+---------+-----------------------------------+------+----------
+-------------------------------------------------------------+
|  1 | PRIMARY     | nation   | ALL  | PRIMARY              | NULL          | NULL    |
 NULL                              |   25 |   100.00 | Using where; Using temporary; Using
 filesort                |
|  1 | PRIMARY     | supplier | ref  | PRIMARY,i_s_nationkey | i_s_nationkey | 5       |
 dbt3_scale_10.nation.n_nationkey | 2057 |   100.00 | Using index
                    |
|  1 | PRIMARY     | partsupp | ref  | i_ps_suppkey         | i_ps_suppkey  | 4       |
 dbt3_scale_10.supplier.s_suppkey |   42 |   100.00 | Using join buffer (Batched Key Access
 with Key Prefetching) |
|  2 | SUBQUERY    | nation   | ALL  | PRIMARY              | NULL          | NULL    |
 NULL                              |   25 |   100.00 | Using where
                    |
|  2 | SUBQUERY    | supplier | ref  | PRIMARY,i_s_nationkey | i_s_nationkey | 5       |
 dbt3_scale_10.nation.n_nationkey | 2057 |   100.00 | Using index
                    |
|  2 | SUBQUERY    | partsupp | ref  | i_ps_suppkey         | i_ps_suppkey  | 4       |
 dbt3_scale_10.supplier.s_suppkey |   42 |   100.00 | Using join buffer (Batched Key Access
 with Key Prefetching) |
+----+-------------+----------+------+----------------------+---------------
+---------+-----------------------------------+------+----------
+-------------------------------------------------------------+
6 rows in set, 1 warning (0.00 sec)
```

For more information about the extended `EXPLAIN` output format, see Extended EXPLAIN Output Format in the MySQL product documentation.

Working with Multi-Threaded Replication Slaves in Amazon Aurora MySQL

By default, Aurora uses single-threaded replication when an Aurora MySQL DB cluster is used as a replication slave. While Amazon Aurora does not prohibit multi-threaded replication, Aurora MySQL has inherited several issues regarding multi-threaded replication from MySQL. We recommend that you do not use multi-threaded replication in production. If you do use multi-threaded replication, we recommend that you test any use thoroughly.

For more information about using replication in Amazon Aurora, see Replication with Amazon Aurora (p. 478).

Using Amazon Aurora to Scale Reads for Your MySQL Database

You can use Amazon Aurora with your MySQL DB instance to take advantage of the read scaling capabilities of Amazon Aurora and expand the read workload for your MySQL DB instance. To use Aurora to read scale your MySQL DB instance, create an Amazon Aurora MySQL DB cluster and make it a replication slave of your MySQL DB instance. This applies to an Amazon RDS MySQL DB instance, or a MySQL database running external to Amazon RDS.

For information on creating an Amazon Aurora DB cluster, see Creating an Amazon Aurora DB Cluster (p. 437).

When you set up replication between your MySQL DB instance and your Amazon Aurora DB cluster, be sure to follow these guidelines:

- Use the Amazon Aurora DB cluster endpoint address when you reference your Amazon Aurora MySQL DB cluster. If a failover occurs, then the Aurora Replica that is promoted to the primary instance for the Aurora MySQL DB cluster will continue to use the DB cluster endpoint address.
- Maintain the binlogs on your master instance until you have verified that they have been applied to the Aurora Replica. This maintenance ensures that you can restore your master instance in the event of a failure.

Important
When using self-managed replication, you're responsible for monitoring and resolving any replication issues that may occur. For more information, see Diagnosing and Resolving Lag Between Read Replicas (p. 1233).

Note
The permissions required to start replication on an Amazon Aurora MySQL DB cluster are restricted and not available to your Amazon RDS master user. Because of this, you must use the Amazon RDS mysql.rds_set_external_master (p. 914) and mysql.rds_start_replication (p. 917) commands to set up replication between your Amazon Aurora MySQL DB cluster and your MySQL DB instance.

Start Replication Between an External Master Instance and a MySQL DB Instance on Amazon RDS

1. Make the source MySQL DB instance read-only:

```
mysql> FLUSH TABLES WITH READ LOCK;
mysql> SET GLOBAL read_only = ON;
```

2. Run the `SHOW MASTER STATUS` command on the source MySQL DB instance to determine the binlog location. You will receive output similar to the following example:

```
File                          Position
-----------------------------------
 mysql-bin-changelog.000031      107
-----------------------------------
```

3. Copy the database from the external MySQL DB instance to the Amazon Aurora MySQL DB cluster using `mysqldump`. For very large databases, you might want to use the procedure in Importing Data to an Amazon RDS MySQL or MariaDB DB Instance with Reduced Downtime (p. 873).

   For Linux, OS X, or Unix:

```
mysqldump \
    --databases <database_name> \
    --single-transaction \
    --compress \
    --order-by-primary \
    -u <local_user> \
    -p <local_password> | mysql \
        --host aurora_cluster_endpoint_address \
        --port 3306 \
        -u <RDS_user_name> \
        -p <RDS_password>
```

   For Windows:

```
mysqldump ^
    --databases <database_name> ^
    --single-transaction ^
```

```
    --compress ^
    --order-by-primary ^
    -u <local_user> ^
    -p <local_password> | mysql ^
        --host aurora_cluster_endpoint_address ^
        --port 3306 ^
        -u <RDS_user_name> ^
        -p <RDS_password>
```

Note
Make sure there is not a space between the `-p` option and the entered password.

Use the `##host`, `##user (-u)`, `##port` and `-p` options in the `mysql` command to specify the hostname, user name, port, and password to connect to your Aurora DB cluster. The host name is the DNS name from the Amazon Aurora DB cluster endpoint, for example, `mydbcluster.cluster-123456789012.us-east-1.rds.amazonaws.com`. You can find the endpoint value in the cluster details in the Amazon RDS Management Console.

4. Make the source MySQL DB instance writeable again:

```
mysql> SET GLOBAL read_only = OFF;
mysql> UNLOCK TABLES;
```

For more information on making backups for use with replication, see Backing Up a Master or Slave by Making It Read Only in the MySQL documentation.

5. In the Amazon RDS Management Console, add the IP address of the server that hosts the source MySQL database to the VPC security group for the Amazon Aurora DB cluster. For more information on modifying a VPC security group, see Security Groups for Your VPC in the *Amazon Virtual Private Cloud User Guide*.

   You might also need to configure your local network to permit connections from the IP address of your Amazon Aurora DB cluster, so that it can communicate with your source MySQL instance. To find the IP address of the Amazon Aurora DB cluster, use the `host` command:

```
host <aurora_endpoint_address>
```

   The host name is the DNS name from the Amazon Aurora DB cluster endpoint.

6. Using the client of your choice, connect to the external MySQL instance and create a MySQL user that will be used for replication. This account is used solely for replication and must be restricted to your domain to improve security. The following is an example:

```
CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY '<password>';
```

7. For the external MySQL instance, grant `REPLICATION CLIENT` and `REPLICATION SLAVE` privileges to your replication user. For example, to grant the `REPLICATION CLIENT` and `REPLICATION SLAVE` privileges on all databases for the '`repl_user`' user for your domain, issue the following command:

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com'
    IDENTIFIED BY '<password>';
```

8. Take a manual snapshot of the Aurora MySQL DB cluster that will be the replication slave prior to setting up replication. If you need to reestablish replication with the DB cluster as a replication slave, you can restore the Aurora MySQL DB cluster from this snapshot instead of having to import the data from your MySQL DB instance into a new Aurora MySQL DB cluster.
9. Make the Amazon Aurora DB cluster the replica. Connect to the Amazon Aurora DB cluster as the master user and identify the source MySQL database as the replication master by using the

mysql.rds_set_external_master (p. 914) command. Use the master log file name and master log position that you determined in Step 2. The following is an example:

```
CALL mysql.rds_set_external_master ('mymasterserver.mydomain.com', 3306,
    'repl_user', '<password>', 'mysql-bin-changelog.000031', 107, 0);
```

10. On the Amazon Aurora DB cluster, issue the mysql.rds_start_replication (p. 917) command to start replication:

```
CALL mysql.rds_start_replication;
```

After you have established replication between your source MySQL DB instance and your Amazon Aurora DB cluster, you can add Aurora Replicas to your Amazon Aurora DB cluster. You can then connect to the Aurora Replicas to read scale your data. For information on creating an Aurora Replica, see Creating an Aurora Replica Using the Console (p. 447).

Using Amazon Aurora for Disaster Recovery with Your MySQL Databases

You can use Amazon Aurora with your MySQL DB instance to create an off-site backup for disaster recovery. To use Aurora for disaster recovery of your MySQL DB instance, create an Amazon Aurora DB cluster and make it a replication slave of your MySQL DB instance. This applies to an Amazon RDS MySQL DB instance, or a MySQL database running external to Amazon RDS.

> **Important**
> When you set up replication between a MySQL DB instance and an Amazon Aurora MySQL DB cluster, the replication is not managed by Amazon RDS. You must monitor the replication to ensure that it remains healthy and repair it if necessary.

For instructions on how to create an Amazon Aurora MySQL DB cluster and make it a replication slave of your MySQL DB instance, follow the procedure in Using Amazon Aurora to Scale Reads for Your MySQL Database (p. 596).

Migrating from MySQL to Amazon Aurora MySQL with Reduced Downtime

When importing data from a MySQL database that supports a live application to an Amazon Aurora MySQL DB cluster, you can use the procedure documented in Importing Data to an Amazon RDS MySQL or MariaDB DB Instance with Reduced Downtime (p. 873) to reduce the amount of time that service to your data is interrupted in order to migrate your data to Aurora MySQL. The procedure can especially help if you are working with a very large database, because you can reduce the cost of the import by minimizing the amount of data that is passed across the network to AWS.

The procedure lists steps to transfer a copy of your database data to an Amazon EC2 instance and import the data into a new Amazon RDS MySQL DB instance. Because Amazon Aurora is compatible with MySQL, you can instead use an Amazon Aurora DB cluster for the target Amazon RDS MySQL DB instance.

Related Topics

- Amazon Aurora on Amazon RDS (p. 428)

Amazon Aurora MySQL Reference

Amazon Aurora MySQL Parameters

You manage your Amazon Aurora MySQL DB cluster in the same way that you manage other Amazon RDS DB instances, by using parameters in a DB parameter group. Amazon Aurora differs from other DB engines in that you have a DB cluster that contains multiple DB instances. As a result, some of the parameters that you use to manage your Aurora MySQL DB cluster apply to the entire cluster, while other parameters apply only to a particular DB instance in the DB cluster.

Cluster-level parameters are managed in DB cluster parameter groups. Instance-level parameters are managed in DB parameter groups. Although each DB instance in an Aurora MySQL DB cluster is compatible with the MySQL database engine, some of the MySQL database engine parameters must be applied at the cluster level, and are managed using DB cluster parameter groups. Cluster-level parameters are not found in the DB parameter group for an instance in an Aurora DB cluster and are listed later in this topic.

You can manage both cluster-level and instance-level parameters using the AWS Management Console, the AWS CLI, or the Amazon RDS API. There are separate commands for managing cluster-level parameters and instance-level parameters. For example, you can use the modify-db-cluster-parameter-group AWS CLI command to manage cluster-level parameters in a DB cluster parameter group and use the modify-db-parameter-group AWS CLI command to manage instance-level parameters in a DB parameter group for a DB instance in a DB cluster.

You can view both cluster-level and instance-level parameters in the AWS Management Console, or by using the AWS CLI or Amazon RDS API. For example, you can use the describe-db-cluster-parameters AWS CLI command to view cluster-level parameters in a DB cluster parameter group and use the describe-db-parameters AWS CLI command to view instance-level parameters in a DB parameter group for a DB instance in a DB cluster.

For more information on DB parameter groups, see Working with DB Parameter Groups (p. 170).

Cluster-level Parameters

The following table shows all of the parameters that apply to the entire Aurora MySQL DB cluster.

Parameter name	Modifiable
aurora_load_from_s3_role	Yes
aurora_select_into_s3_role	Yes
auto_increment_increment	Yes
auto_increment_offset	Yes
aws_default_lambda_role	Yes
aws_default_s3_role	Yes
binlog_checksum	Yes
binlog_format	Yes
binlog_row_image	No
binlog_rows_query_log_events	No
character-set-client-handshake	Yes

Parameter name	Modifiable
character_set_client	Yes
character_set_connection	Yes
character_set_database	Yes
character_set_filesystem	Yes
character_set_results	Yes
collation_connection	Yes
collation_server	Yes
completion_type	Yes
default_storage_engine	No
innodb_autoinc_lock_mode	Yes
innodb_checksum_algorithm	No
innodb_checksums	No
innodb_cmp_per_index_enabled	Yes
innodb_commit_concurrency	Yes
innodb_data_home_dir	No
innodb_doublewrite	No
innodb_file_per_table	Yes
innodb_flush_log_at_trx_commit	Yes
innodb_ft_max_token_size	Yes
innodb_ft_min_token_size	Yes
innodb_ft_num_word_optimize	Yes
innodb_ft_sort_pll_degree	Yes
innodb_online_alter_log_max_size	Yes
innodb_optimize_fulltext_only	Yes
innodb_page_size	No
innodb_purge_batch_size	Yes
innodb_purge_threads	Yes
innodb_rollback_on_timeout	Yes
innodb_rollback_segments	Yes
innodb_spin_wait_delay	Yes
innodb_strict_mode	Yes

Parameter name	Modifiable
innodb_support_xa	Yes
innodb_sync_array_size	Yes
innodb_sync_spin_loops	Yes
innodb_table_locks	Yes
innodb_undo_directory	No
innodb_undo_logs	Yes
innodb_undo_tablespaces	No
lc_time_names	Yes
lower_case_table_names	Yes
master-info-repository	Yes
master_verify_checksum	Yes
server_audit_events	Yes
server_audit_excl_users	Yes
server_audit_incl_users	Yes
server_audit_logging	Yes
server_id	No
skip-character-set-client-handshake	Yes
skip_name_resolve	No
sync_frm	Yes
time_zone	Yes

Instance-level Parameters

The following table shows all of the parameters that apply to a specific DB instance in an Aurora MySQL DB cluster.

Parameter name	Modifiable
allow-suspicious-udfs	No
aurora_lab_mode	Yes
autocommit	Yes
automatic_sp_privileges	Yes
back_log	Yes
basedir	No

Parameter name	Modifiable
binlog_cache_size	Yes
binlog_max_flush_queue_time	Yes
binlog_order_commits	Yes
binlog_stmt_cache_size	Yes
bulk_insert_buffer_size	Yes
concurrent_insert	Yes
connect_timeout	Yes
core-file	No
datadir	No
default_time_zone	No
default_tmp_storage_engine	Yes
default_week_format	Yes
delay_key_write	Yes
delayed_insert_limit	Yes
delayed_insert_timeout	Yes
delayed_queue_size	Yes
div_precision_increment	Yes
end_markers_in_json	Yes
enforce_gtid_consistency	No
eq_range_index_dive_limit	Yes
event_scheduler	Yes
explicit_defaults_for_timestamp	Yes
flush	No
flush_time	Yes
ft_boolean_syntax	No
ft_max_word_len	Yes
ft_min_word_len	Yes
ft_query_expansion_limit	Yes
ft_stopword_file	Yes
general_log	Yes
general_log_file	No

Parameter name	Modifiable
group_concat_max_len	Yes
gtid-mode	No
host_cache_size	Yes
init_connect	Yes
innodb_adaptive_flushing	Yes
innodb_adaptive_hash_index	Yes
innodb_adaptive_max_sleep_delay	Yes
innodb_autoextend_increment	Yes
innodb_buffer_pool_dump_at_shutdown	No
innodb_buffer_pool_dump_now	No
innodb_buffer_pool_filename	No
innodb_buffer_pool_load_abort	No
innodb_buffer_pool_load_at_startup	No
innodb_buffer_pool_load_now	No
innodb_buffer_pool_size	Yes
innodb_change_buffer_max_size	No
innodb_compression_failure_threshold_pct	Yes
innodb_compression_level	Yes
innodb_compression_pad_pct_max	Yes
innodb_concurrency_tickets	Yes
innodb_file_format	Yes
innodb_flush_log_at_timeout	No
innodb_flush_method	Yes
innodb_flush_neighbors	No
innodb_flushing_avg_loops	No
innodb_force_load_corrupted	No
innodb_ft_aux_table	Yes
innodb_ft_cache_size	Yes
innodb_ft_enable_stopword	Yes
innodb_ft_server_stopword_table	Yes
innodb_ft_user_stopword_table	Yes

Parameter name	Modifiable
innodb_io_capacity	No
innodb_io_capacity_max	No
innodb_large_prefix	Yes
innodb_lock_wait_timeout	Yes
innodb_log_compressed_pages	No
innodb_lru_scan_depth	Yes
innodb_max_dirty_pages_pct	Yes
innodb_max_purge_lag	Yes
innodb_max_purge_lag_delay	Yes
innodb_monitor_disable	Yes
innodb_monitor_enable	Yes
innodb_monitor_reset	Yes
innodb_monitor_reset_all	Yes
innodb_old_blocks_pct	Yes
innodb_old_blocks_time	Yes
innodb_open_files	Yes
innodb_print_all_deadlocks	Yes
innodb_random_read_ahead	Yes
innodb_read_ahead_threshold	Yes
innodb_read_io_threads	No
innodb_read_only	No
innodb_replication_delay	Yes
innodb_sort_buffer_size	Yes
innodb_stats_auto_recalc	Yes
innodb_stats_method	Yes
innodb_stats_on_metadata	Yes
innodb_stats_persistent	Yes
innodb_stats_persistent_sample_pages	Yes
innodb_stats_transient_sample_pages	Yes
innodb_thread_concurrency	No
innodb_thread_sleep_delay	Yes

Parameter name	Modifiable
innodb_use_native_aio	No
innodb_write_io_threads	No
interactive_timeout	Yes
join_buffer_size	Yes
keep_files_on_create	Yes
key_buffer_size	Yes
key_cache_age_threshold	Yes
key_cache_block_size	Yes
key_cache_division_limit	Yes
local_infile	Yes
lock_wait_timeout	Yes
log-bin	No
log_bin_trust_function_creators	Yes
log_bin_use_v1_row_events	Yes
log_error	No
log_output	Yes
log_queries_not_using_indexes	Yes
log_slave_updates	No
log_throttle_queries_not_using_indexes	Yes
log_warnings	Yes
long_query_time	Yes
low_priority_updates	Yes
max_allowed_packet	Yes
max_binlog_cache_size	Yes
max_binlog_size	No
max_binlog_stmt_cache_size	Yes
max_connect_errors	Yes
max_connections	Yes
max_delayed_threads	Yes
max_error_count	Yes
max_heap_table_size	Yes

Parameter name	Modifiable
max_insert_delayed_threads	Yes
max_join_size	Yes
max_length_for_sort_data	Yes
max_prepared_stmt_count	Yes
max_seeks_for_key	Yes
max_sort_length	Yes
max_sp_recursion_depth	Yes
max_tmp_tables	Yes
max_user_connections	Yes
max_write_lock_count	Yes
metadata_locks_cache_size	Yes
min_examined_row_limit	Yes
myisam_data_pointer_size	Yes
myisam_max_sort_file_size	Yes
myisam_mmap_size	Yes
myisam_sort_buffer_size	Yes
myisam_stats_method	Yes
myisam_use_mmap	Yes
net_buffer_length	Yes
net_read_timeout	Yes
net_retry_count	Yes
net_write_timeout	Yes
old-style-user-limits	Yes
old_passwords	Yes
optimizer_prune_level	Yes
optimizer_search_depth	Yes
optimizer_switch	Yes
optimizer_trace	Yes
optimizer_trace_features	Yes
optimizer_trace_limit	Yes
optimizer_trace_max_mem_size	Yes

Parameter name	Modifiable
optimizer_trace_offset	Yes
performance_schema	Yes
pid_file	No
plugin_dir	No
port	No
preload_buffer_size	Yes
profiling_history_size	Yes
query_alloc_block_size	Yes
query_cache_limit	Yes
query_cache_min_res_unit	Yes
query_cache_size	Yes
query_cache_type	Yes
query_cache_wlock_invalidate	Yes
query_prealloc_size	Yes
range_alloc_block_size	Yes
read_buffer_size	Yes
read_only	No
read_rnd_buffer_size	Yes
relay-log	No
relay_log_info_repository	Yes
relay_log_recovery	No
safe-user-create	Yes
secure_auth	Yes
secure_file_priv	No
skip-slave-start	No
skip_external_locking	No
skip_show_database	Yes
slave_checkpoint_group	Yes
slave_checkpoint_period	Yes
slave_parallel_workers	Yes
slave_pending_jobs_size_max	Yes

Parameter name	Modifiable
slave_sql_verify_checksum	Yes
slow_launch_time	Yes
slow_query_log	Yes
slow_query_log_file	No
socket	No
sort_buffer_size	Yes
sql_mode	Yes
sql_select_limit	Yes
stored_program_cache	Yes
sync_binlog	No
sync_master_info	Yes
sync_relay_log	Yes
sync_relay_log_info	Yes
sysdate-is-now	Yes
table_cache_element_entry_ttl	No
table_definition_cache	Yes
table_open_cache	Yes
table_open_cache_instances	Yes
temp-pool	Yes
thread_cache_size	Yes
thread_handling	No
thread_stack	Yes
timed_mutexes	Yes
tmp_table_size	Yes
tmpdir	No
transaction_alloc_block_size	Yes
transaction_prealloc_size	Yes
tx_isolation	Yes
updatable_views_with_limit	Yes
validate-password	No
validate_password_dictionary_file	No

Parameter name	Modifiable
validate_password_length	No
validate_password_mixed_case_count	No
validate_password_number_count	No
validate_password_policy	No
validate_password_special_char_count	No
wait_timeout	Yes

Related Topics

- Amazon Aurora on Amazon RDS (p. 428)

Amazon Aurora MySQL Database Engine Updates

Amazon Aurora releases updates regularly. Updates are applied to Aurora DB clusters during system maintenance windows. The timing when updates are applied depends on the region and maintenance window setting for the DB cluster, as well as the type of update. Updates require a database restart, so you will experience 20 to 30 seconds of downtime, after which you can resume using your DB cluster or clusters. You can view or change your maintenance window settings from the AWS Management Console.

Amazon Aurora Versions

Although Amazon Aurora is compatible with the MySQL and PostgreSQL database engines, Aurora includes features that are specific to Amazon Aurora and only available to Aurora DB clusters. Aurora versions use the format <major version>.<minor version>.<patch version>. You can get the version of your Aurora instance by querying for the `AURORA_VERSION` system variable. To get the Amazon Aurora version, use one of the following queries.

Database Engine	Queries
MySQL	`select AURORA_VERSION();`
MySQL	`select @@aurora_version;`
PostgreSQL	`SELECT AURORA_VERSION();`

Amazon Aurora Database Upgrades (Patching)

When a new minor version of the Amazon Aurora MySQL database engine is released, Amazon RDS schedules an automatic upgrade of the database engine for all Aurora DB clusters. We announce automatic upgrades in the Amazon RDS Community Forum.

When a new patch version of the Aurora MySQL database engine is released, no automatic upgrade is required. You can choose to upgrade and apply the patch, otherwise the patch will be applied during the next automatic upgrade for a minor version release.

Before automatic upgrade, new database engine releases show as an **available** maintenance upgrade for your DB cluster. You can manually upgrade the database version for your DB cluster by applying the available maintenance action. We encourage you to apply the update on a non-production instance prior to the automatic upgrade, so that you can see how changes in the new version will affect your instances and applications.

To apply pending maintenance actions

- **By using the Amazon RDS Console** – Log on to the Amazon RDS console and choose **Clusters**. Choose the DB cluster that shows an **available** maintenance upgrade. Choose **Cluster Actions**. Choose **Upgrade Now** to immediately update the database version for your DB cluster, or **Upgrade at Next Window** to update the database version for your DB cluster during the next cluster maintenance window.
- **By using the AWS CLI** – Call the apply-pending-maintenance-action AWS CLI command and specify the Amazon Resource Name (ARN) for your DB cluster for the `--resource-id` option and `system-update` for the `--apply-action` option. Set the `--opt-in-type` option to `immediate` to immediately update the database version for your DB cluster, or `next-maintenance` to update the database version for your DB cluster during the next cluster maintenance window.
- **By using the Amazon RDS API** – Call the ApplyPendingMaintenanceAction API action and specify the ARN for your DB cluster for the `ResourceId` parameter and `system-update` for the `ApplyAction` parameter. Set the `OptInType` parameter to `immediate` to immediately update the database version for your DB cluster, or `next-maintenance` to update the database version for your instance during the next cluster maintenance window.

For more information on how Amazon RDS manages database and operating system updates, see DB Instance and DB Cluster Maintenance (p. 102).

Aurora Lab Mode

Aurora lab mode is used to enable Aurora features that are available in the current Aurora database version, but are not enabled by default. While Aurora lab mode features are not recommended for use in production DB clusters, you can use Aurora lab mode to enable these features for DB clusters in your development and test environments. For more information about Aurora features available when Aurora lab mode is enabled, see Aurora Lab Mode Features (p. 612).

To enable Aurora lab mode, set the `aurora_lab_mode` parameter to `1` in the parameter group for your primary instance or Aurora Replica. The `aurora_lab_mode` parameter is an instance-level parameter that is in the `default.aurora5.6` parameter group by default. For information on modifying a DB parameter group, see Modifying Parameters in a DB Parameter Group (p. 172). For information on parameter groups and Amazon Aurora, see Amazon Aurora MySQL Parameters (p. 600).

Related Topics

- Amazon Aurora MySQL Database Engine Updates 2017-11-20 (p. 612) (Version 1.15.1)
- Amazon Aurora MySQL Database Engine Updates 2017-10-24 (p. 613) (Version 1.15)
- Amazon Aurora MySQL Database Engine Updates: 2017-09-22 (p. 615) (Version 1.14.1)
- Amazon Aurora MySQL Database Engine Updates: 2017-08-07 (p. 616) (Version 1.14)
- Amazon Aurora MySQL Database Engine Updates: 2017-05-15 (p. 617) (Version 1.13)
- Amazon Aurora MySQL Database Engine Updates: 2017-04-05 (p. 619) (Version 1.12)
- Amazon Aurora MySQL Database Engine Updates: 2017-02-23 (p. 620) (Version 1.11)
- Amazon Aurora MySQL Database Engine Updates: 2017-01-12 (p. 622) (Version 1.10.1)
- Amazon Aurora MySQL Database Engine Updates: 2016-12-14 (p. 623) (Version 1.10)
- Amazon Aurora MySQL Database Engine Updates: 2016-11-10 (p. 624) (Versions 1.9.0, 1.9.1)
- Amazon Aurora MySQL Database Engine Updates: 2016-10-26 (p. 625) (Version 1.8.1)

- Amazon Aurora MySQL Database Engine Updates: 2016-10-18 (p. 625) (Version 1.8)
- Amazon Aurora MySQL Database Engine Updates: 2016-09-20 (p. 627) (Version 1.7.1)
- Amazon Aurora MySQL Database Engine Updates: 2016-08-30 (p. 627) (Version 1.7)
- Amazon Aurora MySQL Database Engine Updates: 2016-06-01 (p. 628) (Version 1.6.5)
- Amazon Aurora MySQL Database Engine Updates: 2016-04-06 (p. 628) (Version 1.6)
- Amazon Aurora MySQL Database Engine Updates: 2016-01-11 (p. 630) (Version 1.5)
- Amazon Aurora MySQL Database Engine Updates: 2015-12-03 (p. 630) (Version 1.4)
- Amazon Aurora MySQL Database Engine Updates: 2015-10-16 (p. 632) (Versions 1.2, 1.3)
- Amazon Aurora MySQL Database Engine Updates: 2015-08-24 (p. 634) (Version 1.1)
- MySQL Bugs Fixed by Amazon Aurora MySQL Database Engine Updates (p. 634)

Aurora Lab Mode Features

The following table lists the Aurora features currently available when Aurora lab mode is enabled. You must enable Aurora lab mode before any of these features can be used. For more information about Aurora lab mode, see Aurora Lab Mode (p. 611).

Feature	Description
Fast DDL	This feature allows you to execute an ALTER TABLE *tbl_name* ADD COLUMN *col_name column_definition* operation nearly instantaneously. The operation completes without requiring the table to be copied and without materially impacting other DML statements. Since it does not consume temporary storage for a table copy, it makes DDL statements practical even for large tables on small instance types. Fast DDL is currently only supported for adding a nullable column, without a default value, at the end of a table. For more information about using this feature, see Altering Tables in Amazon Aurora Using Fast DDL (p. 522).
Lock Compression	This feature significantly reduces the amount of memory that lock manager consumes by up to 66%. Lock manager can acquire more row locks without encountering an out-of-memory exception.

Amazon Aurora MySQL Database Engine Updates 2017-11-20

Version: 1.15.1

Amazon Aurora v1.15.1 is generally available. All new database clusters, including those restored from snapshots, will be created in Aurora v1.15.1. You have the option, but are not required, to upgrade existing DB clusters to Aurora v1.15.1. If you wish to create new DB clusters in Aurora v1.14.1, you can do so using the AWS CLI or the Amazon RDS API and specifying the engine version.

With version 1.15.1 of Aurora, we are using a cluster patching model where all nodes in an Aurora DB cluster are patched at the same time. We are enabling zero-downtime patching, which works on a best-effort basis to preserve client connections through the patching process. For more information, see Amazon RDS Maintenance (p. 102).

Should you have any questions or concerns, the AWS Support Team is available on the community forums and through AWS Premium Support at http://aws.amazon.com/support. For more information, see Amazon RDS Maintenance (p. 102).

Zero-Downtime Patching

The zero-downtime patching (ZDP) attempts, on a best-effort basis, to preserve client connections through an engine patch. If ZDP executes successfully, application sessions are preserved and the database engine restarts while patching. The database engine restart can cause a transient (5 second or so) drop in throughput.

ZDP will not execute successfully under the following conditions:

- Long-running queries or transactions are in progress
- Binary logging is enabled or binary log replication is in-progress
- Open SSL connections exist
- Temporary tables or table locks are in use
- Pending parameter changes exist

If no suitable time window for executing ZDP becomes available because of one or more of these conditions, patching reverts to the standard behavior.

> **Note**
> ZDP applies only to the primary instance of a DB cluster. ZDP is not applicable to Aurora Replicas.

Improvements

- Fixed an issue in the adaptive segment selector for a read request that would cause it to choose the same segment twice causing a spike in read latency under certain conditions.
- Fixed an issue that stems from an optimization in Aurora MySQL for the thread scheduler. This problem manifests itself into what are spurious errors while writing to the slow log, while the associated queries themselves perform fine.
- Fixed an issue with stability of read replicas on large (> 5 TB) volumes.
- Fixed an issue where worker thread count increases continuously due to a bogus outstanding connection count.
- Fixed an issue with table locks that caused long semaphore waits during insert workloads.
- Reverted MySQL bug fixes for Full Text Indexes included in release 15.

Integration of MySQL Bug Fixes

None

Amazon Aurora MySQL Database Engine Updates 2017-10-24

Version: 1.15

Amazon Aurora v1.15 is generally available. All new database clusters, including those restored from snapshots, will be created in Aurora v1.15. You have the option, but are not required, to upgrade existing DB clusters to Aurora v1.15. If you wish to create new DB clusters in Aurora v1.14.1, you can do so using the AWS CLI or the Amazon RDS API and specifying the engine version.

With version 1.15 of Aurora, we are using a cluster patching model where all nodes in an Aurora DB cluster are patched at the same time. Updates require a database restart, so you will experience 20 to 30 seconds of downtime, after which you can resume using your DB cluster or clusters. If your DB clusters

are currently running Aurora v1.14 or Aurora v1.14.1, Aurora's zero-downtime patching feature may allow client connections to your Aurora primary instance to persist through the upgrade, depending on your workload.

Should you have any questions or concerns, the AWS Support Team is available on the community forums and through AWS Premium Support at http://aws.amazon.com/support. For more information, see Amazon RDS Maintenance (p. 102).

Zero-Downtime Patching

The zero-downtime patching (ZDP) attempts, on a best-effort basis, to preserve client connections through an engine patch. If ZDP executes successfully, application sessions are preserved and the database engine restarts while patching. The database engine restart can cause a transient (5 second or so) drop in throughput.

ZDP will not execute successfully under the following conditions:

- Long-running queries are in progress
- Open long-running transactions exist
- Binary logging is enabled
- Binary log replication is running
- Pending parameter changes exist
- Temporary tables are in use
- Table locks are in use
- Open SSL connections exist

If no suitable time window for executing ZDP becomes available because of one or more of these conditions, patching reverts to the standard behavior.

> **Note**
> ZDP applies only to the primary instance of a DB cluster. ZDP is not applicable to Aurora Replicas.

New Features

- **Asynchronous Key Prefetch** – Asynchronous key prefetch (AKP) is a feature targeted to improve the performance of non-cached index joins, by prefetching keys in memory ahead of when they are needed. The primary use case targeted by AKP is an index join between a small outer and large inner table, where the index is highly selective on the larger table. Also, when the Multi-Range Read (MRR) interface is enabled, AKP will be leveraged for a secondary to primary index lookup. Smaller instances which have memory constraints might in some cases be able to leverage AKP, given the right key cardinality. For more information, see Working with Asynchronous Key Prefetch in Amazon Aurora (p. 594).
- **Fast DDL** – We have extended the feature that was released in Aurora v1.13 (p. 617) to operations that include default values. With this extension, Fast DDL is applicable for operations that add a nullable column at the end of a table, with or without default values. The feature remains under Aurora lab mode. For more information, see Altering Tables in Amazon Aurora Using Fast DDL (p. 522).

Improvements

- Fixed a calculation error during optimization of WITHIN/CONTAINS spatial queries which previously resulted in an empty result set.
- Fixed `SHOW VARIABLE` command to show the updated `innodb_buffer_pool_size` parameter value whenever it is changed in the parameter group.

- Improved stability of primary instance during bulk insert into a table altered using Fast DDL when adaptive hash indexing is disabled and the record to be inserted is the first record of a page.
- Improved stability of Aurora when the user attempts to set **server_audit_events** DB cluster parameter value to **`default`**.
- Fixed an issue in which a database charset change for an ALTER TABLE statement that ran on the Aurora primary instance was not being replicated on the Aurora Replicas until they were restarted.
- Improved stability by fixing a race condition on the primary instance which previously allowed it to register an Aurora Replica even if the primary instance had closed its own volume.
- Improved performance of the primary instance during index creation on a large table by changing the locking protocol to enable concurrent DMLs during index build.
- Fixed InnoDB metadata inconsistency during ALTER TABLE RENAME query which improved stability. Example: When columns of table t1(c1, c2) are renamed cyclically to t1(c2,c3) within the same ALTER statement.
- Improved stability of Aurora Replicas for the scenario where an Aurora Replica has no active workload and the primary instance is unresponsive.
- Improved availability of Aurora Replicas for a scenario in which the Aurora Replica holds an explicit lock on a table and blocks the replication thread from applying any DDL changes received from the primary instance.
- Improved stability of the primary instance when a foreign key and a column are being added to a table from two separate sessions at the same time and Fast DDL has been enabled.
- Improved stability of the purge thread on the primary instance during a heavy write workload by blocking truncate of undo records until they have been purged.
- Improved stability by fixing the lock release order during commit process of transactions which drop tables.
- Fixed a defect for Aurora Replicas in which the DB instance could not complete startup and complained that port 3306 was already in use.
- Fixed a race condition in which a SELECT query run on certain information_schema tables (innodb_trx, innodb_lock, innodb_lock_waits) increased cluster instability.

Integration of MySQL Bug Fixes

- CREATE USER accepts plugin and password hash, but ignores the password hash (Bug #78033
- Ignorable events do not work and are not tested (Bug #74683
- NEW->OLD ASSERT FAILURE `GTID_MODE > 0' IN 5.6.24 AT LOG_EVENT.CC:13555 (Bug#20436436
- The partitioning engine adds fields to the read bit set to be able to return entries sorted from a partitioned index. This leads to the join buffer will try to read unneeded fields. Fixed by not adding all partitioning fields to the read_set,but instead only sort on the already set prefix fields in the read_set. Added a DBUG_ASSERT that if doing key_cmp, at least the first field must be read (Bug#16367691
- MySQL instance stalling “doing SYNC index” (Bug #73816
- ASSERT RBT_EMPTY(INDEX_CACHE->WORDS) IN ALTER TABLE CHANGE COLUMN (Bug #17536995
- InnoDB Fulltext search doesn't find records when savepoints are involved (Bug #70333

Amazon Aurora MySQL Database Engine Updates: 2017-09-22

Version: 1.14.1

Amazon Aurora MySQL v1.14.1 is generally available. All new database clusters, including those restored from snapshots, will be created in Aurora MySQL v1.14.1. Aurora MySQL v1.14.1 is also a mandatory upgrade for existing Aurora MySQL DB clusters. For more information, see Announcement: Extension to Mandatory Upgrade Schedule for Amazon Aurora on the AWS Developer Forums website.

With version 1.14.1 of Aurora MySQL, we are using a cluster patching model where all nodes in an Aurora MySQL DB cluster are patched at the same time. Updates require a database restart, so you will experience 20 to 30 seconds of downtime, after which you can resume using your DB cluster or clusters. If your DB clusters are currently running version 1.13 or greater, Aurora MySQL's zero-downtime patching feature may allow client connections to your Aurora MySQL primary instance to persist through the upgrade, depending on your workload.

Should you have any questions or concerns, the AWS Support Team is available on the community forums and through AWS Premium Support at http://aws.amazon.com/support.

Improvements

- Fixed race conditions associated with inserts and purge to improve the stability of the Fast DDL feature, which remains in Aurora MySQL lab mode.

Amazon Aurora MySQL Database Engine Updates: 2017-08-07

Version: 1.14

Amazon Aurora MySQL 1.14 is generally available. All new database clusters, including those restored from snapshots, will be created in Aurora MySQL v1.14. Aurora MySQL v1.14 is also a mandatory upgrade for existing Aurora MySQL DB clusters. We will send a separate announcement with the timeline for deprecating earlier versions of Aurora MySQL.

With version 1.14 of Aurora MySQL, we are using a cluster patching model where all nodes in an Aurora DB cluster are patched at the same time. Updates require a database restart, so you will experience 20 to 30 seconds of downtime, after which you can resume using your DB cluster or clusters. If your DB clusters are currently running version 1.13, Aurora's zero-downtime patching feature may allow client connections to your Aurora primary instance to persist through the upgrade, depending on your workload.

Should you have any questions or concerns, the AWS Support Team is available on the community forums and through AWS Premium Support at http://aws.amazon.com/support.

Zero-Downtime Patching

The zero-downtime patching (ZDP) attempts, on a best-effort basis, to preserve client connections through an engine patch. If ZDP executes successfully, application sessions are preserved and the database engine restarts while patching. The database engine restart can cause a transient (5 second or so) drop in throughput.

ZDP will not execute successfully under the following conditions:

- Long-running queries are in progress
- Open long-running transactions exist
- Binary logging is enabled
- Binary log replication is running
- Pending parameter changes exist
- Temporary tables are in use
- Table locks are in use
- Open SSL connections exist

If no suitable time window for executing ZDP becomes available because of one or more of these conditions, patching reverts to the standard behavior.

Note
ZDP applies only to the primary instance of an Aurora DB cluster. ZDP is not applicable to Aurora Replicas.

Improvements

- Fixed an incorrect "record not found" error when a record is found in the secondary index but not in the primary index.
- Fixed a stability issue that can occur due to a defensive assertion (added in 1.12) that was too strong in the case when an individual write spans over 32 pages. Such a situation can occur, for instance, with large BLOB values.
- Fixed a stability issue because of inconsistencies between the tablespace cache and the dictionary cache.
- Fixed an issue in which an Aurora Replica becomes unresponsive after it has exceeded the maximum number of attempts to connect to the primary instance. An Aurora Replica now restarts if the period of inactivity is more than the heartbeat time period used for health check by the primary instance.
- Fixed a livelock that can occur under very high concurrency when one connection tries to acquire an exclusive meta data lock (MDL) while issuing a command, such as `ALTER TABLE`.
- Fixed a stability issue in an Aurora Read Replica in the presence of logical/parallel read ahead.
- Improved `LOAD FROM S3` in two ways:
  1. Better handling of Amazon S3 timeout errors by using the SDK retry in addition to the existing retry.
  2. Performance optimization when loading very big files or large numbers of files by caching and reusing client state.
- Fixed the following stability issues with Fast DDL for `ALTER TABLE` operations:
  1. When the `ALTER TABLE` statement has multiple `ADD COLUMN` commands and the column names are not in ascending order.
  2. When the name string of the column to be updated and its corresponding name string, fetched from the internal system table, differs by a null terminating character (/0).
  3. Under certain B-tree split operations.
  4. When the table has a variable length primary key.
- Fixed a stability issue with Aurora Replicas when it takes too long to make its Full Text Search (FTS) index cache consistent with that of the primary instance. This can happen if a large portion of the newly created FTS index entries on the primary instance have not yet been flushed to disk.
- Fixed a stability issue that can happen during index creation.
- New infrastructure that tracks memory consumption per connection and associated telemetry that will be used for building out Out-Of-Memory (OOM) avoidance strategies.
- Fixed an issue where `ANALYZE TABLE` was incorrectly allowed on Aurora Replicas. This has now been blocked.
- Fixed a stability issue caused by a rare deadlock as a result of a race condition between logical read-ahead and purge.

Integration of MySQL Bug Fixes

- A full-text search combined with derived tables (subqueries in the `FROM` clause) caused a server exit. Now, if a full-text operation depends on a derived table, the server produces an error indicating that a full-text search cannot be done on a materialized table. (Bug #68751, Bug #16539903)

Amazon Aurora MySQL Database Engine Updates: 2017-05-15

Version: 1.13

Note
We enabled a new feature – SELECT INTO OUTFILE S3 – in Amazon Aurora MySQL version 1.13 after the initial release, and have updated the release notes to reflect that change.

Amazon Aurora MySQL 1.13 is generally available. All new database clusters, including those restored from snapshots, will be created in Aurora MySQL v1.13. You have the option, but are not required, to upgrade existing database clusters to Aurora MySQL v1.13. With version 1.13 of Aurora, we are using a cluster patching model where all nodes in an Aurora DB cluster are patched at the same time. We are enabling zero-downtime patching, which works on a best-effort basis to preserve client connections through the patching process. For more information, see Amazon RDS Maintenance (p. 102).

Zero-Downtime Patching

The zero-downtime patching (ZDP) attempts, on a best-effort basis, to preserve client connections through an engine patch. If ZDP executes successfully, application sessions are preserved and the database engine restarts while patching. The database engine restart can cause a transient (5 second or so) drop in throughput.

ZDP will not execute successfully under the following conditions:

- Long-running queries are in progress
- Open long-running transactions exist
- Binary logging is enabled
- Binary log replication is running
- Pending parameter changes exist
- Temporary tables are in use
- Table locks are in use
- Open SSL connections exist

If no suitable time window for executing ZDP becomes available because of one or more of these conditions, patching reverts to the standard behavior.

Note
ZDP applies only to the primary instance of an Aurora DB cluster. ZDP is not applicable to Aurora Replicas.

New Features:

- **SELECT INTO OUTFILE S3** – Amazon Aurora MySQL now allows you to upload the results of a query to one or more files in an Amazon S3 bucket. For more information, see Saving Data from an Amazon Aurora MySQL DB Cluster into Text Files in an Amazon S3 Bucket (p. 567).

Improvements:

- Implemented truncation of CSV format log files at engine startup to avoid long recovery time. The `general_log_backup`, `general_log`, `slow_log_backup`, and `slow_log` tables now do not survive a database restart.
- Fixed an issue where migration of a database named **`test`** would fail.
- Improved stability in the lock manager's garbage collector by reusing the correct lock segments.
- Improved stability of the lock manager by removing invalid assertions during deadlock detection algorithm.
- Re-enabled asynchronous replication, and fixed an associated issue which caused incorrect replica lag to be reported under no-load or read-only workload. The replication pipeline improvements that were

introduced in version 1.10. These improvements were introduced in order to apply log stream updates to the buffer cache of an Aurora Replica. which helps to improve read performance and stability on Aurora Replicas.
- Fixed an issue where autocommit=OFF leads to scheduled events being blocked and long transactions being held open until the server reboots.
- Fixed an issue where general, audit, and slow query logs could not log queries handled by asynchronous commit.
- Improved the performance of the logical read ahead (LRA) feature by up to 2.5 times. This was done by allowing pre-fetches to continue across intermediate pages in a B-tree.
- Added parameter validation for audit variables to trim unnecessary spaces.
- Fixed a regression, introduced in Aurora MySQL version 1.11, in which queries can return incorrect results when using the SQL_CALC_FOUND_ROWS option and invoking the FOUND_ROWS() function.
- Fixed a stability issue when the Metadata Lock list was incorrectly formed.
- Improved stability when sql_mode is set to PAD_CHAR_TO_FULL_LENGTH and the command `SHOW FUNCTION STATUS WHERE Db='string'` is executed.
- Fixed a rare case when instances would not come up after Aurora version upgrade because of a false volume consistency check.
- Fixed the performance issue, introduced in Aurora MySQL version 1.12, where the performance of the Aurora writer was reduced when users have a large number of tables.
- Improved stability issue when the Aurora writer is configured as a binlog slave and the number of connections approaches 16,000.
- Fixed a rare issue where an Aurora Replica could restart when a connection gets blocked waiting for Metadata Lock when running DDL on the Aurora master.

Integration of MySQL Bug Fixes

- With an empty InnoDB table, it's not possible to decrease the auto_increment value using an ALTER TABLE statement, even when the table is empty. (Bug #69882)
- MATCH() ... AGAINST queries that use a long string as an argument for AGAINST() could result in an error when run on an InnoDB table with a full-text search index. (Bug #17640261)
- Handling of SQL_CALC_FOUND_ROWS in combination with ORDER BY and LIMIT could lead to incorrect results for FOUND_ROWS(). (Bug #68458, Bug # 16383173)
- ALTER TABLE does not allow to change nullability of the column if foreign key exists. (Bug #77591)

Amazon Aurora MySQL Database Engine Updates: 2017-04-05

Version: 1.12

Amazon Aurora MySQL 1.12 is now the preferred version for the creation of new DB clusters, including restores from snapshots.

This is not a mandatory upgrade for existing clusters. You will have the option to upgrade existing clusters to version 1.12 after we complete the fleet-wide patch to 1.11 (see Aurora 1.11 release notes (p. 620) and corresponding forum announcement). With version 1.12 of Aurora, we are using a cluster patching model where all nodes in an Aurora DB cluster are patched at the same time. For more information, see Amazon RDS Maintenance (p. 102).

New Features

- **Fast DDL** – Amazon Aurora MySQL now allows you to execute an ALTER TABLE *tbl_name* ADD COLUMN *col_name column_definition* operation nearly instantaneously. The operation completes without requiring the table to be copied and without materially impacting other DML statements.

Since it does not consume temporary storage for a table copy, it makes DDL statements practical even for large tables on small instance types. Fast DDL is currently only supported for adding a nullable column, without a default value, at the end of a table. This feature is currently available in Aurora lab mode. For more information, see Altering Tables in Amazon Aurora Using Fast DDL (p. 522).

- **Show volume status** – We have added a new monitoring command, SHOW VOLUME STATUS, to display the number of nodes and disks in a volume. For more information, see Displaying Volume Status for an Aurora DB Cluster (p. 523).

Improvements

- Implemented changes to lock compression to further reduce memory allocated per lock object. This improvement is available in lab mode.
- Fixed an issue where the `trx_active_transactions` metric decrements rapidly even when the database is idle.
- Fixed an invalid error message regarding fault injection query syntax when simulating failure in disks and nodes.
- Fixed multiple issues related to race conditions and dead latches in the lock manager.
- Fixed an issue causing a buffer overflow in the query optimizer.
- Fixed a stability issue in Aurora read replicas when the underlying storage nodes experience low available memory.
- Fixed an issue where idle connections persisted past the `wait_timeout` parameter setting.
- Fixed an issue where `query_cache_size` returns an unexpected value after reboot of the instance.
- Fixed a performance issue that is the result of a diagnostic thread probing the network too often in the event that writes are not progressing to storage.

Integration of MySQL Bug Fixes

- Reloading a table that was evicted while empty caused an AUTO_INCREMENT value to be reset. (Bug #21454472, Bug #77743)
- An index record was not found on rollback due to inconsistencies in the purge_node_t structure. The inconsistency resulted in warnings and error messages such as “error in sec index entry update”, “unable to purge a record”, and “tried to purge sec index entry not marked for deletion”. (Bug #19138298, Bug #70214, Bug #21126772, Bug #21065746)
- Wrong stack size calculation for qsort operation leads to stack overflow. (Bug #73979)
- Record not found in an index upon rollback. (Bug #70214, Bug #72419)
- ALTER TABLE add column TIMESTAMP on update CURRENT_TIMESTAMP inserts ZERO-datas (Bug #17392)

Amazon Aurora MySQL Database Engine Updates: 2017-02-23

Version: 1.11

We will patch all Amazon Aurora MySQL DB clusters with the latest version over a short period following the release. DB clusters are patched using the legacy procedure with a downtime of about 5-30 seconds.

Patching occurs during the system maintenance window that you have specified for each of your database instances. You can view or change this window using the AWS Management Console. For more information, see Amazon RDS Maintenance (p. 102).

Alternatively, you can apply the patch immediately in the AWS Management Console by choosing a DB cluster, choosing **Cluster Actions**, and then choosing **Upgrade Now**.

With version 1.11 of Aurora MySQL, we are using a cluster patching model where all nodes in an Aurora DB cluster are patched at the same time.

New Features

- **MANIFEST option for LOAD DATA FROM S3** – LOAD DATA FROM S3 was released in version 1.8. The options for this command have been expanded, and you can now specify a list of files to be loaded into an Aurora DB cluster from Amazon S3 by using a manifest file. This makes it easy to load data from specific files in one or more locations, as opposed to loading data from a single file by using the FILE option or loading data from multiple files that have the same location and prefix by using the PREFIX option. The manifest file format is the same as that used by Amazon Redshift. For more information about using LOAD DATA FROM S3 with the MANIFEST option, see Using a Manifest to Specify Data Files to Load (p. 563).
- **Spatial indexing enabled by default** – This feature was released in lab mode in version 1.10, and is now turned on by default. Spatial indexing improves query performance on large datasets for queries that use spatial data. For more information about using spatial indexing, see Amazon Aurora MySQL and Spatial Data (p. 485).
- **Throughput improvement for workloads with hot row contention** – This feature was released in lab mode in version 1.10, and is now available outside of lab mode. Throughput for workloads with hot row contention was improved by changing the lock release algorithm used by Aurora. This change improves TPC-C benchmark performance by up to 16x relative to MySQL 5.7.
- **Advanced Auditing timing change** – This feature was released in version 1.10.1 to provide a high-performance facility for auditing database activity. In this release, the precision of audit log timestamps has been changed from one second to one microsecond. The more accurate timestamps allow you to better understand when an audit event happened. For more information about audit, see Using Advanced Auditing with an Amazon Aurora MySQL DB Cluster (p. 524).

Improvements

- Modified the `thread_handling` parameter to prevent you from setting it to anything other than **`multiple-connections-per-thread`**, which is the only supported model with Aurora's thread pool.
- Fixed an issue caused when you set either the `buffer_pool_size` or the `query_cache_size` parameter to be larger than the DB cluster's total memory. In this circumstance, Aurora sets the modified parameter to the default value, so the DB cluster can start up and not crash.
- Fixed an issue in the query cache where a transaction gets stale read results if the table is invalidated in another transaction.
- Fixed an issue where binlog files marked for deletion are removed after a small delay rather than right away.
- Fixed an issue where a database created with the name **`tmp`** is treated as a system database stored on ephemeral storage and not persisted to Aurora distributed storage.
- Modified the behavior of SHOW TABLES to exclude certain internal system tables. This change helps avoid an unnecessary failover caused by mysqldump locking all files listed in SHOW TABLES, which in turn prevents writes on the internal system table, causing the failover.
- Fixed an issue where an Aurora Replica incorrectly restarts when creating a temporary table from a query that invokes a function whose argument is a column of an InnoDB table.
- Fixed an issue related to a metadata lock conflict in an Aurora Replica node that causes the Aurora Replica to fall behind the primary DB cluster and eventually get restarted.
- Fixed a dead latch in the replication pipeline in reader nodes, which causes an Aurora Replica to fall behind and eventually get restarted.
- Fixed an issue where an Aurora Replica lags too much with encrypted volumes larger than 1 terabyte (TB).

- Improved Aurora Replica dead latch detection by using an improved way to read the system clock time.
- Fixed an issue where an Aurora Replica can restart twice instead of once following de-registration by the writer.
- Fixed a slow query performance issue on Aurora Replicas that occurs when transient statistics cause statistics discrepancy on non-unique index columns.
- Fixed an issue where an Aurora Replica can crash when a DDL statement is replicated on the Aurora Replica at the same time that the Aurora Replica is processing a related query.
- Changed the replication pipeline improvements that were introduced in version 1.10 from enabled by default to disabled by default. These improvements were introduced in order to apply log stream updates to the buffer cache of an Aurora Replica, and although this feature helps to improve read performance and stability on Aurora Replicas, it increases replica lag in certain workloads.
- Fixed an issue where the simultaneous occurrence of an ongoing DDL statement and pending Parallel Read Ahead on the same table causes an assertion failure during the commit phase of the DDL transaction.
- Enhanced the general log and slow query log to survive DB cluster restart.
- Fixed an out-of-memory issue for certain long running queries by reducing memory consumption in the ACL module.
- Fixed a restart issue that occurs when a table has non-spatial indexes, there are spatial predicates in the query, the planner chooses to use a non-spatial index, and the planner incorrectly pushes the spatial condition down to the index.
- Fixed an issue where the DB cluster restarts when there is a delete, update, or purge of very large geospatial objects that are stored externally (like LOBs).
- Fixed an issue where fault simulation using ALTER SYSTEM SIMULATE … FOR INTERVAL isn't working properly.
- Fixed a stability issue caused by an invalid assertion on an incorrect invariant in the lock manager.
- Disabled the following two improvements to InnoDB Full-Text Search that were introduced in version 1.10 because they introduce stability issues for some demanding workloads:
  - Updating the cache only after a read request to an Aurora Replica in order to improve full-text search index cache replication speed.
  - Offloading the cache sync task to a separate thread as soon as the cache size crosses 10% of the total size, in order to avoid MySQL queries stalling for too long during FTS cache sync to disk. (Bugs #22516559, #73816).

Integration of MySQL Bug Fixes

- Running ALTER table DROP foreign key simultaneously with another DROP operation causes the table to disappear. (Bug #16095573)
- Some INFORMATION_SCHEMA queries that used ORDER BY did not use a filesort optimization as they did previously. (Bug #16423536)
- FOUND_ROWS () returns the wrong count of rows on a table. (Bug #68458)
- The server fails instead of giving an error when too many temp tables are open. (Bug #18948649)

Amazon Aurora MySQL Database Engine Updates: 2017-01-12

Version: 1.10.1

Version 1.10.1 of Amazon Aurora MySQL is an opt-in version and is not used to patch your database instances. It is available for creating new Aurora instances and for upgrading existing instances. You can apply the patch by choosing a cluster in the Amazon RDS console, choosing **Cluster Actions**, and then choosing **Upgrade Now**. Patching requires a database restart with downtime typically lasting 5-30

seconds, after which you can resume using your DB clusters. This patch is using a cluster patching model where all nodes in an Aurora cluster are patched at the same time.

New Features

- **Advanced Auditing** – Amazon Aurora MySQL provides a high-performance Advanced Auditing feature, which you can use to audit database activity. For more information about enabling and using Advanced Auditing, see Using Advanced Auditing with an Amazon Aurora MySQL DB Cluster (p. 524).

Improvements

- Fixed an issue with spatial indexing when creating a column and adding an index on it in the same statement.
- Fixed an issue where spatial statistics aren't persisted across DB cluster restart.

Amazon Aurora MySQL Database Engine Updates: 2016-12-14

Version: 1.10

New Features

- **Zero downtime patch** – This feature allows a DB instance to be patched without any downtime. That is, database upgrades are performed without disconnecting client applications, or rebooting the database. This approach increases the availability of your Aurora DB clusters during the maintenance window. Note that temporary data like that in the performance schema is reset during the upgrade process. This feature applies to service-delivered patches during a maintenance window as well as user-initiated patches.

  When a patch is initiated, the service ensures there are no open locks, transactions or temporary tables, and then waits for a suitable window during which the database can be patched and restarted. Application sessions are preserved, although there is a drop in throughput while the patch is in progress (for approximately 5 seconds). If no suitable window can be found, then patching defaults to the standard patching behavior.

  Zero downtime patching takes place on a best-effort basis, subject to certain limitations as described following:

  - This feature is currently applicable for patching single-node DB clusters or writer instances in multi-node DB clusters.
  - SSL connections are not supported in conjunction with this feature. If there are active SSL connections, Amazon Aurora MySQL won't perform a zero downtime patch, and instead will retry periodically to see if the SSL connections have terminated. If they have, zero downtime patching proceeds. If the SSL connections persist after more than a couple seconds, standard patching with downtime proceeds.
  - The feature is available in Aurora release 1.10 and beyond. Going forward, we will identify any releases or patches that can't be applied by using zero downtime patching.
  - This feature is not applicable if replication based on binary logging is active.
- **Spatial indexing** – Spatial indexing improves query performance on large datasets for queries that use spatial data. For more information about using spatial indexing, see Amazon Aurora MySQL and Spatial Data (p. 485).

  This feature is disabled by default and can be activated by enabling Aurora lab mode. For information, see Aurora Lab Mode (p. 611).
- **Replication pipeline improvements** – Amazon Aurora MySQL now uses an improved mechanism to apply log stream updates to the buffer cache of an Aurora Replica. This feature improves the read

performance and stability on Aurora Replicas when there is a heavy write load on the master as well as a significant read load on the Replica. This feature is enabled by default.

- **Throughput improvement for workloads with cached reads** – Amazon Aurora MySQL now uses a latch-free concurrent algorithm to implement read views, which leads to better throughput for read queries served by the buffer cache. As a result of this and other improvements, Amazon Aurora MySQL can achieve throughput of up to 625K reads per second compared to 164K reads per second by MySQL 5.7 for a sysbench SELECT-only workload.
- **Throughput improvement for workloads with hot row contention** – Amazon Aurora MySQL uses a new lock release algorithm that improves performance, particularly when there is hot page contention (that is, many transactions contending for the rows on the same page). In tests with the TPC-C benchmark, this can result in up to16x throughput improvement in transactions per minute relative to MySQL 5.7. This feature is disabled by default and can be activated by enabling Aurora lab mode. For information, see Aurora Lab Mode (p. 611).

Improvements

- Full-text search index cache replication speed has been improved by updating the cache only after a read request to an Aurora Replica. This approach avoids any reads from disk by the replication thread.
- Fixed an issue where dictionary cache invalidation does not work on an Aurora Replica for tables that have a special character in the database name or table name.
- Fixed a `STUCK IO` issue during data migration for distributed storage nodes when storage heat management is enabled.
- Fixed an issue in the lock manager where an assertion check fails for the transaction lock wait thread when preparing to rollback or commit a transaction.
- Fixed an issue when opening a corrupted dictionary table by correctly updating the reference count to the dictionary table entries.
- Fixed a bug where the DB cluster minimum read point can be held by slow Aurora Replicas.
- Fixed a potential memory leak in the query cache.
- Fixed a bug where an Aurora Replica places a row-level lock on a table when a query is used in an `IF` statement of a stored procedure.

Integration of MySQL Bug Fixes

- UNION of derived tables returns wrong results with '1=0/false'-clauses. (Bug #69471)
- Server crashes in ITEM_FUNC_GROUP_CONCAT::FIX_FIELDS on 2nd execution of stored procedure. (Bug #20755389)
- Avoid MySQL queries from stalling for too long during FTS cache sync to disk by offloading the cache sync task to a separate thread, as soon as the cache size crosses 10% of the total size. (Bug #22516559, #73816)

Amazon Aurora MySQL Database Engine Updates: 2016-11-10

Version: 1.9.0, 1.9.1

New Features

- **Improved index build** – The implementation for building secondary indexes now operates by building the index in a bottom-up fashion, which eliminates unnecessary page splits. This can reduce the time needed to create an index or rebuild a table by up to 75% (based on an `db.r3.8xlarge` DB instance class). This feature was in lab mode in Aurora MySQL version 1.7 and is enabled by default in Aurora version 1.9 and later. For information, see Aurora Lab Mode (p. 611).

- **Lock compression (lab mode)** – This implementation significantly reduces the amount of memory that lock manager consumes by up to 66%. Lock manager can acquire more row locks without encountering an out-of-memory exception. This feature is disabled by default and can be activated by enabling Aurora lab mode. For information, see Aurora Lab Mode (p. 611).
- **Performance schema** – Amazon Aurora MySQL now includes support for performance schema with minimal impact on performance. In our testing using SysBench, enabling performance schema could degrade MySQL performance by up to 60%.

  SysBench testing of an Aurora DB cluster showed an impact on performance that is 4x less than MySQL. Running the `db.r3.8xlarge` DB instance class resulted in 100K SQL writes/sec and over 550K SQL reads/sec, even with performance schema enabled.
- **Hot row contention improvement** – This feature reduces CPU utilization and increases throughput when a small number of hot rows are accessed by a large number of connections. This feature also eliminates `error 188` when there is hot row contention.
- **Improved out-of-memory handling** – When non-essential, locking SQL statements are executed and the reserved memory pool is breached, Aurora forces rollback of those SQL statements. This feature frees memory and prevents engine crashes due to out-of-memory exceptions.
- **Smart read selector** – This implementation improves read latency by choosing the optimal storage segment among different segments for every read, resulting in improved read throughput. SysBench testing has shown up to a 27% performance increase for write workloads .

Improvements

- Fixed an issue where an Aurora Replica encounters a shared lock during engine start up.
- Fixed a potential crash on an Aurora Replica when the read view pointer in the purge system is NULL.

Amazon Aurora MySQL Database Engine Updates: 2016-10-26

Version: 1.8.1

Improvements

- Fixed an issue where bulk inserts that use triggers that invoke AWS Lambda procedures fail.
- Fixed an issue where catalog migration fails when autocommit is off globally.
- Resolved a connection failure to Aurora when using SSL and improved Diffie-Hellman group to deal with LogJam attacks.

Integration of MySQL Bug Fixes

- OpenSSL changed the Diffie-Hellman key length parameters due to the LogJam issue. (Bug #18367167)

Amazon Aurora MySQL Database Engine Updates: 2016-10-18

Version: 1.8

New Features

- **AWS Lambda integration** – You can now asynchronously invoke an AWS Lambda function from an Aurora DB cluster using the `mysql.lambda_async` procedure. For more information, see Invoking a Lambda Function from an Amazon Aurora MySQL DB Cluster (p. 572).

- **Load data from Amazon S3** – You can now load text or XML files from an Amazon S3 bucket into your Aurora DB cluster using the `LOAD DATA FROM S3` or `LOAD XML FROM S3` commands. For more information, see Loading Data into an Amazon Aurora MySQL DB Cluster from Text Files in an Amazon S3 Bucket (p. 560).
- **Catalog migration** – Aurora now persists catalog metadata in the cluster volume to support versioning. This enables seamless catalog migration across versions and restores.
- **Cluster-level maintenance and patching** – Aurora now manages maintenance updates for an entire DB cluster. For more information, see Amazon RDS Maintenance (p. 102).

Improvements

- Fixed an issue where an Aurora Replica crashes when not granting a metadata lock to an inflight DDL table.
- Allowed Aurora Replicas to modify non-InnoDB tables to facilitate rotation of the slow and general log CSV files where `log_output=TABLE`.
- Fixed a lag when updating statistics from the primary instance to an Aurora Replica. Without this fix, the statistics of the Aurora Replica can get out of sync with the statistics of the primary instance and result in a different (and possibly under-performing) query plan on an Aurora Replica.
- Fixed a race condition that ensures that an Aurora Replica does not acquire locks.
- Fixed a rare scenario where an Aurora Replica that registers or de-registers with the primary instance could fail.
- Fixed a race condition that could lead to a deadlock on `db.r3.large` instances when opening or closing a volume.
- Fixed an out-of-memory issue that can occur due to a combination of a large write workload and failures in the Aurora Distributed Storage service.
- Fixed an issue with high CPU consumption because of the purge thread spinning in the presence of a long-running transaction.
- Fixed an issue when running information schema queries to get information about locks under heavy load.
- Fixed an issue with a diagnostics process that could in rare cases cause Aurora writes to storage nodes to stall and restart/fail-over.
- Fixed a condition where a successfully created table may be deleted during crash recovery if the crash occurred while a `CREATE TABLE [if not exists]` statement was being handled.
- Fixed a case where the log rotation procedure is broken when the general log and slow log are not stored on disk using catalog mitigation.
- Fixed a crash when a user creates a temporary table within a user defined function, and then uses the user defined function in the select list of the query.
- Fixed a crash that occurred when replaying GTID events. GTID is not supported by Amazon Aurora MySQL.

Integration of MySQL Bug Fixes:

- When dropping all indexes on a column with multiple indexes, InnoDB failed to block a DROP INDEX operation when a foreign key constraint requires an index. (Bug #16896810)
- Solve add foreign key constraint crash. (Bug #16413976)
- Fixed a crash when fetching a cursor in a stored procedure, and analyzing or flushing the table at the same time. (Bug # 18158639)
- Fixed an auto-increment bug when a user alters a table to change the AUTO_INCREMENT value to less than the maximum auto-increment column value. (Bug # 16310273)

Amazon Aurora MySQL Database Engine Updates: 2016-09-20

Version: 1.7.1

Improvements

- Fixes an issue where an Aurora Replica crashes if the InnoDB full-text search cache is full.
- Fixes an issue where the database engine crashes if a worker thread in the thread pool waits for itself.
- Fixes an issue where an Aurora Replica crashes if a metadata lock on a table causes a deadlock.
- Fixes an issue where the database engine crashes due to a race condition between two worker threads in the thread pool.
- Fixes an issue where an unnecessary failover occurs under heavy load if the monitoring agent doesn't detect the advancement of write operations to the distributed storage subsystem.

Amazon Aurora MySQL Database Engine Updates: 2016-08-30

Version: 1.7.0

New Features

- **NUMA aware scheduler** – The task scheduler for the Amazon Aurora MySQL engine is now Non-Uniform Memory Access (NUMA) aware. This minimizes cross-CPU socket contention resulting in improved performance throughput for the `db.r3.8xlarge` DB instance class.
- **Parallel read-ahead operates asynchronously in the background** – Parallel read-ahead has been revised to improve performance by using a dedicated thread to reduce thread contention.
- **Improved index build (lab mode)** – The implementation for building secondary indexes now operates by building the index in a bottom-up fashion, which eliminates unnecessary page splits. This can reduce the time needed to create an index or rebuild a table. This feature is disabled by default and can be activated by enabling Aurora lab mode. For information, see Aurora Lab Mode (p. 611).

Improvements

- Fixed an issue where establishing a connection was taking a long time if there was a surge in the number of connections requested for an instance.
- Fixed an issue where a crash occurred if ALTER TABLE was run on a partitioned table that did not use InnoDB.
- Fixed an issue where heavy write workload can cause a failover.
- Fixed an erroneous assertion that caused a failure if RENAME TABLE was run on a partitioned table.
- Improved stability when rolling back a transaction during insert-heavy workload.
- Fixed an issue where full-text search indexes were not viable on an Aurora Replica.

Integration of MySQL Bug Fixes

- Improve scalability by partitioning LOCK_grant lock. (Port WL #8355)
- Opening cursor on SELECT in stored procedure causes segfault. (Port Bug#16499751)
- MySQL gives the wrong result with some special usage. (Bug #11751794)
- Crash in GET_SEL_ARG_FOR_KEYPART – caused by patch for bug #11751794. (Bug #16208709)
- Wrong results for a simple query with GROUP BY. (Bug #17909656)
- Extra rows on semijoin query with range predicates. (Bug #16221623)

- Adding an ORDER BY clause following an IN subquery could cause duplicate rows to be returned. (Bug #16308085)
- Crash with explain for a query with loose scan for GROUP BY, MyISAM. (Bug #16222245)
- Loose index scan with quoated int predicate returns random data. (Bug #16394084)
- If the optimizer was using a loose index scan, the server could exit while attempting to create a temporary table. (Bug #16436567)
- COUNT(DISTINCT) should not count NULL values, but they were counted when the optimizer used loose index scan. (Bug #17222452)
- If a query had both MIN()/MAX() and aggregate_function(DISTINCT) (for example, SUM(DISTINCT)) and was executed using loose index scan, the result values of MIN()/MAX() were set improperly. (Bug #17217128)

Amazon Aurora MySQL Database Engine Updates: 2016-06-01

Version: 1.6.5

New Features

- **Efficient storage of Binary Logs** – Efficient storage of binary logs is now enabled by default for all Amazon Aurora MySQL DB clusters, and is not configurable. Efficient storage of binary logs was introduced in the April 2016 update. For more information, see Amazon Aurora MySQL Database Engine Updates: 2016-04-06 (p. 628).

Improvements

- Improved stability for Aurora Replicas when the primary instance is encountering a heavy workload.
- Improved stability for Aurora Replicas when running queries on partitioned tables and tables with special characters in the table name.
- Fixed connection issues when using secure connections.

Integration of MySQL Bug Fixes

- SLAVE CAN'T CONTINUE REPLICATION AFTER MASTER'S CRASH RECOVERY (Port Bug #17632285)

Amazon Aurora MySQL Database Engine Updates: 2016-04-06

Version: 1.6

This update includes the following improvements:

New Features

- **Parallel read-ahead** – Parallel read-ahead is now enabled by default for all Amazon Aurora MySQL DB clusters, and is not configurable. Parallel read-ahead was introduced in the December 2015 update. For more information, see Amazon Aurora MySQL Database Engine Updates: 2015-12-03 (p. 630).

  In addition to enabling parallel read-ahead by default, this release includes the following improvements to parallel read-ahead:

  - Improved logic so that parallel read-ahead is less aggressive, which is beneficial when your DB cluster encounters many parallel workloads.
  - Improved stability on smaller tables.

- **Efficient storage of Binary Logs (lab mode)** – MySQL binary log files are now stored more efficiently in Amazon Aurora MySQL. The new storage implementation enables binary log files to be deleted much earlier and improves system performance for an instance in an Amazon Aurora MySQL DB cluster that is a binary log replication master.

  To enable efficient storage of binary logs, set the `aurora_lab_mode` parameter to `1` in the parameter group for your primary instance or Aurora Replica. The `aurora_lab_mode` parameter is an instance-level parameter that is in the `default.aurora5.6` parameter group by default. For information on modifying a DB parameter group, see Modifying Parameters in a DB Parameter Group (p. 172). For information on parameter groups and Aurora MySQL, see Amazon Aurora MySQL Parameters (p. 600).

  Only turn on efficient storage of binary logs for instances in an Amazon Aurora MySQL DB cluster that are MySQL binary log replication master instances.
- **AURORA_VERSION system variable** – You can now get the Aurora version of your Amazon Aurora MySQL DB cluster by querying for the `AURORA_VERSION` system variable.

  To get the Aurora version, use one of the following queries:

```
select AURORA_VERSION();
```

```
select @@aurora_version;
```

```
show variables like '%version';
```

  You can also see the Aurora version in the AWS Management Console when you modify a DB cluster, or by calling the describe-db-engine-versions AWS CLI command or the DescribeDBEngineVersions API action.
- **Lock manager memory usage metric** – Information about lock manager memory usage is now available as a metric.

  To get the lock manager memory usage metric, use one of the following queries:

```
show global status where variable_name in ('aurora_lockmgr_memory_used');
```

```
select * from INFORMATION_SCHEMA.GLOBAL_STATUS where variable_name in
 ('aurora_lockmgr_memory_used');
```

Improvements

- Improved stability during binlog and XA transaction recovery.
- Fixed a memory issue resulting from a large number of connections.
- Improved accuracy in the following metrics: `Read Throughput`, `Read IOPS`, `Read Latency`, `Write Throughput`, `Write IOPS`, `Write Latency`, and `Disk Queue Depth`.
- Fixed a stability issue causing slow startup for large instances after a crash.
- Improved concurrency in the data dictionary regarding synchronization mechanisms and cache eviction.
- Stability and performance improvements for Aurora Replicas:
  - Fixed a stability issue for Aurora Replicas during heavy or burst write workloads for the primary instance.
  - Improved replica lag for db.r3.4xlarge and db.r3.8xlarge instances.

- Improved performance by reducing contention between application of log records and concurrent reads on an Aurora Replica.
- Fixed an issue for refreshing statistics on Aurora Replicas for newly created or updated statistics.
- Improved stability for Aurora Replicas when there are many transactions on the primary instance and concurrent reads on the Aurora Replicas across the same data.
- Improved stability for Aurora Replicas when running `UPDATE` and `DELETE` statements with `JOIN` statements.
- Improved stability for Aurora Replicas when running `INSERT ... SELECT` statements.

Integration of MySQL Bug Fixes

- BACKPORT BUG#18694052 FIX FOR ASSERTION `!M_ORDERED_REC_BUFFER' FAILED TO 5.6 (Port Bug #18305270)
- SEGV IN MEMCPY(), HA_PARTITION::POSITION (Port Bug # 18383840)
- WRONG RESULTS WITH PARTITIONING,INDEX_MERGE AND NO PK (Port Bug # 18167648)
- FLUSH TABLES FOR EXPORT: ASSERTION IN HA_PARTITION::EXTRA (Port Bug # 16943907)
- SERVER CRASH IN VIRTUAL HA_ROWS HANDLER::MULTI_RANGE_READ_INFO_CONST (Port Bug # 16164031)
- RANGE OPTIMIZER CRASHES IN SEL_ARG::RB_INSERT() (Port Bug # 16241773)

Amazon Aurora MySQL Database Engine Updates: 2016-01-11

Version: 1.5

This update includes the following improvements:

Improvements

- Fixed a 10 second pause of write operations for idle instances during Aurora storage deployments.
- Logical read-ahead now works when `innodb_file_per_table` is set to `No`. For more information on logical read-ahead, see Amazon Aurora MySQL Database Engine Updates: 2015-12-03 (p. 630).
- Fixed issues with Aurora Replicas reconnecting with the primary instance. This improvement also fixes an issue when you specify a large value for the `quantity` parameter when testing Aurora Replica failures using fault-injection queries. For more information, see Testing an Aurora Replica Failure (p. 520).
- Improved monitoring of Aurora Replicas falling behind and restarting.
- Fixed an issue that caused an Aurora Replica to lag, become deregistered, and then restart.
- Fixed an issue when you run the `show innodb status` command during a deadlock.
- Fixed an issue with failovers for larger instances during high write throughput.

Integration of MySQL Bug Fixes

- Addressed incomplete fix in MySQL full text search affecting tables where the database name begins with a digit. (Port Bug #17607956)

Amazon Aurora MySQL Database Engine Updates: 2015-12-03

Version: 1.4

This update includes the following improvements:

New Features

- **Fast Insert** – Accelerates parallel inserts sorted by primary key. For more information, see Amazon Aurora Performance Enhancements (p. 433).
- **Large dataset read performance** – Amazon Aurora MySQL automatically detects an IO heavy workload and launches more threads in order to boost the performance of the DB cluster. The Aurora scheduler looks into IO activity and decides to dynamically adjust the optimal number of threads in the system, quickly adjusting between IO heavy and CPU heavy workloads with low overhead.
- **Parallel read-ahead** – Improves the performance of B-Tree scans that are too large for the memory available on your primary instance or Aurora Replica (including range queries). Parallel read-ahead automatically detects page read patterns and pre-fetches pages into the buffer cache in advance of when they are needed. Parallel read-ahead works multiple tables at the same time within the same transaction.

Improvements:

- Fixed brief Aurora database availability issues during Aurora storage deployments.
- Correctly enforce the `max_connection` limit.
- Improve binlog purging where Aurora is the binlog master and the database is restarting after a heavy data load.
- Fixed memory management issues with the table cache.
- Add support for huge pages in shared memory buffer cache for faster recovery.
- Fixed an issue with thread-local storage not being initialized.
- Allow 16K connections by default.
- Dynamic thread pool for IO heavy workloads.
- Fixed an issue with properly invalidating views involving UNION in the query cache.
- Fixed a stability issue with the dictionary stats thread.
- Fixed a memory leak in the dictionary subsystem related to cache eviction.
- Fixed high read latency issue on Aurora Replicas when there is very low write load on the master.
- Fixed stability issues on Aurora Replicas when performing operations on DDL partitioned tables such as ALTER TABLE ... REORGANIZE PARTITION on the master.
- Fixed stability issues on Aurora Replicas during volume growth.
- Fixed performance issue for scans on non-clustered indexes in Aurora Replicas.
- Fix stability issue that makes Aurora Replicas lag and eventually get deregistered and re-started.

Integration of MySQL Bug Fixes

- SEGV in FTSPARSE(). (Bug #16446108)
- InnoDB data dictionary is not updated while renaming the column. (Bug #19465984)
- FTS crash after renaming table to different database. (Bug #16834860)
- Failed preparing of trigger on truncated tables cause error 1054. (Bug #18596756)
- Metadata changes might cause problems with trigger execution. (Bug #18684393)
- Materialization is not chosen for long UTF8 VARCHAR field. (Bug #17566396)
- Poor execution plan when ORDER BY with limit X. (Bug #16697792)
- Backport bug #11765744 TO 5.1, 5.5 AND 5.6. (Bug #17083851)
- Mutex issue in SQL/SQL_SHOW.CC resulting in SIG6. Source likely FILL_VARIABLES. (Bug #20788853)

- Backport bug #18008907 to 5.5+ versions. (Bug #18903155)
- Adapt fix for a stack overflow error in MySQL 5.7. (Bug #19678930)

Amazon Aurora MySQL Database Engine Updates: 2015-10-16

Versions: 1.2, 1.3

This update includes the following improvements:

Fixes

- Resolved out-of-memory issue in the new lock manager with long-running transactions
- Resolved security vulnerability when replicating with non-RDS MySQL databases
- Updated to ensure that quorum writes retry correctly with storage failures
- Updated to report replica lag more accurately
- Improved performance by reducing contention when many concurrent transactions are trying to modify the same row
- Resolved query cache invalidation for views that are created by joining two tables
- Disabled query cache for transactions with `UNCOMMITTED_READ` isolation

Improvements

- Better performance for slow catalog queries on warm caches
- Improved concurrency in dictionary statistics
- Better stability for the new query cache resource manager, extent management, files stored in Amazon Aurora smart storage, and batch writes of log records

Integration of MySQL Bug Fixes

- Killing a query inside innodb causes it to eventually crash with an assertion. (Bug #1608883)
- For failure to create a new thread for the event scheduler, event execution, or new connection, no message was written to the error log. (Bug #16865959)
- If one connection changed its default database and simultaneously another connection executed SHOW PROCESSLIST, the second connection could access invalid memory when attempting to display the first connection's default database memory. (Bug #11765252)
- PURGE BINARY LOGS by design does not remove binary log files that are in use or active, but did not provide any notice when this occurred. (Bug #13727933)
- For some statements, memory leaks could result when the optimizer removed unneeded subquery clauses. (Bug #15875919)
- During shutdown, the server could attempt to lock an uninitialized mutex. (Bug #16016493)
- A prepared statement that used GROUP_CONCAT() and an ORDER BY clause that named multiple columns could cause the server to exit. (Bug #16075310)
- Performance Schema instrumentation was missing for slave worker threads. (Bug #16083949)
- STOP SLAVE could cause a deadlock when issued concurrently with a statement such as SHOW STATUS that retrieved the values for one or more of the status variables Slave_retried_transactions, Slave_heartbeat_period, Slave_received_heartbeats, Slave_last_heartbeat, or Slave_running. (Bug #16088188)
- A full-text query using Boolean mode could return zero results in some cases where the search term was a quoted phrase. (Bug #16206253)

- The optimizer's attempt to remove redundant subquery clauses raised an assertion when executing a prepared statement with a subquery in the ON clause of a join in a subquery. (Bug #16318585)
- GROUP_CONCAT unstable, crash in ITEM_SUM::CLEAN_UP_AFTER_REMOVAL. (Bug #16347450)
- Attempting to replace the default InnoDB full-text search (FTS) stopword list by creating an InnoDB table with the same structure as INFORMATION_SCHEMA.INNODB_FT_DEFAULT_STOPWORD would result in an error. (Bug #16373868)
- After the client thread on a slave performed a FLUSH TABLES WITH READ LOCK and was followed by some updates on the master, the slave hung when executing SHOW SLAVE STATUS. (Bug #16387720)
- When parsing a delimited search string such as "abc-def" in a full-text search, InnoDB now uses the same word delimiters as MyISAM. (Bug #16419661)
- Crash in FTS_AST_TERM_SET_WILDCARD. (Bug #16429306)
- SEGFAULT in FTS_AST_VISIT() for FTS RQG test. (Bug # 16435855)
- For debug builds, when the optimizer removed an Item_ref pointing to a subquery, it caused a server exit. (Bug #16509874)
- Full-text search on InnoDB tables failed on searches for literal phrases combined with + or - operators. (Bug #16516193)
- START SLAVE failed when the server was started with the options --master-info-repository=TABLE relay-log-info-repository=TABLE and with autocommit set to 0, together with --skip-slave-start. (Bug #16533802)
- Very large InnoDB full-text search (FTS) results could consume an excessive amount of memory. (Bug #16625973)
- In debug builds, an assertion could occur in OPT_CHECK_ORDER_BY when using binary directly in a search string, as binary may include NULL bytes and other non-meaningful characters. (Bug #16766016)
- For some statements, memory leaks could result when the optimizer removed unneeded subquery clauses. (Bug #16807641)
- It was possible to cause a deadlock after issuing FLUSH TABLES WITH READ LOCK by issuing STOP SLAVE in a new connection to the slave, then issuing SHOW SLAVE STATUS using the original connection. (Bug #16856735)
- GROUP_CONCAT() with an invalid separator could cause a server exit. (Bug #16870783)
- The server did excessive locking on the LOCK_active_mi and active_mi->rli->data_lock mutexes for any SHOW STATUS LIKE 'pattern' statement, even when the pattern did not match status variables that use those mutexes (Slave_heartbeat_period, Slave_last_heartbeat, Slave_received_heartbeats, Slave_retried_transactions, Slave_running). (Bug #16904035)
- A full-text search using the IN BOOLEAN MODE modifier would result in an assertion failure. (Bug #16927092)
- Full-text search on InnoDB tables failed on searches that used the + boolean operator. (Bug #17280122)
- 4-way deadlock: zombies, purging binlogs, show processlist, show binlogs. (Bug #17283409)
- When an SQL thread which was waiting for a commit lock was killed and restarted it caused a transaction to be skipped on slave. (Bug #17450876)
- An InnoDB full-text search failure would occur due to an "unended" token. The string and string length should be passed for string comparison. (Bug #17659310)
- Large numbers of partitioned InnoDB tables could consume much more memory when used in MySQL 5.6 or 5.7 than the memory used by the same tables used in previous releases of the MySQL Server. (Bug #17780517)
- For full-text queries, a failure to check that num_token is less than max_proximity_item could result in an assertion. (Bug #18233051)
- Certain queries for the INFORMATION_SCHEMA TABLES and COLUMNS tables could lead to excessive memory use when there were large numbers of empty InnoDB tables. (Bug #18592390)

- When committing a transaction, a flag is now used to check whether a thread has been created, rather than checking the thread itself, which uses more resources, particularly when running the server with master_info_repository=TABLE. (Bug #18684222)
- If a client thread on a slave executed FLUSH TABLES WITH READ LOCK while the master executed a DML, executing SHOW SLAVE STATUS in the same client became blocked, causing a deadlock. (Bug #19843808)
- Ordering by a GROUP_CONCAT() result could cause a server exit. (Bug #19880368)

Amazon Aurora MySQL Database Engine Updates: 2015-08-24

Version: 1.1

This update includes the following improvements:

- Replication stability improvements when replicating with a MySQL database (binlog replication). For information on Amazon Aurora MySQL replication with MySQL, see Replication with Amazon Aurora (p. 478).
- A 1 gigabyte (GB) limit on the size of the relay logs accumulated for an Amazon Aurora MySQL DB cluster that is a replication slave. This improves the file management for the Aurora DB clusters.
- Stability improvements in the areas of read ahead, recursive foreign-key relationships, and Aurora replication.
- Integration of MySQL bug fixes.
  - InnoDB databases with names beginning with a digit cause a full-text search (FTS) parser error. (Bug #17607956)
  - InnoDB full-text searches fail in databases whose names began with a digit. (Bug #17161372)
  - For InnoDB databases on Windows, the full-text search (FTS) object ID is not in the expected hexadecimal format. (Bug #16559254)
  - A code regression introduced in MySQL 5.6 negatively impacted DROP TABLE and ALTER TABLE performance. This could cause a performance drop between MySQL Server 5.5.x and 5.6.x. (Bug #16864741)
- Simplified logging to reduce the size of log files and the amount of storage that they require.

MySQL Bugs Fixed by Amazon Aurora MySQL Database Engine Updates

The following table identifies MySQL bugs that have been fixed by Amazon Aurora MySQL database engine updates, and which update they were fixed in.

Database engine update	Version	MySQL bugs fixed
Amazon Aurora MySQL Database Engine Updates: 2015-08-24 (p. 634)	1.1	• InnoDB databases with names beginning with a digit cause a full-text search (FTS) parser error. (Bug #17607956) • InnoDB full-text searches fail in databases whose names began with a digit. (Bug #17161372) • For InnoDB databases on Windows, the full-text search (FTS) object ID is not in the expected hexadecimal format. (Bug #16559254) • A code regression introduced in MySQL 5.6 negatively impacted DROP TABLE and ALTER TABLE performance. This could cause a

Database engine update	Version	MySQL bugs fixed
		performance drop between MySQL Server 5.5.x and 5.6.x. (Bug #16864741)
Amazon Aurora MySQL Database Engine Updates: 2015-10-16 (p. 632)	1.2, 1.3	• Killing a query inside innodb causes it to eventually crash with an assertion. (Bug #1608883) • For failure to create a new thread for the event scheduler, event execution, or new connection, no message was written to the error log. (Bug #16865959) • If one connection changed its default database and simultaneously another connection executed SHOW PROCESSLIST, the second connection could access invalid memory when attempting to display the first connection's default database memory. (Bug #11765252) • PURGE BINARY LOGS by design does not remove binary log files that are in use or active, but did not provide any notice when this occurred. (Bug #13727933) • For some statements, memory leaks could result when the optimizer removed unneeded subquery clauses. (Bug #15875919) • During shutdown, the server could attempt to lock an uninitialized mutex. (Bug #16016493) • A prepared statement that used GROUP_CONCAT() and an ORDER BY clause that named multiple columns could cause the server to exit. (Bug #16075310) • Performance Schema instrumentation was missing for slave worker threads. (Bug #16083949) • STOP SLAVE could cause a deadlock when issued concurrently with a statement such as SHOW STATUS that retrieved the values for one or more of the status variables Slave_retried_transactions, Slave_heartbeat_period, Slave_received_heartbeats, Slave_last_heartbeat, or Slave_running. (Bug #16088188) • A full-text query using Boolean mode could return zero results in some cases where the search term was a quoted phrase. (Bug #16206253) • The optimizer's attempt to remove redundant subquery clauses raised an assertion when executing a prepared statement with a subquery in the ON clause of a join in a subquery. (Bug #16318585) • GROUP_CONCAT unstable, crash in ITEM_SUM::CLEAN_UP_AFTER_REMOVAL. (Bug #16347450) • Attempting to replace the default InnoDB full-text search (FTS) stopword list by creating an InnoDB table with the same structure as INFORMATION_SCHEMA.INNODB_FT_DEFAULT_STOPWORD would result in an error. (Bug #16373868) • After the client thread on a slave performed a FLUSH TABLES WITH READ LOCK and was followed by some updates on the master, the slave hung when executing SHOW SLAVE STATUS. (Bug #16387720) • When parsing a delimited search string such as "abc-def" in a full-text search, InnoDB now uses the same word delimiters as MyISAM. (Bug #16419661)

Database engine update	Version	MySQL bugs fixed
		• Crash in FTS_AST_TERM_SET_WILDCARD. (Bug #16429306) • SEGFAULT in FTS_AST_VISIT() for FTS RQG test. (Bug # 16435855) • For debug builds, when the optimizer removed an Item_ref pointing to a subquery, it caused a server exit. (Bug #16509874) • Full-text search on InnoDB tables failed on searches for literal phrases combined with + or - operators. (Bug #16516193) • START SLAVE failed when the server was started with the options --master-info-repository=TABLE relay-log-info-repository=TABLE and with autocommit set to 0, together with --skip-slave-start. (Bug #16533802) • Very large InnoDB full-text search (FTS) results could consume an excessive amount of memory. (Bug #16625973) • In debug builds, an assertion could occur in OPT_CHECK_ORDER_BY when using binary directly in a search string, as binary may include NULL bytes and other non-meaningful characters. (Bug #16766016) • For some statements, memory leaks could result when the optimizer removed unneeded subquery clauses. (Bug #16807641) • It was possible to cause a deadlock after issuing FLUSH TABLES WITH READ LOCK by issuing STOP SLAVE in a new connection to the slave, then issuing SHOW SLAVE STATUS using the original connection. (Bug #16856735) • GROUP_CONCAT() with an invalid separator could cause a server exit. (Bug #16870783) • The server did excessive locking on the LOCK_active_mi and active_mi->rli->data_lock mutexes for any SHOW STATUS LIKE 'pattern' statement, even when the pattern did not match status variables that use those mutexes (Slave_heartbeat_period, Slave_last_heartbeat, Slave_received_heartbeats, Slave_retried_transactions, Slave_running). (Bug #16904035) • A full-text search using the IN BOOLEAN MODE modifier would result in an assertion failure. (Bug #16927092) • Full-text search on InnoDB tables failed on searches that used the + boolean operator. (Bug #17280122) • 4-way deadlock: zombies, purging binlogs, show processlist, show binlogs. (Bug #17283409) • When an SQL thread which was waiting for a commit lock was killed and restarted it caused a transaction to be skipped on slave. (Bug #17450876) • An InnoDB full-text search failure would occur due to an "unended" token. The string and string length should be passed for string comparison. (Bug #17659310) • Large numbers of partitioned InnoDB tables could consume much more memory when used in MySQL 5.6 or 5.7 than the memory used by the same tables used in previous releases of the MySQL Server. (Bug #17780517) • For full-text queries, a failure to check that num_token is less than max_proximity_item could result in an assertion. (Bug #18233051)

Database engine update	Version	MySQL bugs fixed
		• Certain queries for the INFORMATION_SCHEMA TABLES and COLUMNS tables could lead to excessive memory use when there were large numbers of empty InnoDB tables. (Bug #18592390) • When committing a transaction, a flag is now used to check whether a thread has been created, rather than checking the thread itself, which uses more resources, particularly when running the server with master_info_repository=TABLE. (Bug #18684222) • If a client thread on a slave executed FLUSH TABLES WITH READ LOCK while the master executed a DML, executing SHOW SLAVE STATUS in the same client became blocked, causing a deadlock. (Bug #19843808) • Ordering by a GROUP_CONCAT() result could cause a server exit. (Bug #19880368)
Amazon Aurora MySQL Database Engine Updates: 2015-12-03 (p. 630)	1.4	• SEGV in FTSPARSE(). (Bug #16446108) • InnoDB data dictionary is not updated while renaming the column. (Bug #19465984) • FTS crash after renaming table to different database. (Bug #16834860) • Failed preparing of trigger on truncated tables cause error 1054. (Bug #18596756) • Metadata changes might cause problems with trigger execution. (Bug #18684393) • Materialization is not chosen for long UTF8 VARCHAR field. (Bug #17566396) • Poor execution plan when ORDER BY with limit X. (Bug #16697792) • Backport bug #11765744 TO 5.1, 5.5 AND 5.6. (Bug #17083851) • Mutex issue in SQL/SQL_SHOW.CC resulting in SIG6. Source likely FILL_VARIABLES. (Bug #20788853) • Backport bug #18008907 to 5.5+ versions. (Bug #18903155) • Adapt fix for a stack overflow error in MySQL 5.7. (Bug #19678930)
Amazon Aurora MySQL Database Engine Updates: 2016-01-11 (p. 630)	1.5	• Addressed incomplete fix in MySQL full text search affecting tables where the database name begins with a digit. (Port Bug #17607956)

Database engine update	Version	MySQL bugs fixed
Amazon Aurora MySQL Database Engine Updates: 2016-04-06 (p. 628)	1.6	• BACKPORT BUG#18694052 FIX FOR ASSERTION `! M_ORDERED_REC_BUFFER' FAILED TO 5.6 (Port Bug #18305270) • SEGV IN MEMCPY(), HA_PARTITION::POSITION (Port Bug # 18383840) • WRONG RESULTS WITH PARTITIONING,INDEX_MERGE AND NO PK (Port Bug # 18167648) • FLUSH TABLES FOR EXPORT: ASSERTION IN HA_PARTITION::EXTRA (Port Bug # 16943907) • SERVER CRASH IN VIRTUAL HA_ROWS HANDLER::MULTI_RANGE_READ_INFO_CONST (Port Bug # 16164031) • RANGE OPTIMIZER CRASHES IN SEL_ARG::RB_INSERT() (Port Bug # 16241773)
Amazon Aurora MySQL Database Engine Updates: 2016-06-01 (p. 628)	1.6.5	• SLAVE CAN'T CONTINUE REPLICATION AFTER MASTER'S CRASH RECOVERY (Port Bug #17632285)
Amazon Aurora MySQL Database Engine Updates: 2016-08-30 (p. 627)	1.7	• Improve scalability by partitioning LOCK_grant lock. (Port WL #8355) • Opening cursor on SELECT in stored procedure causes segfault. (Port Bug#16499751) • MySQL gives the wrong result with some special usage. (Bug #11751794) • Crash in GET_SEL_ARG_FOR_KEYPART – caused by patch for bug #11751794. (Bug #16208709) • Wrong results for a simple query with GROUP BY. (Bug #17909656) • Extra rows on semijoin query with range predicates. (Bug #16221623) • Adding an ORDER BY clause following an IN subquery could cause duplicate rows to be returned. (Bug #16308085) • Crash with explain for a query with loose scan for GROUP BY, MyISAM. (Bug #16222245) • Loose index scan with quoated int predicate returns random data. (Bug #16394084) • If the optimizer was using a loose index scan, the server could exit while attempting to create a temporary table. (Bug #16436567) • COUNT(DISTINCT) should not count NULL values, but they were counted when the optimizer used loose index scan. (Bug #17222452) • If a query had both MIN()/MAX() and aggregate_function(DISTINCT) (for example, SUM(DISTINCT)) and was executed using loose index scan, the result values of MIN()/MAX() were set improperly. (Bug #17217128)

Database engine update	Version	MySQL bugs fixed
Amazon Aurora MySQL Database Engine Updates: 2016-09-20 (p. 627)	1.7.1	
Amazon Aurora MySQL Database Engine Updates: 2016-10-18 (p. 625)	1.8	• When dropping all indexes on a column with multiple indexes, InnoDB failed to block a DROP INDEX operation when a foreign key constraint requires an index. (Bug #16896810) • Solve add foreign key constraint crash. (Bug #16413976) • Fixed a crash when fetching a cursor in a stored procedure, and analyzing or flushing the table at the same time. (Bug # 18158639) • Fixed an auto-increment bug when a user alters a table to change the AUTO_INCREMENT value to less than the maximum auto-increment column value. (Bug # 16310273)
Amazon Aurora MySQL Database Engine Updates: 2016-10-26 (p. 625)	1.8.1	• OpenSSL changed the Diffie-Hellman key length parameters due to the LogJam issue. (Bug #18367167)
Amazon Aurora MySQL Database Engine Updates: 2016-11-10 (p. 624)	1.9.0, 1.9.1	
Amazon Aurora MySQL Database Engine Updates: 2016-12-14 (p. 623)	1.10	• UNION of derived tables returns wrong results with '1=0/false'-clauses. (Bug #69471) • Server crashes in ITEM_FUNC_GROUP_CONCAT::FIX_FIELDS on 2nd execution of stored procedure. (Bug #20755389) • Avoid MySQL queries from stalling for too long during FTS cache sync to disk by offloading the cache sync task to a separate thread, as soon as the cache size crosses 10% of the total size. (Bugs #22516559, #73816)
Amazon Aurora MySQL Database Engine Updates: 2017-01-12 (p. 622)	1.10.1	
Amazon Aurora MySQL Database Engine Updates: 2017-02-23 (p. 620)	1.11	• Running ALTER table DROP foreign key simultaneously with another DROP operation causes the table to disappear. (Bug #16095573) • Some INFORMATION_SCHEMA queries that used ORDER BY did not use a filesort optimization as they did previously. (Bug #16423536) • FOUND_ROWS () returns the wrong count of rows on a table. (Bug #68458) • The server fails instead of giving an error when too many temp tables are open. (Bug #18948649)

Database engine update	Version	MySQL bugs fixed
Amazon Aurora MySQL Database Engine Updates: 2017-04-05 (p. 619)	1.12	• Reloading a table that was evicted while empty caused an AUTO_INCREMENT value to be reset. (Bug #21454472, Bug #77743) • An index record was not found on rollback due to inconsistencies in the purge_node_t structure. The inconsistency resulted in warnings and error messages such as “error in sec index entry update”, “unable to purge a record”, and “tried to purge sec index entry not marked for deletion”. (Bug #19138298, Bug #70214, Bug #21126772, Bug #21065746) • Wrong stack size calculation for qsort operation leads to stack overflow. (Bug #73979) • Record not found in an index upon rollback. (Bug #70214, Bug #72419) • ALTER TABLE add column TIMESTAMP on update CURRENT_TIMESTAMP inserts ZERO-datas (Bug #17392)
Amazon Aurora MySQL Database Engine Updates: 2017-05-15 (p. 617)	1.13	• Reloading a table that was evicted while empty caused an AUTO_INCREMENT value to be reset. (Bug #21454472, Bug #77743) • An index record was not found on rollback due to inconsistencies in the purge_node_t structure. The inconsistency resulted in warnings and error messages such as “error in sec index entry update”, “unable to purge a record”, and “tried to purge sec index entry not marked for deletion”. (Bug #19138298, Bug #70214, Bug #21126772, Bug #21065746) • Wrong stack size calculation for qsort operation leads to stack overflow. (Bug #73979) • Record not found in an index upon rollback. (Bug #70214, Bug #72419) • ALTER TABLE add column TIMESTAMP on update CURRENT_TIMESTAMP inserts ZERO-datas (Bug #17392)
Amazon Aurora MySQL Database Engine Updates: 2017-08-07 (p. 616)	1.14	A full-text search combined with derived tables (subqueries in the `FROM` clause) caused a server exit. Now, if a full-text operation depends on a derived table, the server produces an error indicating that a full-text search cannot be done on a materialized table. (Bug #68751, Bug #16539903)

Working with Amazon Aurora PostgreSQL

Amazon Aurora PostgreSQL is a fully managed, PostgreSQL-compatible, relational database engine that combines the speed and reliability of high-end commercial databases with the simplicity and cost-effectiveness of open-source databases. Aurora PostgreSQL is a drop-in replacement for PostgreSQL and makes it simple and cost-effective to set up, operate, and scale your new and existing PostgreSQL deployments, thus freeing you to focus on your business and applications. Amazon RDS provides administration for Aurora by handling routine database tasks such as provisioning, patching, backup, recovery, failure detection, and repair. Amazon RDS also provides push-button migration tools to convert your existing Amazon RDS for PostgreSQL applications to Aurora PostgreSQL.

Aurora PostgreSQL can work with many industry standards. For example, you can use Aurora PostgreSQL databases to build HIPAA-compliant applications and to store healthcare related information, including

protected health information (PHI), under an executed Business Associate Agreement (BAA) with AWS. For more information about BAAs with AWS, see HIPAA Compliance.

Availability for Amazon Aurora PostgreSQL

The following table shows the regions where Aurora PostgreSQL is currently available.

Region	Console Link
US East (Ohio)	https://console.aws.amazon.com/rds/home?region=us-east-2
US East (N. Virginia)	https://console.aws.amazon.com/rds/home?region=us-east-1
US West (Oregon)	https://console.aws.amazon.com/rds/home?region=us-west-2
Asia Pacific (Mumbai)	https://console.aws.amazon.com/rds/home?region=ap-south-1
Asia Pacific (Sydney)	https://console.aws.amazon.com/rds/home?region=ap-southeast-2
Canada (Central)	https://console.aws.amazon.com/rds/home?region=ca-central-1
EU (Frankfurt)	https://console.aws.amazon.com/rds/home?region=eu-central-1
EU (Ireland)	https://console.aws.amazon.com/rds/home?region=eu-west-1

Comparison of Amazon Aurora PostgreSQL and Amazon RDS for PostgreSQL

Although Aurora instances are compatible with PostgreSQL client applications, Aurora has advantages over PostgreSQL and also limitations to the PostgreSQL features that Aurora supports. This functionality can influence your decision about whether Amazon Aurora PostgreSQL or PostgreSQL on Amazon RDS is the best cloud database for your solution. The following table shows the differences between Amazon Aurora PostgreSQL and Amazon RDS for PostgreSQL.

Feature	Amazon Aurora PostgreSQL	Amazon RDS for PostgreSQL
`wal2json` plugin support	Currently, the plugin is not supported.	The plugin is supported.

Migrating Data to Amazon Aurora PostgreSQL

You have several options for migrating data from your existing database to an Amazon Aurora PostgreSQL DB cluster. Your migration options also depend on the database that you are migrating from and the size of the data that you are migrating. The following table describes your options.

Migrating From	Solution
An RDS PostgreSQL DB instance	• You can migrate data directly from an Amazon RDS PostgreSQL DB snapshot to an Amazon Aurora PostgreSQL DB cluster. For more information, see Migrating Data from a PostgreSQL DB Instance to an Amazon Aurora PostgreSQL DB Cluster by Using a DB Snapshot (p. 642).
A database that is not PostgreSQL-compatible	You can use AWS Database Migration Service (AWS DMS) to migrate data from a database that is not PostgreSQL-compatible. For more information on AWS DMS, see What Is AWS Database Migration Service?

Migrating Data from a PostgreSQL DB Instance to an Amazon Aurora PostgreSQL DB Cluster by Using a DB Snapshot

You can migrate (copy) data to an Amazon Aurora PostgreSQL DB cluster from an Amazon RDS PostgreSQL DB snapshot, as described following.

Migrating an RDS PostgreSQL Snapshot to Aurora

You can migrate a DB snapshot of an Amazon RDS PostgreSQL DB instance to create an Aurora PostgreSQL DB cluster. The new Aurora PostgreSQL DB cluster is populated with the data from the original Amazon RDS PostgreSQL DB instance. The DB snapshot must have been made from an Amazon RDS DB instance running PostgreSQL version 9.6.1 or later.

You can migrate either a manual or automated DB snapshot. After the DB cluster is created, you can then create optional Aurora Replicas.

The general steps you must take are as follows:

1. Determine the amount of space to provision for your Aurora PostgreSQL DB cluster.
2. Use the console to create the snapshot in the region where the Amazon RDS PostgreSQL 9.6.1 instance is located. For information about creating a DB snapshot, see Creating a DB Snapshot.
3. If the DB snapshot is not in the same region as your DB cluster, use the Amazon RDS console to copy the DB snapshot to that region. For information about copying a DB snapshot, see Copying a DB Snapshot.
4. Use the console to migrate the DB snapshot and create an Aurora PostgreSQL DB cluster with the same databases as the original PostgreSQL DB instance.

Warning
Amazon RDS limits each AWS account to one snapshot copy into each AWS Region at a time.

AWS Management Console

You can migrate a DB snapshot of an Amazon RDS PostgreSQL DB instance to create an Aurora PostgreSQL DB cluster. The new Aurora PostgreSQL DB cluster will be populated with the data from the original Amazon RDS PostgreSQL DB instance. The DB snapshot must have been made from an Amazon RDS DB instance running PostgreSQL 9.6.1 or later and must not be encrypted. For information about creating a DB snapshot, see Creating a DB Snapshot.

If the DB snapshot is not in the AWS Region where you want to locate your data, use the Amazon RDS console to copy the DB snapshot to that AWS Region. For information about copying a DB snapshot, see Copying a DB Snapshot.

When you migrate the DB snapshot by using the AWS Management Console, the console takes the actions necessary to create both the DB cluster and the primary instance.

You can also choose for your new Aurora PostgreSQL DB cluster to be encrypted at rest using an AWS Key Management Service (AWS KMS) encryption key. This option is available only for unencrypted DB snapshots.

To migrate a PostgreSQL DB snapshot by using the AWS Management Console

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. Choose **Snapshots**.
3. On the **Snapshots** page, choose the snapshot that you want to migrate into an Aurora PostgreSQL DB cluster.
4. Choose **Migrate Database**.
5. Set the following values on the **Migrate Database** page:

   - **DB Instance Class**: Select a DB instance class that has the required storage and capacity for your database, for example `db.r4.large`. Aurora cluster volumes automatically grow as the amount of data in your database increases, up to a maximum size of 64 terabytes (TB). So you only need to select a DB instance class that meets your current storage requirements. For more information, see Amazon Aurora Storage (p. 432).
   - **DB Instance Identifier**: Type a name for the DB cluster that is unique for your account in the region you selected. This identifier is used in the endpoint addresses for the instances in your DB cluster. You might choose to add some intelligence to the name, such as including the region and DB engine you selected, for example **`aurora-cluster1`**.

     The DB instance identifier has the following constraints:

     - It must contain from 1 to 63 alphanumeric characters or hyphens.
     - Its first character must be a letter.
     - It cannot end with a hyphen or contain two consecutive hyphens.
     - It must be unique for all DB instances per AWS account, per AWS Region.
   - **VPC**: If you have an existing VPC, then you can use that VPC with your Aurora PostgreSQL DB cluster by selecting your VPC identifier, for example `vpc-a464d1c1`. For information on using an existing VPC, see How to Create a VPC for Use with Amazon Aurora (p. 452).

     Otherwise, you can choose to have Amazon RDS create a VPC for you by selecting **Create a new VPC**.
   - **Subnet Group**: If you have an existing subnet group, then you can use that subnet group with your Aurora PostgreSQL DB cluster by selecting your subnet group identifier, for example `gs-subnet-group1`.

     Otherwise, you can choose to have Amazon RDS create a subnet group for you by selecting **Create a new subnet group**.
   - **Publicly Accessible**: Select **No** to specify that instances in your DB cluster can only be accessed by resources inside of your VPC. Select **Yes** to specify that instances in your DB cluster can be accessed by resources on the public network. The default is **Yes**.

     Note
     Your production DB cluster might not need to be in a public subnet, because only your application servers will require access to your DB cluster. If your DB cluster doesn't need to be in a public subnet, set **Publicly Accessible** to **No**.
   - **Availability Zone**: Select the Availability Zone to host the primary instance for your Aurora PostgreSQL DB cluster. To have Amazon RDS select an Availability Zone for you, select **No Preference**.

- **Database Port**: Type the default port to be used when connecting to instances in the Aurora PostgreSQL DB cluster. The default is `5432`.

  Note
  You might be behind a corporate firewall that doesn't allow access to default ports such as the PostgreSQL default port, 5432. In this case, provide a port value that your corporate firewall allows. Remember that port value later when you connect to the Aurora PostgreSQL DB cluster.
- **Enable Encryption**: Choose **Yes** for your new Aurora PostgreSQL DB cluster to be encrypted at rest. If you choose **Yes**, you will be required to choose an AWS KMS encryption key as the **Master Key** value.
- **Auto Minor Version Upgrade**: Select **Yes** if you want to enable your Aurora PostgreSQL DB cluster to receive minor PostgreSQL DB engine version upgrades automatically when they become available.

  The **Auto Minor Version Upgrade** option only applies to upgrades to PostgreSQL minor engine versions for your Amazon Aurora PostgreSQL DB cluster. It doesn't apply to regular patches applied to maintain system stability.

6. Choose **Migrate** to migrate your DB snapshot.
7. Choose **Instances**, and then choose the arrow icon to show the DB cluster details and monitor the progress of the migration. On the details page, you will find the cluster endpoint used to connect to the primary instance of the DB cluster. For more information on connecting to an Aurora PostgreSQL DB cluster, see Connecting to an Amazon Aurora DB Cluster (p. 457).

Related Topics

- Amazon Aurora on Amazon RDS (p. 428)
- Migrating Data to an Amazon Aurora DB Cluster (p. 466)

Managing Amazon Aurora PostgreSQL

The following sections discuss managing performance and scaling for an Amazon Aurora PostgreSQL DB cluster.

Managing Performance and Scaling for Amazon Aurora PostgreSQL

Scaling Aurora PostgreSQL DB Instances

You can scale Aurora PostgreSQL DB instances in two ways, instance scaling and read scaling. For more information about read scaling, see Read Scaling (p. 467).

You can scale your Aurora PostgreSQL DB cluster by modifying the DB instance class for each DB instance in the DB cluster. Aurora PostgreSQL supports several DB instance classes optimized for Aurora. The following table describes the specifications of the DB instance classes supported by Aurora PostgreSQL.

Instance Class	vCPU	Memory (GiB)
db.r4.large	2	15.25
db.r4.xlarge	4	30.5

Instance Class	vCPU	Memory (GiB)
db.r4.2xlarge	8	61
db.r4.4xlarge	16	122
db.r4.8xlarge	32	244
db.r4.16xlarge	64	488

Maximum Connections to an Aurora PostgreSQL DB Instance

The maximum number of connections allowed to an Aurora PostgreSQL DB instance is determined by the `max_connections` parameter in the instance-level parameter group for the DB instance. By default, this value is set to the following equation (the log function represents log base 2):

`LEAST({DBInstanceClassMemory/9531392},2500).`

Setting the `max_connections` parameter to this equation makes sure that the number of allowed connection scales well with the size of the instance. For example, suppose your DB instance class is db.r4.large, which has 15.25 gibibytes (GiB) of memory. Then the maximum connections allowed is 1660, as shown in the following equation:

```
LEAST( (15.25 * 1000000000) / 9531392 ), 2500) = 1660
```

The following table lists the resulting default value of `max_connections` for each DB instance class available to Aurora PostgreSQL. You can increase the maximum number of connections to your Aurora PostgreSQL DB instance by scaling the instance up to a DB instance class with more memory, or by setting a larger value for the `max_connections` parameter, up to 262,143.

Instance Class	max_connec Default Value		
db.r4.large	1660		
db.r4.xlarge	2500		
db.r4.2xlarge	2500		
db.r4.4xlarge	2500		
db.r4.8xlarge	2500		
db.r4.16xlarge	2500		

Replication with Amazon Aurora PostgreSQL

Using Aurora Replicas

Aurora Replicas are independent endpoints in an Aurora DB cluster, best used for scaling read operations and increasing availability. Up to 15 Aurora Replicas can be distributed across the Availability Zones that a DB cluster spans within an AWS Region. Although the DB cluster volume is made up of multiple copies of the data for the DB cluster, the data in the cluster volume is represented as a single, logical volume to the primary instance and to Aurora Replicas in the DB cluster. For more information about Aurora Replicas, see Aurora Replicas (p. 478).

Aurora Replicas work well for read scaling because they are fully dedicated to read operations on your cluster volume. Write operations are managed by the primary instance. Because the cluster volume is shared among all instances in your Aurora PostgreSQL DB cluster, no additional work is required to replicate a copy of the data for each Aurora Replica. In contrast, PostgreSQL Read Replicas must apply, on a single thread, all write operations from the master DB instance to their local data store. This limitation can affect the ability of PostgreSQL Read Replicas to support large volumes of read traffic.

Replication Options for Amazon Aurora PostgreSQL

Note
Rebooting the primary instance of an Amazon Aurora DB cluster also automatically reboots the Aurora Replicas for that DB cluster, in order to re-establish an entry point that guarantees read/write consistency across the DB cluster.

Monitoring Amazon Aurora PostgreSQL Replication

Read scaling and high availability depend on minimal lag time. You can monitor how far an Aurora Replica is lagging behind the primary instance of your Aurora PostgreSQL DB cluster by monitoring the Amazon CloudWatch `ReplicaLag` metric. Because Aurora Replicas read from the same cluster volume as the primary instance, the `ReplicaLag` metric has a different meaning for an Aurora PostgreSQL DB cluster. The `ReplicaLag` metric for an Aurora Replica indicates the lag for the page cache of the Aurora Replica compared to that of the primary instance.

For more information on monitoring RDS instances and CloudWatch metrics, see Monitoring Amazon RDS (p. 245).

Security with Amazon Aurora PostgreSQL

Security for Amazon Aurora PostgreSQL is managed at three levels:

- To control who can perform Amazon RDS management actions on Aurora DB clusters and DB instances, you use AWS Identity and Access Management (IAM). When you connect to AWS using IAM credentials, your IAM account must have IAM policies that grant the permissions required to perform Amazon RDS management operations. For more information, see Authentication and Access Control for Amazon RDS (p. 327).

  If you are using an IAM account to access the Amazon RDS console, you must first log on to the AWS Management Console with your IAM account, and then go to the Amazon RDS console at https://console.aws.amazon.com/rds.
- Aurora DB clusters must be created in an Amazon Virtual Private Cloud (VPC). To control which devices and Amazon EC2 instances can open connections to the endpoint and port of the DB instance for Aurora DB clusters in a VPC, you use a VPC security group. These endpoint and port connections can be made using Secure Sockets Layer (SSL). In addition, firewall rules at your company can control whether devices running at your company can open connections to a DB instance. For more information on VPCs, see Amazon Virtual Private Cloud (VPCs) and Amazon RDS (p. 390).
- To authenticate login and permissions for an Amazon Aurora DB cluster, you can take the same approach as with a stand-alone instance of PostgreSQL.

  Commands such as `CREATE ROLE`, `ALTER ROLE`, `GRANT`, and `REVOKE` work just as they do in on-premises databases, as does directly modifying database schema tables. For more information, see Client Authentication in the PostgreSQL documentation.

When you create an Amazon Aurora PostgreSQL DB instance, the master user has the following default privileges:

- `LOGIN`

- `NOSUPERUSER`
- `INHERIT`
- `CREATEDB`
- `CREATEROLE`
- `NOREPLICATION`
- `VALID UNTIL 'infinity'`

To provide management services for each DB cluster, the `rdsadmin` user is created when the DB cluster is created. Attempting to drop, rename, change the password, or change privileges for the `rdsadmin` account will result in an error.

Integrating Amazon Aurora PostgreSQL with Other AWS Services

Amazon Aurora integrates with other AWS services so that you can extend your Aurora PostgreSQL DB cluster to use additional capabilities in the AWS Cloud. Your Aurora PostgreSQL DB cluster can use AWS services to do the following:

- Quickly collect, view, and assess performance for your Aurora PostgreSQL DB instances with Amazon RDS Performance Insights. Performance Insights expands on existing Amazon RDS monitoring features to illustrate your database's performance and help you analyze any issues that affect it. With the Performance Insights dashboard, you can visualize the database load and filter the load by waits, SQL statements, hosts, or users.

  For more information about Performance Insights, see Preview: Using Amazon Performance Insights (p. 269).

Related Topics

- Amazon Aurora on Amazon RDS (p. 428)

Best Practices with Amazon Aurora PostgreSQL

This topic includes information on best practices and options for using or migrating data to an Amazon Aurora PostgreSQL DB cluster.

Fast Failover with Amazon Aurora PostgreSQL

There are several things you can do to make a failover perform faster with Aurora PostgreSQL. This section discusses each of the following ways:

- Aggressively set TCP keepalives to ensure that longer running queries that are waiting for a server response will be killed before the read timeout expires in the event of a failure.
- Set the Java DNS caching timeouts aggressively to ensure the Aurora Read-Only Endpoint can properly cycle through read-only nodes on subsequent connection attempts.
- Set the timeout variables used in the JDBC connection string as low as possible. Use separate connection objects for short and long running queries.
- Use the provided read and write Aurora endpoints to establish a connection to the cluster.
- Use RDS APIs to test application response on server side failures and use a packet dropping tool to test application response for client-side failures.

Setting TCP Keepalives Parameters

The TCP keepalive process is simple: when you set up a TCP connection, you associate a set of timers. When the keepalive timer reaches zero, you send a keepalive probe packet. If you receive a reply to your keepalive probe, you can assume that the connection is still up and running.

Enabling TCP keepalive parameters and setting them aggressively ensures that if your client is no longer able to connect to the database, then any active connections are quickly closed. This action allows the application to react appropriately, such as by picking a new host to connect to.

The following TCP keepalive parameters need to be set:

- `tcp_keepalive_time` controls the time, in seconds, after which a keepalive packet is sent when no data has been sent by the socket (ACKs are not considered data). We recommend the following setting:

  ```
  tcp_keepalive_time = 1
  ```
- `tcp_keepalive_intvl` controls the time, in seconds, between sending subsequent keepalive packets after the initial packet is sent (set using the `tcp_keepalive_time` parameter). We recommend the following setting:

  ```
  tcp_keepalive_intvl = 1
  ```
- `tcp_keepalive_probes` is the number of unacknowledged keepalive probes that occur before the application is notified. We recommend the following setting:

  ```
  tcp_keepalive_probes = 5
  ```

These settings should notify the application within five seconds when the database stops responding. A higher *tcp_keepalive_probes* value can be set if keepalive packets are often dropped within the application's network. This subsequently increases the time it takes to detect an actual failure, but allows for more buffer in less reliable networks.

Setting TCP keepalive parameters on Linux

1. When testing how to configure the TCP keepalive parameters, we recommend doing so via the command line with the following commands: This suggested configuration is system wide, meaning that it affects all other applications that create sockets with the SO_KEEPALIVE option on.

   ```
   sudo sysctl net.ipv4.tcp_keepalive_time=1
   sudo sysctl net.ipv4.tcp_keepalive_intvl=1
   sudo sysctl net.ipv4.tcp_keepalive_probes=5
   ```
2. Once you've found a configuration that works for your application, these settings must be persisted by adding the following lines (including any changes you made) to */etc/sysctl.conf*:

   ```
   tcp_keepalive_time = 1
   tcp_keepalive_intvl = 1
   tcp_keepalive_probes = 5
   ```

For information on setting TCP keepalive parameters on Windows, see Things You May Want to Know About TCP Keepalive.

Configuring Your Application for Fast Failover

This section discusses several Aurora PostgreSQL specific configuration changes you can make. Documentation for general setup and configuration of the JDBC driver is available from the PostgreSQL JDBC site.

Reducing DNS Cache Timeouts

When your application tries to establish a connectionafter a failover, the new Aurora PostgreSQL writer will be a previous reader, which can be found using the Aurora **read only** endpoint before DNS updates have fully propagated. Setting the java DNS TTL to a low value helps cycle between reader nodes on subsequent connection attempts.

```
// Sets internal TTL to match the Aurora RO Endpoint TTL
java.security.Security.setProperty("networkaddress.cache.ttl" , "1");
// If the lookup fails, default to something like small to retry
java.security.Security.setProperty("networkaddress.cache.negative.ttl" , "3");
```

Setting an Aurora PostgreSQL Connection String for Fast Failover

To make use of Aurora PostgreSQL fast failover, your application's connection string should have a list of hosts (highlighted in bold in the following example) instead of just a single host. Here is an example connection string you could use to connect to an Aurora PostgreSQL cluster:

```
jdbc:postgresql://myauroracluster.cluster-c9bfei4hjlrd.us-east-1-
beta.rds.amazonaws.com:5432,
myauroracluster.cluster-ro-c9bfei4hjlrd.us-east-1-beta.rds.amazonaws.com:5432
/postgres?user=<masteruser>&password=<masterpw>&loginTimeout=2
&connectTimeout=2&cancelSignalTimeout=2&socketTimeout=60
&tcpKeepAlive=true&targetServerType=master&loadBalanceHosts=true
```

For best availability and to avoid a dependency on the RDS API, the best option for connecting is to maintain a file with a host string that your application reads from when you establish a connection to the database. This host string would have all the Aurora endpoints available for the cluster. For more information about Aurora endpoints, see Aurora Endpoints (p. 431). For example, you could store the endpoints in a file locally like the following:

```
myauroracluster.cluster-c9bfei4hjlrd.us-east-1-beta.rds.amazonaws.com:5432,
myauroracluster.cluster-ro-c9bfei4hjlrd.us-east-1-beta.rds.amazonaws.com:5432
```

Your application would read from this file to populate the host section of the JDBC connection string. Renaming the DB Cluster causes these endpoints to change; ensure that your application handles that event should it occur.

Another option is to use a list of DB instance nodes:

```
my-node1.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com:5432,
my-node2.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com:5432,
my-node3.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com:5432,
my-node4.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com:5432
```

The benefit of this approach is that the PostgreSQL JDBC connection driver will loop through all nodes on this list to find a valid connection, whereas when using the Aurora endpoints only two nodes will be tried per connection attempt. The downside of using DB instance nodes is that if you add or remove nodes from your cluster and the list of instance endpoints becomes stale, the connection driver may never find the correct host to connect to.

Setting the following parameters aggressively helps ensure that your application doesn't wait too long to connect to any one host.

- `loginTimeout` - Controls how long your application waits to login to the database *after* a socket connection has been established.
- `connectTimeout` - Controls how long the socket waits to establish a connection to the database.

Other application parameters can be modified to speed up the connection process, depending on how aggressive you want your application to be.

- `cancelSignalTimeout` - In some applications, you may want to send a "best effort" cancel signal on a query that has timed out. If this cancel signal is in your failover path, you should consider setting it aggressively to avoid sending this signal to a dead host.
- `socketTimeout` - This parameter controls how long the socket waits for read operations. This parameter can be used as a global "query timeout" to ensure no query waits longer than this value. A good practice is to have one connection handler that runs short lived queries and sets this value lower, and to have another connection handler for long running queries with this value set much higher. Then, you can rely on TCP keepalive parameters to kill long running queries if the server goes down.
- `tcpKeepAlive` - Enable this parameter to ensure the TCP keepalive parameters that you set are respected.
- `targetServerType`- This parameter can be used to control whether the driver connects to a read (slave) or write (master) node. Possible values are: `any`, `master`, `slave` and `preferSlave`. The `preferSlave` value attempts to establish a connection to a reader first but falls back and connects to the writer if no reader connection can be established.
- `loadBalanceHosts` - When set to `true`, this parameter has the application connect to a random host chosen from a list of candidate hosts.

Other Options for Obtaining The Host String

You can get the host string from several sources, including the `aurora_replica_status` function and by using the Amazon RDS API.

Your application can connect to any DB instance in the DB Cluster and query the `aurora_replica_status` function to determine who the writer of the cluster is, or to find any other reader nodes in the cluster. You can use this function to reduce the amount of time it takes to find a host to connect to, though in certain scenarios the `aurora_replica_status` function may show out of date or incomplete information in certain network failure scenarios.

A good way to ensure your application can find a node to connect to is to attempt to connect to the **cluster writerendpoint** and then the **cluster readerendpoint** until you can establish a readable connection. These endpoints do not change unless you rename your DB Cluster, and thus can generally be left as static members of your application or stored in a resource file that your application reads from.

Once you establish a connection using one of these endpoints, you can call the `aurora_replica_status` function to get information about the rest of the cluster. For example, the following command retrieves information with the `aurora_replica_status` function.

```
postgres=> select server_id, session_id, highest_lsn_rcvd,
cur_replay_latency_in_usec, now(), last_update_timestamp from
aurora_replica_status();
    server_id     |              session_id              |
vdl    | highest_lsn_rcvd | cur_replay_latency |
now               |    last_update_time
--------------------------------------+--------------------------
-----------+-----------+------------------+--------------------+-
------------------------------+-------
         mynode-1 | 3e3c5044-02e2-11e7-b70d-95172646d6ca |
594220999 |        594221001 |             201421 | 2017-03-07
19:50:24.695322+00 | 2017-03-07 19:50:23+00
         mynode-2 | 1efdd188-02e4-11e7-becd-f12d7c88a28a |
594220999 |        594221001 |             201350 | 2017-03-07
19:50:24.695322+00 | 2017-03-07 19:50:23+00
         mynode-3 |                     MASTER_SESSION_ID |
594220999 |                  |                    | 2017-03-07
19:50:24.695322+00 | 2017-03-07 19:50:23+00
```

```
(3 rows)
```

So for example, the hosts section of your connection string could start with both the writer and reader cluster endpoints:

```
myauroracluster.cluster-c9bfei4hjlrd.us-east-1-beta.rds.amazonaws.com:5432,
myauroracluster.cluster-ro-c9bfei4hjlrd.us-east-1-beta.rds.amazonaws.com:5432
```

In this scenario, your application would attempt to establish a connection to any node type, master or slave. Once connected, a good practice is to first examine the read-write status of the node by querying for the result of the command `SHOW transaction_read_only`.

If the return value of the query is `OFF`, then you've successfully connected to the master node. If the return value is `ON`, and your application requires a read-write connection, you can then call the `aurora_replica_status` function to determine the `server_id` that has `session_id='MASTER_SESSION_ID'`. This function gives you the name of the master node. You can use this in conjunction with the 'endpointPostfix' described below.

One thing to watch out for is when you connect to a replica that has data that has become stale. When this happens, the `aurora_replica_status` function may show out-of-date information. A threshold for staleness can be set at the application level and examined by looking at the difference between the server time and the last_update_time. In general, your application should be sure to avoid flip-flopping between two hosts due to conflicting information returned by the aurora_replica_status function. That is, your application should err on the side of trying all known hosts first instead of blindly following the data returned by the aurora_replica_status function.

API

You can programatically find the list of instances by using the AWS Java SDK, specifically the DescribeDbClusters API. Here's a small example of how you might do this in java 8:

```
AmazonRDS client = AmazonRDSClientBuilder.defaultClient();
DescribeDBClustersRequest request = new DescribeDBClustersRequest()
   .withDBClusterIdentifier(clusterName);
DescribeDBClustersResult result =
rdsClient.describeDBClusters(request);

DBCluster singleClusterResult = result.getDBClusters().get(0);

String pgJDBCEndpointStr =
singleClusterResult.getDBClusterMembers().stream()
   .sorted(Comparator.comparing(DBClusterMember::getIsClusterWriter)
   .reversed()) // This puts the writer at the front of the list
   .map(m -> m.getDBInstanceIdentifier() + endpointPostfix + ":" +
 singleClusterResult.getPort()))
   .collect(Collectors.joining(","));
```

pgJDBCEndpointStr will contain a formatted list of endpoints, e.g:

```
my-node1.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com:5432,
my-node2.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com:5432
```

The variable 'endpointPostfix' can be a constant that your application sets, or can be obtained by querying the DescribeDBInstances API for a single instance in your cluster. This value remains constant within a region and for an individual customer, so it would save an API call to simply keep this constant in a resource file that your application reads from. In the example above, it would be set to:

```
.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com
```

For availability purposes, a good practice would be to default to using the Aurora Endpoints of your DB Cluster if the API is not responding, or taking too long to respond. The endpoints are guaranteed to be up to date within the time it takes to update the DNS record (typically less than 30 seconds). This again can be stored in a resource file that your application consumes.

Testing Failover

In all cases you must have a DB Cluster with >= 2 DB instances in it.

From the server side, certain APIs can cause an outage that can be used to test how your applications responds:

- FailoverDBCluster - Will attempt to promote a new DB Instance in your DB Cluster to writer

```
public void causeFailover() {
    /*
     * See http://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/basics.html for
 more details on setting up an RDS client
     */
    final AmazonRDS rdsClient = AmazonRDSClientBuilder.defaultClient();

    FailoverDBClusterRequest request = new FailoverDBClusterRequest();
    request.setDBClusterIdentifier("cluster-identifier");

    rdsClient.failoverDBCluster(request);
}
```

- RebootDBInstance - Failover is not guaranteed in this API. It will shutdown the database on the writer, though, and can be used to test how your application responds to connections dropping (note that the **ForceFailover** parameter is not applicable for Aurora engines and instead should use the FailoverDBCluster API)
- ModifyDBCluster - Modifying the **Port** will cause an outage when the nodes in the cluster begin listening on a new port. In general your application can respond to this failure by ensuring that only your application controls port changes and can appropriately update the endpoints it depends on, by having someone manually update the port when they make modifications at the API level, or by querying the RDS API in your application to determine if the port has changed.
- ModifyDBInstance - Modifying the **DBInstanceClass** will cause an outage
- DeleteDBInstance - Deleting the master/writer will cause a new DB Instance to be promoted to writer in your DB Cluster

From the application/client side, if using Linux, you can test how the application responds to sudden packet drops based on port, host, or if tcp keepalive packets are not sent or received by using iptables.

Fast Failover Example

The following code sample shows how an application might set up an Aurora PostgreSQL driver manager. The application would call `getConnection(...)` when it needed a connection. A call to that function can fail to find a valid host, such as when no writer is found but the targetServerType was set to "master"), and the calling application should simply retry. This can easily be wrapped into a connection pooler to avoid pushing the retry behavior onto the application. Most connection poolers allow you to specify a JDBC connection string, so your application could call into `getJdbcConnectionString(...)` and pass that to the connection pooler to make use of faster failover on Aurora PostgreSQL.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
```

```
import java.util.List;
import java.util.stream.Collectors;
import java.util.stream.IntStream;

import org.joda.time.Duration;

public class FastFailoverDriverManager {
   private static Duration LOGIN_TIMEOUT = Duration.standardSeconds(2);
   private static Duration CONNECT_TIMEOUT = Duration.standardSeconds(2);
   private static Duration CANCEL_SIGNAL_TIMEOUT = Duration.standardSeconds(1);
   private static Duration DEFAULT_SOCKET_TIMEOUT = Duration.standardSeconds(5);

   public FastFailoverDriverManager() {
       try {
            Class.forName("org.postgresql.Driver");
       } catch (ClassNotFoundException e) {
            e.printStackTrace();
       }

       /*
         * RO endpoint has a TTL of 1s, we should honor that here. Setting this
 aggressively makes sure that when
         * the PG JDBC driver creates a new connection, it will resolve a new different RO
 endpoint on subsequent attempts
         * (assuming there is > 1 read node in your cluster)
         */
        java.security.Security.setProperty("networkaddress.cache.ttl" , "1");
       // If the lookup fails, default to something like small to retry
       java.security.Security.setProperty("networkaddress.cache.negative.ttl" , "3");
   }

   public Connection getConnection(String targetServerType) throws SQLException {
       return getConnection(targetServerType, DEFAULT_SOCKET_TIMEOUT);
   }

   public Connection getConnection(String targetServerType, Duration queryTimeout) throws
 SQLException {
        Connection conn =
 DriverManager.getConnection(getJdbcConnectionString(targetServerType, queryTimeout));

       /*
         * A good practice is to set socket and statement timeout to be the same thing
 since both
         * the client AND server will kill the query at the same time, leaving no running
 queries
         * on the backend
         */
        Statement st = conn.createStatement();
        st.execute("set statement_timeout to " + queryTimeout.getMillis());
        st.close();

       return conn;
   }

   private static String urlFormat = "jdbc:postgresql://%s"
           + "/postgres"
           + "?user=%s"
           + "&password=%s"
           + "&loginTimeout=%d"
           + "&connectTimeout=%d"
           + "&cancelSignalTimeout=%d"
           + "&socketTimeout=%d"
           + "&targetServerType=%s"
           + "&tcpKeepAlive=true"
           + "&ssl=true"
           + "&loadBalanceHosts=true";
```

```
   public String getJdbcConnectionString(String targetServerType, Duration queryTimeout) {
       return String.format(urlFormat,
                getFormattedEndpointList(getLocalEndpointList()),
                CredentialManager.getUsername(),
                CredentialManager.getPassword(),
                LOGIN_TIMEOUT.getStandardSeconds(),
                CONNECT_TIMEOUT.getStandardSeconds(),
                CANCEL_SIGNAL_TIMEOUT.getStandardSeconds(),
                queryTimeout.getStandardSeconds(),
                targetServerType
       );
   }

   private List<String> getLocalEndpointList() {
       /*
         * As mentioned in the best practices doc, a good idea is to read a local resource
 file and parse the cluster endpoints.
         * For illustration purposes, the endpoint list is hardcoded here
         */
        List<String> newEndpointList = new ArrayList<>();
        newEndpointList.add("myauroracluster.cluster-c9bfei4hjlrd.us-east-1-
beta.rds.amazonaws.com:5432");
        newEndpointList.add("myauroracluster.cluster-ro-c9bfei4hjlrd.us-east-1-
beta.rds.amazonaws.com:5432");

       return newEndpointList;
   }

   private static String getFormattedEndpointList(List<String> endpoints) {
       return IntStream.range(0, endpoints.size())
                .mapToObj(i -> endpoints.get(i).toString())
                .collect(Collectors.joining(","));
   }
}
```

Related Topics

- Amazon Aurora on Amazon RDS (p. 428)

Amazon Aurora PostgreSQL Reference

Amazon Aurora PostgreSQL Parameters

You manage your Amazon Aurora DB cluster in the same way that you manage other Amazon RDS DB instances, by using parameters in a DB parameter group. Amazon Aurora differs from other DB engines in that you have a DB cluster that contains multiple DB instances. As a result, some of the parameters that you use to manage your Amazon Aurora DB cluster apply to the entire cluster, while other parameters apply only to a particular DB instance in the DB cluster.

Cluster-level parameters are managed in DB cluster parameter groups. Instance-level parameters are managed in DB parameter groups. Although each DB instance in an Aurora PostgreSQL DB cluster is compatible with the PostgreSQL database engine, some of the PostgreSQL database engine parameters must be applied at the cluster level, and are managed using DB cluster parameter groups. Cluster-level parameters are not found in the DB parameter group for a DB instance in an Aurora PostgreSQL DB cluster and are listed later in this topic.

You can manage both cluster-level and instance-level parameters using the Amazon RDS console, the AWS CLI, or the Amazon RDS API. There are separate commands for managing cluster-level parameters and instance-level parameters. For example, you can use the modify-db-cluster-parameter-group AWS

CLI command to manage cluster-level parameters in a DB cluster parameter group and use the modify-db-parameter-group AWS CLI command to manage instance-level parameters in a DB parameter group for a DB instance in a DB cluster.

You can view both cluster-level and instance-level parameters in the Amazon RDS console, or by using the AWS CLI or Amazon RDS API. For example, you can use the describe-db-cluster-parameters AWS CLI command to view cluster-level parameters in a DB cluster parameter group and use the describe-db-parameters AWS CLI command to view instance-level parameters in a DB parameter group for a DB instance in a DB cluster.

For more information about parameter groups, see Working with DB Parameter Groups (p. 170).

Cluster-level Parameters

The following table shows all of the parameters that apply to the entire Aurora PostgreSQL DB cluster.

Parameter name	Modifiable
archive_command	No
archive_timeout	No
array_nulls	Yes
autovacuum	Yes
autovacuum_analyze_scale_factor	Yes
autovacuum_analyze_threshold	Yes
autovacuum_freeze_max_age	Yes
autovacuum_max_workers	Yes
autovacuum_multixact_freeze_max_age	Yes
autovacuum_naptime	Yes
autovacuum_vacuum_cost_delay	Yes
autovacuum_vacuum_cost_limit	Yes
autovacuum_vacuum_scale_factor	Yes
autovacuum_vacuum_threshold	Yes
autovacuum_work_mem	Yes
backslash_quote	Yes
client_encoding	Yes
data_directory	No
datestyle	Yes
default_tablespace	Yes
default_with_oids	Yes
extra_float_digits	Yes

Parameter name	Modifiable
huge_pages	No
intervalstyle	Yes
lc_monetary	Yes
lc_numeric	Yes
lc_time	Yes
log_autovacuum_min_duration	Yes
max_prepared_transactions	Yes
password_encryption	No
port	No
rds.extensions	No
rds.force_autovacuum_logging_level	Yes
rds.force_ssl	Yes
server_encoding	No
ssl	Yes
synchronous_commit	Yes
timezone	Yes
track_commit_timestamp	Yes
vacuum_cost_delay	Yes
vacuum_cost_limit	Yes
vacuum_cost_page_hit	Yes
vacuum_cost_page_miss	Yes
vacuum_defer_cleanup_age	Yes
vacuum_freeze_min_age	Yes
vacuum_freeze_table_age	Yes
vacuum_multixact_freeze_min_age	Yes
vacuum_multixact_freeze_table_age	Yes
wal_buffers	Yes

Instance-level Parameters

The following table shows all of the parameters that apply to a specific DB instance in an Aurora PostgreSQL DB cluster.

Parameter name	Modifiable
application_name	Yes
authentication_timeout	Yes
auto_explain.log_analyze	Yes
auto_explain.log_buffers	Yes
auto_explain.log_format	Yes
auto_explain.log_min_duration	Yes
auto_explain.log_nested_statements	Yes
auto_explain.log_timing	Yes
auto_explain.log_triggers	Yes
auto_explain.log_verbose	Yes
auto_explain.sample_rate	Yes
backend_flush_after	Yes
bgwriter_flush_after	Yes
bytea_output	Yes
check_function_bodies	Yes
checkpoint_flush_after	Yes
checkpoint_timeout	No
client_min_messages	Yes
config_file	No
constraint_exclusion	Yes
cpu_index_tuple_cost	Yes
cpu_operator_cost	Yes
cpu_tuple_cost	Yes
cursor_tuple_fraction	Yes
db_user_namespace	No
deadlock_timeout	Yes
debug_pretty_print	Yes
debug_print_parse	Yes
debug_print_plan	Yes
debug_print_rewritten	Yes
default_statistics_target	Yes

Parameter name	Modifiable
default_transaction_deferrable	Yes
default_transaction_isolation	Yes
default_transaction_read_only	Yes
effective_cache_size	Yes
effective_io_concurrency	Yes
enable_bitmapscan	Yes
enable_hashagg	Yes
enable_hashjoin	Yes
enable_indexonlyscan	Yes
enable_indexscan	Yes
enable_material	Yes
enable_mergejoin	Yes
enable_nestloop	Yes
enable_seqscan	Yes
enable_sort	Yes
enable_tidscan	Yes
escape_string_warning	Yes
exit_on_error	No
force_parallel_mode	Yes
from_collapse_limit	Yes
geqo	Yes
geqo_effort	Yes
geqo_generations	Yes
geqo_pool_size	Yes
geqo_seed	Yes
geqo_selection_bias	Yes
geqo_threshold	Yes
gin_fuzzy_search_limit	Yes
gin_pending_list_limit	Yes
hba_file	No
hot_standby_feedback	Yes

Parameter name	Modifiable
ident_file	No
idle_in_transaction_session_timeout	Yes
join_collapse_limit	Yes
lc_messages	Yes
listen_addresses	No
lo_compat_privileges	No
log_connections	Yes
log_destination	Yes
log_directory	No
log_disconnections	Yes
log_duration	Yes
log_error_verbosity	Yes
log_executor_stats	Yes
log_file_mode	No
log_filename	Yes
log_hostname	Yes
log_line_prefix	No
log_lock_waits	Yes
log_min_duration_statement	Yes
log_min_error_statement	Yes
log_min_messages	Yes
log_parser_stats	Yes
log_planner_stats	Yes
log_replication_commands	Yes
log_rotation_age	Yes
log_rotation_size	Yes
log_statement	Yes
log_statement_stats	Yes
log_temp_files	Yes
log_timezone	No
log_truncate_on_rotation	No

Parameter name	Modifiable
logging_collector	No
maintenance_work_mem	Yes
max_connections	Yes
max_files_per_process	Yes
max_locks_per_transaction	Yes
max_replication_slots	Yes
max_stack_depth	Yes
max_standby_archive_delay	Yes
max_standby_streaming_delay	Yes
max_wal_senders	Yes
max_worker_processes	Yes
min_parallel_relation_size	Yes
old_snapshot_threshold	Yes
operator_precedence_warning	Yes
parallel_setup_cost	Yes
parallel_tuple_cost	Yes
pg_hint_plan.debug_print	Yes
pg_hint_plan.enable_hint	Yes
pg_hint_plan.enable_hint_table	Yes
pg_hint_plan.message_level	Yes
pg_hint_plan.parse_messages	Yes
pg_stat_statements.max	Yes
pg_stat_statements.save	Yes
pg_stat_statements.track	Yes
pg_stat_statements.track_utility	Yes
pgaudit.log	Yes
pgaudit.log_catalog	Yes
pgaudit.log_level	Yes
pgaudit.log_parameter	Yes
pgaudit.log_relation	Yes
pgaudit.log_statement_once	Yes

Parameter name	Modifiable
pgaudit.role	Yes
postgis.gdal_enabled_drivers	Yes
quote_all_identifiers	Yes
random_page_cost	Yes
rds.force_admin_logging_level	Yes
rds.log_retention_period	No
rds.rds_superuser_reserved_connections	Yes
rds.superuser_variables	No
replacement_sort_tuples	Yes
restart_after_crash	No
row_security	Yes
search_path	Yes
seq_page_cost	Yes
session_replication_role	Yes
shared_buffers	Yes
shared_preload_libraries	Yes
sql_inheritance	Yes
ssl_ca_file	No
ssl_cert_file	No
ssl_ciphers	No
ssl_key_file	No
standard_conforming_strings	Yes
statement_timeout	Yes
stats_temp_directory	No
superuser_reserved_connections	No
synchronize_seqscans	Yes
syslog_facility	No
tcp_keepalives_count	Yes
tcp_keepalives_idle	Yes
tcp_keepalives_interval	Yes
temp_buffers	Yes

Parameter name	Modifiable
temp_tablespaces	Yes
track_activities	Yes
track_activity_query_size	Yes
track_counts	Yes
track_functions	Yes
track_io_timing	Yes
transaction_deferrable	Yes
transaction_read_only	Yes
transform_null_equals	Yes
unix_socket_directories	No
unix_socket_group	No
unix_socket_permissions	No
update_process_title	Yes
wal_receiver_status_interval	Yes
wal_receiver_timeout	Yes
wal_sender_timeout	Yes
work_mem	Yes
xmlbinary	Yes
xmloption	Yes

Related Topics

- Amazon Aurora on Amazon RDS (p. 428)

Amazon Aurora PostgreSQL Database Engine Updates

In the following topic, you can find version and update information specific to Amazon Aurora PostgreSQL. For more information about updates that apply generally to Aurora, see Amazon Aurora Updates (p. 610).

Amazon Aurora PostgreSQL Versions

Amazon Aurora includes certain features that are general to Aurora and available to all Aurora DB clusters. Aurora includes other features that are specific to a particular database engine that Aurora supports. These features are available only to those Aurora DB clusters that use that database engine, such as Aurora PostgreSQL.

An Aurora DB instance provides two version numbers, the Aurora version number and the Aurora database engine version number. For more information about the Aurora version number, see Amazon Aurora Versions (p. 664).

You can get the Aurora database engine version number for an Aurora PostgreSQL DB instance by querying for the `SERVER_VERSION` run-time parameter. To get the Aurora database engine version number, use one of the following queries.

```
SELECT SERVER_VERSION();
```

```
SHOW SERVER_VERSION;
```

Amazon Aurora PostgreSQL Database Upgrades (Patching)

When a new minor version of the Amazon Aurora PostgreSQL database engine is released, Amazon RDS schedules an automatic upgrade of the database engine for all Aurora DB clusters. We announce automatic upgrades in the Amazon RDS Community Forum.

When a new patch version of the Aurora PostgreSQL database engine is released, no automatic upgrade is required. You can choose to upgrade and apply the patch. If you don't, the patch is applied during the next automatic upgrade for a minor version release.

Before automatic upgrade, new database engine releases show as an **available** maintenance upgrade for your DB cluster. You can manually upgrade the database version for your DB cluster by applying the available maintenance action. We encourage you to apply the update on a nonproduction instance before the automatic upgrade. That way, you can see how changes in the new version affect your instances and applications.

To apply pending maintenance actions

- **By using the Amazon RDS Management Console** – Log on to the Amazon RDS console and choose **Clusters**. Choose the DB cluster that shows an **available** maintenance upgrade. Choose **Cluster Actions**. Choose **Upgrade Now** to immediately update the database version for your DB cluster. Or choose **Upgrade at Next Window** to update the database version for your DB cluster during the next cluster maintenance window.
- **By using the AWS CLI** – Call the apply-pending-maintenance-action AWS CLI command and specify the Amazon Resource Name (ARN) for your DB cluster for the `--resource-id` option and `system-update` for the `--apply-action` option. Set the `--opt-in-type` option to `immediate` to immediately update the database version for your DB cluster. Or set it to `next-maintenance` to update the database version for your DB cluster during the next cluster maintenance window.
- **By using the Amazon RDS API** – Call the ApplyPendingMaintenanceAction API action and specify the Amazon Resource Name (ARN) for your DB cluster for the `ResourceId` parameter and `system-update` for the `ApplyAction` parameter. Set the `OptInType` parameter to `immediate` to immediately update the database version for your DB cluster, or `next-maintenance` to update the database version for your instance during the next cluster maintenance window.

For more information on how Amazon RDS manages database and operating system updates, see DB Instance and DB Cluster Maintenance (p. 102).

Best Practices with Amazon Aurora

This topic includes information on general best practices and options for using or migrating data to an Amazon Aurora DB cluster.

Some of the best practices for Amazon Aurora are specific to a particular database engine. For more information about Aurora best practices specific to a database engine, see the following:

Database Engine	Best Practices
Amazon Aurora MySQL	See Best Practices with Amazon Aurora MySQL (p. 592)
Amazon Aurora PostgreSQL	See Best Practices with Amazon Aurora PostgreSQL (p. 647)

Related Topics

- Amazon Aurora on Amazon RDS (p. 428)

Amazon Aurora Updates

Amazon Aurora releases updates regularly. Updates are applied to Amazon Aurora DB clusters during system maintenance windows. The timing when updates are applied depends on the region and maintenance window setting for the DB cluster, and also the type of update. Updates require a database restart, so you experience 20 to 30 seconds of downtime. After this downtime, you can resume using your DB cluster or clusters. You can view or change your maintenance window settings from the AWS Management Console.

Following, you can find information on general updates to Amazon Aurora. Some of the updates applied to Amazon Aurora are specific to a database engine supported by Aurora. For more information about database engine updates for Aurora, see the following table.

Database Engine	Updates
Amazon Aurora MySQL	See Amazon Aurora MySQL Database Engine Updates (p. 610)
Amazon Aurora PostgreSQL	See Amazon Aurora PostgreSQL Database Engine Updates (p. 662)

Amazon Aurora Versions

Amazon Aurora includes certain features that are general to Aurora and available to all Aurora DB clusters. Aurora includes other features that are specific to a particular database engine that Aurora supports. These features are available only to those Aurora DB clusters that use that database engine, such as Aurora PostgreSQL.

An Aurora DB instance provides two version numbers, the Aurora version number and the Aurora database engine version number. Aurora version numbers use the following format.

```
<major version>.<minor version>.<patch version>
```

To get the Aurora version number from an Aurora DB instance using a particular database engine, use one of the following queries.

Database Engine	Queries
Amazon Aurora MySQL	`SELECT AURORA_VERSION();`

Database Engine	Queries
	`SHOW @@aurora_version;`
Amazon Aurora PostgreSQL	`SELECT AURORA_VERSION();`

Related Topics

- Amazon Aurora on Amazon RDS (p. 428)

MariaDB on Amazon RDS

Amazon RDS supports DB instances running several versions of MariaDB. You can use the following major versions:

- MariaDB 10.1
- MariaDB 10.0

For more information about minor version suport, see MariaDB on Amazon RDS Versions (p. 667).

You first use the Amazon RDS management tools or interfaces to create an Amazon RDS MariaDB DB instance. You can then use the Amazon RDS tools to perform management actions for the DB instance, such as reconfiguring or resizing the DB instance, authorizing connections to the DB instance, creating and restoring from backups or snapshots, creating Multi-AZ secondaries, creating Read Replicas, and monitoring the performance of the DB instance. You use standard MariaDB utilities and applications to store and access the data in the DB instance.

MariaDB is available in all of the AWS regions. For more information about AWS Regions, see Regions and Availability Zones (p. 97).

You can use Amazon RDS for MariaDB databases to build HIPAA-compliant applications. You can store healthcare-related information, including protected health information (PHI), under an executed Business Associate Agreement (BAA) with AWS. For more information, see HIPAA Compliance. AWS Services in Scope have been fully assessed by a third-party auditor and result in a certification, attestation of compliance, or Authority to Operate (ATO). For more information, see AWS Services in Scope by Compliance Program.

Common Management Tasks for MariaDB on Amazon RDS

These are the common management tasks you perform with an Amazon RDS MariaDB DB instance, with links to information about each task:

- Before creating a DB instance, you should complete the steps in the Setting Up for Amazon RDS (p. 5) section of this guide.
- After you have met your prerequisites, such as creating security groups or DB parameter groups, you can create an Amazon RDS MariaDB DB instance. For information on this process, see Creating a DB Instance Running the MariaDB Database Engine (p. 678).
- After creating your security group and DB instance, you can connect to the DB instance from MariaDB applications and utilities. For information, see Connecting to a DB Instance Running the MariaDB Database Engine (p. 688).
- A newly created Amazon RDS DB instance has one empty database with the name you specified when you created the DB instance, and one master user account with the name and password you specified. You must use a tool or utility compatible with MariaDB to log in as the master user, and then use MariaDB commands and SQL statements to add all of the users and elements required for your applications to store and retrieve data in the DB instance, such as the following:

  - Create all user IDs and grant them the appropriate permissions. For information, see MariaDB User Account Management in the MariaDB documentation.
  - Create any required databases and objects such as tables and views. For information, see Data Definition in the MariaDB documentation.
  - Establish procedures for importing or exporting data. For information on some recommended procedures, see Importing Data into a MariaDB DB Instance (p. 706).
- You might need to periodically change your DB instance, such as to resize or reconfigure the DB instance. For information on doing so, see Modifying a DB Instance Running the MariaDB Database Engine (p. 691). For additional information on specific tasks, see the following:
  - Renaming a DB Instance (p. 116)
  - Deleting a DB Instance (p. 126)
  - Rebooting a DB Instance (p. 119)
  - Tagging Amazon RDS Resources (p. 129)
  - DB Instance and DB Cluster Maintenance (p. 102)
  - Adjusting the Preferred DB Instance Maintenance Window (p. 104)
- You can configure your DB instance to take automated backups, or take manual snapshots, and then restore instances from the backups or snapshots. For information, see Backing Up and Restoring Amazon RDS DB Instances (p. 200).
- You can monitor an instance through actions such as viewing the MariaDB logs, Amazon CloudWatch metrics for Amazon RDS, and events. For information, see Monitoring Amazon RDS (p. 245).
- You can offload read traffic from your primary MariaDB DB instance by creating Read Replicas. For information, see Working with PostgreSQL, MySQL, and MariaDB Read Replicas (p. 134).
- Several Amazon RDS features that you can use with MariaDB DB instances are common across the Amazon RDS database engines. For information on these, see the following:
  - Working with Reserved DB Instances (p. 189)
  - Provisioned IOPS Storage (p. 413)

Also, several appendices include useful information about working with Amazon RDS MariaDB DB instances:

- Appendix: Parameters for MariaDB (p. 712)
- Appendix: MariaDB on Amazon RDS SQL Reference (p. 717)

MariaDB on Amazon RDS Versions

For MariaDB, version numbers are organized as version X.Y.Z. In Amazon RDS terminology, X.Y denotes the major version, and Z is the minor version number. For Amazon RDS implementations, a version change is considered major if the major version number changes, for example going from version 10.0 to 10.1. A version change is considered minor if only the minor version number changes, for example going from version 10.0.17 to 10.0.24.

Amazon RDS currently supports the following versions of MariaDB:

Major Version	Minor Version
MariaDB 10.1	• 10.1.26 (supported in all AWS Regions) • 10.1.23 (supported in all AWS Regions) • 10.1.19 (supported in all AWS Regions) • 10.1.14 (supported in all AWS Regions except us-east-2)

Major Version	Minor Version
MariaDB 10.0	• 10.0.32 (supported in all AWS Regions) • 10.0.31 (supported in all AWS Regions) • 10.0.28 (supported in all AWS Regions) • 10.0.24 (supported in all AWS Regions) • 10.0.17 (supported in all AWS Regions except us-east-2, ca-central-1, eu-west-2)

Over time, we plan to support additional MariaDB versions for Amazon RDS. The number of new version releases supported in a given year will vary based on the frequency and content of the MariaDB version releases and the outcome of a thorough vetting of the release by our database engineering team. However, as a general guidance, we aim to support new MariaDB versions within 3-5 months of their General Availability release.

You can specify any currently supported MariaDB version when creating a new DB Instance. If no version is specified, Amazon RDS will default to a supported version, typically the most recent version. If a major version (e.g. MariaDB 10.0) is specified but a minor version is not, Amazon RDS will default to a recent release of the major version you have specified. To see a list of supported versions, as well as defaults for newly created DB Instances, use the DescribeDBEngineVersions API.

With Amazon RDS, you control when to upgrade your MariaDB instance to a new version supported by Amazon RDS. You can maintain compatibility with specific MariaDB versions, test new versions with your application before deploying in production, and perform version upgrades at times that best fit your schedule.

Unless you specify otherwise, your DB instance is automatically upgraded to new MariaDB minor versions as they are supported by Amazon RDS. This patching occurs during your scheduled maintenance window, and it is announced on the Amazon RDS Community Forum in advance. To turn off automatic version upgrades, change the **Auto Minor Version Upgrade** setting for the DB instance to **No**. For more information on modifying the DB instance, see Modifying a DB Instance Running the MariaDB Database Engine (p. 691).

If you opt out of automatic minor version upgrades, you can manually upgrade to a supported minor version release by following the same procedure as for a major version update. For information, see DB Instance and DB Cluster Maintenance (p. 102).

You cannot set major version upgrades to occur automatically, because they involve some compatibility risk. Instead, you must make a request to upgrade the DB instance to a different major version. You should thoroughly test your databases and applications against the new target version before upgrading your production instances. For information about upgrading a DB instance, see DB Instance and DB Cluster Maintenance (p. 102).

You can test a DB instance against a new version before upgrading by creating a DB snapshot of your existing DB instance, restoring from the DB snapshot to create a new DB instance, and then initiating a version upgrade for the new DB instance. You can then experiment safely on the upgraded clone of your DB instance before deciding whether or not to upgrade your original DB instance.

The Amazon RDS deprecation policy for MariaDB includes the following:

- We intend to support major MariaDB version releases, starting with MariaDB 10.0.17, for 3 years after they are initially supported by Amazon RDS.
- We intend to support minor MariaDB version releases for at least 1 year after they are initially supported by Amazon RDS.
- After a MariaDB major or minor version has been deprecated, we expect to provide a three-month grace period for you to initiate an upgrade to a supported version prior to an automatic upgrade being applied during your scheduled maintenance window.

MariaDB, MySQL, and Amazon Aurora Feature Comparison

Use the following table to compare MariaDB, MySQL, and Aurora features to determine which DB engine is the best choice for your DB instance.

Feature	MariaDB	MySQL	Amazon Aurora
Storage engines	Supports XtraDB fully, and Aria with some limitations.	Supports both MyISAM and InnoDB.	Supports only InnoDB. Tables from other storage engines are automatically converted to InnoDB. Because Amazon Aurora only supports the InnoDB engine, the NO_ENGINE_SUBSTITUTION option of the SQL_MODE database parameter is enabled. Enabling this option disables the ability to create an in-memory table, unless that table is specified as TEMPORARY.
Plugins	Supports plugins. For more information, see Appendix: Options for MariaDB Database Engine (p. 709).	Supports plugins. For more information, see Options for MySQL DB Instances (p. 897).	Doesn't support plugins.
Join and subquery performance	Includes query optimizer improvements for joins and subqueries faster than those in MySQL 5.5 and 5.6. For more information, see Optimizer Feature Comparison Matrix in the MariaDB documentation.	Query optimizer performance in keeping with MySQL 5.5, 5.6, or 5.7, depending on the version you selected for your Amazon RDS MySQL DB instance.	Query optimizer performance in keeping with MySQL 5.6.
Group commit	Supports group commits. For more information, see Optimizer Feature Comparison Matrix in the MariaDB documentation. Supports additional tuning of group commits by setting the `binlog_commit_wait_count` parameter to determine the number of transactions that must complete before performing a group commit, and by setting the `binlog_commit_wait_usec`	Supports group commits.	Supports group commits.

Feature	MariaDB	MySQL	Amazon Aurora
	parameter to delay performing a group commit by a specified number of milliseconds. For more information on these parameters, see binlog_commit_wait_count or binlog_commit_wait_usec in the MariaDB documentation. For more information on setting parameters for a DB instance, see Working with DB Parameter Groups (p. 170).		
Progress reporting	Supports progress reporting for some long-running commands. For more information, see Progress Reporting in the MariaDB documentation.	Doesn't support progress reporting.	Doesn't support progress reporting.
Roles	Support creation of custom roles for easily assigning sets of privileges to groups of users. For more information, see Roles in the MariaDB documentation.	Doesn't support roles.	Doesn't support roles.
SHOW EXPLAIN	Supports the SHOW EXPLAIN command, using which you can get a description of the query plan for a query running in a specified thread. For more information, see SHOW EXPLAIN in the MariaDB documentation.	Doesn't support SHOW EXPLAIN.	Doesn't support SHOW EXPLAIN.
Table elimination	Supports table elimination, which sometimes allows the DB instance to improve performance by resolving a query without accessing some of the tables that the query refers to. For more information, see Table Elimination in the MariaDB documentation.	Doesn't support table elimination.	Doesn't support table elimination.

Feature	MariaDB	MySQL	Amazon Aurora
Thread pooling	Supports thread pooling to enable the DB instance to handle more connections without performance degradation. For more information, see Thread Pool in MariaDB in the MariaDB documentation.	Doesn't support thread pooling.	Doesn't support thread pooling.
Virtual columns	Supports virtual columns. These table columns have their values automatically calculated using a deterministic expression, typically based on the values of other columns in the table. For more information, see Virtual (Computed) Columns in the MariaDB documentation.	Doesn't support virtual columns.	Doesn't support virtual columns.
Global transaction IDs	Supports the MariaDB implementation of global transaction IDs (GTIDs). For more information, see Global Transaction ID in the MariaDB documentation. **Note** Amazon RDS doesn't permit changes to the domain ID portion of a MariaDB GTID.	Doesn't support the MySQL implementation of global transaction IDs.	Doesn't support the MySQL implementation of global transaction IDs.
Parallel replication	Supports parallel replication, which increases replication performance by allowing queries to process in parallel on the replica. For more information, see Parallel Replication in the MariaDB documentation. Although parallel replication is similar to multithreaded replication in MySQL 5.6, it has some enhancements, such as not requiring partitioning across schemas and permitting group commits to replicate in parallel.	MySQL 5.6 and 5.7 support multithreaded replication. For more information, see Replication Slave Options and Variables in the MySQL documentation. MySQL 5.5 doesn't support multithreaded replication.	Supports the MySQL 5.6 implementation of multithreaded replication.

Feature	MariaDB	MySQL	Amazon Aurora
Database engine parameters	Parameters apply to each individual DB instance or Read Replica and are managed by DB parameter groups.	Parameters apply to each individual DB instance or Read Replica and are managed by DB parameter groups.	Some parameters apply to the entire Aurora DB cluster and are managed by DB cluster parameter groups. Other parameters apply to each individual DB instance in a DB cluster and are managed by DB parameter groups.
Read Replicas with a different storage engine than the master instance	Read Replicas can use XtraDB.	Read Replicas can use both MyISAM and InnoDB.	MySQL (non-RDS) Read Replicas that replicate with an Aurora DB cluster can only use InnoDB.
Read scaling	Supports up to 5 Read Replicas with some impact on the performance of write operations.	Supports up to 5 Read Replicas with some impact on the performance of write operations.	Supports up to 15 Aurora Replicas with minimal impact on the performance of write operations.
Failover target	You can manually promote Read Replicas to the master DB instance with potential data loss.	You can manually promote Read Replicas to the master DB instance with potential data loss.	Aurora Replicas are automatic failover targets with no data loss.
AWS region	Available in all AWS regions.	Available in all AWS regions.	Aurora DB clusters are not available in all AWS Regions. For more information, see Availability (p. 431).

MariaDB Features Supported in Version 10.1

Amazon RDS supports the following features in MariaDB DB instances running MariaDB version 10.1 or later:

- Optimistic in-order parallel replication
- Page Compression
- XtraDB data scrubbing and defragmentation

MariaDB Features Not Supported by Amazon RDS

Amazon RDS currently doesn't support the following MariaDB features:

- Data at Rest Encryption
- MariaDB Galera Cluster
- HandlerSocket

- JSON table type
- Multi-source Replication
- Password validation plugin, `simple_password_check`, and `cracklib_password_check`
- Replication Filters
- Storage engine-specific object attributes, as described in Engine-defined New Table/Field/Index Attributes.
- Table and Tablespace Encryption

To deliver a managed service experience, Amazon RDS doesn't provide shell access to DB instances, and it restricts access to certain system procedures and tables that require advanced privileges. Amazon RDS supports access to databases on a DB instance using any standard SQL client application. Amazon RDS doesn't allow direct host access to a DB instance by using Telnet, Secure Shell (SSH), or Windows Remote Desktop Connection.

Supported Storage Engines for MariaDB on Amazon RDS

While MariaDB supports multiple storage engines with varying capabilities, not all of them are optimized for recovery and data durability. Amazon RDS fully supports the XtraDB storage engine for MariaDB DB instances. Amazon RDS features such as Point-In-Time Restore and snapshot restore require a recoverable storage engine and are supported for the XtraDB storage engine only. Amazon RDS also supports Aria, although using Aria might have a negative impact on recovery in the event of an instance failure. However, if you need to use spatial indexes to handle geographic data, you should use Aria because spatial indexes are not supported by XtraDB.

Other storage engines are not currently supported by Amazon RDS for MariaDB.

MariaDB Security on Amazon RDS

Security for Amazon RDS MariaDB DB instances is managed at three levels:

- AWS Identity and Access Management controls who can perform Amazon RDS management actions on DB instances. When you connect to AWS using IAM credentials, your IAM account must have IAM policies that grant the permissions required to perform Amazon RDS management operations. For more information, see Authentication and Access Control for Amazon RDS (p. 327).
- When you create a DB instance, you use either a VPC security group or a DB security group to control which devices and Amazon EC2 instances can open connections to the endpoint and port of the DB instance. These connections can be made using Secure Socket Layer (SSL). In addition, firewall rules at your company can control whether devices running at your company can open connections to the DB instance.
- Once a connection has been opened to a MariaDB DB instance, authentication of the login and permissions are applied the same way as in a stand-alone instance of MariaDB. Commands such as `CREATE USER`, `RENAME USER`, `GRANT`, `REVOKE`, and `SET PASSWORD` work just as they do in stand-alone databases, as does directly modifying database schema tables.

When you create an Amazon RDS DB instance, the master user has the following default privileges:

- `alter`
- `alter routine`

- `create`
- `create routine`
- `create temporary tables`
- `create user`
- `create view`
- `delete`
- `drop`
- `event`
- `execute`
- `grant option`
- `index`
- `insert`
- `lock tables`
- `process`
- `references`
- `reload`

  This privilege is limited on Amazon RDS MariaDB DB instances. It doesn't grant access to the FLUSH LOGS or FLUSH TABLES WITH READ LOCK operations.
- `replication client`
- `replication slave`
- `select`
- `show databases`
- `show view`
- `trigger`
- `update`

For more information about these privileges, see User Account Management in the MariaDB documentation.

> **Note**
> Although you can delete the master user on a DB instance, we don't recommend doing so. To recreate the master user, use the `ModifyDBInstance` API or the `modify-db-instance` AWS command line tool and specify a new master user password with the appropriate parameter. If the master user does not exist in the instance, the master user is created with the specified password.

To provide management services for each DB instance, the `rdsadmin` user is created when the DB instance is created. Attempting to drop, rename, change the password for, or change privileges for the `rdsadmin` account results in an error.

To allow management of the DB instance, the standard `kill` and `kill_query` commands have been restricted. The Amazon RDS commands `mysql.rds_kill`, `mysql.rds_kill_query`, and `mysql.rds_kill_query_id` are provided for use in MariaDB and also MySQL so that you can terminate user sessions or queries on DB instances.

SSL Support for MariaDB DB Instances

Amazon RDS supports SSL connections with DB instances running the MariaDB database engine.

Amazon RDS creates an SSL certificate and installs the certificate on the DB instance when Amazon RDS provisions the instance. These certificates are signed by a certificate authority. The SSL certificate includes the DB instance endpoint as the Common Name (CN) for the SSL certificate to guard against spoofing attacks. The public key is stored at https://s3.amazonaws.com/rds-downloads/rds-combined-ca-bundle.pem.

To encrypt connections using the default mysql client, launch the mysql client using the `--ssl-ca parameter` to reference the public key, for example:

```
mysql -h mymariadbinstance.abcd1234.rds-us-east-1.amazonaws.com --ssl-ca=[full path]rds-combined-ca-bundle.pem --ssl-verify-server-cert
```

You can use the GRANT statement to require SSL connections for specific users accounts. For example, you can use the following statement to require SSL connections on the user account **`encrypted_user`**:

```
GRANT USAGE ON *.* TO 'encrypted_user'@'%' REQUIRE SSL
```

Note
For more information on SSL connections with MariaDB, see the SSL Overview in the MariaDB documentation.

XtraDB Cache Warming

XtraDB cache warming can provide performance gains for your MariaDB DB instance by saving the current state of the buffer pool when the DB instance is shut down, and then reloading the buffer pool from the saved information when the DB instance starts up. This approach bypasses the need for the buffer pool to "warm up" from normal database use and instead preloads the buffer pool with the pages for known common queries. For more information on XtraDB cache warming, see Dumping and restoring the buffer pool in the MariaDB documentation.

To enable XtraDB cache warming, set the `innodb_buffer_pool_dump_at_shutdown` and `innodb_buffer_pool_restore_at_startup` parameters to 1 in the parameter group for your DB instance. Changing these parameter values in a parameter group affects all MariaDB DB instances that use that parameter group. To enable XtraDB cache warming for specific MariaDB DB instances, you might need to create a new parameter group for those instances. For information on parameter groups, see Working with DB Parameter Groups (p. 170).

XtraDB cache warming primarily provides a performance benefit for DB instances that use standard storage. If you use PIOPS storage, you don't commonly see a significant performance benefit.

Important
If your MariaDB DB instance doesn't shut down normally, such as during a failover, then the buffer pool state isn't saved to disk. In this case, MariaDB loads whatever buffer pool file is available when the DB instance is restarted. No harm is done, but the restored buffer pool might not reflect the most recent state of the buffer pool prior to the restart. To ensure that you have a recent state of the buffer pool available to warm the XtraDB cache on startup, we recommend that you periodically dump the buffer pool "on demand." You can dump or load the buffer pool on demand.
You can create an event to dump the buffer pool automatically and at a regular interval. For example, the following statement creates an event named `periodic_buffer_pool_dump` that dumps the buffer pool every hour.

```
CREATE EVENT periodic_buffer_pool_dump
   ON SCHEDULE EVERY 1 HOUR
   DO CALL mysql.rds_innodb_buffer_pool_dump_now();
```

For more information, see Events in the MariaDB documentation.

Dumping and Loading the Buffer Pool on Demand

You can save and load the XtraDB cache on demand using the following stored procedures:

- To dump the current state of the buffer pool to disk, call the mysql.rds_innodb_buffer_pool_dump_now (p. 921) stored procedure.
- To load the saved state of the buffer pool from disk, call the mysql.rds_innodb_buffer_pool_load_now (p. 922) stored procedure.
- To cancel a load operation in progress, call the mysql.rds_innodb_buffer_pool_load_abort (p. 922) stored procedure.

Database Parameters for MariaDB

By default, a MariaDB DB instance uses a DB parameter group that is specific to a MariaDB database. This parameter group contains some but not all of the parameters contained in the Amazon RDS DB parameter groups for the MySQL database engine. It also contains a number of new, MariaDB-specific parameters. For more information on the parameters available for the Amazon RDS MariaDB DB engine, see Appendix: Parameters for MariaDB (p. 712).

Common DBA Tasks for MariaDB

Killing sessions or queries, skipping replication errors, working with XtraDB tablespaces to improve crash recovery times, and managing the global status history are common DBA tasks you might perform in a MariaDB DB instance. You can handle these tasks just as in an Amazon RDS MySQL DB instance, as described in Common DBA Tasks for MySQL DB Instances (p. 905). The crash recovery instructions there refer to the MySQL InnoDB engine, but they are applicable to a MariaDB instance running XtraDB as well.

Local Time Zone for MariaDB DB Instances

By default, the time zone for an RDS MariaDB DB instance is Universal Time Coordinated (UTC). You can set the time zone for your DB instance to the local time zone for your application instead.

To set the local time zone for a DB instance, set the `time_zone` parameter in the parameter group for your DB instance to one of the supported values listed later in this section. When you set the `time_zone` parameter for a parameter group, all DB instances and Read Replicas that are using that parameter group change to use the new local time zone. For information on setting parameters in a parameter group, see Working with DB Parameter Groups (p. 170).

After you set the local time zone, all new connections to the database reflect the change. If you have any open connections to your database when you change the local time zone, you won't see the local time zone update until after you close the connection and open a new connection.

You can set a different local time zone for a DB instance and one or more of its Read Replicas. To do this, use a different parameter group for the DB instance and the replica or replicas and set the `time_zone` parameter in each parameter group to a different local time zone.

If you are replicating across regions, then the replication master DB instance and the Read Replica use different parameter groups (parameter groups are unique to a region). To use the same local time zone for each instance, you must set the `time_zone` parameter in the instance's and Read Replica's parameter groups.

When you restore a DB instance from a DB snapshot, the local time zone is set to UTC. You can update the time zone to your local time zone after the restore is complete. If you restore a DB instance to a point in time, then the local time zone for the restored DB instance is the time zone setting from the parameter group of the restored DB instance.

You can set your local time zone to one of the following values.

Africa/Cairo	Asia/Bangkok	Australia/Darwin
Africa/Casablanca	Asia/Beirut	Australia/Hobart
Africa/Harare	Asia/Calcutta	Australia/Perth
Africa/Monrovia	Asia/Damascus	Australia/Sydney
Africa/Nairobi	Asia/Dhaka	Brazil/East
Africa/Tripoli	Asia/Irkutsk	Canada/Newfoundland
Africa/Windhoek	Asia/Jerusalem	Canada/Saskatchewan
America/Araguaina	Asia/Kabul	Europe/Amsterdam
America/Asuncion	Asia/Karachi	Europe/Athens
America/Bogota	Asia/Kathmandu	Europe/Dublin
America/Caracas	Asia/Krasnoyarsk	Europe/Helsinki
America/Chihuahua	Asia/Magadan	Europe/Istanbul
America/Cuiaba	Asia/Muscat	Europe/Kaliningrad
America/Denver	Asia/Novosibirsk	Europe/Moscow
America/Fortaleza	Asia/Riyadh	Europe/Paris
America/Guatemala	Asia/Seoul	Europe/Prague
America/Halifax	Asia/Shanghai	Europe/Sarajevo
America/Manaus	Asia/Singapore	Pacific/Auckland
America/Matamoros	Asia/Taipei	Pacific/Fiji
America/Monterrey	Asia/Tehran	Pacific/Guam
America/Montevideo	Asia/Tokyo	Pacific/Honolulu
America/Phoenix	Asia/Ulaanbaatar	Pacific/Samoa
America/Santiago	Asia/Vladivostok	US/Alaska
America/Tijuana	Asia/Yakutsk	US/Central
Asia/Amman	Asia/Yerevan	US/Eastern
Asia/Ashgabat	Atlantic/Azores	US/East-Indiana
Asia/Baghdad	Australia/Adelaide	US/Pacific
Asia/Baku	Australia/Brisbane	UTC

Creating a DB Instance Running the MariaDB Database Engine

The basic building block of Amazon RDS is the DB instance. The DB instance is where you create your MariaDB databases.

> **Important**
> You must complete the tasks in the Setting Up for Amazon RDS (p. 5) section before you can create or connect to a DB instance.

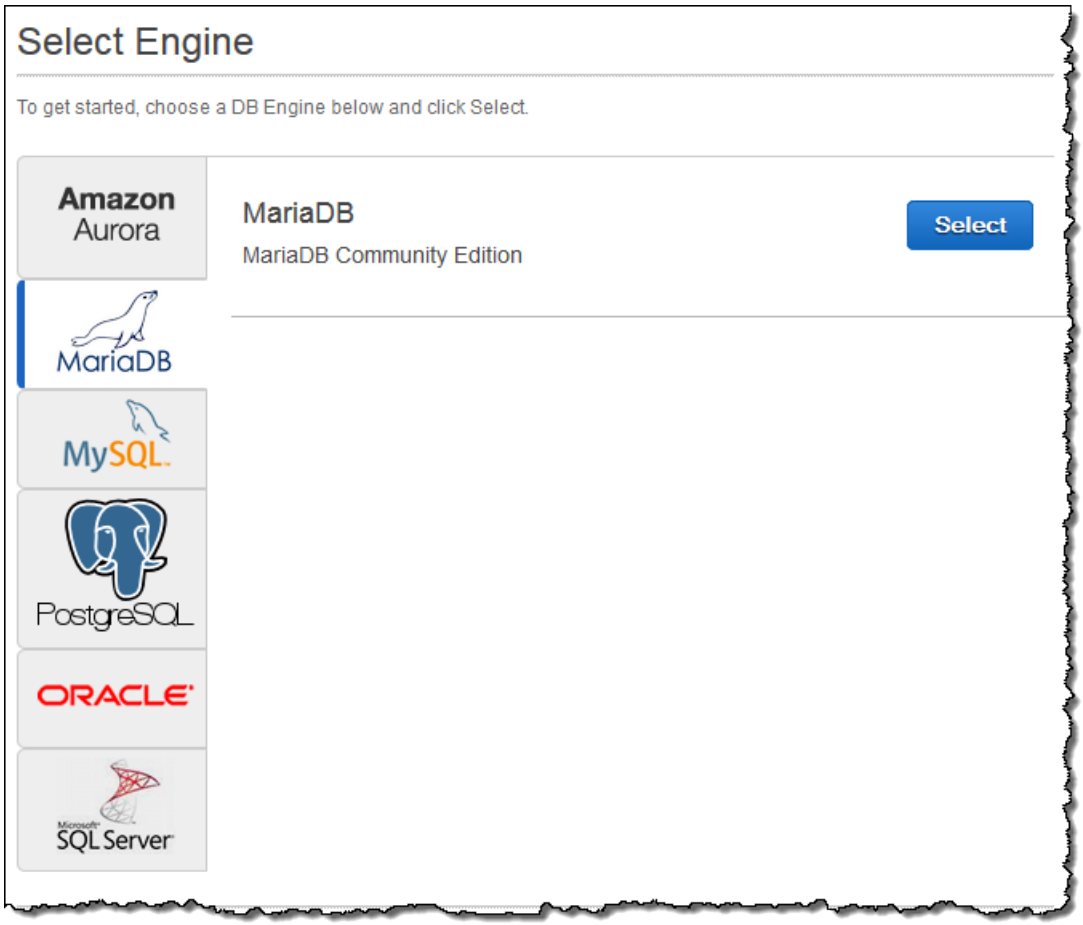
For an example that walks you through the process of creating and connecting to a sample DB instance, see Creating a MariaDB DB Instance and Connecting to a Database on a MariaDB DB Instance (p. 17).

AWS Management Console

To launch a MariaDB DB instance

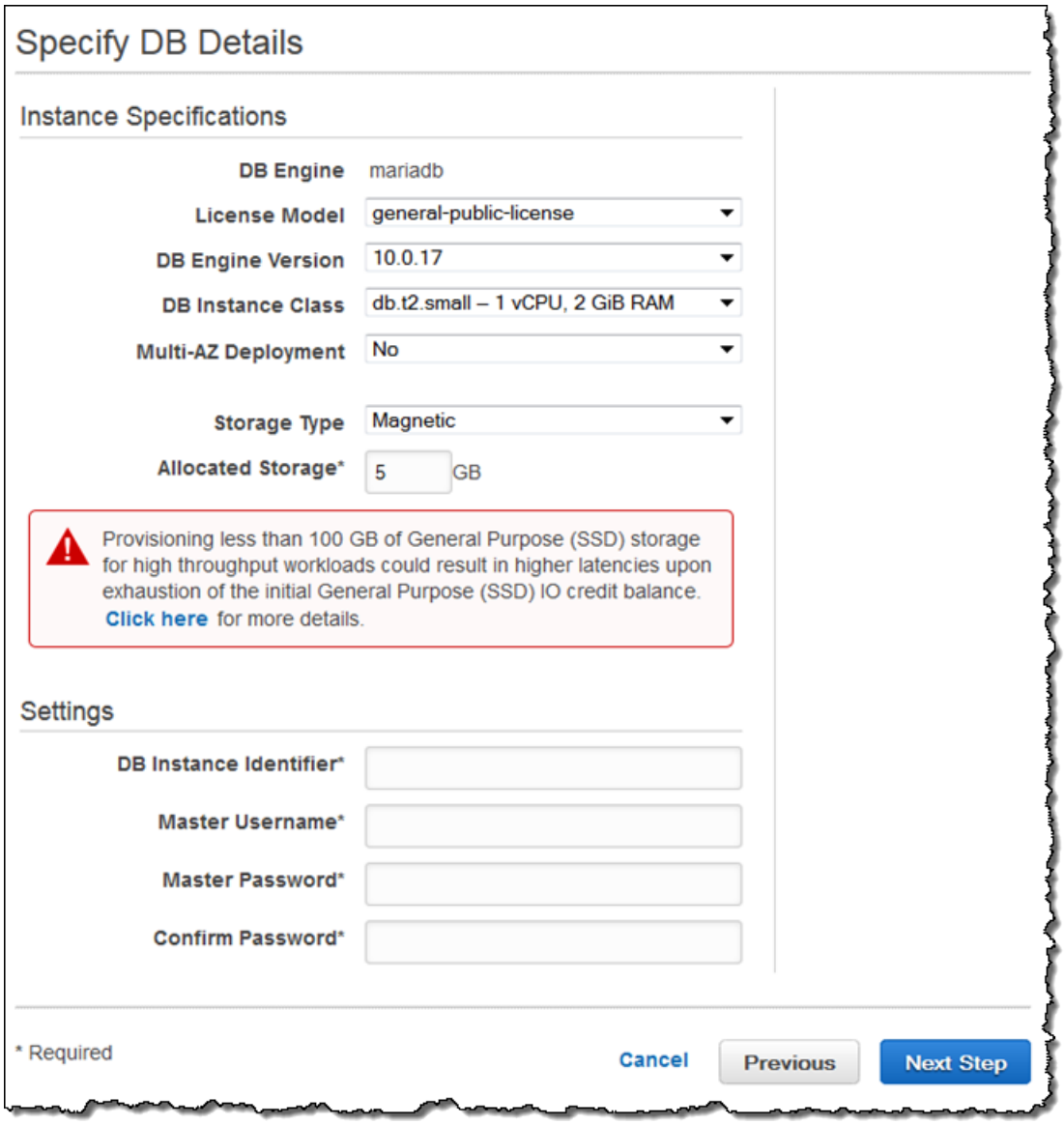
1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the top right corner of the AWS Management Console, choose the region in which you want to create the DB instance.
3. In the navigation pane, choose **Instances**.
4. Choose **Launch DB Instance** to start the **Launch DB Instance Wizard**.

   The wizard opens on the **Select Engine** page.



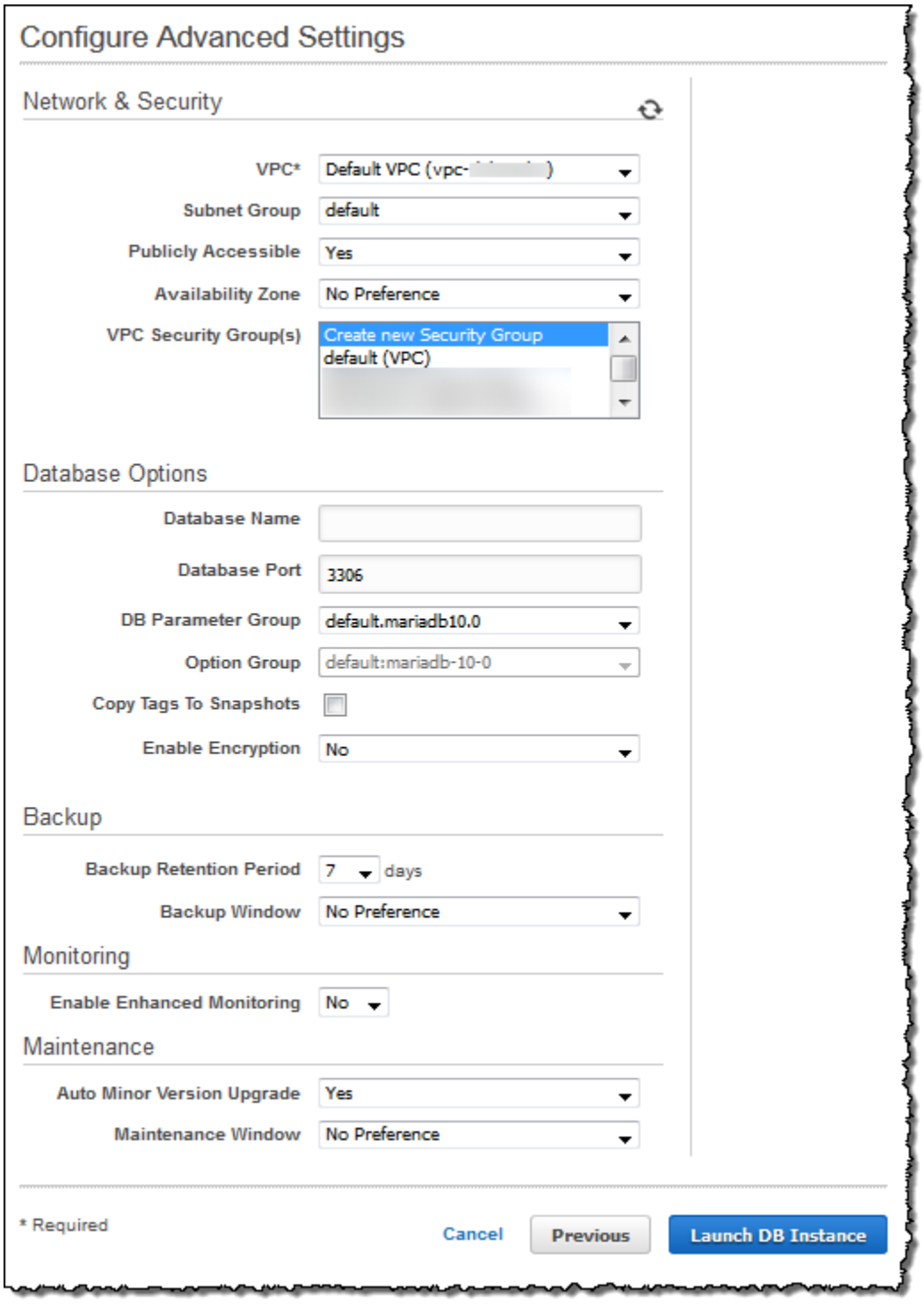
5. In the **Launch DB Instance Wizard** window, choose **Select** for the MariaDB DB engine.
6. The **Production?** step asks if you are planning to use the DB instance you are creating for production. If you are, choose **Yes**. If you choose **Yes**, the failover option **Multi-AZ** and the **Provisioned IOPS** storage option are preselected in the following step. We recommend these features for any production environment.
7. Choose **Next** to continue. The **Specify DB Details** page appears.

   On the **Specify DB Details** page, specify your DB instance information. For information about each setting, see Settings for MariaDB DB Instances (p. 683).



8. Choose **Next** to continue. The **Configure Advanced Settings** page appears.

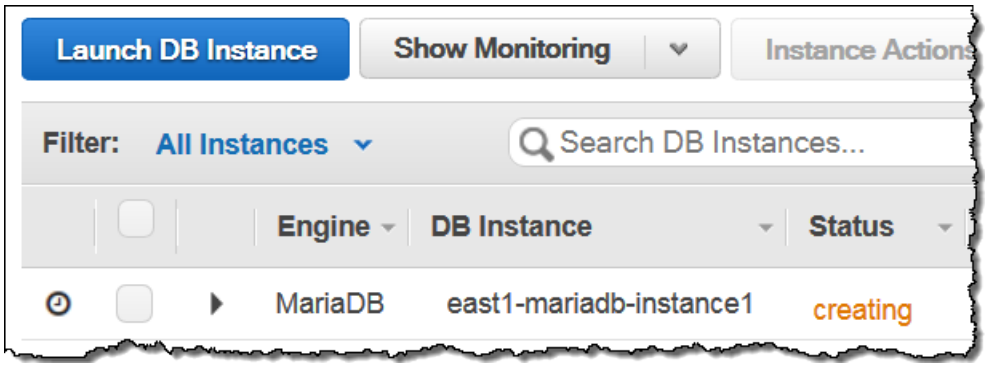
   On the **Configure Advanced Settings** page, provide additional information that Amazon RDS needs to launch the DB instance. For information about each setting, see Settings for MySQL DB Instances (p. 835).



9. Choose **Launch DB Instance** to create your MariaDB DB instance.
10. On the final page of the wizard, choose **Close**.

On the Amazon RDS console, the new DB instance appears in the list of DB instances. The DB instance has a status of **creating** until the DB instance is created and ready for use. When the state changes to

available, you can connect to the DB instance. Depending on the DB instance class and store allocated, it can take several minutes for the new instance to be available.



CLI

To create a MariaDB DB instance by using the AWS CLI, call the create-db-instance command with the parameters below. For information about each setting, see Settings for MariaDB DB Instances (p. 683).

- `--db-instance-identifier`
- `--db-instance-class`
- `--db-security-groups`
- `--db-subnet-group`
- `--engine`
- `--master-user-name`
- `--master-user-password`
- `--allocated-storage`
- `--backup-retention-period`

Example

The following command creates a MariaDB instance named *mydbinstance*.

For Linux, OS X, or Unix:

```
aws rds create-db-instance \
    --db-instance-identifier mydbinstance \
    --db-instance-class db.m1.small \
    --engine mariadb \
    --allocated-storage 20 \
    --master-username masteruser \
    --master-user-password masteruserpassword \
    --backup-retention-period 3
```

For Windows:

```
aws rds create-db-instance ^
    --db-instance-identifier mydbinstance ^
    --db-instance-class db.m1.small ^
    --engine mariadb ^
    --allocated-storage 20 ^
    --master-username masteruser ^
    --master-user-password masteruserpassword ^
    --backup-retention-period 3
```

This command should produce output similar to the following:

```
DBINSTANCE  mydbinstance  db.m1.small  mariadb  20  sa  creating  3  ****  n  10.0.17
SECGROUP  default  active
PARAMGRP  default.mariadb10.0  in-sync
```

API

To create a MariaDB DB instance by using the Amazon RDS API, call the CreateDBInstance action with the parameters below. For information about each setting, see Settings for MariaDB DB Instances (p. 683).

- `AllocatedStorage`
- `BackupRetentionPeriod`
- `DBInstanceClass`
- `DBInstanceIdentifier`
- `DBSecurityGroups`
- `DBSubnetGroup`
- `Engine`
- `MasterUsername`
- `MasterUserPassword`

Example

```
https://rds.us-west-2.amazonaws.com/
    ?Action=CreateDBInstance
    &AllocatedStorage=20
    &BackupRetentionPeriod=3
    &DBInstanceClass=db.m3.medium
    &DBInstanceIdentifier=mydbinstance
    &DBName=mydatabase
    &DBSecurityGroups.member.1=mysecuritygroup
    &DBSubnetGroup=mydbsubnetgroup
    &Engine=mariadb
    &MasterUserPassword=masteruserpassword
    &MasterUsername=masterawsuser
    &Version=2014-10-31
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20140213/us-west-2/rds/aws4_request
    &X-Amz-Date=20140213T162136Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=8052a76dfb18469393c5f0182cdab0ebc224a9c7c5c949155376c1c250fc7ec3
```

Settings for MariaDB DB Instances

The following table contains details about settings that you choose when you create a Maria DB instance.

Setting	Setting Description
Allocated Storage	The amount of storage to allocate for your DB instance (in gigabytes). In some cases, allocating a higher amount of storage for your DB instance than the size of your database can improve I/O performance. For more information, see Storage for Amazon RDS (p. 410).

Setting	Setting Description
Auto Minor Version Upgrade	**Yes** to enable your DB instance to receive minor DB engine version upgrades automatically when they become available.
Availability Zone	The availability zone for your DB instance. Use the default value of **No Preference** unless you want to specify an Availability Zone. For more information, see Regions and Availability Zones (p. 97).
Backup Retention Period	The number of days that you want automatic backups of your DB instance to be retained. For any non-trivial DB instance, you should set this value to **1** or greater. For more information, see Working With Backups (p. 201).
Backup Window	The time period during which Amazon RDS automatically takes a backup of your DB instance. Unless you have a specific time that you want to have your database backup, use the default of **No Preference**. For more information, see Working With Backups (p. 201).
Copy Tags To Snapshots	Select this option to copy any DB instance tags to a DB snapshot when you create a snapshot. For more information, see Tagging Amazon RDS Resources (p. 129).
Database Name	The name for the database on your DB instance. The name must contain 1 to 64 alpha-numeric characters. If you do not provide a name, Amazon RDS does not create a database on the DB instance you are creating. To create additional databases on your DB instance, connect to your DB instance and use the SQL command CREATE DATABASE. For more information, see Connecting to a DB Instance Running the MariaDB Database Engine (p. 688).
Database Port	The port that you want to access the DB instance through. MariaDB installations default to port 3306. If you use a DB security group with your DB instance, this must be the same port value you provided when creating the DB security group. The firewalls at some companies block connections to the default MariaDB port. If your company firewall blocks the default port, choose another port for your DB instance.
DB Engine Version	The version of MariaDB that you want to use.

Setting	Setting Description
DB Instance Class	The configuration for your DB instance. For example, a **db.m1.small** instance class equates to 1.7 GB memory, 1 ECU (1 virtual core with 1 ECU), 64-bit platform, and moderate I/O capacity. If possible, choose an instance class large enough that a typical query working set can be held in memory. When working sets are held in memory the system can avoid writing to disk, and this improves performance. For more information, see DB Instance Class (p. 92).
DB Instance Identifier	The name for your DB instance. Your DB instance identifier can contain up to 63 alphanumeric characters, and must be unique for your account in the region you chose. You can add some intelligence to the name, such as including the region you chose, for example **`mariadb-instance1`**.
DB Parameter Group	A parameter group for your DB instance. You can choose the default parameter group or you can create a custom parameter group. For more information, see Working with DB Parameter Groups (p. 170).
Enable Encryption	**Yes** to enable encryption at rest for this DB instance. For more information, see Encrypting Amazon RDS Resources (p. 355).
Enable Enhanced Monitoring	**Yes** to gather metrics in real time for the operating system that your DB instance runs on. For more information, see Enhanced Monitoring (p. 258).
License Model	MariaDB has only one license model, **`General-Public-License`** the general license agreement for MariaDB.
Maintenance Window	The 30 minute window in which pending modifications to your DB instance are applied. If the time period doesn't matter, choose **No Preference**. For more information, see The Amazon RDS Maintenance Window (p. 103).
Master Username	The name that you use as the master user name to log on to your DB Instance. For more information, and a list of the default privileges for the master user, see MariaDB Security on Amazon RDS (p. 673).
Master User Password	The password for your master user account. The password must contain from 8 to 41 printable ASCII characters (excluding /,", a space, and @).

Setting	Setting Description
Multi-AZ Deployment	**Yes** to create a standby mirror of your DB instance in another Availability Zone for failover support. We recommend Multi-AZ for production workloads to maintain high availability. For development and testing, you can choose **No**. For more information, see High Availability (Multi-AZ) (p. 99).
Option Group	An option group for your DB instance. You can choose the default option group or you can create a custom option group. For more information, see Working with Option Groups (p. 153).
Publicly Accessible	**Yes** to give your DB instance a public IP address. This means that it is accessible outside the VPC (the DB instance also needs to be in a public subnet in the VPC). Choose **No** if you want the DB instance to only be accessible from inside the VPC. For more information, see Hiding a DB Instance in a VPC from the Internet (p. 401).
Storage Type	The storage type for your DB instance. For more information, see Amazon RDS Storage Types (p. 410).
Subnet Group	This setting depends on the platform you are on. If you are a new customer to AWS, choose **default**, which is the default DB subnet group that was created for your account. If you are creating a DB instance on the previous E2-Classic platform and you want your DB instance in a specific VPC, choose the DB subnet group you created for that VPC.
VPC	This setting depends on the platform you are on. If you are a new customer to AWS, choose the default VPC shown. If you are creating a DB instance on the previous E2-Classic platform that does not use a VPC, choose **Not in VPC**. For more information, see Amazon Virtual Private Cloud (VPCs) and Amazon RDS (p. 390).
VPC Security Group	If you are a new customer to AWS, choose the default VPC. Otherwise, choose the VPC security group you previously created. For more information, see Working with DB Security Groups (EC2-Classic Platform) (p. 380).

Related Topics

- Tutorial: Create an Amazon VPC for Use with an Amazon RDS DB Instance (p. 406)
- Connecting to a DB Instance Running the MariaDB Database Engine (p. 688)

- Modifying a DB Instance Running the MariaDB Database Engine (p. 691)
- Deleting a DB Instance (p. 126)

Connecting to a DB Instance Running the MariaDB Database Engine

Once Amazon RDS provisions your DB instance, you can use any standard MariaDB client application or utility to connect to the instance. In the connection string, you specify the DNS address from the DB instance endpoint as the host parameter, and specify the port number from the DB instance endpoint as the port parameter.

You can use the AWS Management Console, the AWS CLI describe-db-instances command, or the Amazon RDS API DescribeDBInstances action to list the details of an Amazon RDS DB instance, including its endpoint. If an endpoint value is `myinstance.123456789012.us-east-1.rds.amazonaws.com:3306`, then you specify the following values in a MariaDB connection string:

- For host or host name, specify `myinstance.123456789012.us-east-1.rds.amazonaws.com`
- For port, specify `3306`

You can connect to an Amazon RDS MariaDB DB instance by using tools like the `mysql` command line utility. For more information on using the `mysql` utility, go to mysql Command-line Client in the MariaDB documentation. One GUI-based application you can use to connect is HeidiSQL; for more information, go to the Download HeidiSQL page.

Two common causes of connection failures to a new DB instance are the following:

- The DB instance was created using a security group that does not authorize connections from the device or Amazon EC2 instance where the MariaDB application or utility is running. If the DB instance was created in an Amazon VPC, it must have a VPC security group that authorizes the connections. If the DB instance was created outside of a VPC, it must have a DB security group that authorizes the connections.
- The DB instance was created using the default port of 3306, and your company has firewall rules blocking connections to that port from devices in your company network. To fix this failure, recreate the instance with a different port.

You can use SSL encryption on connections to an Amazon RDS MariaDB DB instance. For information, see SSL Support for MariaDB DB Instances (p. 674).

Connecting from the mysql Utility

To connect to a DB instance using the mysql utility, type the following command at a command prompt on a client computer to connect to a database on a MariaDB DB instance. Substitute the DNS name for your DB instance for *<endpoint>*, the master user name you used for *<mymasteruser>*, and provide the master password you used when prompted for a password.

```
mysql -h <endpoint> -P 3306 -u <mymasteruser> -p
```

You will see output similar to the following.

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 272
Server version: 5.5.5-10.0.17-MariaDB-log MariaDB Server

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
```

```
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql >
```

Connecting with SSL

Amazon RDS creates an SSL certificate for your DB instance when the instance is created. If you enable SSL certificate verification, then the SSL certificate includes the DB instance endpoint as the Common Name (CN) for the SSL certificate to guard against spoofing attacks. To connect to your DB instance using SSL, follow these steps:

To connect to a DB instance with SSL using the mysql utility

1. Download a root certificate that works for all regions from here.
2. Type the following command at a command prompt to connect to a DB instance with SSL using the `mysql` utility. For the `-h` parameter, substitute the DNS name for your DB instance. For the `--ssl-ca` parameter, substitute the SSL certificate file name as appropriate.

   ```
   mysql -h myinstance.123456789012.us-east-1.rds.amazonaws.com --ssl-ca=rds-ca-2015-
   root.pem
   ```

3. Include the `--ssl-verify-server-cert` parameter so that the SSL connection verifies the DB instance endpoint against the endpoint in the SSL certificate. For example:

   ```
   mysql -h myinstance.123456789012.us-east-1.rds.amazonaws.com --ssl-ca=rds-ca-2015-
   root.pem --ssl-verify-server-cert
   ```

4. Type the master user password when prompted.

You will see output similar to the following.

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 272
Server version: 5.5.5-10.0.17-MariaDB-log MariaDB Server

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql >
```

Maximum MariaDB Connections

The maximum number of connections allowed to an Amazon RDS MariaDB DB instance is based on the amount of memory available for the DB instance class of the DB instance. A DB instance class with more memory available results in a larger number of connections available. For more information on DB instance classes, see DB Instance Class (p. 92).

The connection limit for a DB instance is set by default to the maximum for the DB instance class for the DB instance. You can limit the number of concurrent connections to any value up to the maximum

number of connections allowed using the `max_connections` parameter in the parameter group for the DB instance. For more information, see Working with DB Parameter Groups (p. 170).

You can retrieve the maximum number of connections allowed for an Amazon RDS MariaDB DB instance by executing the following query on your DB instance:

```
SELECT @@max_connections;
```

You can retrieve the number of active connections to an Amazon RDS MariaDB DB instance by executing the following query on your DB instance:

```
SHOW STATUS WHERE `variable_name` = 'Threads_connected';
```

Related Topics

- Amazon RDS DB Instances (p. 90)
- Creating a DB Instance Running the MariaDB Database Engine (p. 678)
- Amazon RDS Security Groups (p. 375)
- Deleting a DB Instance (p. 126)

Modifying a DB Instance Running the MariaDB Database Engine

You can change the settings of a DB instance to accomplish tasks such as adding additional storage or changing the DB instance class. This topic guides you through modifying an Amazon RDS MariaDB DB instance, and describes the settings for MariaDB instances.

We recommend that you test any changes on a test instance before modifying a production instance, so that you fully understand the impact of each change. This is especially important when upgrading database versions.

After you modify your DB instance settings, you can apply the changes immediately, or apply them during the next maintenance window for the DB instance. Some modifications cause an interruption by restarting the DB instance.

AWS Management Console

To modify a MariaDB DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **DB Instances**, and then select the DB instance that you want to modify.
3. Choose **Instance Actions**, and then choose **Modify**. The **Modify DB Instance** page appears.
4. Change any of the settings that you want. For information about each setting, see Settings for MariaDB DB Instances (p. 692).
5. To apply the changes immediately, select **Apply Immediately**. Selecting this option can cause an outage in some cases. For more information, see The Impact of Apply Immediately (p. 114).
6. When all the changes are as you want them, choose **Continue**.
7. On the confirmation page, review your changes. If they are correct, choose **Modify DB Instance** to save your changes.

   Alternatively, choose **Back** to edit your changes, or choose **Cancel** to cancel your changes.

CLI

To modify a MariaDB DB instance by using the AWS CLI, call the modify-db-instance command. Specify the DB instance identifier, and the parameters for the settings that you want to modify. For information about each parameter, see Settings for MariaDB DB Instances (p. 692).

Example

The following code modifies `mydbinstance` by setting the backup retention period to 1 week (7 days). The code disables automatic minor version upgrades by using `--no-auto-minor-version-upgrade`. To allow automatic minor version upgrades, use `--auto-minor-version-upgrade`. The changes are applied during the next maintenance window by using `--no-apply-immediately`. Use `--apply-immediately` to apply the changes immediately. For more information, see The Impact of Apply Immediately (p. 114).

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
```

```
    --db-instance-identifier mydbinstance \
    --backup-retention-period 7 \
    --no-auto-minor-version-upgrade \
    --no-apply-immediately
```

For Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --backup-retention-period 7 ^
    --no-auto-minor-version-upgrade ^
    --no-apply-immediately
```

API

To modify a MariaDB instance by using the Amazon RDS API, call the ModifyDBInstance action. Specify the DB instance identifier, and the parameters for the settings that you want to modify. For information about each parameter, see Settings for MariaDB DB Instances (p. 692).

Example

The following code modifies `mydbinstance` by setting the backup retention period to 1 week (7 days) and disabling automatic minor version upgrades. These changes are applied during the next maintenance window.

```
https://rds.amazonaws.com/
    ?Action=ModifyDBInstance
    &ApplyImmediately=false
    &AutoMinorVersionUpgrade=false
    &BackupRetentionPeriod=7
    &DBInstanceIdentifier=mydbinstance
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-10-31
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20131016/us-west-1/rds/aws4_request
    &X-Amz-Date=20131016T233051Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=087a8eb41cb1ab0fc9ec1575f23e73757ffc6a1e42d7d2b30b9cc0be988cff97
```

Settings for MariaDB DB Instances

The following table contains details about which settings you can modify, which settings you can't modify, when the changes can be applied, and whether the changes cause downtime for the DB instance.

Setting	Setting Description	When the Change Occurs	Downtime Notes
Allocated Storage	The storage, in gigabytes, that you want to allocate for your DB instance. You can't modify allocated storage if the DB instance status is `storage-optimization` or if the allocated storage for the DB instance has been modified in the last six hours.	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	No downtime. Performance may be degraded during the change.

Setting	Setting Description	When the Change Occurs	Downtime Notes
	The maximum storage allowed depends on the storage type. For more information, see Storage for Amazon RDS (p. 410).		
Auto Minor Version Upgrade	**Yes** if you want your DB instance to receive minor engine version upgrades automatically when they become available. Upgrades are installed only during your scheduled maintenance window.	–	–
Backup Retention Period	The number of days that automatic backups are retained. To disable automatic backups, set the backup retention period to 0. For more information, see Working With Backups (p. 201).	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false and you change the setting from a non-zero value to another non-zero value, the change is applied asynchronously, as soon as possible. Otherwise, the change occurs during the next maintenance window.	An outage occurs if you change from 0 to a non-zero value, or from a non-zero value to 0.
Backup Window	The time range during which automated backups of your databases occur. The backup window is a start time in Universal Coordinated Time (UTC), and a duration in hours. For more information, see Working With Backups (p. 201).	The change is applied asynchronously, as soon as possible.	–
Certificate Authority	The certificate that you want to use.	–	–
Copy Tags to Snapshots	If you have any DB instance tags, this option copies them when you create a DB snapshot. For more information, see Tagging Amazon RDS Resources (p. 129).	–	–
Database Port	The port that you want to use to access the database. The port value must not match any of the port values specified for options in the option group for the DB instance.	The change occurs immediately. This setting ignores the **Apply Immediately** setting.	The DB instance is rebooted immediately.

Setting	Setting Description	When the Change Occurs	Downtime Notes
DB Engine Version	The version of the MariaDB database engine that you want to use. Before you upgrade your production DB instances, we recommend that you test the upgrade process on a test instance to verify its duration and to validate your applications. For more information, see Upgrading the MariaDB DB Engine (p. 699).	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	An outage occurs during this change.
DB Instance Class	The DB instance class that you want to use. For more information, see DB Instance Class (p. 92).	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	An outage occurs during this change.
DB Instance Identifier	The DB instance identifier. This value is stored as a lowercase string. For more information about the effects of renaming a DB instance, see Renaming a DB Instance (p. 116).	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	An outage occurs during this change. The DB instance is rebooted.
DB Parameter Group	The parameter group that you want associated with the DB instance. For more information, see Working with DB Parameter Groups (p. 170).	The parameter group change occurs immediately. However, parameter changes only occur when you reboot the DB instance manually without failover. For more information, see Rebooting a DB Instance (p. 119).	An outage doesn't occur during this change. However, parameter changes only occur when you reboot the DB instance manually without failover.
Enable Enhanced Monitoring	**Yes** to enable gathering metrics in real time for the operating system that your DB instance runs on. For more information, see Enhanced Monitoring (p. 258).	–	–

Setting	Setting Description	When the Change Occurs	Downtime Notes
Maintenance Window	The time range during which system maintenance occurs. System maintenance includes upgrades, if applicable. The maintenance window is a start time in Universal Coordinated Time (UTC), and a duration in hours. If you set the window to the current time, there must be at least 30 minutes between the current time and end of the window to ensure any pending changes are applied. For more information, see The Amazon RDS Maintenance Window (p. 103).	The change occurs immediately. This setting ignores the **Apply Immediately** setting.	If there are one or more pending actions that cause an outage, and the maintenance window is changed to include the current time, then those pending actions are applied immediately, and an outage occurs.
Multi-AZ Deployment	**Yes** to deploy your DB instance in multiple Availability Zones; otherwise, **No**. For more information, see Regions and Availability Zones (p. 97).	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	–
New Master Password	The password for your master user. The password must contain from 8 to 41 alphanumeric characters.	The change is applied asynchronously, as soon as possible. This setting ignores the **Apply Immediately** setting.	–
Option Group	The option group that you want associated with the DB instance. For more information, see Working with Option Groups (p. 153).	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	–
Publicly Accessible	**Yes** to give the DB instance a public IP address, meaning that it is accessible outside the VPC. To be publicly accessible, the DB instance also has to be in a public subnet in the VPC. **No** to make the DB instance accessible only from inside the VPC. For more information, see Hiding a DB Instance in a VPC from the Internet (p. 401).	The change occurs immediately. This setting ignores the **Apply Immediately** setting.	–

Setting	Setting Description	When the Change Occurs	Downtime Notes
Security Group	The security group you want associated with the DB instance. For more information, see Working with DB Security Groups (EC2-Classic Platform) (p. 380).	The change is applied asynchronously, as soon as possible. This setting ignores the **Apply Immediately** setting.	–

Setting	Setting Description	When the Change Occurs	Downtime Notes
Storage Type	The storage type that you want to use. For more information, see Amazon RDS Storage Types (p. 410).	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	The following changes all result in a brief outage while the process starts. After that, you can use your database normally while the change takes place. • From **General Purpose (SSD)** to **Magnetic**. • From **General Purpose (SSD)** to **Provisioned IOPS (SSD)**, if the DB instance is single-AZ or if you are using a custom parameter group and the DB instance is a read replica. There is no outage for a multi-AZ DB instance or for the source DB instance of a read replica. • From **Magnetic** to **General Purpose (SSD)**. • From **Magnetic** to **Provisioned IOPS (SSD)**. • From **Provisioned IOPS (SSD)** to **Magnetic**. • From **Provisioned IOPS (SSD)** to **General Purpose (SSD)**, if the DB instance is single-AZ or if you are using a custom parameter group and the DB instance is a read replica. There is no outage for a multi-AZ

Setting	Setting Description	When the Change Occurs	Downtime Notes
			DB instance or for the source DB instance of a read replica.
Subnet Group	The subnet group for the DB instance. You can use this setting to move your DB instance to a different VPC. If your DB instance is not in a VPC, you can use this setting to move your DB instance into a VPC. For more information, see Moving a DB Instance Not in a VPC into a VPC (p. 405).	–	–

Related Topics

- Rebooting a DB Instance (p. 119)
- Connecting to a DB Instance Running the MariaDB Database Engine (p. 688)
- Upgrading the MariaDB DB Engine (p. 699)
- Deleting a DB Instance (p. 126)

Upgrading the MariaDB DB Engine

When Amazon Relational Database Service (Amazon RDS) supports a new version of a database engine, you can upgrade your DB instances to the new version. There are two kinds of upgrades: major version upgrades and minor version upgrades. You must modify the DB instance manually to perform a major version upgrade.

For more information about MariaDB supported versions and version management, see MariaDB on Amazon RDS Versions (p. 667).

Overview of Upgrading

Major version upgrades can contain database changes that are not backward-compatible with existing applications. As a result, Amazon Relational Database Service (Amazon RDS) doesn't apply major version upgrades automatically; you must manually modify your DB instance. You should thoroughly test any upgrade before applying it to your production instances.

Minor version upgrades that contain database changes that are backward-compatible with the previous version might be applied automatically. Amazon RDS doesn't automatically upgrade an Amazon RDS DB instance until after posting an announcement to the forums announcement page, and sending customers an e-mail notification. Automatic upgrades are scheduled so that you can plan around them, because downtime is required to upgrade a DB instance, even for Multi-AZ instances.

Amazon RDS takes two DB snapshots during the upgrade process. The first DB snapshot is of the DB instance before any upgrade changes have been made. If the upgrade doesn't work for your databases, you can restore this snapshot to create a DB instance running the old version. The second DB snapshot is taken when the upgrade completes.

After the upgrade is complete, you can't revert to the previous version of the database engine. If you want to return to the previous version, restore the first DB snapshot taken to create a new DB instance.

You control when to upgrade your DB instance to a new version supported by Amazon RDS. This level of control helps you maintain compatibility with specific database versions and test new versions with your application before deploying in production. When you are ready, you can perform version upgrades at the times that best fit your schedule.

If your DB instance is using read replication, you must upgrade all of the Read Replicas before upgrading the source instance.

If your DB instance is in a Multi-AZ deployment, both the primary and standby replicas are upgraded. The primary and standby DB instances are upgraded at the same time and you will experience an outage until the upgrade is complete. The time for the outage varies based on your database engine, engine version, and the size of your DB instance.

AWS Management Console

To upgrade the engine version of a DB instance by using the AWS Management Console

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Instances**.
3. Choose the check box for the DB instance that you want to upgrade.
4. Choose **Instance Actions**, and then choose **Modify**.
5. For **DB Engine Version**, choose the new version.

6. To upgrade immediately, select **Apply Immediately**. To delay the upgrade to the next maintenance window, clear **Apply Immediately**.
7. Choose **Continue**.
8. Review the modification summary information. To proceed with the upgrade, choose **Modify DB Instance**. To cancel the upgrade, choose **Cancel** or **Back**.

CLI

To upgrade the engine version of a DB instance, use the AWS CLI modify-db-instance command. Specify the following parameters:

- `--db-instance-identifier` – the name of the db instance.
- `--engine-version` – the version number of the database engine to upgrade to.
- `--no-apply-immediately` – apply changes during the next maintenance window. To apply changes immediately, use `--apply-immediately`.

Example

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier <mydbinstance> \
    --engine-version <new_version> \
    --allow-major-version-upgrade \
    --apply-immediately
```

For Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier <mydbinstance> ^
    --engine-version <new_version> ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

API

To upgrade the engine version of a DB instance, use the ModifyDBInstance action. Specify the following parameters:

- `DBInstanceIdentifier` – the name of the db instance, for example *`mydbinstance`*.
- `EngineVersion` – the version number of the database engine to upgrade to.
- `ApplyImmediately` – whether to apply changes immediately or during the next maintenance window. To apply changes immediately, set the value to *`true`*. To apply changes during the next maintenance window, set the value to *`false`*.

Example

```
https://rds.us-east-1.amazonaws.com/
   ?Action=ModifyDBInstance
   &ApplyImmediately=false
   &DBInstanceIdentifier=mydbinstance
   &EngineVersion=new_version
   &SignatureMethod=HmacSHA256
```

```
&SignatureVersion=4
&Version=2013-09-09
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20131016/us-east-1/rds/aws4_request
&X-Amz-Date=20131016T233051Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=087a8eb41cb1ab5f99e81575f23e73757ffc6a1e42d7d2b30b9cc0be988cff97
```

Related Topics

- Amazon RDS Maintenance (p. 102)
- Updating the Operating System for a DB Instance or DB Cluster (p. 108)

Migrating Data from a MySQL DB Snapshot to a MariaDB DB Instance

You can migrate an Amazon RDS MySQL DB snapshot to a new DB instance running MariaDB 10.1 using the AWS Management Console, AWS CLI, or Amazon RDS API. You must create the DB snapshot from an Amazon RDS DB instance running MySQL 5.6. To learn how to create an RDS MySQL DB snapshot, see Creating a DB Snapshot (p. 207).

After you migrate from MySQL to MariaDB, the MariaDB DB instance will be associated with the default DB parameter group and option group. After you restore the DB snapshot, you can associate a custom DB parameter group for the new DB instance. However, a MariaDB parameter group has a different set of configurable system variables. For information about the differences between MySQL and MariaDB system variables, see System Variable Differences Between MariaDB 10.0 and MySQL 5.6. To learn about DB parameter groups, see Working with DB Parameter Groups (p. 170). To learn about option groups, see Working with Option Groups (p. 153).

Incompatibilities Between MariaDB and MySQL

Incompatibilities between MySQL and MariaDB include the following:

- You cannot migrate a DB snapshot created with MySQL 5.7 or MySQL 5.5 to MariaDB 10.1.
- If the source MySQL database uses a SHA256 password hash, you need to reset user passwords that are SHA256 hashed before you can connect to the MariaDB database. The following code shows how to reset a password that is SHA256 hashed:

```
SET old_passwords = 0;
UPDATE mysql.user SET plugin = 'mysql_native_password',
Password = PASSWORD('new_password')
WHERE (User, Host) = ('master_user_name', %);
FLUSH PRIVILEGES;
```

- If your RDS master user account uses the SHA-256 password hash, the password has to be reset using the rds `modify-db-instance` AWS CLI command, ModifyDBInstance API action, or the AWS Management Console. For information about modifying a MariaDB DB instance, see Modifying a DB Instance Running the MariaDB Database Engine (p. 691).
- MariaDB doesn't support the Memcached plugin; however, the data used by the Memcached plugin is stored as InnoDB tables. After you migrate a MySQL DB snapshot, you can access the data used by the Memcached plugin using SQL. For more information about the innodb_memcache database, see InnoDB memcached Plugin Internals.

AWS Management Console

To migrate a MySQL DB snapshot to a MariaDB DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Snapshots**.
3. Choose **Migrate Snapshot**.
4. For **Migrate to DB Engine**, choose **MariaDB**.
5. On the **Migrate Database** page, provide additional information that RDS needs to launch the MariaDB DB instance.

- **DB Instance Class**: Choose a DB instance class that has the required storage and capacity for your database, for example db.r3.large. For any production application that requires fast and consistent I/O performance, we recommend Provisioned IOPS storage. For more information, see Provisioned IOPS Storage (p. 413). MariaDB 10.1 does not support previous generation DB instance classes. For more information, see DB Instance Class (p. 92).
- **DB Snapshot ID**: Type a name for the DB snapshot identifier.

  The DB snapshot identifier has the following constraints:

  - It must contain from 1 to 255 alphanumeric characters or hyphens.
  - The character must be a letter.
  - It cannot end with a hyphen or contain two consecutive hyphens.

  If you are restoring from a shared manual DB snapshot, the DB snapshot identifier must be the Amazon Resource Name (ARN) of the shared DB snapshot.
- **DB Instance Identifier**: Type a name for the DB instance that is unique for your account in the AWS Region where the DB instance will reside. This identifier is used in the endpoint addresses for the instances in your DB instance.

  The DB instance identifier has the following constraints:

  - It must contain from 1 to 63 alphanumeric characters or hyphens.
  - Its first character must be a letter.
  - It cannot end with a hyphen or contain two consecutive hyphens.
  - It must be unique for all DB instances for your AWS account, within an AWS Region.
- **VPC**: If you have an existing VPC, then you can use that VPC with your MariaDB DB instance by selecting your VPC identifier, for example `vpc-a464d1c1`. For more information about VPC, see Amazon Virtual Private Cloud (VPCs) and Amazon RDS (p. 390) .

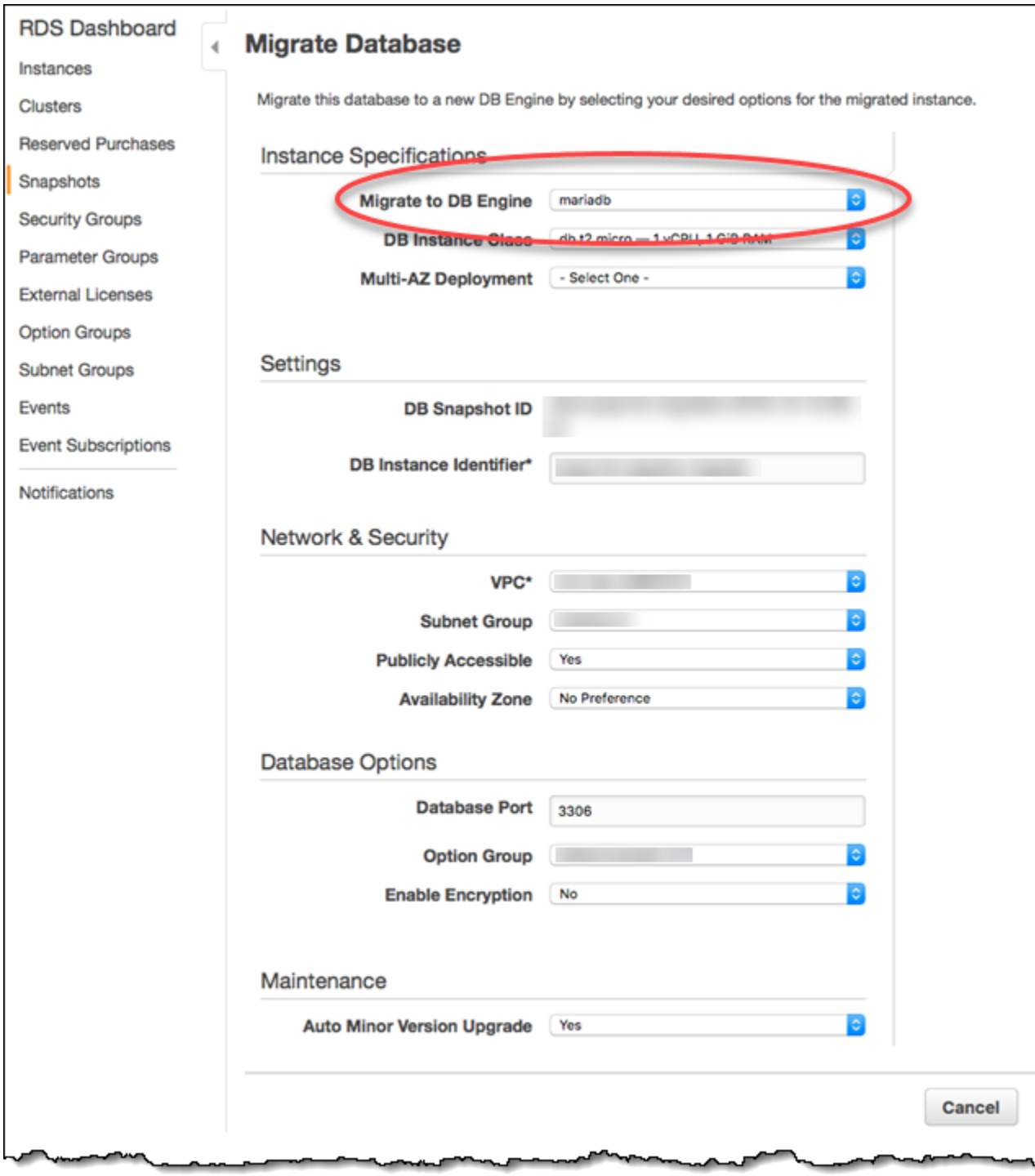
  Otherwise, you can choose to have Amazon RDS create a VPC for you by selecting Create a new VPC.

  You cannot create MariaDB instances in the EC2 Classic Network.
- **Subnet Group**: If you have an existing subnet group, then you can use that subnet group with your MariaDB DB instance by selecting your subnet group identifier, for example `gs-subnet-group1`.

  Otherwise, you can choose to have Amazon RDS create a subnet group for you by selecting Create a new subnet group.
- **Publicly Accessible**: Choose **No** to specify that instances in your DB instance can only be accessed by resources inside your VPC. Choose **Yes** to specify that instances in your DB instance can be accessed by resources on the public network. The default is **Yes**.
- **Availability Zone**: Choose the **Availability Zone** to host the primary instance for your MariaDB DB instance. To have Amazon RDS choose an **Availability Zone** for you, choose **No Preference**.
- **Database Port**: Type the default port to be used when connecting to instances in the DB instance. The default is `3306`.

  You might be behind a corporate firewall that doesn't allow access to default ports such as the MySQL default port `3306`. In this case, provide a port value that your corporate firewall allows.
- **Enable Encryption**: Choose **Yes** for your new MariaDB DB instance to be encrypted "at rest." If you choose **Yes**, you will be required to choose an AWS KMS encryption key as the **Master Key** value.
- **Auto Minor Version Upgrade**: Choose **Yes** if you want to enable your MariaDB DB instance to receive minor MySQL DB engine version upgrades automatically when they become available. The **Auto Minor Version Upgrade** option only applies to upgrades to MySQL minor engine versions for your MariaDB DB instance. It doesn't apply to regular patches applied to maintain system stability.

6. Choose **Migrate**.



CLI

To migrate data from a MySQL DB snapshot to a MariaDB DB instance, use the AWS CLI `restore-db-instance-from-db-snapshot` command with the following parameters:

- --db-instance-identifier – Name of the DB instance to create from the DB snapshot.
- --db-snapshot-identifier – The identifier for the DB snapshot to restore from.
- --engine – The database engine to use for the new instance.

Example

For Linux, OS X, or Unix:

```
aws rds restore-db-instance-from-db-snapshot \
    --db-instance-identifier newmariadbinstance \
    --db-snapshot-identifier mysqlsnapshot \
    --engine mariadb
```

For Windows:

```
aws rds restore-db-instance-from-db-snapshot \
    --db-instance-identifier newmariadbinstance ^
    --db-snapshot-identifier mysqlsnapshot ^
    --engine mariadb
```

API

To migrate data from a MySQL DB snapshot to a MariaDB DB instance, call the Amazon RDS API action `RestoreDBInstanceFromDBSnapshot`.

Example

```
https://rds.us-west-2.amazonaws.com/
   ?Action=RestoreDBInstanceFromDBSnapshot
   &DBInstanceIdentifier= newmariadbinstance
   &DBSnapshotIdentifier= mysqlsnapshot
   &Engine= mariadb
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &Version=2013-09-09
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20140428/us-west-2/rds/aws4_request
   &X-Amz-Date=20140428T232655Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
   &X-Amz-Signature=78ac761e8c8f54a8c0727f4e67ad0a766fbb0024510b9aa34ea6d1f7df52fe92
```

Related Topics

- Creating a DB Snapshot (p. 207)
- System Variable Differences Between MariaDB 10.0 and MySQL 5.6
- Working with DB Parameter Groups (p. 170)
- Working with Option Groups (p. 153)

Importing Data into a MariaDB DB Instance

Following, you can find information about methods to import your MariaDB data to an Amazon RDS DB instance running MariaDB.

To do an initial data import into a MariaDB DB instance, you can use the procedures documented in Importing Data into an Amazon RDS MySQL DB Instance (p. 860), as follows:

- To move data from an Amazon RDS MySQL DB instance, a MariaDB or MySQL instance in Amazon Elastic Compute Cloud (Amazon EC2) in the same VPC as your Amazon RDS MariaDB DB instance, or a small on-premises instance of MariaDB or MySQL, you can use the procedure documented in Importing Data from a MySQL or MariaDB DB to an Amazon RDS MySQL or MariaDB DB Instance (p. 872).
- To move data from a large or production on-premises instance of MariaDB or MySQL, you can use the procedure documented in Importing Data to an Amazon RDS MySQL or MariaDB DB Instance with Reduced Downtime (p. 873).
- To move data from an instance of MariaDB or MySQL that is in EC2 in a different VPC than your Amazon RDS MariaDB DB instance, or to move data from any data source that can output delimited text files, you can use the procedure documented in Importing Data From Any Source to a MySQL or MariaDB DB Instance (p. 886).

You can also use AWS Database Migration Service (AWS DMS) to import data into an Amazon RDS DB instance. AWS DMS can migrate databases without downtime and, for many database engines, continue ongoing replication until you are ready to switch over to the target database. You can migrate to MariaDB from either the same database engine or a different database engine using AWS DMS. If you are migrating from a different database engine, you can use the AWS Schema Conversion Tool to migrate schema objects that are not migrated by AWS DMS. For more information about AWS DMS, see see What is AWS Database Migration Service.

You can configure replication into an Amazon RDS MariaDB DB instance using MariaDB global transaction identifiers (GTIDs) when the external instance is MariaDB version 10.0.24 or greater, or using binary log coordinates for MySQL instances or MariaDB instances on earlier versions than 10.0.24. Note that MariaDB GTIDs are implemented differently than MySQL GTIDs, which are not supported by Amazon RDS.

To configure replication into a MariaDB DB instance, you can use the following procedures:

- To configure replication into a MariaDB DB instance from an external MySQL instance or an external MariaDB instance running a version prior to 10.0.24, you can use the procedure documented in Replication with a MySQL or MariaDB Instance Running External to Amazon RDS (p. 890).
- To configure replication into a MariaDB DB instance from an external MariaDB instance running version 10.0.24 or greater, you can use the procedure documented in Configuring GTID-Based Replication into an Amazon RDS MariaDB DB instance (p. 707).

Note
The mysql system database contains authentication and authorization information required to log into your DB instance and access your data. Dropping, altering, renaming, or truncating tables, data, or other contents of the mysql database in your DB instance can result in errors and might render the DB instance and your data inaccessible. If this occurs, the DB instance can be restored from a snapshot using the AWS CLI `restore-db-instance-from-db-snapshot` or recovered using `restore-db-instance-to-point-in-time` commands.

Configuring GTID-Based Replication into an Amazon RDS MariaDB DB instance

You can set up GTID-based replication from an external MariaDB instance of version 10.0.24 or greater into an Amazon RDS MariaDB DB instance. Be sure to follow these guidelines when you set up an external replication master and a replica on Amazon RDS:

- Monitor failover events for the Amazon RDS MariaDB DB instance that is your replica. If a failover occurs, then the DB instance that is your replica might be recreated on a new host with a different network address. For information on how to monitor failover events, see Using Amazon RDS Event Notification (p. 279).
- Maintain the binlogs on your master instance until you have verified that they have been applied to the replica. This maintenance ensures that you can restore your master instance in the event of a failure.
- Turn on automated backups on your MariaDB DB instance on Amazon RDS. Turning on automated backups ensures that you can restore your replica to a particular point in time if you need to re-synchronize your master and replica. For information on backups and Point-In-Time Restore, see Backing Up and Restoring Amazon RDS DB Instances (p. 200).

Note
The permissions required to start replication on an Amazon RDS MariaDB DB instance are restricted and not available to your Amazon RDS master user. Because of this, you must use the Amazon RDS mysql.rds_set_external_master_gtid (p. 717) and mysql.rds_start_replication (p. 917) commands to set up replication between your live database and your Amazon RDS MariaDB database.

To start replication between an external master instance and a MariaDB DB instance on Amazon RDS, use the following procedure.

To Start Replication

1. Make the source MariaDB instance read-only:

```
mysql> FLUSH TABLES WITH READ LOCK;
mysql> SET GLOBAL read_only = ON;
```

2. Get the current GTID of the external MariaDB instance. You can do this by using `mysql` or the query editor of your choice to run `SELECT @@gtid_current_pos;`.

   The GTID is formatted as `<domain-id>-<server-id>-<sequence-id>`. A typical GTID looks something like **`0-1234510749-1728`**. For more information about GTIDs and their component parts, see Global Transaction ID in the MariaDB documentation.

3. Copy the database from the external MariaDB instance to the Amazon RDS MariaDB DB instance using `mysqldump`. For very large databases, you might want to use the procedure in Importing Data to an Amazon RDS MySQL or MariaDB DB Instance with Reduced Downtime (p. 873).

   Note
   Make sure there is not a space between the `-p` option and the entered password.

   For Linux, OS X, or Unix:

```
mysqldump \
    --databases <database_name> \
    --single-transaction \
    --compress \
    --order-by-primary \
```

```
    -u <local_user> \
    -p<local_password> | mysql \
        --host=hostname \
        --port=3306 \
        -u <RDS_user_name> \
        -p <RDS_password>
```

For Windows:

```
mysqldump ^
    --databases <database_name> ^
    --single-transaction ^
    --compress ^
    --order-by-primary \
    -u <local_user> \
    -p<local_password> | mysql ^
        --host=hostname ^
        --port=3306 ^
        -u <RDS_user_name> ^
        -p <RDS_password>
```

Use the `--host`, `--user (-u)`, `--port` and `-p` options in the `mysql` command to specify the host name, user name, port, and password to connect to your Amazon RDS MariaDB DB instance. The host name is the DNS name from the Amazon RDS MariaDB DB instance endpoint, for example `myinstance.123456789012.us-east-1.rds.amazonaws.com`. You can find the endpoint value in the instance details in the Amazon RDS Management Console.

4. Make the source MariaDB instance writeable again:

```
mysql> SET GLOBAL read_only = OFF;
mysql> UNLOCK TABLES;
```

5. In the Amazon RDS Management Console, add the IP address of the server that hosts the external MariaDB database to the VPC security group for the Amazon RDS MariaDB DB instance. For more information on modifying a VPC security group, go to Security Groups for Your VPC in the *Amazon Virtual Private Cloud User Guide*.

   You might also need to configure your local network to permit connections from the IP address of your Amazon RDS MariaDB DB instance, so that it can communicate with your external MariaDB instance. To find the IP address of the Amazon RDS MariaDB DB instance, use the `host` command:

```
host <RDS_MariaDB_DB_host_name>
```

   The host name is the DNS name from the Amazon RDS MariaDB DB instance endpoint.

6. Using the client of your choice, connect to the external MariaDB instance and create a MariaDB user to be used for replication. This account is used solely for replication and must be restricted to your domain to improve security. The following is an example:

```
CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY '<password>';
```

7. For the external MariaDB instance, grant `REPLICATION CLIENT` and `REPLICATION SLAVE` privileges to your replication user. For example, to grant the `REPLICATION CLIENT` and `REPLICATION SLAVE` privileges on all databases for the '`repl_user`' user for your domain, issue the following command:

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.*
   TO 'repl_user'@'mydomain.com'
   IDENTIFIED BY '<password>';
```

8. Make the Amazon RDS MariaDB DB instance the replica. Connect to the Amazon RDS MariaDB DB instance as the master user and identify the external MariaDB database as the replication master by using the mysql.rds_set_external_master_gtid (p. 717) command. Use the GTID that you determined in Step 2. The following is an example:

```
CALL mysql.rds_set_external_master_gtid ('mymasterserver.mydomain.com', 3306,
    'repl_user', '<password>', '<GTID>', 0);
```

9. On the Amazon RDS MariaDB DB instance, issue the mysql.rds_start_replication (p. 917) command to start replication:

```
CALL mysql.rds_start_replication;
```

Appendix: Options for MariaDB Database Engine

This appendix describes options, or additional features, that are available for Amazon RDS instances running the MariaDB DB engine. To enable these options, you add them to a custom option group, and then associate the option group with your DB instance. For more information about working with option groups, see Working with Option Groups (p. 153).

Amazon RDS supports the following options for MariaDB:

Option ID	Engine Versions
MARIADB_AUDIT_PLUGIN	MariaDB 10.0.24 and later

MariaDB Audit Plugin Support

Amazon RDS supports using the MariaDB Audit Plugin on MariaDB database instances. The MariaDB Audit Plugin records database activity such as users logging on to the database, queries run against the database, and more. The record of database activity is stored in a log file.

Audit Plugin Option Settings

Amazon RDS supports the following settings for the MariaDB Audit Plugin option.

Option Setting	Valid Values	Default Value	Description
SERVER_AUDIT_FILE_PATH	/rdsdbdata/log/audit/	/rdsdbdata/log/audit/	The location of the log file. The log file contains the record of the activity specified in SERVER_AUDIT_EVENTS. For more information, see Viewing and Listing Database Log Files (p. 303) and MariaDB Database Log Files (p. 306).
SERVER_AUDIT_FILE_ROTATE_SIZE	1–1000000000	1000000	The size in bytes that when reached, causes the file to rotate. For more information, see Log File Size (p. 307).
SERVER_AUDIT_FILE_ROTATIONS	0–100	9	The number of log rotations to save. For more information, see Log File Size (p. 307) and Downloading a Database Log File (p. 304).

Option Setting	Valid Values	Default Value	Description
SERVER_AUDIT_EVENTS	CONNECT, QUERY, TABLE	CONNECT, QUERY	The types of activity to record in the log. Installing the MariaDB Audit Plugin is itself logged. • CONNECT: Log successful and unsuccessful connections to the database, and disconnections from the database. • QUERY: Log the text of all queries run against the database. • TABLE: Log tables affected by queries when the queries are run against the database. For MariaDB, CONNECT, QUERY, and TABLE are supported. For MySQL, CONNECT and QUERY are supported.
SERVER_AUDIT_INCL_USERS	Multiple comma-separated values	None	Include only activity from the specified users. By default, activity is recorded for all users. If a user is specified in both SERVER_AUDIT_EXCL_USERS and SERVER_AUDIT_INCL_USERS, then activity is recorded for the user.
SERVER_AUDIT_EXCL_USERS	Multiple comma-separated values	None	Exclude activity from the specified users. By default, activity is recorded for all users. If a user is specified in both SERVER_AUDIT_EXCL_USERS and SERVER_AUDIT_INCL_USERS, then activity is recorded for the user. The rdsadmin user queries the database every second to check the health of the database. Depending on your other settings, this activity can possibly cause the size of your log file to grow very large, very quickly. If you don't need to record this activity, add the rdsadmin user to the SERVER_AUDIT_EXCL_USERS list.
SERVER_AUDIT_LOGGING	ON	ON	Logging is active. The only valid value is ON. Amazon RDS does not support deactivating logging. If you want to deactivate logging, remove the MariaDB Audit Plugin. For more information, see Removing the MariaDB Audit Plugin (p. 711).

Adding the MariaDB Audit Plugin

The general process for adding the MariaDB Audit Plugin to a DB instance is the following:

1. Create a new option group, or copy or modify an existing option group.
2. Add the option to the option group.
3. Associate the option group with the DB instance.

After you add the MariaDB Audit Plugin, you don't need to restart your DB instance. As soon as the option group is active, auditing begins immediately.

To add the MariaDB Audit Plugin

1. Determine the option group you want to use. You can create a new option group or use an existing option group. If you want to use an existing option group, skip to the next step. Otherwise, create a custom DB option group. Choose **mariadb** for **Engine**, and choose **10.0** or later for **Major Engine Version**. For more information, see Creating an Option Group (p. 154).
2. Add the **MARIADB_AUDIT_PLUGIN** option to the option group, and configure the option settings. For more information about adding options, see Adding an Option to an Option Group (p. 157). For more information about each setting, see Audit Plugin Option Settings (p. 709).
3. Apply the option group to a new or existing DB instance.

   - For a new DB instance, you apply the option group when you launch the instance. For more information, see Creating a DB Instance Running the MariaDB Database Engine (p. 678).
   - For an existing DB instance, you apply the option group by modifying the instance and attaching the new option group. For more information, see Modifying a DB Instance Running the MariaDB Database Engine (p. 691).

Viewing and Downloading the MariaDB Audit Plugin Log

After you enable the MariaDB Audit Plugin, you access the results in the log files the same way you access any other text-based log files. The audit log files are located at `/rdsdbdata/log/audit/`. For information about viewing the log file in the console, see Viewing and Listing Database Log Files (p. 303). For information about downloading the log file, see Downloading a Database Log File (p. 304).

Modifying MariaDB Audit Plugin Settings

After you enable the MariaDB Audit Plugin, you can modify settings for the plugin. For more information about how to modify option settings, see Modifying an Option Setting (p. 163). For more information about each setting, see Audit Plugin Option Settings (p. 709).

Removing the MariaDB Audit Plugin

Amazon RDS doesn't support turning off logging in the MariaDB Audit Plugin. However, you can remove the plugin from a DB instance. After you remove the MariaDB Audit Plugin, you need to restart your DB instance to stop auditing.

To remove the MariaDB Audit Plugin from a DB instance, do one of the following:

- Remove the MariaDB Audit Plugin option from the option group it belongs to. This change affects all DB instances that use the option group. For more information, see Removing an Option from an Option Group (p. 167)
- Modify the DB instance and specify a different option group that doesn't include the plugin. This change affects a single DB instance. You can specify the default (empty) option group, or a different custom option group. For more information, see Modifying a DB Instance Running the MariaDB Database Engine (p. 691).

Appendix: Parameters for MariaDB

By default, a MariaDB DB instance uses a DB parameter group that is specific to a MariaDB database. This parameter group contains some but not all of the parameters contained in the Amazon RDS DB parameter groups for the MySQL database engine. It also contains a number of new, MariaDB-specific parameters. The following MySQL parameters are not available in MariaDB-specific DB parameter groups:

- bind_address
- binlog_error_action
- binlog_gtid_simple_recovery
- binlog_max_flush_queue_time
- binlog_order_commits
- binlog_row_image
- binlog_rows_query_log_events
- binlogging_impossible_mode
- block_encryption_mode
- core_file
- default_tmp_storage_engine
- div_precision_increment
- end_markers_in_json
- enforce_gtid_consistency
- eq_range_index_dive_limit
- explicit_defaults_for_timestamp
- gtid_executed
- gtid-mode
- gtid_next
- gtid_owned
- gtid_purged
- log_bin_basename
- log_bin_index
- log_bin_use_v1_row_events
- log_slow_admin_statements
- log_slow_slave_statements
- log_throttle_queries_not_using_indexes
- master-info-repository
- optimizer_trace
- optimizer_trace_features
- optimizer_trace_limit
- optimizer_trace_max_mem_size
- optimizer_trace_offset
- relay_log_info_repository
- rpl_stop_slave_timeout
- slave_parallel_workers
- slave_pending_jobs_size_max
- slave_rows_search_algorithms

- storage_engine
- table_open_cache_instances
- timed_mutexes
- transaction_allow_batching
- validate-password
- validate_password_dictionary_file
- validate_password_length
- validate_password_mixed_case_count
- validate_password_number_count
- validate_password_policy
- validate_password_special_char_count

For more information on MySQL 5.6 parameters, go to the MySQL 5.6 documentation.

The MariaDB-specific DB parameter groups also contain the following modifiable parameters that are applicable to MariaDB only. Acceptable ranges for all modifiable parameters are the same as specified in the MariaDB documentation except where noted. Amazon RDS MariaDB parameters are set to the default values of the storage engine you have selected.

- aria_block_size
- aria_checkpoint_interval
- aria_checkpoint_log_activity
- aria_force_start_after_recovery_failures
- aria_group_commit
- aria_group_commit_interval
- aria_log_dir_path
- aria_log_file_size
- aria_log_purge_type
- aria_max_sort_file_size
- aria_page_checksum
- aria_pagecache_age_threshold
- aria_pagecache_division_limit
- aria_recover

  Amazon RDS MariaDB supports the values of NORMAL, OFF, and QUICK, but not FORCE or BACKUP.
- aria_repair_threads
- aria_sort_buffer_size
- aria_stats_method
- aria_sync_log_dir
- binlog_annotate_row_events
- binlog_commit_wait_count
- binlog_commit_wait_usec
- binlog_row_image (MariaDB version 10.1 and later)
- deadlock_search_depth_long
- deadlock_search_depth_short
- deadlock_timeout_long
- deadlock_timeout_short
- explicit_defaults_for_timestamp (MariaDB version 10.1 and later)

- extra_max_connections
- extra_port
- feedback
- feedback_send_retry_wait
- feedback_send_timeout
- feedback_url
- feedback_user_info
- gtid_domain_id
- gtid_strict_mode
- histogram_size
- histogram_type
- innodb_adaptive_hash_index_partitions
- innodb_background_scrub_data_check_interval (MariaDB version 10.1 and later)
- innodb_background_scrub_data_compressed (MariaDB version 10.1 and later)
- innodb_background_scrub_data_interval (MariaDB version 10.1 and later)
- innodb_background_scrub_data_uncompressed (MariaDB version 10.1 and later)
- innodb_buf_dump_status_frequency (MariaDB version 10.1 and later)
- innodb_buffer_pool_populate
- innodb_cleaner_lsn_age_factor
- innodb_compression_algorithm (MariaDB version 10.1 and later)
- innodb_corrupt_table_action
- innodb_defragment (MariaDB version 10.1 and later)
- innodb_defragment_fill_factor (MariaDB version 10.1 and later)
- innodb_defragment_fill_factor_n_recs (MariaDB version 10.1 and later)
- innodb_defragment_frequency (MariaDB version 10.1 and later)
- innodb_defragment_n_pages (MariaDB version 10.1 and later)
- innodb_defragment_stats_accuracy (MariaDB version 10.1 and later)
- innodb_empty_free_List_algorithm
- innodb_fake_changes
- innodb_fatal_semaphore_wait_threshold (MariaDB version 10.1 and later)
- innodb_foreground_preflush
- innodb_idle_flush_pct (MariaDB version 10.1 and later)
- innodb_immediate_scrub_data_uncompressed (MariaDB version 10.1 and later)
- innodb_instrument_semaphores (MariaDB version 10.1 and later)
- innodb_locking_fake_changes
- innodb_log_arch_dir
- innodb_log_arch_expire_sec
- innodb_log_archive
- innodb_log_block_size
- innodb_log_checksum_algorithm
- innodb_max_bitmap_file_size
- innodb_max_changed_pages
- innodb_prefix_index_cluster_optimization (MariaDB version 10.1 and later)
- innodb_sched_priority_cleaner
- innodb_scrub_log (MariaDB version 10.1 and later)
- innodb_scrub_log_speed (MariaDB version 10.1 and later)

- innodb_show_locks_held
- innodb_show_verbose_locks
- innodb_simulate_comp_failures
- innodb_stats_modified_counter
- innodb_stats_traditional
- innodb_use_atomic_writes
- innodb_use_fallocate
- innodb_use_global_flush_log_at_trx_commit
- innodb_use_stacktrace
- innodb_use_trim (MariaDB version 10.1 and later)
- join_buffer_space_limit
- join_cache_level
- key_cache_file_hash_size
- key_cache_segments
- max_digest_length (MariaDB version 10.1 and later)
- max_statement_time (MariaDB version 10.1 and later)
- mysql56_temporal_format (MariaDB version 10.1 and later)
- progress_report_time
- query_cache_strip_comments
- replicate_annotate_row_events
- replicate_do_db
- replicate_do_table
- replicate_events_marked_for_skip
- replicate_ignore_db
- replicate_ignore_table
- replicate_wild_ignore_table
- slave_domain_parallel_threads
- slave_parallel_max_queued
- slave_parallel_mode (MariaDB version 10.1 and later)
- slave_parallel_threads
- slave_run_triggers_for_rbr (MariaDB version 10.1 and later)
- sql_error_log_filename
- sql_error_log_rate
- sql_error_log_rotate
- sql_error_log_rotations
- sql_error_log_size_limit
- thread_handling
- thread_pool_idle_timeout
- thread_pool_max_threads
- thread_pool_min_threads
- thread_pool_oversubscribe
- thread_pool_size
- thread_pool_stall_limit
- transaction_write_set_extraction
- use_stat_tables
- userstat

For more information on MariaDB parameters, go to the MariaDB documentation.

Appendix: MariaDB on Amazon RDS SQL Reference

This appendix describes system stored procedures that are available for Amazon RDS instances running the MariaDB DB engine.

You can use all of the system stored procedures that are available for Amazon RDS MySQL DB instances for MariaDB DB instances also. These stored procedures are documented at Appendix: MySQL on Amazon RDS SQL Reference (p. 913).

Additionally, the following system stored procedures are supported only for Amazon RDS DB instances running MariaDB:

- mysql.rds_set_external_master_gtid (p. 717)
- mysql.rds_kill_query_id (p. 719)

mysql.rds_set_external_master_gtid

Configures GTID-based replication from a MariaDB instance running external to Amazon RDS to an Amazon RDS MariaDB DB instance. This stored procedure is supported only where the external MariaDB instance is version 10.0.24 or greater. When setting up replication where one or both instances do not support MariaDB global transaction identifiers (GTIDs), use mysql.rds_set_external_master (p. 914).

Using GTIDs for replication provides crash-safety features not offered by binary log replication, so we recommend it in cases where the replicating instances support it.

Syntax

```
CALL mysql.rds_set_external_master_gtid(
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , gtid
  , ssl_encryption
);
```

Parameters

host_name

String. The host name or IP address of the MariaDB instance running external to Amazon RDS that will become the replication master.

host_port

Integer. The port used by the MariaDB instance running external to Amazon RDS to be configured as the replication master. If your network configuration includes SSH port replication that converts the port number, specify the port number that is exposed by SSH.

replication_user_name

String. The ID of a user with REPLICATION SLAVE permissions in the MariaDB DB instance to be configured as the Read Replica.

replication_user_password

String. The password of the user ID specified in `replication_user_name`.

gtid

String. The global transaction ID on the master that replication should start from.

You can use `@@gtid_current_pos` to get the current GTID if the replication master has been locked while you are configuring replication, so the binary log doesn't change between the points when you get the GTID and when replication starts.

Otherwise, if you are using `mysqldump` version 10.0.13 or greater to populate the slave instance prior to starting replication, you can get the GTID position in the output by using the `--master-data` or `--dump-slave` options. If you are not using `mysqldump` version 10.0.13 or greater, you can run the `SHOW MASTER STATUS` or use those same `mysqldump` options to get the binary log file name and position, then convert them to a GTID by running `BINLOG_GTID_POS` on the external MariaDB instance:

```
SELECT BINLOG_GTID_POS('<binary log file name>', <binary log file position>);
```

For more information about the MariaDB implementation of GTIDs, go to Global Transaction ID in the MariaDB documentation.

ssl_encryption

Integer. This option is not currently implemented. The default is 0.

Usage Notes

The `mysql.rds_set_external_master_gtid` procedure must be run by the master user. It must be run on the MariaDB DB instance that you are configuring as the replication slave of a MariaDB instance running external to Amazon RDS. Before running `mysql.rds_set_external_master_gtid`, you must have configured the instance of MariaDB running external to Amazon RDS as a replication master. For more information, see Importing Data into a MariaDB DB Instance (p. 706).

Warning
Do not use `mysql.rds_set_external_master_gtid` to manage replication between two Amazon RDS DB instances. Use it only when replicating with a MariaDB instance running external to RDS. For information about managing replication between Amazon RDS DB instances, see Working with PostgreSQL, MySQL, and MariaDB Read Replicas (p. 134).

After calling `mysql.rds_set_external_master_gtid` to configure an Amazon RDS DB instance as a Read Replica, you can call mysql.rds_start_replication (p. 917) on the replica to start the replication process. You can call mysql.rds_reset_external_master (p. 916) to remove the Read Replica configuration.

When `mysql.rds_set_external_master_gtid` is called, Amazon RDS records the time, user, and an action of "set master" in the `mysql.rds_history` and `mysql.rds_replication_status` tables.

Examples

When run on a MariaDB DB instance, the following example configures it as the replication slave of an instance of MariaDB running external to Amazon RDS.

```
call mysql.rds_set_external_master_gtid
 ('Sourcedb.some.com',3306,'ReplicationUser','SomePassW0rd','0-123-456',0);
```

Related Topics

- mysql.rds_reset_external_master (p. 916)

- mysql.rds_start_replication (p. 917)
- mysql.rds_stop_replication (p. 918)

mysql.rds_kill_query_id

Terminates a query running against the MariaDB server.

Syntax

```
CALL mysql.rds_kill_query_id(queryID);
```

Parameters

queryID

Integer. The identity of the query to be terminated.

Usage Notes

To terminate a query running against the MariaDB server, use the `mysql.rds_kill_query_id` procedure and pass in the ID of that query. To obtain the query ID, query the MariaDB Information Schema PROCESSLIST Table, as shown following:

```
SELECT USER, HOST, COMMAND, TIME, STATE, INFO, QUERY_ID FROM
                INFORMATION_SCHEMA.PROCESSLIST WHERE USER = '<user name>';
```

The connection to the MariaDB server is retained.

Related Topics

- mysql.rds_kill (p. 924)
- mysql.rds_kill_query (p. 925)

Examples

The following example terminates a query with a query ID of 230040:

```
call mysql.rds_kill_query_id(230040);
```

Microsoft SQL Server on Amazon RDS

Amazon RDS supports DB instances running several versions and editions of Microsoft SQL Server. The most recent supported version of each major version is shown following. For the full list of supported versions and editions, see Version and Feature Support on Amazon RDS (p. 726).

- SQL Server 2017, version 14.00.1000.169, RTM, for all editions, and all AWS Regions
- SQL Server 2016, version 13.00.4451.0, SP1 CU5, for all editions, and all AWS Regions
- SQL Server 2014, version 12.00.5546.0, SP2 CU5, for all editions and all AWS Regions
- SQL Server 2012version 11.00.6594.0, SP3 CU8, for all editions and all AWS Regions
- SQL Server 2008 R2, version 10.50.6529.0, SP3 QFE, for all editions, and all AWS Regions except US East (Ohio), Canada (Central), and EU (London)

For information about licensing options for the different SQL Server versions and editions, see Licensing Microsoft SQL Server on Amazon RDS (p. 735).

With Amazon RDS, you can create DB instances and DB snapshots, point-in-time restores, and automated or manual backups. DB instances running SQL Server can be used inside a VPC. You can also use SSL to connect to a DB instance running SQL Server, and you can use TDE to encrypt data at rest. Amazon RDS currently supports Multi-AZ deployments for SQL Server using SQL Server Mirroring as a high-availability, failover solution.

In order to deliver a managed service experience, Amazon RDS does not provide shell access to DB instances, and it restricts access to certain system procedures and tables that require advanced privileges. Amazon RDS supports access to databases on a DB instance using any standard SQL client application such as Microsoft SQL Server Management Studio. Amazon RDS does not allow direct host access to a DB instance via Telnet, Secure Shell (SSH), or Windows Remote Desktop Connection. When you create a DB instance, you are assigned to the *db_owner* role for all databases on that instance, and you have all database-level permissions except for those that are used for backups. Amazon RDS manages backups for you.

Before creating your first DB instance, you should complete the steps in the setting up section of this guide. For more information, see Setting Up for Amazon RDS (p. 5).

Common Management Tasks for Microsoft SQL Server on Amazon RDS

The following are the common management tasks you perform with an Amazon RDS SQL Server DB instance, with links to relevant documentation for each task.

Task Area	Relevant Documentation
Instance Classes, Storage, and PIOPS If you are creating a DB instance for production purposes, you should understand how instance classes, storage types, and Provisioned IOPS work in Amazon RDS.	DB Instance Class Support for Microsoft SQL Server (p. 723) Amazon RDS Storage Types (p. 410)
Multi-AZ Deployments A production DB instance should use Multi-AZ deployments. Multi-AZ deployments provide increased availability, data durability, and fault tolerance for DB instances. Multi-AZ deployments for SQL Server are implemented using SQL Server's native Mirroring technology.	High Availability (Multi-AZ) (p. 99) Multi-AZ Deployments Using Microsoft SQL Server Mirroring (p. 730)
Amazon Virtual Private Cloud (VPC) If your AWS account has a default VPC, then your DB instance is automatically created inside the default VPC. If your account does not have a default VPC, and you want the DB instance in a VPC, you must create the VPC and subnet groups before you create the DB instance.	Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform (p. 391) Working with an Amazon RDS DB Instance in a VPC (p. 399)
Security Groups By default, DB instances are created with a firewall that prevents access to them. You therefore must create a security group with the correct IP addresses and network configuration to access the DB instance. The security group you create depends on what Amazon EC2 platform your DB instance is on, and whether you will access your DB instance from an Amazon EC2 instance. In general, if your DB instance is on the *EC2-Classic* platform, you will need to create a DB security group; if your DB instance is on the *EC2-VPC* platform, you will need to create a VPC security group.	Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform (p. 391) Amazon RDS Security Groups (p. 375)
Parameter Groups If your DB instance is going to require specific database parameters, you should create a parameter group before you create the DB instance.	Working with DB Parameter Groups (p. 170)
Option Groups If your DB instance is going to require specific database options, you should create an option group before you create the DB instance.	Options for the Microsoft SQL Server Database Engine (p. 795)
Connecting to Your DB Instance After creating a security group and associating it to a DB instance, you can connect to the DB instance using any standard SQL client application such as Microsoft SQL Server Management Studio.	Connecting to a DB Instance Running the Microsoft SQL Server Database Engine (p. 749)
Backup and Restore	Working With Backups (p. 201) Importing and Exporting SQL Server Databases (p. 769)

Task Area	Relevant Documentation
When you create your DB instance, you can configure it to take automated backups. You can also back up and restore your databases manually by using full backup files (.bak files).	
Monitoring You can monitor your SQL Server DB instance by using CloudWatch Amazon RDS metrics, events, and enhanced monitoring.	Viewing DB Instance Metrics (p. 254) Viewing Amazon RDS Events (p. 301)
Log Files You can access the log files for your SQL Server DB instance.	Amazon RDS Database Log Files (p. 303) Microsoft SQL Server Database Log Files (p. 312)

There are also advanced administrative tasks for working with SQL Server DB instances. For more information, see the following documentation:

- Common DBA Tasks for Microsoft SQL Server (p. 800).
- Using Windows Authentication with a SQL Server DB Instance (p. 812)
- Accessing the tempdb Database (p. 801)

Limits for Microsoft SQL Server DB Instances

The Amazon RDS implementation of Microsoft SQL Server on a DB instance have some limitations you should be aware of:

- You can create up to 30 databases on each of your DB instances running Microsoft SQL Server. The Microsoft system databases, such as `master` and `model`, don't count toward this limit.
- Some ports are reserved for Amazon RDS use and you can't use them when you create a DB instance.
- Amazon RDS for SQL Server does not support importing data into the msdb database.
- You can't rename databases on a DB instance in a SQL Server Multi-AZ with Mirroring deployment.
- The maximum storage size for SQL Server DB instances is the following:
  - General Purpose (SSD) storage: 16 TB for all editions
  - Provisioned IOPS storage: 16 TB for all editions
  - Magnetic storage: 1 TB for all editions

  If you have a scenario that requires a larger amount of storage, you can use sharding across multiple DB instances to get around the limit. This approach requires data-dependent routing logic in applications that connect to the sharded system. You can use an existing sharding framework, or you can write custom code to enable sharding. If you use an existing framework, the framework can't install any components on the same server as the DB instance.
- The minimum storage size for SQL Server DB instances is the following:
  - General Purpose (SSD) storage: 200 GB for Enterprise and Standard editions, 20 GB for Web and Express editions
  - Provisioned IOPS storage: 200 GB for Enterprise and Standard editions, 100 GB for Web and Express editions
  - Magnetic storage: 200 GB for Enterprise and Standard editions, 20 GB for Web and Express editions

- Amazon RDS doesn't support running SQL Server Analysis Services, SQL Server Integration Services, SQL Server Reporting Services, Data Quality Services, or Master Data Services on the same server as your Amazon RDS DB instance. To use these features, we recommend that you install SQL Server on an Amazon EC2 instance, or use an on-premise SQL Server instance, to act as the Reporting, Analysis, Integration, or Master Data Services server for your SQL Server DB instance on Amazon RDS. You can install SQL Server on an Amazon EC2 instance with Amazon EBS storage, pursuant to Microsoft licensing policies.
- Because of limitations in Microsoft SQL Server, restoring to a point in time before successful execution of a DROP DATABASE might not reflect the state of that database at that point in time. For example, the dropped database is typically restored to its state up to 5 minutes before the DROP DATABASE command was issued, which means that you can't restore the transactions made during those few minutes on your dropped database. To work around this, you can reissue the DROP DATABASE command after the restore operation is completed. Dropping a database removes the transaction logs for that database.

DB Instance Class Support for Microsoft SQL Server

The computation and memory capacity of a DB instance is determined by its DB instance class. The DB instance class you need depends on your processing power and memory requirements. For more information, see DB Instance Class (p. 92).

The following are the latest-generation DB instance classes supported for Microsoft SQL Server.

SQL Server Edition	2017 and 2016 Support	2014, 2012, and 2008 R2 Support
Enterprise Edition (Bring Your Own License)	db.m4.large–16xlarge db.r4.large–16xlarge db.t2.small–large	db.m4.large–10xlarge db.r4.large–8xlarge db.t2.small–large
Enterprise Edition (License Included)	db.m4.xlarge–16xlarge db.r4.xlarge–16xlarge —	db.m4.xlarge–10xlarge db.r4.xlarge–8xlarge —
Standard Edition (Bring Your Own License)	db.m4.large–16xlarge db.r4.large–16xlarge db.t2.small–large	db.m4.large–10xlarge db.r4.large–8xlarge db.t2.small–large
Standard Edition	db.m4.large–16xlarge, except db.m4.10xlarge db.r4.large–16xlarge	db.m4.large–4xlarge db.r4.large–8xlarge —

SQL Server Edition	2017 and 2016 Support	2014, 2012, and 2008 R2 Support
(License Included)	—	
Web Edition	db.m4.large–4xlarge db.r4.large–2xlarge db.t2.small–medium	db.m4.large–4xlarge db.r4.large–2xlarge db.t2.small–medium
Express Edition	— — db.t2.micro–medium	db.m1.small–small — db.t2.micro–medium

Microsoft SQL Server Security

The Microsoft SQL Server database engine uses role-based security. The master user name you use when you create a DB instance is a SQL Server Authentication login that is a member of the `processadmin`, `public`, and `setupadmin` fixed server roles.

Any user who creates a database is assigned to the db_owner role for that database and has all database-level permissions except for those that are used for backups. Amazon RDS manages backups for you.

The following server-level roles are not currently available in Amazon RDS:

- bulkadmin
- dbcreator
- diskadmin
- securityadmin
- serveradmin
- sysadmin

The following server-level permissions are not available on SQL Server DB instances:

- ADMINISTER BULK OPERATIONS
- ALTER ANY CREDENTIAL
- ALTER ANY EVENT NOTIFICATION
- ALTER ANY EVENT SESSION
- ALTER ANY SERVER AUDIT
- ALTER RESOURCES
- ALTER SETTINGS (You can use the DB Parameter Group APIs to modify parameters. For more information, see Working with DB Parameter Groups (p. 170).
- AUTHENTICATE SERVER
- CONTROL_SERVER
- CREATE DDL EVENT NOTIFICATION
- CREATE ENDPOINT

- CREATE TRACE EVENT NOTIFICATION
- EXTERNAL ACCESS ASSEMBLY
- SHUTDOWN (You can use the RDS reboot option instead)
- UNSAFE ASSEMBLY
- ALTER ANY AVAILABILITY GROUP (SQL Server 2012 only)
- CREATE ANY AVAILABILITY GROUP (SQL Server 2012 only)

Compliance Program Support for Microsoft SQL Server DB Instances

AWS Services in Scope have been fully assessed by a third-party auditor and result in a certification, attestation of compliance, or Authority to Operate (ATO). For more information, see AWS Services in Scope by Compliance Program.

HIPAA Support for Microsoft SQL Server DB Instances

You can use Amazon RDS for Microsoft SQL Server databases to build HIPAA-compliant applications. You can store healthcare-related information, including protected health information (PHI), under an executed Business Associate Agreement (BAA) with AWS. For more information, see HIPAA Compliance.

Amazon RDS for SQL Server supports HIPAA for the following versions and editions:

- SQL Server 2017, 2016, 2014, and 2012: Enterprise, Standard, and Web Editions
- SQL Server 2008 R2: Enterprise Edition

To enable HIPAA support on your DB instance, set up the following three components.

Component	Details
Auditing	To set up auditing, set the parameter `rds.sqlserver_audit` to the value `fedramp_hipaa`. If your DB instance is not already using a custom DB parameter group, you must create a custom parameter group and attach it to your DB instance before you can modify the `rds.sqlserver_audit` parameter. For more information, see Working with DB Parameter Groups (p. 170).
Transport Encryption	To set up transport encryption, force all connections to your DB instance to use Secure Sockets Layer (SSL). For more information, see Forcing Connections to Your DB Instance to Use SSL (p. 791).

Component	Details
Encryption at Rest	To set up encryption at rest, you have two options: 1. If you are running Enterprise Edition, you can choose to use Transparent Data Encryption (TDE) to achieve encryption at rest. For more information, see Microsoft SQL Server Transparent Data Encryption Support (p. 797). 2. You can set up encryption at rest by using AWS Key Management Service (AWS KMS) encryption keys. For more information, see Encrypting Amazon RDS Resources (p. 355).

SSL Support for Microsoft SQL Server DB Instances

You can use SSL to encrypt connections between your applications and your Amazon RDS DB instances running Microsoft SQL Server. You can also force all connections to your DB instance to use SSL. If you force connections to use SSL, it happens transparently to the client, and the client doesn't have to do any work to use SSL.

SSL is supported in all AWS Regions and for all supported SQL Server editions. For more information, see Using SSL with a Microsoft SQL Server DB Instance (p. 791).

Version and Feature Support on Amazon RDS

Microsoft SQL Server 2017 Support on Amazon RDS

Amazon RDS supports the following versions of SQL Server 2017:

- Version 14.00.1000.169, RTM, for all editions, and all AWS Regions

SQL Server 2017 includes many new features, such as the following:

- Adaptive query processing
- Automatic plan correction
- GraphDB
- Resumable index rebuilds

For the full list of SQL Server 2017 features, see What's New in SQL Server 2017 in the Microsoft documentation.

For a list of unsupported features, see Features Not Supported (p. 729).

Microsoft SQL Server 2016 Support on Amazon RDS

Amazon RDS supports the following versions of SQL Server 2016:

- Version 13.00.4451.0, SP1 CU5, for all editions, and all AWS Regions
- Version 13.00.4422.0, SP1 CU2, for all editions, and all AWS Regions
- Version 13.00.2164.0, RTM CU2, for all editions, and all AWS Regions

SQL Server 2016 includes many new features, such as the following:

- Query store
- Operational Analytics
- Temporal tables
- Always encrypted (Supported for all editions on 13.00.4422.0 SP1 CU2 and later.)
- JSON support

For the full list of SQL Server 2016 features, see What's New in SQL Server 2016 and SQL Server 2016 Service Pack 1 (SP1) Released in the Microsoft documentation.

For a list of unsupported features, see Features Not Supported (p. 729).

Microsoft SQL Server 2014 Support on Amazon RDS

Amazon RDS supports the following versions of SQL Server 2014:

- Version 12.00.5546.0, SP2 CU5, for all editions and all AWS Regions
- Version 12.00.5000.0, SP2, for all editions and all AWS Regions
- Version 12.00.4422.0, SP1 CU2, for all editions except Enterprise Edition, and all AWS Regions except Canada (Central), and EU (London)

In addition to supported features of SQL Server 2012, Amazon RDS supports the new query optimizer available in SQL Server 2014, and also the delayed durability feature.

For a list of unsupported features, see Features Not Supported (p. 729).

SQL Server 2014 supports all the parameters from SQL Server 2012 and uses the same default values. SQL Server 2014 includes one new parameter, backup checksum default. For more information, see How to enable the CHECKSUM option if backup utilities do not expose the option in the Microsoft documentation.

Microsoft SQL Server 2012 Support on Amazon RDS

Amazon RDS supports the following versions of SQL Server 2012:

- Version 11.00.6594.0, SP3 CU8, for all editions and all AWS Regions
- Version 11.00.6020.0, SP3, for all editions and all AWS Regions
- Version 11.00.5058.0, SP2, for all editions, and all AWS Regions except US East (Ohio), Canada (Central), and EU (London)
- Version 11.00.2100.60, RTM, for all editions, and all AWS Regions except US East (Ohio), Canada (Central), and EU (London)

For more information about SQL Server 2012, see Features Supported by the Editions of SQL Server 2012 in the Microsoft documentation.

In addition to supported features of SQL Server 2008 R2, Amazon RDS supports the following SQL Server 2012 features:

- Columnstore indexes (Enterprise Edition)
- Online Index Create, Rebuild and Drop for XML, varchar(max), nvarchar(max), and varbinary(max) data types (Enterprise Edition)
- Flexible Server Roles
- Service Broker (note that Service Broker Endpoints are not supported)
- Partially Contained Databases
- Sequences
- Transparent Data Encryption (Enterprise Edition only)
- THROW statement
- New and enhanced spatial types
- UTF-16 Support
- ALTER ANY SERVER ROLE server-level permission

For a list of unsupported features, see Features Not Supported (p. 729).

Some SQL Server parameters have changed in SQL Server 2012.

- The following parameters have been removed from SQL Server 2012: `awe enabled`, `precompute rank`, and `sql mail xps`. These parameters were not modifiable in SQL Server DB Instances and their removal should have no impact on your SQL Server use.
- A new `contained database authentication` parameter in SQL Server 2012 supports partially contained databases. When you enable this parameter and then create a partially contained database, an authorized user's user name and password is stored within the partially contained database instead of in the master database. For more information about partially contained databases, see Contained Databases in the Microsoft documentation.

Microsoft SQL Server 2008 R2 Support on Amazon RDS

Amazon RDS supports the following versions of SQL Server 2008 R2:

- Version 10.50.6529.0, SP3 QFE, for all editions, and all AWS Regions except US East (Ohio), Canada (Central), and EU (London)
- Version 10.50.6000.34, SP3, for all editions, and all AWS Regions except US East (Ohio), Canada (Central), and EU (London)
- Version 10.50.2789.0, SP1, for all editions, and all AWS Regions except US East (Ohio), Canada (Central), and EU (London)

For more information about SQL Server 2008 R2, see Features Supported by the Editions of SQL Server 2008 R2 in the Microsoft documentation.

Amazon RDS supports the following SQL Server 2008 R2 features:

- Core database engine features

- SQL Server development tools:
  - Visual Studio integration
  - IntelliSense
- SQL Server management tools:
  - SQL Server Management Studio (SMS)
  - sqlcmd
  - SQL Server Profiler (client side traces; workaround available for server side)
  - SQL Server Migration Assistant (SSMA)
  - Database Engine Tuning Advisor
  - SQL Server Agent
- Safe CLR
- Full-text search (except semantic search)
- SSL
- Transparent Data Encryption (Enterprise Edition only)
- Spatial and location features
- Service Broker (note that Service Broker Endpoints are not supported)
- Change Tracking
- Database Mirroring
- The ability to use an Amazon RDS SQL DB instance as a data source for Reporting, Analysis, and Integration Services that are running on a separate server.

For a list of unsupported features, see Features Not Supported (p. 729).

Features Not Supported

The following Microsoft SQL Server features are not supported on Amazon RDS:

- Always On (2012 Enterprise Edition)
- Stretch database
- Backing up to Microsoft Azure Blob Storage
- Buffer pool extension
- BULK INSERT and OPENROWSET(BULK...) features
- Change Data Capture (CDC) - Consider using Change Tracking as an alternative to CDC.
- Data Quality Services
- Database Log Shipping
- Database Mail
- Distributed Queries (i.e., Linked Servers)
- Distribution Transaction Coordinator (MSDTC)
- File tables
- FILESTREAM support
- Instant file initialization
- Maintenance Plans
- Performance Data Collector
- Policy-Based Management

- PolyBase
- Replication
- Resource Governor
- Server-level triggers
- Service Broker or additional T-SQL endpoints (all operations using CREATE ENDPOINT are unavailable)
- SQL Server Audit
- WCF Data Services

Microsoft SQL Server Engine Version Management

With Amazon RDS, you control when to upgrade your SQL Server DB instance to new versions supported by Amazon RDS. You can maintain compatibility with specific SQL Server versions, test new versions with your application before deploying in production, and perform version upgrades on your own terms and timelines.

Currently, you perform all SQL Server database upgrades manually. For more information about upgrading a SQL Server DB instance, see Upgrading the Microsoft SQL Server DB Engine (p. 764).

Multi-AZ Deployments Using Microsoft SQL Server Mirroring

Amazon RDS supports Multi-AZ deployments for DB instances running Microsoft SQL Server by using SQL Server Database Mirroring. Multi-AZ deployments provide increased availability, data durability, and fault tolerance for DB instances. In the event of planned database maintenance or unplanned service disruption, Amazon RDS automatically fails over to the up-to-date standby so database operations can resume quickly without manual intervention. The primary and standby instances use the same endpoint, whose physical network address transitions to the mirror as part of the failover process. You don't have to reconfigure your application when a failover occurs.

Amazon RDS manages failover by actively monitoring your Multi-AZ deployment and initiating a failover when a problem with your primary occurs. Failover doesn't occur unless the standby and primary are fully in sync. Amazon RDS actively maintains your Multi-AZ deployment by automatically repairing unhealthy DB instances and reestablishing synchronous replication. You don't have to manage anything; Amazon RDS handles the primary, the Mirroring witness, and the standby instance for you. When you set up SQL Server Multi-AZ, all databases on the instance are mirrored automatically.

For more information, see Multi-AZ Deployments for Microsoft SQL Server with Database Mirroring (p. 787).

Using Transparent Data Encryption to Encrypt Data at Rest

Amazon RDS supports Microsoft SQL Server Transparent Data Encryption (TDE), which transparently encrypts stored data. Amazon RDS uses option groups to enable and configure these features. For more information about the TDE option, see Microsoft SQL Server Transparent Data Encryption Support (p. 797).

Local Time Zone for Microsoft SQL Server DB Instances

The time zone of an Amazon RDS DB instance running Microsoft SQL Server is set by default. The current default is Universal Coordinated Time (UTC). You can set the time zone of your DB instance to a local time zone instead, to match the time zone of your applications.

You set the time zone when you first create your DB instance. You can create your DB instance by using the AWS Management Console, the Amazon RDS API CreateDBInstance action, or the AWS CLI create-db-instance command.

If your DB instance is part of a Multi-AZ deployment (using SQL Server Mirroring), then when you fail over, your time zone remains the local time zone that you set. For more information, see Multi-AZ Deployments Using Microsoft SQL Server Mirroring (p. 730).

When you request a point-in-time restore, you specify the time to restore to in UTC. During the restore process, the time is translated to the time zone of the DB instance. For more information, see Restoring a DB Instance to a Specified Time (p. 237).

The following are limitations to setting the local time zone on your DB instance:

- You can't modify the time zone of an existing SQL Server DB instance.
- You can't restore a snapshot from a DB instance in one time zone to a DB instance in a different time zone.
- We strongly recommend that you don't restore a backup file from one time zone to a different time zone. If you restore a backup file from one time zone to a different time zone, you must audit your queries and applications for the effects of the time zone change. For more information, see Importing and Exporting SQL Server Databases (p. 769).

Supported Time Zones

You can set your local time zone to one of the values listed in the following table.

Time Zone	Standard Time Offset	Description	Notes
Afghanistan Standard Time	(UTC+04:30)	Kabul	
Alaskan Standard Time	(UTC–09:00)	Alaska	
Arabian Standard Time	(UTC+04:00)	Abu Dhabi, Muscat	
Atlantic Standard Time	(UTC–04:00)	Atlantic Time (Canada)	
AUS Central Standard Time	(UTC+09:30)	Darwin	
AUS Eastern Standard Time	(UTC+10:00)	Canberra, Melbourne, Sydney	
Belarus Standard Time	(UTC+03:00)	Minsk	This time zone does not observe daylight savings time.
Canada Central Standard Time	(UTC–06:00)	Saskatchewan	

Time Zone	Standard Time Offset	Description	Notes
Cape Verde Standard Time	(UTC–01:00)	Cabo Verde Is.	
Cen. Australia Standard Time	(UTC+09:30)	Adelaide	
Central America Standard Time	(UTC–06:00)	Central America	
Central Asia Standard Time	(UTC+06:00)	Astana	
Central Brazilian Standard Time	(UTC–04:00)	Cuiaba	
Central Europe Standard Time	(UTC+01:00)	Belgrade, Bratislava, Budapest, Ljubljana, Prague	
Central European Standard Time	(UTC+01:00)	Sarajevo, Skopje, Warsaw, Zagreb	
Central Pacific Standard Time	(UTC+11:00)	Solomon Islands, New Caledonia	
Central Standard Time	(UTC–06:00)	Central Time (US and Canada)	
Central Standard Time (Mexico)	(UTC–06:00)	Guadalajara, Mexico City, Monterrey	
China Standard Time	(UTC+08:00)	Beijing, Chongqing, Hong Kong, Urumqi	
E. Africa Standard Time	(UTC+03:00)	Nairobi	This time zone does not observe daylight savings time.
E. Australia Standard Time	(UTC+10:00)	Brisbane	
E. Europe Standard Time	(UTC+02:00)	Chisinau	
E. South America Standard Time	(UTC–03:00)	Brasilia	
Eastern Standard Time	(UTC–05:00)	Eastern Time (US and Canada)	
Georgian Standard Time	(UTC+04:00)	Tbilisi	
GMT Standard Time	(UTC)	Dublin, Edinburgh, Lisbon, London	This time zone is not the same as Greenwich Mean Time. This time zone does observe daylight savings time.
Greenland Standard Time	(UTC–03:00)	Greenland	
Greenwich Standard Time	(UTC)	Monrovia, Reykjavik	This time zone does not observe daylight savings time.
GTB Standard Time	(UTC+02:00)	Athens, Bucharest	

Time Zone	Standard Time Offset	Description	Notes
Hawaiian Standard Time	(UTC–10:00)	Hawaii	
India Standard Time	(UTC+05:30)	Chennai, Kolkata, Mumbai, New Delhi	
Jordan Standard Time	(UTC+02:00)	Amman	
Korea Standard Time	(UTC+09:00)	Seoul	
Middle East Standard Time	(UTC+02:00)	Beirut	
Mountain Standard Time	(UTC–07:00)	Mountain Time (US and Canada)	
Mountain Standard Time (Mexico)	(UTC–07:00)	Chihuahua, La Paz, Mazatlan	
New Zealand Standard Time	(UTC+12:00)	Auckland, Wellington	
Newfoundland Standard Time	(UTC–03:30)	Newfoundland	
Pacific SA Standard Time	(UTC–03:00)	Santiago	
Pacific Standard Time	(UTC–08:00)	Pacific Time (US and Canada)	
Pacific Standard Time (Mexico)	(UTC–08:00)	Baja California	
Russian Standard Time	(UTC+03:00)	Moscow, St. Petersburg, Volgograd	This time zone does not observe daylight savings time.
SA Pacific Standard Time	(UTC–05:00)	Bogota, Lima, Quito, Rio Branco	This time zone does not observe daylight savings time.
SE Asia Standard Time	(UTC+07:00)	Bangkok, Hanoi, Jakarta	
Singapore Standard Time	(UTC+08:00)	Kuala Lumpur, Singapore	
Tokyo Standard Time	(UTC+09:00)	Osaka, Sapporo, Tokyo	
US Eastern Standard Time	(UTC–05:00)	Indiana (East)	
UTC	UTC	Coordinated Universal Time	This time zone does not observe daylight savings time.
UTC–02	(UTC–02:00)	Coordinated Universal Time–02	
UTC–08	(UTC–08:00)	Coordinated Universal Time–08	

Time Zone	Standard Time Offset	Description	Notes
UTC–09	(UTC–09:00)	Coordinated Universal Time–09	
UTC–11	(UTC–11:00)	Coordinated Universal Time–11	
UTC+12	(UTC+12:00)	Coordinated Universal Time+12	
W. Australia Standard Time	(UTC+08:00)	Perth	
W. Central Africa Standard Time	(UTC+01:00)	West Central Africa	
W. Europe Standard Time	(UTC+01:00)	Amsterdam, Berlin, Bern, Rome, Stockholm, Vienna	

Licensing Microsoft SQL Server on Amazon RDS

There are two licensing options available for Amazon RDS for Microsoft SQL Server: License Included and Bring Your Own License (BYOL). After you create a Microsoft SQL Server DB instance on Amazon RDS, you can change the licensing model by using the AWS Management Console, the Amazon RDS API ModifyDBInstance action, or the AWS CLI modify-db-instance command.

In accordance with Microsoft's usage rights, SQL Server Web Edition can be used only to support public and Internet-accessible webpages, websites, web applications, and web services. For more information, see AWS Service Terms.

License Included

In the License Included model, you don't need to purchase SQL Server licenses separately. AWS holds the license for the SQL Server database software. License Included pricing includes the software license, underlying hardware resources, and Amazon RDS management capabilities.

The License Included model is supported on Amazon RDS for the following Microsoft SQL Server database editions:

- Microsoft SQL Server Enterprise Edition (2008 R2, 2012, 2014, 2016, 2017)
- Microsoft SQL Server Standard Edition (2008 R2, 2012, 2014, 2016, 2017)
- Microsoft SQL Server Web Edition (2008 R2, 2012, 2014, 2016, 2017)
- Microsoft SQL Server Express Edition (2008 R2, 2012, 2014, 2016, 2017)

Bring Your Own License (BYOL)

If you participate in Microsoft's License Mobility program, you can bring your own license to Amazon RDS. Microsoft's License Mobility program allows Microsoft customers to easily move current on-premises Microsoft Server application workloads to Amazon Web Services (AWS), without any additional Microsoft software license fees. This benefit is available to Microsoft Volume Licensing customers with eligible server applications covered by active Microsoft Software Assurance contracts. For the latest licensing terms, see Microsoft's Product Use Rights.

The Bring Your Own License model is supported on Amazon RDS for the following Microsoft SQL Server database editions:

- Microsoft SQL Server Enterprise Edition (2008 R2, 2012, 2014, 2016, 2017)
- Microsoft SQL Server Standard Edition (2008 R2, 2012, 2014, 2016, 2017)

For more information about License Mobility, see SQL License Mobility.

Licensing Microsoft SQL Server Multi-AZ Deployments

Amazon RDS supports Multi-AZ deployments for DB instances running Microsoft SQL Server by using SQL Server Database Mirroring. We recommend Multi-AZ for production workloads. For more information, see Multi-AZ Deployments for Microsoft SQL Server with Database Mirroring (p. 787).

There are no additional licensing requirements for Multi-AZ deployments.

- License Included — Multi-AZ deployments are already included in the software license.

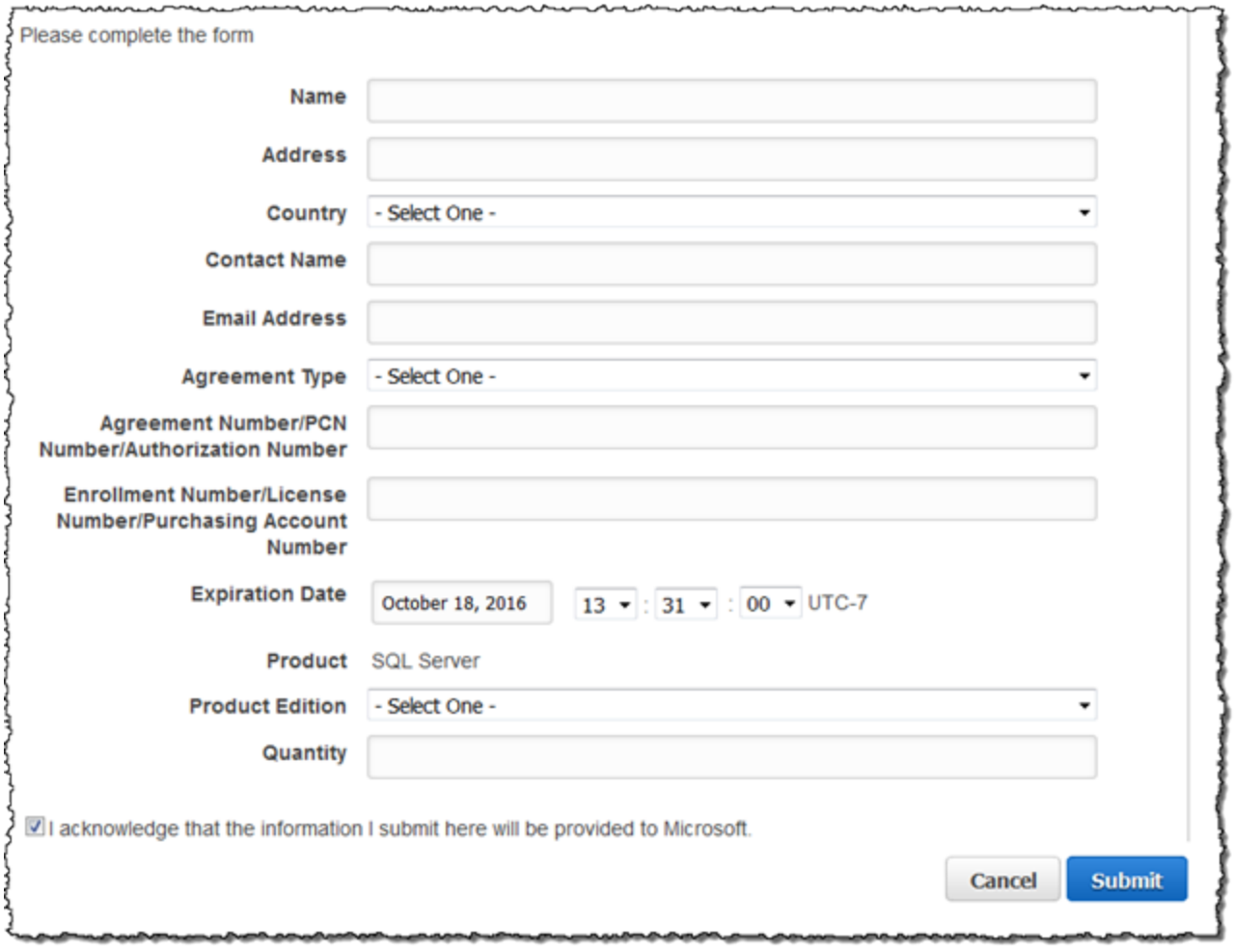
- Bring Your Own License — The secondary instance of a SQL Server Multi-AZ deployment is passive. The secondary instance does not take writes or provide reads until a failover occurs. Therefore, you don't need a license for this secondary instance.

Providing External License Information

To use the Bring Your Own License model, you must provide your Microsoft License Mobility Agreement information in the **External Licenses** section of the Amazon RDS console. Fill out the form once for each License Mobility Agreement you have with Microsoft.

> **Note**
> In some cases, you provide your License Mobility Agreement information on the AWS website instead of the Amazon RDS console. Use the AWS website form instead of the console form if your DB instance is in the US East (Ohio), AWS GovCloud (US), or Asia Pacific (Mumbai) region.

The following image shows the License Mobility Agreement verification form on the Amazon RDS console.



Restoring License-Terminated DB Instances

Microsoft has requested that some Amazon RDS customers who did not report their Microsoft License Mobility information terminate their DB instance. Amazon RDS takes snapshots of these DB instances, and you can restore from the snapshot to a new DB instance that has the License Included model.

For more information, see Restoring License-Terminated DB Instances (p. 807).

Related Topics

- Microsoft SQL Server on Amazon RDS (p. 720)
- Creating a DB Instance Running the Microsoft SQL Server Database Engine (p. 738)

Creating a DB Instance Running the Microsoft SQL Server Database Engine

The basic building block of Amazon RDS is the DB instance. Your Amazon RDS DB instance is similar to your on-premises Microsoft SQL Server. After you create your SQL Server DB instance, you can add one or more custom databases to it.

> **Important**
> You must complete the tasks in the Setting Up for Amazon RDS (p. 5) section before you can create or connect to a DB instance.

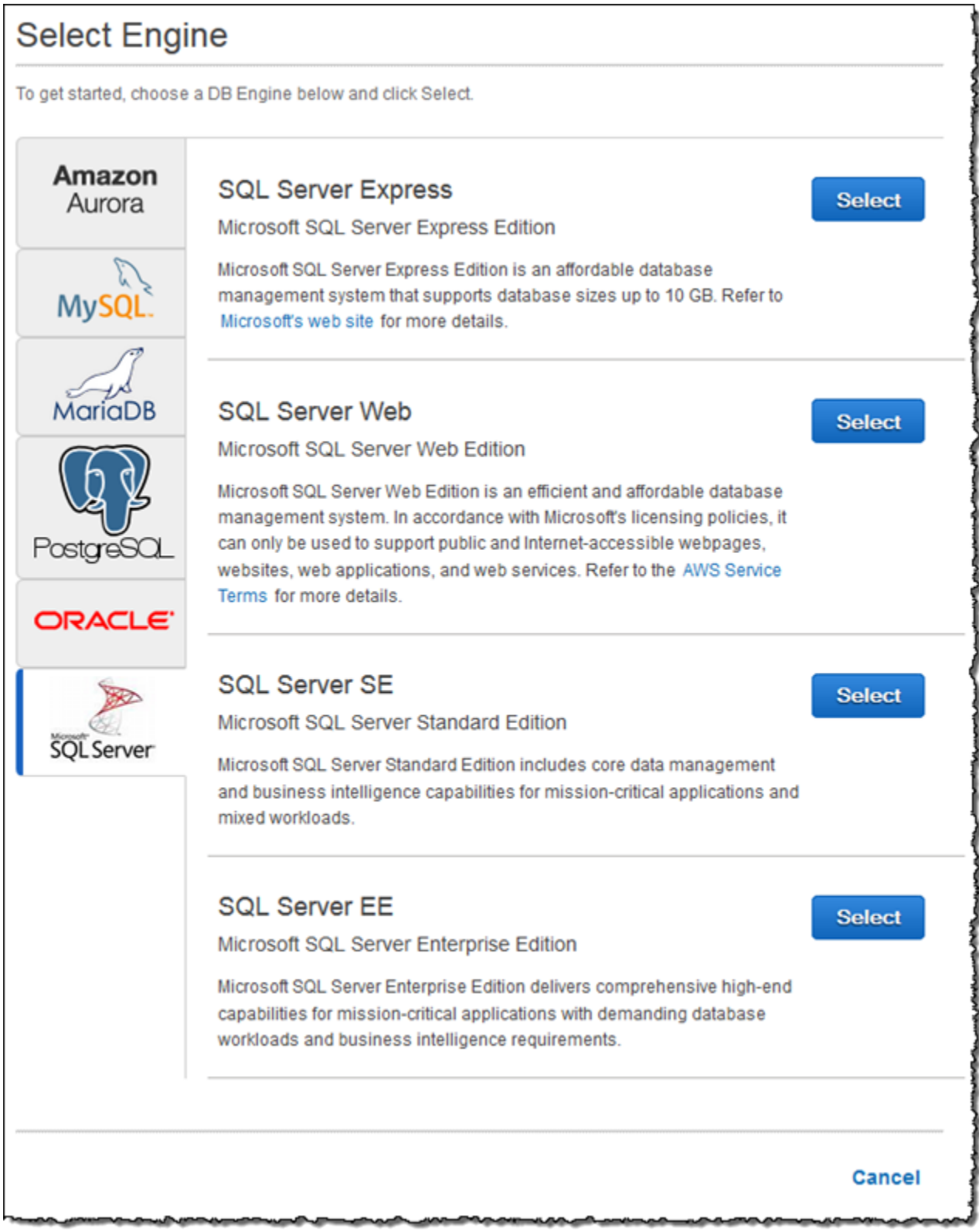
For an example that walks you through the process of creating and connecting to a sample DB instance, see Creating a Microsoft SQL Server DB Instance and Connecting to a DB Instance (p. 25).

AWS Management Console

To launch a SQL Server DB instance

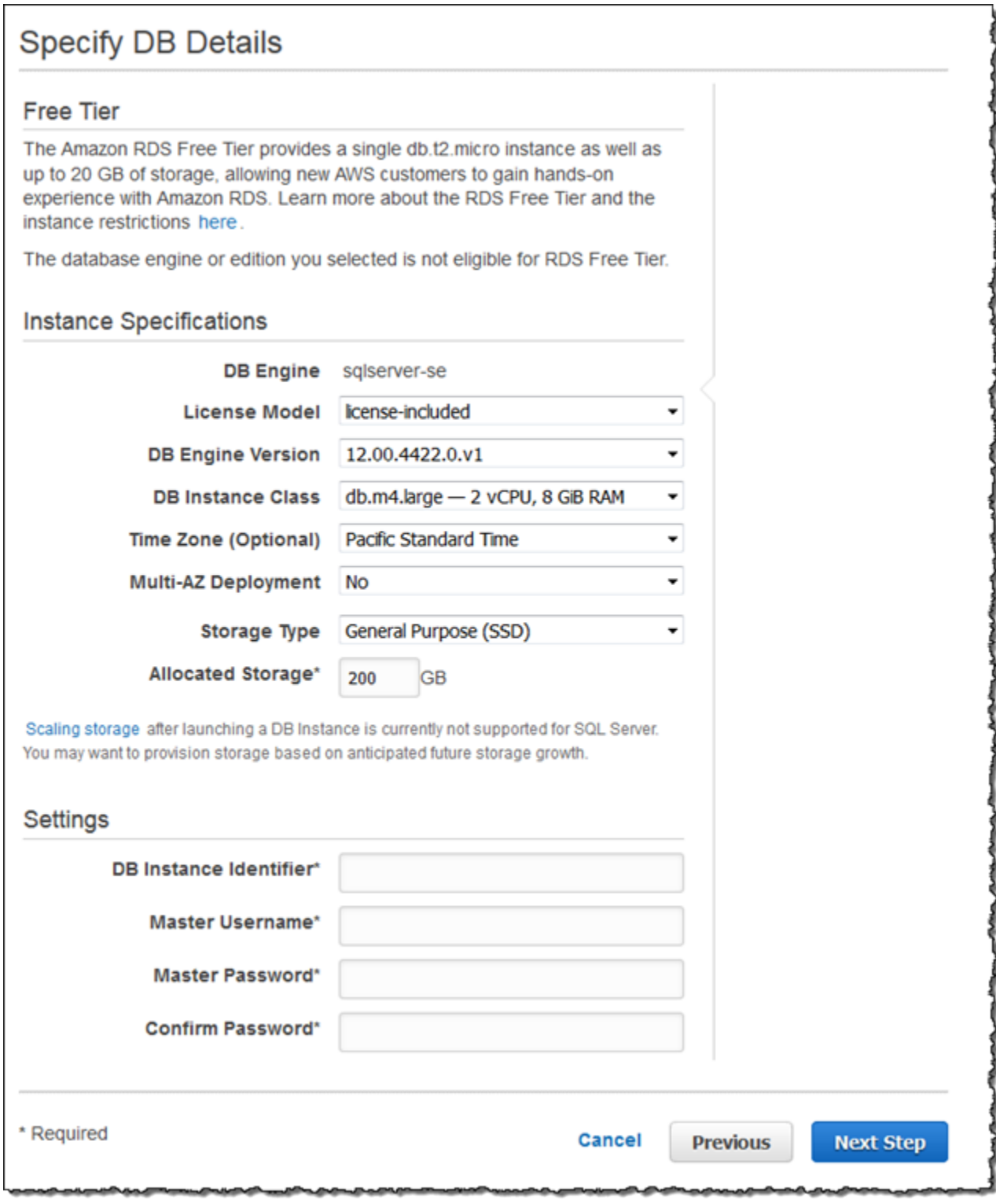
1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the top right corner of the Amazon RDS console, choose the region in which you want to create the DB instance.
3. In the navigation pane, choose **Instances**.
4. Choose **Launch DB Instance** to start the **Launch DB Instance Wizard**.

   The wizard opens on the **Select Engine** page. The SQL Server editions that are available vary by region.


Select Engine
To get started, choose a DB Engine below and click Select.
Amazon Aurora
MySQL
MariaDB
PostgreSQL
ORACLE
SQL Server
SQL Server Express
Microsoft SQL Server Express Edition
Microsoft SQL Server Express Edition is an affordable database management system that supports database sizes up to 10 GB. Refer to Microsoft's web site for more details.
Select
SQL Server Web
Microsoft SQL Server Web Edition
Microsoft SQL Server Web Edition is an efficient and affordable database management system. In accordance with Microsoft's licensing policies, it can only be used to support public and Internet-accessible webpages, websites, web applications, and web services. Refer to the AWS Service Terms for more details.
Select
SQL Server SE
Microsoft SQL Server Standard Edition
Microsoft SQL Server Standard Edition includes core data management and business intelligence capabilities for mission-critical applications and mixed workloads.
Select
SQL Server EE
Microsoft SQL Server Enterprise Edition
Microsoft SQL Server Enterprise Edition delivers comprehensive high-end capabilities for mission-critical applications with demanding database workloads and business intelligence requirements.
Select
Cancel

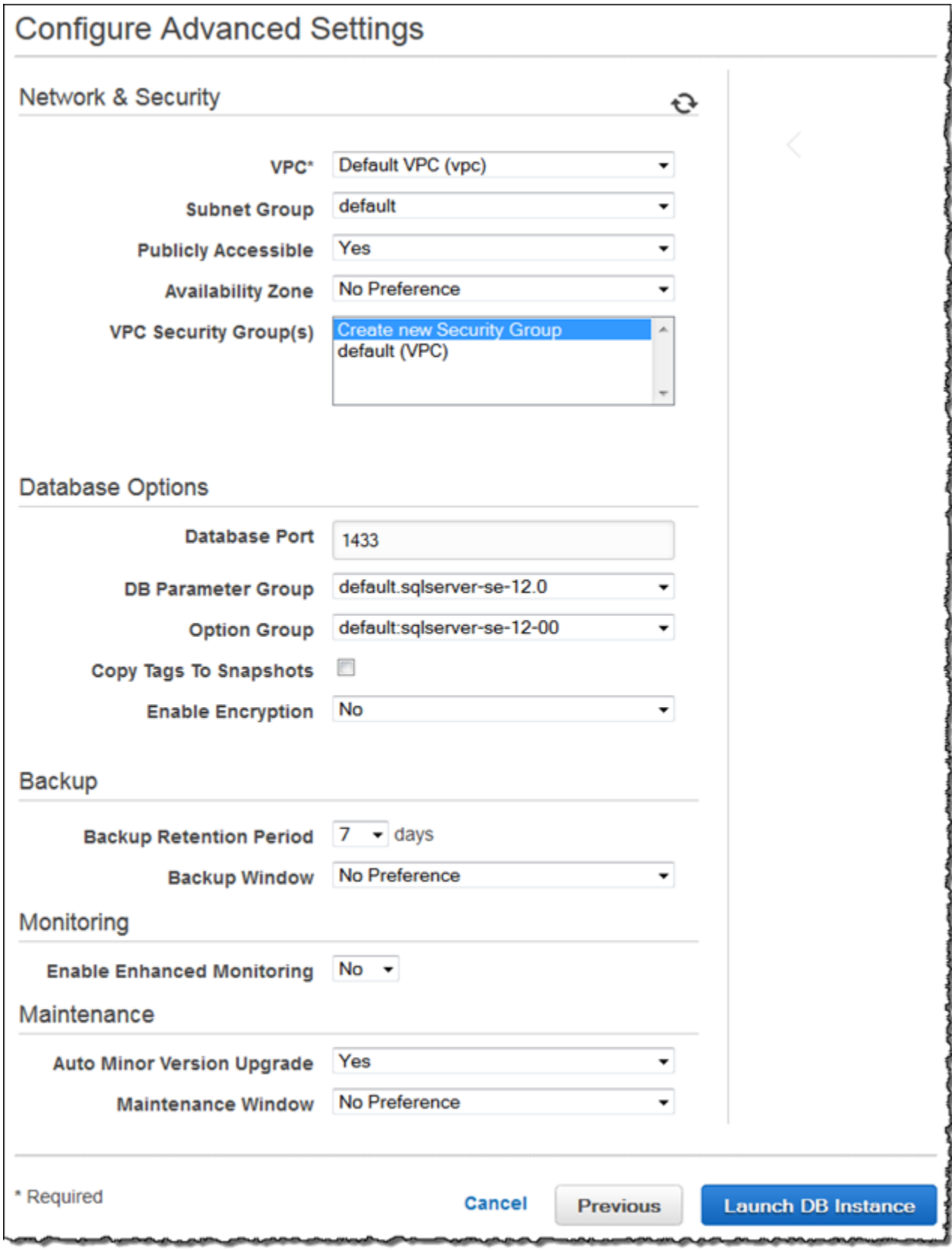
5. In the **Select Engine** window, choose the SQL Server icon and then choose the **Select** button for the SQL Server DB engine edition that you want to use. The SQL Server editions that are available vary by region.
6. The **Production?** step asks if you are planning to use the DB instance you are creating for production. If you are, choose **Yes**. If you choose **Yes**, then the failover option, **Multi-AZ**, and **Provisioned IOPS** are all preselected in the following step. We recommend these features for any production environment.
7. Choose **Next** to continue. The **Specify DB Details** page appears.

   On the **Specify DB Details** page, specify your DB instance information. For information about each setting, see Settings for Microsoft SQL Server DB Instances (p. 745).



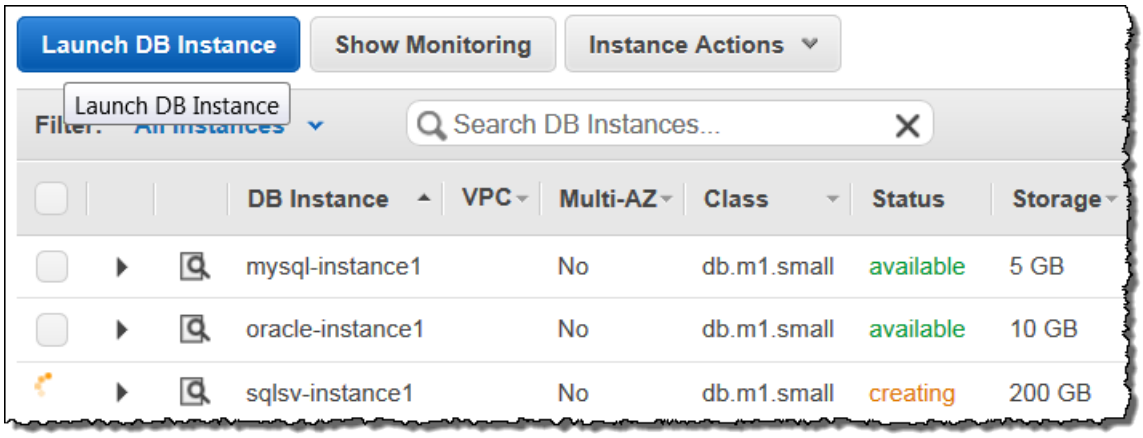
8. Choose **Next** to continue. The **Configure Advanced Settings** page appears.

   On the **Configure Advanced Settings** page, provide additional information that Amazon RDS needs to launch the DB instance. For information about each setting, see Settings for Microsoft SQL Server DB Instances (p. 745).


Configure Advanced Settings
Network & Security
VPC*
Default VPC (vpc)
Subnet Group
default
Publicly Accessible
Yes
Availability Zone
No Preference
VPC Security Group(s)
Create new Security Group
default (VPC)
Database Options
Database Port
1433
DB Parameter Group
default.sqlserver-se-12.0
Option Group
default:sqlserver-se-12-00
Copy Tags To Snapshots
Enable Encryption
No
Backup
Backup Retention Period
7
days
Backup Window
No Preference
Monitoring
Enable Enhanced Monitoring
No
Maintenance
Auto Minor Version Upgrade
Yes
Maintenance Window
No Preference
* Required
Cancel
Previous
Launch DB Instance

9. Choose **Launch DB Instance**.
10. On the final page of the wizard, choose **Close**.

On the RDS console, the new DB instance appears in the list of DB instances. The DB instance has a status of **creating** until the DB instance is ready to use. When the state changes to **available**, you can connect to the DB instance. Depending on the DB instance class and the amount of storage, it can take up to 20 minutes before the new instance is available.



CLI

To create a Microsoft SQL Server DB instance by using the AWS CLI, call the create-db-instance command with the parameters below. For information about each setting, see Settings for Microsoft SQL Server DB Instances (p. 745).

- `--db-instance-identifier`
- `--db-instance-class`
- `--db-security-groups`
- `--db-subnet-group`
- `--engine`
- `--master-user-name`
- `--master-user-password`
- `--allocated-storage`
- `--backup-retention-period`

Example

For Linux, OS X, or Unix:

```
aws rds create-db-instance
    --engine sqlserver-se \
    --db-instance-identifier mymsftsqlserver \
    --allocated-storage 250 \
    --db-instance-class db.m1.large \
    --db-security-groups mydbsecuritygroup \
    --db-subnet-group mydbsubnetgroup \
    --master-user-name masterawsuser \
    --master-user-password masteruserpassword \
    --backup-retention-period 3
```

For Windows:

```
aws rds create-db-instance ^
    --engine sqlserver-se ^
    --db-instance-identifier mydbinstance ^
    --allocated-storage 250 ^
    --db-instance-class db.m1.large ^
    --db-security-groups mydbsecuritygroup ^
    --db-subnet-group mydbsubnetgroup ^
    --master-user-name masterawsuser ^
    --master-user-password masteruserpassword ^
    --backup-retention-period 3
```

This command should produce output similar to the following:

```
DBINSTANCE  mydbinstance  db.m1.large  sqlserver-se  250  sa  creating  3  ****  n
 10.50.2789
SECGROUP  default  active
PARAMGRP  default.sqlserver-se-10.5  in-sync
```

API

To create a Microsoft SQL Server DB instance by using the Amazon RDS API, call the CreateDBInstance action with the parameters below. For information about each setting, see Settings for Microsoft SQL Server DB Instances (p. 745).

- AllocatedStorage
- BackupRetentionPeriod
- DBInstanceClass
- DBInstanceIdentifier
- DBSecurityGroups
- DBSubnetGroup
- Engine
- MasterUsername
- MasterUserPassword

Example

```
https://rds.amazonaws.com/
    ?Action=CreateDBInstance
    &AllocatedStorage=250
    &BackupRetentionPeriod=3
    &DBInstanceClass=db.m1.large
    &DBInstanceIdentifier=mydbinstance
    &DBSecurityGroups.member.1=mysecuritygroup
    &DBSubnetGroup=mydbsubnetgroup
    &Engine=sqlserver-se
    &MasterUserPassword=masteruserpassword
    &MasterUsername=masterawsuser
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-10-31
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20140305/us-west-1/rds/aws4_request
    &X-Amz-Date=20140305T185838Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
```

```
&X-Amz-Signature=b441901545441d3c7a48f63b5b1522c5b2b37c137500c93c45e209d4b3a064a3
```

Settings for Microsoft SQL Server DB Instances

The following table contains details about settings that you choose when you create a SQL Server DB instance.

Setting	Setting Description
Allocated Storage	The amount of storage to allocate for your DB instance (in gigabytes). In some cases, allocating a higher amount of storage for your DB instance than the size of your database can improve I/O performance. For more information, see Storage for Amazon RDS (p. 410).
Auto Minor Version Upgrade	**Yes** to enable your DB instance to receive minor DB engine version upgrades automatically when they become available.
Availability Zone	The availability zone for your DB instance. Use the default value of **No Preference** unless you want to specify an Availability Zone. For more information, see Regions and Availability Zones (p. 97).
Backup Retention Period	The number of days that you want automatic backups of your DB instance to be retained. For any non-trivial DB instance, you should set this value to `1` or greater. For more information, see Working With Backups (p. 201).
Backup Window	The time period during which Amazon RDS automatically takes a backup of your DB instance. Unless you have a specific time that you want to have your database backup, use the default of **No Preference**. For more information, see Working With Backups (p. 201).
Copy Tags To Snapshots	Select this option to copy any DB instance tags to a DB snapshot when you create a snapshot. For more information, see Tagging Amazon RDS Resources (p. 129).
Database Port	The port that you want to access the DB instance through. SQL Server installations default to port 1433. If you use a DB security group with your DB instance, this must be the same port value you provided when creating the DB security group.
DB Engine Version	The version of Microsoft SQL Server that you want to use.
DB Instance Class	The configuration for your DB instance. For example, a **db.m1.small** instance class equates to 1.7 GB memory, 1 ECU (1 virtual core with 1 ECU), 64-bit platform, and moderate I/O capacity.

Setting	Setting Description
	If possible, choose an instance class large enough that a typical query working set can be held in memory. When working sets are held in memory the system can avoid writing to disk, and this improves performance. For more information, see DB Instance Class (p. 92) and DB Instance Class Support for Microsoft SQL Server (p. 723).
DB Instance Identifier	The name for your DB instance. Name your DB instances in the same way that you would name your on-premises servers. Your DB instance identifier can contain up to 63 alphanumeric characters, and must be unique for your account in the region you chose. You can add some intelligence to the name, such as including the region and DB engine you chose, for example **`sqlsv-instance1`**.
DB Parameter Group	A parameter group for your DB instance. You can choose the default parameter group or you can create a custom parameter group. For more information, see Working with DB Parameter Groups (p. 170).
Enable Encryption	**Yes** to enable encryption at rest for this DB instance. For more information, see Encrypting Amazon RDS Resources (p. 355).
Enable Enhanced Monitoring	**Yes** to gather metrics in real time for the operating system that your DB instance runs on. For more information, see Enhanced Monitoring (p. 258).
License Model	The license model that you want to use. Choose **license-included** to use the general license agreement for Microsoft SQL Server. Choose **bring-your-own-license** to use your existing license. To use the Bring Your Own License model, you must provide your Microsoft License Mobility Agreement information in the **External Licenses** section of the Amazon RDS console. For more information, see Providing External License Information (p. 736).
Maintenance Window	The 30 minute window in which pending modifications to your DB instance are applied. If the time period doesn't matter, choose **No Preference**. For more information, see The Amazon RDS Maintenance Window (p. 103).

Setting	Setting Description
Master Username	The name that you use as the master user name to log on to your DB Instance with all database privileges. The master user name is a SQL Server Authentication login that is a member of the `processadmin`, `public`, and `setupadmin` fixed server roles. For more information, see Microsoft SQL Server Security (p. 724).
Master User Password	The password for your master user account. The password must contain from 8 to 128 printable ASCII characters (excluding /,", a space, and @).
Multi-AZ Deployment	**Yes** to create a standby mirror of your DB instance in another Availability Zone for failover support. We recommend Multi-AZ for production workloads to maintain high availability. For development and testing, you can choose **No**. For more information, see Multi-AZ Deployments for Microsoft SQL Server with Database Mirroring (p. 787).
Option Group	An option group for your DB instance. You can choose the default option group or you can create a custom option group. For more information, see Working with Option Groups (p. 153).
Publicly Accessible	**Yes** to give your DB instance a public IP address. This means that it is accessible outside the VPC (the DB instance also needs to be in a public subnet in the VPC). Choose **No** if you want the DB instance to only be accessible from inside the VPC. For more information, see Hiding a DB Instance in a VPC from the Internet (p. 401).
Storage Type	The storage type for your DB instance. For more information, see Amazon RDS Storage Types (p. 410).
Subnet Group	This setting depends on the platform you are on. If you are a new customer to AWS, choose **default**, which is the default DB subnet group that was created for your account. If you are creating a DB instance on the previous E2-Classic platform and you want your DB instance in a specific VPC, choose the DB subnet group you created for that VPC.
Time Zone	The time zone for your DB instance. If you don't choose a time zone, your DB instance uses the default time zone. For more information, see Local Time Zone for Microsoft SQL Server DB Instances (p. 731).

Setting	Setting Description
VPC	This setting depends on the platform you are on. If you are a new customer to AWS, choose the default VPC shown. If you are creating a DB instance on the previous E2-Classic platform that does not use a VPC, choose **Not in VPC**. For more information, see Amazon Virtual Private Cloud (VPCs) and Amazon RDS (p. 390).
VPC Security Group	If you are a new customer to AWS, choose the default VPC. Otherwise, choose the VPC security group you previously created. For more information, see Working with DB Security Groups (EC2-Classic Platform) (p. 380).

Related Topics

- Tutorial: Create an Amazon VPC for Use with an Amazon RDS DB Instance (p. 406)
- Connecting to a DB Instance Running the Microsoft SQL Server Database Engine (p. 749)
- Modifying a DB Instance Running the Microsoft SQL Server Database Engine (p. 756)
- Deleting a DB Instance (p. 126)

Connecting to a DB Instance Running the Microsoft SQL Server Database Engine

After Amazon RDS provisions your DB instance, you can use any standard SQL client application to connect to the DB instance. In this topic you connect to your DB instance by using either Microsoft SQL Server Management Studio (SSMS) or SQL Workbench/J.

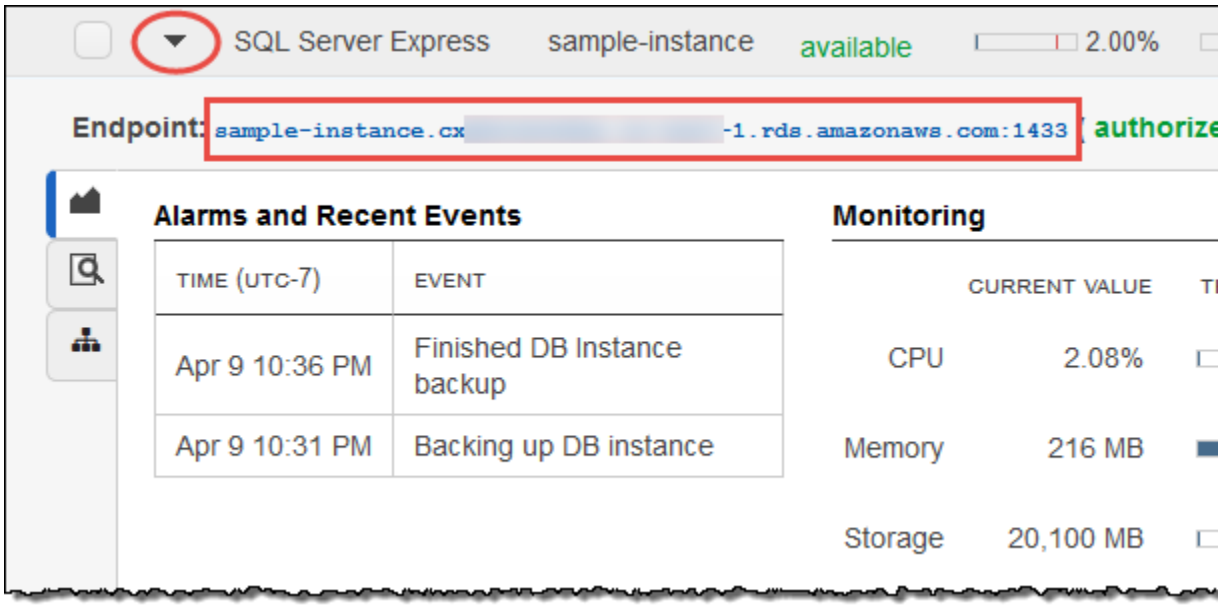
For an example that walks you through the process of creating and connecting to a sample DB instance, see Creating a Microsoft SQL Server DB Instance and Connecting to a DB Instance (p. 25).

Connecting to Your DB Instance with Microsoft SQL Server Management Studio

In this procedure you connect to your sample DB instance by using Microsoft SQL Server Management Studio (SSMS). To download a stand-alone version of this utility, see Download SQL Server Management Studio (SSMS) in the Microsoft documentation.

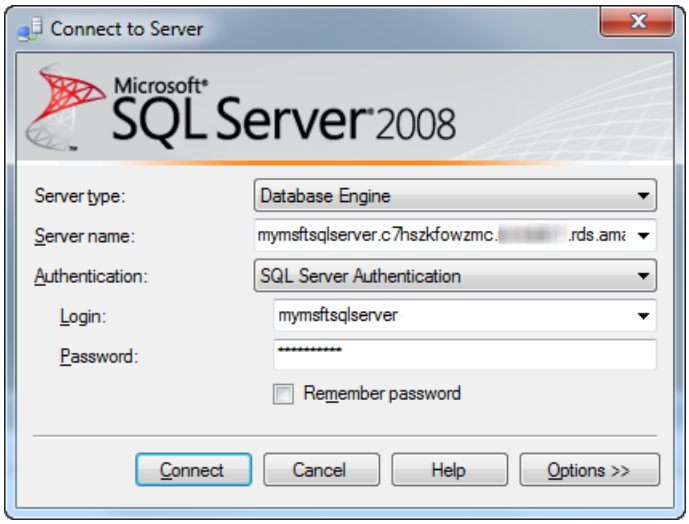
To connect to a DB Instance using SSMS

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the top right corner of the Amazon RDS console, select the region of your DB instance.
3. Find the DNS name and port number for your DB Instance.
   a. Open the RDS console and then choose **Instances** to display a list of your DB instances.
   b. Choose the row for your SQL Server DB instance to display the summary information for the instance.



   c. Copy the endpoint. The **Endpoint** field has two parts separated by a colon (:). The part before the colon is the DNS name for the instance, the part following the colon is the port number. Copy both parts.
4. Start SQL Server Management Studio.

The **Connect to Server** dialog box appears.



5. Provide the information for your DB instance.

   a. For **Server type**, choose **Database Engine**.

   b. For **Server name**, type or paste the DNS name and port number of your DB Instance, separated by a comma.

      Important
      Change the colon between the DNS name and port number to a comma.

      For example, your server name should look like the following:

      ```
      sample-instance.cg034hpkmmjt.us-east-1.rds.amazonaws.com,1433
      ```

   c. For **Authentication**, choose **SQL Server Authentication**.

   d. For **Login**, type the master user name for your DB instance.

   e. For **Password**, type the password for your DB instance.

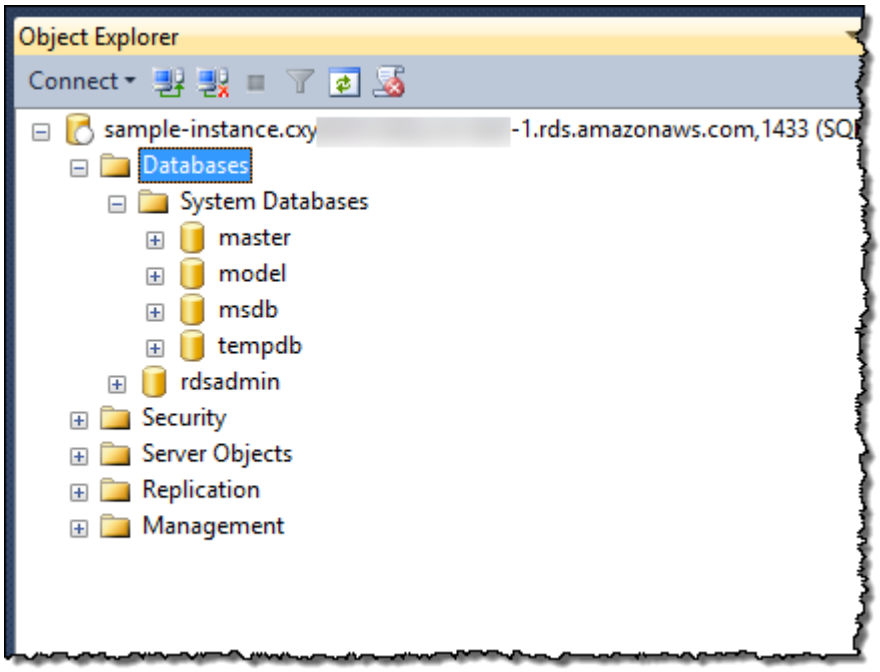
6. Choose **Connect**.

   After a few moments, SSMS connects to your DB instance. If you can't connect to your DB instance, see Security Group Considerations (p. 754) and Troubleshooting the Connection to Your SQL Server DB Instance (p. 754).

7. Your SQL Server DB instance comes with SQL Server's standard built-in system databases (master, model, msdb, and tempdb). To explore the system databases, do the following:

   a. In SSMS, on the **View** menu, choose **Object Explorer**.

   b. Expand your DB instance, expand **Databases**, and then expand **System Databases** as shown following.

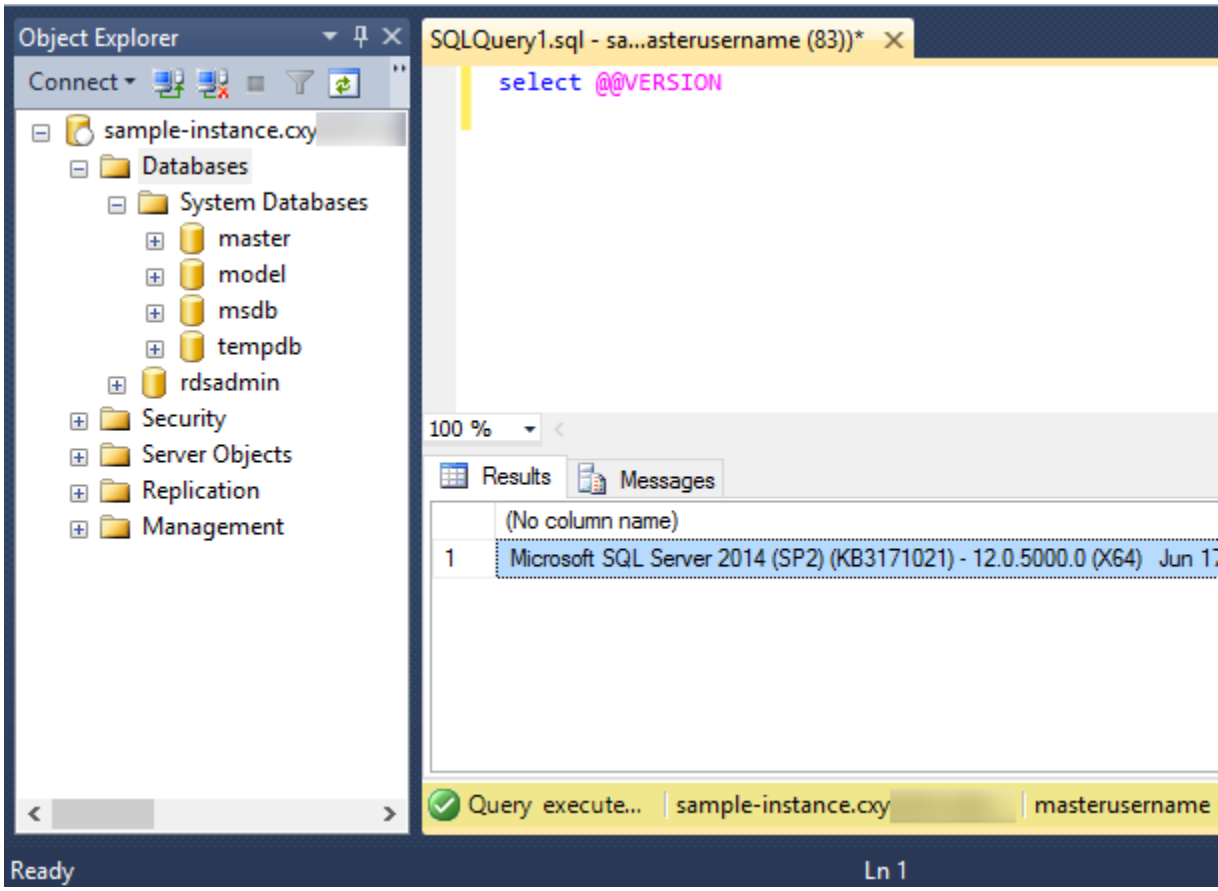


8. Your SQL Server DB instance also comes with a database named `rdsadmin`. Amazon RDS uses this database to store the objects that it uses to manage your database. The `rdsadmin` database also includes stored procedures that you can run to perform advanced tasks. For more information, see Common DBA Tasks for Microsoft SQL Server (p. 800).
9. You can now start creating your own databases and running queries against your DB instance and databases as usual. To run a test query against your DB instance, do the following:

   a. In SSMS, on the **File** menu point to **New** and then choose **Query with Current Connection**.
   b. Type the following SQL query:

   ```
   select @@VERSION
   ```

   c. Run the query. SSMS returns the SQL Server version of your Amazon RDS DB instance.



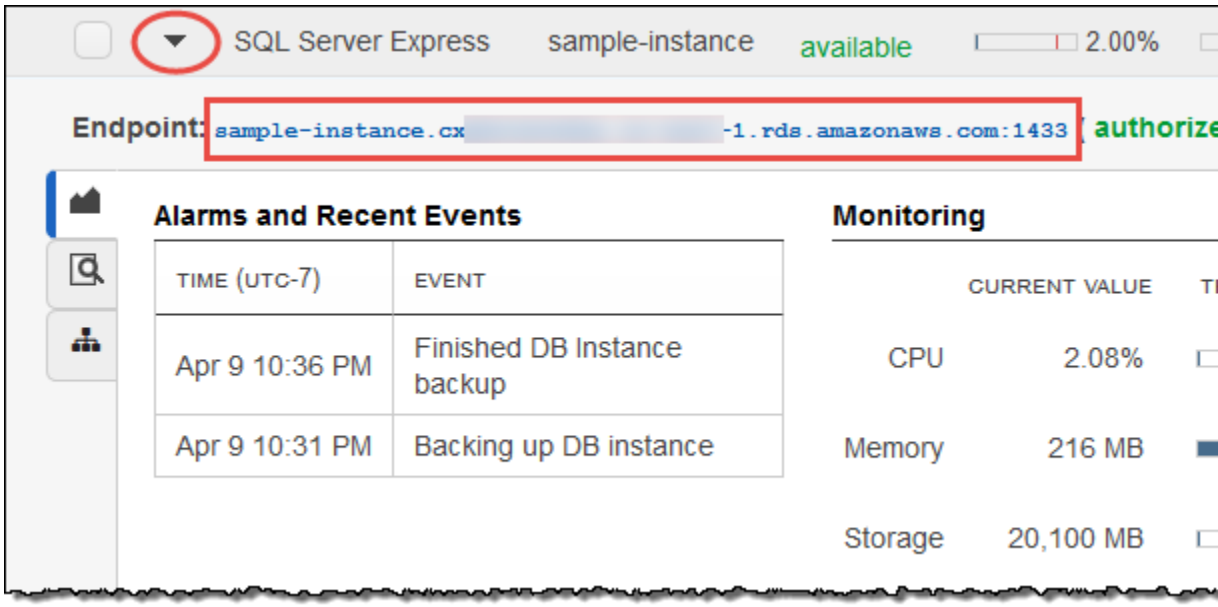
Connecting to Your DB Instance with SQL Workbench/J

This example shows how to connect to a DB instance running the Microsoft SQL Server database engine by using the SQL Workbench/J database tool. To download SQL Workbench/J, see SQL Workbench/J.

SQL Workbench/J uses JDBC to connect to your DB instance. You also need the JDBC driver for SQL Server. To download this driver, see Microsoft JDBC Drivers 4.1 (Preview) and 4.0 for SQL Server.

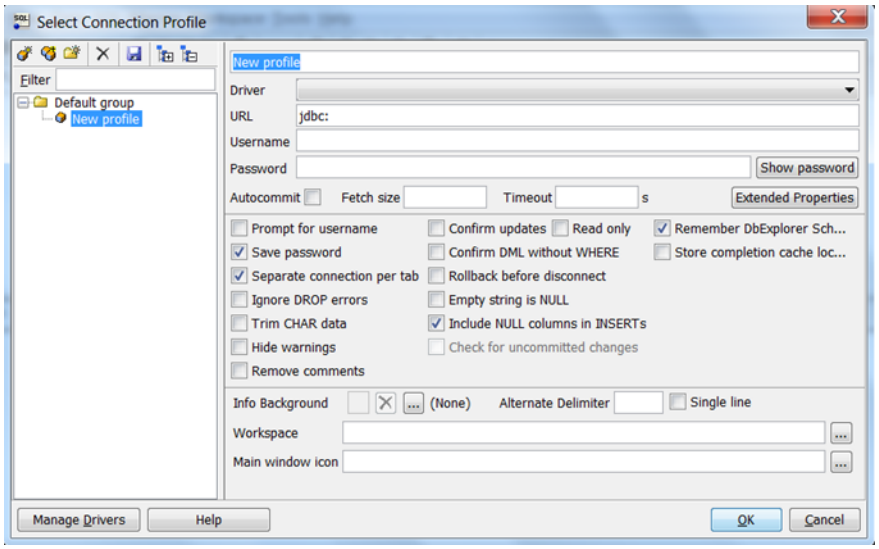
To connect to a DB instance using SQL Workbench

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the top right corner of the Amazon RDS console, select the region of your DB instance.
3. Find the DNS name and port number for your DB Instance.
   a. Open the RDS console and then choose **Instances** to display a list of your DB instances.
   b. Choose the row for your SQL Server DB instance to display the summary information for the instance.



c. Copy the endpoint. The **Endpoint** field has two parts separated by a colon (:). The part before the colon is the DNS name for the instance, the part following the colon is the port number. Copy both parts.

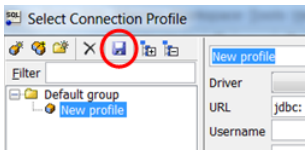
4. Open SQL Workbench/J. The **Select Connection Profile** dialog box appears, as shown following:



5. In the first box at the top of the dialog box, enter a name for the profile.
6. For **Driver**, select **`SQL JDBC 4.0`**.
7. For **URL**, type **`jdbc:sqlserver://`**, then type or paste the endpoint of your DB instance. For example, the URL value could be the following:

```
jdbc:sqlserver://sqlsvr-pdz.abcd12340.us-west-2.rds.amazonaws.com:1433
```

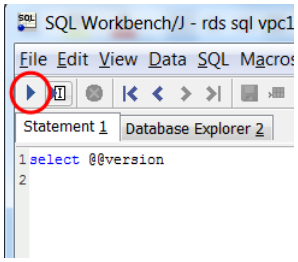
8. For **Username**, type or paste the master user name for the DB instance.
9. For **Password**, type the password for the master user.
10. Choose the save icon in the dialog toolbar, as shown following:



11. Choose **OK**. After a few moments, SQL Workbench/J connects to your DB instance. If you can't connect to your DB instance, see Security Group Considerations (p. 754) and Troubleshooting the Connection to Your SQL Server DB Instance (p. 754).
12. In the query pane, type the following SQL query:

```
select @@VERSION
```

13. Choose the execute icon in the toolbar, as shown following:



The query returns the version information for your DB instance, similar to the following:

```
Microsoft SQL Server 2012 - 11.0.2100.60 (X64)
```

Security Group Considerations

To connect to your DB instance, your DB instance must be associated with a security group that contains the IP addresses and network configuration that you use to access the DB instance. You may have associated your DB instance with an appropriate security group when you created your DB instance. If you assigned a default, non-configured security group when you created your DB instance, your DB instance firewall prevents connections.

If you need to create a new security group to enable access, the type of security group that you create will depend on what Amazon EC2 platform your DB instance is on. To determine your platform, see Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform (p. 391). In general, if your DB instance is on the *EC2-Classic* platform, you create a DB security group; if your DB instance is on the *VPC* platform, you create a VPC security group. For instructions on creating a new security group, see Amazon RDS Security Groups (p. 375).

After you have created the new security group, you modify your DB instance to associate it with the security group. For more information, see Modifying a DB Instance Running the Microsoft SQL Server Database Engine (p. 756).

You can enhance security by using SSL to encrypt connections to your DB instance. For more information, see Using SSL with a Microsoft SQL Server DB Instance (p. 791).

Troubleshooting the Connection to Your SQL Server DB Instance

The following are issues you might encounter when you attempt to connect to your SQL Server DB instance.

Issue	Troubleshooting Suggestions
Unable to connect to your DB instance.	For a newly-created DB instance, the DB instance has a status of **creating** until the DB instance is ready to use. When the state changes to **available**, you can connect to the DB instance. Depending on the DB instance class and the amount of storage, it can take up to 20 minutes before the new instance is available.
Unable to connect to your DB instance.	If you can't send or receive communications over the port that you specified when you created the DB instance, you can't connect to the DB instance. Check with your network administrator to verify that the port you specified for your DB instance allows inbound and outbound communication.
Unable to connect to your DB instance.	The access rules enforced by your local firewall and the IP addresses you authorized to access your DB instance in the security group for the DB instance might not not match. The problem is most likely the egress or ingress rules on your firewall. For more information about security groups, see Amazon RDS Security Groups (p. 375). For a topic that walks you through the process of setting up rules for your security group, see Tutorial: Create an Amazon VPC for Use with an Amazon RDS DB Instance (p. 406).
Could not open a connection to SQL Server – Microsoft SQL Server, Error: 53	Make sure specified the server name correctly. For **Server name**, type or paste the DNS name and port number of your sample DB Instance, separated by a comma. **Important** Change the colon between the DNS name and port number to a comma. For example, your server name should look like the following: `sample-instance.cg034hpkmmjt.us-east-1.rds.amazonaws.com,1433`
No connection could be made because the target machine actively refused it – Microsoft SQL Server, Error: 10061	You were able to reach the DB instance but the connection was refused. This is usually caused by specifying the user name or password incorrectly. Verify the user name and password and then retry.

Related Topics

- Creating a DB Instance Running the Microsoft SQL Server Database Engine (p. 738)
- Modifying a DB Instance Running the Microsoft SQL Server Database Engine (p. 756)
- Deleting a DB Instance (p. 126)

Modifying a DB Instance Running the Microsoft SQL Server Database Engine

You can change the settings of a DB instance to accomplish tasks such as changing the instance class or renaming the instance. This topic guides you through modifying an Amazon RDS DB instance running Microsoft SQL Server, and describes the settings for SQL Server DB instances.

We recommend that you test any changes on a test instance before modifying a production instance, so that you fully understand the impact of each change. This is especially important when upgrading database versions.

After you modify your DB instance settings, you can apply the changes immediately, or apply them during the next maintenance window for the DB instance. Some modifications cause an interruption by restarting the DB instance.

AWS Management Console

To modify an SQL Server DB Instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **DB Instances**, and then select the DB instance that you want to modify.
3. Choose **Instance Actions**, and then choose **Modify**. The **Modify DB Instance** page appears.
4. Change any of the settings that you want. For information about each setting, see Settings for Microsoft SQL Server DB Instances (p. 757).
5. To apply the changes immediately, select **Apply Immediately**. Selecting this option can cause an outage in some cases. For more information, see The Impact of Apply Immediately (p. 114).
6. When all the changes are as you want them, choose **Continue**.
7. On the confirmation page, review your changes. If they are correct, choose **Modify DB Instance** to save your changes.

   Alternatively, choose **Back** to edit your changes, or choose **Cancel** to cancel your changes.

CLI

To modify a Microsoft SQL Server DB instance by using the AWS CLI, call the modify-db-instance command. Specify the DB instance identifier, and the parameters for the settings that you want to modify. For information about each parameter, see Settings for Microsoft SQL Server DB Instances (p. 757).

Example

The following code modifies `mydbinstance` by setting the backup retention period to 1 week (7 days). The code disables automatic minor version upgrades by using `--no-auto-minor-version-upgrade`. To allow automatic minor version upgrades, use `--auto-minor-version-upgrade`. The changes are applied during the next maintenance window by using `--no-apply-immediately`. Use `--apply-immediately` to apply the changes immediately. For more information, see The Impact of Apply Immediately (p. 114).

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
```

```
    --db-instance-identifier mydbinstance \
    --backup-retention-period 7 \
    --no-auto-minor-version-upgrade \
    --no-apply-immediately
```

For Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --backup-retention-period 7 ^
    --no-auto-minor-version-upgrade ^
    --no-apply-immediately
```

API

To modify a Microsoft SQL Server DB instance by using the Amazon RDS API, call the ModifyDBInstance action. Specify the DB instance identifier, and the parameters for the settings that you want to modify. For information about each parameter, see Settings for Microsoft SQL Server DB Instances (p. 757).

Example

The following code modifies `mydbinstance` by setting the backup retention period to 1 week (7 days) and disabling automatic minor version upgrades. These changes are applied during the next maintenance window.

```
https://rds.amazonaws.com/
    ?Action=ModifyDBInstance
    &ApplyImmediately=false
    &AutoMinorVersionUpgrade=false
    &BackupRetentionPeriod=7
    &DBInstanceIdentifier=mydbinstance
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-10-31
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20131016/us-west-1/rds/aws4_request
    &X-Amz-Date=20131016T233051Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=087a8eb41cb1ab0fc9ec1575f23e73757ffc6a1e42d7d2b30b9cc0be988cff97
```

Settings for Microsoft SQL Server DB Instances

The following table contains details about which settings you can modify, which settings you can't modify, when the changes can be applied, and whether the changes cause downtime for the DB instance.

Setting	Setting Description	When the Change Occurs	Downtime Notes
Allocated Storage	The storage, in gigabytes, that you want to allocate for your DB instance. You can only increase the allocated storage, you can't reduce the allocated storage. The maximum storage allowed is 16 TB. **Warning** Once Amazon RDS begins to modify your DB instance	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	A short outage of a few minutes may occur. After that, the DB instance is online but in the storage-optimization state. Performance may be degraded during storage

Setting	Setting Description	When the Change Occurs	Downtime Notes
	to increase the storage size or type, you can't submit another request to increase the storage size or type for 6 hours. You can't modify the storage of some older DB instances, and DB instances restored from older DB snapshots. The Allocated Storage option is disabled in the console if your DB instance isn't eligible. You can also check eligibility by using the AWS CLI command describe-valid-db-instance-modifications which returns the valid storage options for your DB instance. For more information, see Storage for Amazon RDS (p. 410).		optimization. The storage optimization process is usually short, but can sometimes take up to and even beyond 24 hours.
Auto Minor Version Upgrade	**Yes** if you want your DB instance to receive minor engine version upgrades automatically when they become available. Upgrades are installed only during your scheduled maintenance window.	–	–
Backup Retention Period	The number of days that automatic backups are retained. To disable automatic backups, set the backup retention period to 0. For more information, see Working With Backups (p. 201).	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false and you change the setting from a nonzero value to another nonzero value, the change is applied asynchronously, as soon as possible. Otherwise, the change occurs during the next maintenance window.	An outage occurs if you change from 0 to a nonzero value, or from a nonzero value to 0.
Backup Window	The time range during which automated backups of your databases occur. The backup window is a start time in Universal Coordinated Time (UTC), and a duration in hours. For more information, see Working With Backups (p. 201).	The change is applied asynchronously, as soon as possible.	–

Setting	Setting Description	When the Change Occurs	Downtime Notes
Certificate Authority	The certificate that you want to use.	–	–
Copy Tags to Snapshots	If you have any DB instance tags, this option copies them when you create a DB snapshot. For more information, see Tagging Amazon RDS Resources (p. 129).	–	–
Database Port	The port that you want to use to access the database. The port value must not match any of the port values specified for options in the option group for the DB instance.	The change occurs immediately. This setting ignores the **Apply Immediately** setting.	The DB instance is rebooted immediately.
DB Engine Version	The version of the SQL Server database engine that you want to use. Before you upgrade your production DB instances, we recommend that you test the upgrade process on a test instance to verify its duration and to validate your applications. For more information, see Upgrading the Microsoft SQL Server DB Engine (p. 764).	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	An outage occurs during this change.
DB Instance Class	The DB instance class that you want to use. For more information, see DB Instance Class (p. 92) and DB Instance Class Support for Microsoft SQL Server (p. 723).	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	An outage occurs during this change.
DB Instance Identifier	The DB instance identifier. For more information about the effects of renaming a DB instance, see Renaming a DB Instance (p. 116).	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	An outage occurs during this change. The DB instance is rebooted.

Setting	Setting Description	When the Change Occurs	Downtime Notes
DB Parameter Group	The parameter group that you want associated with the DB instance. For more information, see Working with DB Parameter Groups (p. 170).	The parameter group change occurs immediately. However, parameter changes only occur when you reboot the DB instance manually without failover. For more information, see Rebooting a DB Instance (p. 119).	An outage doesn't occur during this change. However, parameter changes only occur when you reboot the DB instance manually without failover.
Domain	The Active Directory Domain to move the instance to. Specify none to remove the instance from its current domain. The domain must exist prior to this operation. For more information, see Using Windows Authentication with a Microsoft SQL Server DB Instance (p. 812).	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	A brief outage occurs during this change. For single-AZ DB instances, the outage is approximately 5-10 minutes. For multi-AZ DB instances, the outage is approximately 1 minute.
Domain IAM Role Name	The name of the IAM role to use when accessing the Active Directory Service. For more information, see Using Windows Authentication with a Microsoft SQL Server DB Instance (p. 812).	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	A brief outage occurs during this change.
Enable Enhanced Monitoring	**Yes** to enable gathering metrics in real time for the operating system that your DB instance runs on. For more information, see Enhanced Monitoring (p. 258).	–	–
License Model	**license-included** to use the general license agreement for Microsoft SQL Server. **bring-your-own-license** to use your existing license. To use the Bring Your Own License model, you must provide your Microsoft License Mobility Agreement information in the **External Licenses** section of the Amazon RDS console. For more information, see Providing External License Information (p. 736).	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	An outage occurs during this change.

Setting	Setting Description	When the Change Occurs	Downtime Notes
Maintenance Window	The time range during which system maintenance occurs. System maintenance includes upgrades, if applicable. The maintenance window is a start time in Universal Coordinated Time (UTC), and a duration in hours. If you set the window to the current time, there must be at least 30 minutes between the current time and end of the window to ensure any pending changes are applied. For more information, see The Amazon RDS Maintenance Window (p. 103).	The change occurs immediately. This setting ignores the **Apply Immediately** setting.	If there are one or more pending actions that cause an outage, and the maintenance window is changed to include the current time, then those pending actions are applied immediately, and an outage occurs.
Multi-AZ Deployment	**Yes** to have a standby mirror of your DB instance created in another Availability Zone for failover support. We recommend Multi-AZ for production workloads to maintain high availability. **No** for development and testing. If your DB instance is running SQL Server 2014, 2016, or 2017 Enterprise Edition, and has in-memory optimization enabled, you can't add Multi-AZ. To add Multi-AZ, first disable in-memory optimization. For more information, see Multi-AZ Deployments for Microsoft SQL Server with Database Mirroring (p. 787).	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	–
New Master Password	The password for your master user. The password must contain from 8 to 128 printable ASCII characters (excluding /,", a space, and @). By resetting the master password, you also reset permissions for the DB instance. For more information, see Resetting the DB Instance Owner Role Password (p. 1229).	The change is applied asynchronously, as soon as possible. This setting ignores the **Apply Immediately** setting.	–

Setting	Setting Description	When the Change Occurs	Downtime Notes
Option Group	The option group that you want associated with the DB instance. For more information, see Working with Option Groups (p. 153).	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	–
Publicly Accessible	**Yes** to give the DB instance a public IP address, meaning that it is accessible outside the VPC. To be publicly accessible, the DB instance also has to be in a public subnet in the VPC. **No** to make the DB instance accessible only from inside the VPC. For more information, see Hiding a DB Instance in a VPC from the Internet (p. 401).	The change occurs immediately. This setting ignores the **Apply Immediately** setting.	–
Security Group	The security group you want associated with the DB instance. For more information, see Working with DB Security Groups (EC2-Classic Platform) (p. 380).	The change is applied asynchronously, as soon as possible. This setting ignores the **Apply Immediately** setting.	–
Storage Type	The storage type that you want to use. You can't change from or to magnetic storage. For more information, see Amazon RDS Storage Types (p. 410). **Warning** Once Amazon RDS begins to modify your DB instance to change the storage size or type, you can't submit another request to change the storage size or type for 6 hours.	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	The following changes all result in a brief outage while the process starts. After that, you can use your database normally while the change takes place. • From **General Purpose (SSD)** to **Provisioned IOPS (SSD)**. • From **Provisioned IOPS (SSD)** to **General Purpose (SSD)**.

Setting	Setting Description	When the Change Occurs	Downtime Notes
Subnet Group	The subnet group for the DB instance. You can use this setting to move your DB instance to a different VPC. If your DB instance is not in a VPC, you can use this setting to move your DB instance into a VPC. For more information, see Moving a DB Instance Not in a VPC into a VPC (p. 405).	–	–

Related Topics

- Rebooting a DB Instance (p. 119)
- Connecting to a DB Instance Running the Microsoft SQL Server Database Engine (p. 749)
- Upgrading the Microsoft SQL Server DB Engine (p. 764)
- Deleting a DB Instance (p. 126)

Upgrading the Microsoft SQL Server DB Engine

When Amazon RDS supports a new version of Microsoft SQL Server, you can upgrade your DB instances to the new version. Amazon RDS supports the following upgrades to a Microsoft SQL Server DB instance:

- Major Version Upgrades
- Minor Version Upgrades

You must perform all upgrades manually, and an outage occurs while the upgrade takes place. The time for the outage varies based on your engine version and the size of your DB instance.

For information about what SQL Server versions are available on Amazon RDS, see Microsoft SQL Server on Amazon RDS (p. 720).

Overview of Upgrading

Amazon RDS takes two DB snapshots during the upgrade process. The first DB snapshot is of the DB instance before any upgrade changes have been made. If the upgrade doesn't work for your databases, you can restore this snapshot to create a DB instance running the old version. The second DB snapshot is taken after the upgrade completes.

> **Note**
> Amazon RDS only takes DB snapshots if you have set the backup retention period for your DB instance to a number greater than 0. To change your backup retention period, see Modifying a DB Instance Running the Microsoft SQL Server Database Engine (p. 756).

After an upgrade is complete, you can't revert to the previous version of the database engine. If you want to return to the previous version, restore the DB snapshot that was taken before the upgrade to create a new DB instance.

During a minor or major version upgrade of SQL Server, the **Free Storage Space** and **Disk Queue Depth** metrics will display `-1`. After the upgrade is complete, both metrics will return to normal.

Major Version Upgrades

Amazon RDS currently supports the following major version upgrades to a Microsoft SQL Server DB instance.

> **Note**
> Currently, you can't upgrade your existing DB instance to SQL Server 2017.

Current Version	Supported Upgrade Versions
SQL Server 2014	SQL Server 2016
SQL Server 2012	SQL Server 2016 SQL Server 2014
SQL Server 2008 R2	SQL Server 2016 SQL Server 2014 SQL Server 2012

Database Compatibility Level

You can use Microsoft SQL Server database compatibility levels to adjust some database behaviors to mimic previous versions of SQL Server. For more information, see Compatibility Level in the Microsoft documentation.

The following are the compatibility levels of the SQL Server versions:

- SQL Server 2016 – compatibility level 130
- SQL Server 2014 – compatibility level 120
- SQL Server 2012 – compatibility level 110

When you upgrade your DB instance, all existing databases remain at their original compatibility level. For example, if you upgrade from SQL Server 2012 to SQL Server 2014, all existing databases have a compatibility level of 110. Any new database created after the upgrade have compatibility level 120.

You can change the compatibility level of a database by using the ALTER DATABASE command. For example, to change a database named `customeracct` to be compatible with SQL Server 2014, issue the following command:

```
ALTER DATABASE customeracct SET COMPATIBILITY_LEVEL = 120
```

Multi-AZ and In-Memory Optimization Considerations

Amazon RDS supports Multi-AZ deployments for DB instances running Microsoft SQL Server by using SQL Server Database Mirroring. For more information, see Multi-AZ Deployments for Microsoft SQL Server with Database Mirroring (p. 787).

If your DB instance is in a Multi-AZ deployment, both the primary and standby instances are upgraded. The primary and standby instances are upgraded at the same time, and you experience an outage until the upgrade is complete.

SQL Server 2014 Enterprise Edition and SQL Server 2016 Enterprise Edition support in-memory optimization. Multi-AZ deployments are not supported on DB instances that have in-memory optimization enabled. When you upgrade your DB instance with Multi-AZ enabled, Amazon RDS automatically disables in-memory optimization.

Option and Parameter Group Considerations

Option Group Considerations

If your DB instance uses a custom option group, in some cases Amazon RDS can't automatically assign your DB instance a new option group. For example, when you upgrade to a new major version. In that case, you must specify a new option group when you upgrade. We recommend that you create a new option group, and add the same options to it as your existing custom option group.

For more information, see Creating an Option Group (p. 154) or Making a Copy of an Option Group (p. 156).

Parameter Group Considerations

If your DB instance uses a custom parameter group, in some cases Amazon RDS can't automatically assign your DB instance a new parameter group. For example, when you upgrade to a new major version.

In that case, you must specify a new parameter group when you upgrade. We recommend that you create a new parameter group, and configure the parameters as in your existing custom parameter group.

For more information, see Creating a DB Parameter Group (p. 171) or Copying a DB Parameter Group (p. 175).

Testing an Upgrade

Before you perform a major version upgrade on your DB instance, you should thoroughly test your database, and all applications that access the database, for compatibility with the new version. We recommend that you use the following procedure.

To test a major version upgrade

1. Review the upgrade documentation for the new version of the database engine to see if there are compatibility issues that might affect your database or applications:
   - Upgrade to SQL Server 2016
   - Upgrade to SQL Server 2014
   - Upgrade to SQL Server 2012
2. If your DB instance uses a custom option group, create a new option group compatible with the new version you are upgrading to. For more information, see Option Group Considerations (p. 765).
3. If your DB instance uses a custom parameter group, create a new parameter group compatible with the new version you are upgrading to. For more information, see Parameter Group Considerations (p. 765).
4. Create a DB snapshot of the DB instance to be upgraded. For more information, see Creating a DB Snapshot (p. 207).
5. Restore the DB snapshot to create a new test DB instance. For more information, see Restoring from a DB Snapshot (p. 209).
6. Modify this new test DB instance to upgrade it to the new version, by using one of the following methods:
   - AWS Management Console (p. 766)
   - CLI (p. 767)
   - API (p. 767)
7. Evaluate the storage used by the upgraded instance to determine if the upgrade requires additional storage.
8. Run as many of your quality assurance tests against the upgraded DB instance as needed to ensure that your database and application work correctly with the new version. Implement any new tests needed to evaluate the impact of any compatibility issues you identified in step 1. Test all stored procedures and functions. Direct test versions of your applications to the upgraded DB instance.
9. If all tests pass, then perform the upgrade on your production DB instance. We recommend that you do not allow write operations to the DB instance until you confirm that everything is working correctly.

AWS Management Console

To upgrade a Microsoft SQL Server DB instance by using the AWS Management Console, you follow the same procedure as when you modify the DB instance. For detailed instructions, see Modifying a DB Instance Running the Microsoft SQL Server Database Engine (p. 756).

CLI

To upgrade a Microsoft SQL Server DB instance by using the AWS CLI, call the modify-db-instance command with the following parameters:

- `--db-instance-identifier` – the name of the db instance.
- `--engine-version` – the version number of the database engine to upgrade to.
- `--allow-major-version-upgrade` – to upgrade major version.
- `--no-apply-immediately` – apply changes during the next maintenance window. To apply changes immediately, use `--apply-immediately`. For more information, see The Impact of Apply Immediately (p. 114).

You might also need to include the following parameters. For more information, see Option Group Considerations (p. 765) and Parameter Group Considerations (p. 765).

- `--option-group-name` – the option group for the upgraded db instance.
- `--db-parameter-group-name` – the parameter group for the upgraded db instance.

Example

The following code upgrades a DB instance. These changes are applied during the next maintenance window.

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier <mydbinstance> \
    --engine-version <11.00.6020.0.v1> \
    --option-group-name <default:sqlserver-se-11-00> \
    --db-parameter-group-name <default.sqlserver-se-11.0> \
    --allow-major-version-upgrade \
    --no-apply-immediately
```

For Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier <mydbinstance> ^
    --engine-version <11.00.6020.0.v1> ^
    --option-group-name <default:sqlserver-se-11-00> ^
    --db-parameter-group-name <default.sqlserver-se-11.0> ^
    --allow-major-version-upgrade ^
    --no-apply-immediately
```

API

To upgrade a Microsoft SQL Server DB instance by using the Amazon RDS API, call the ModifyDBInstance action with the following parameters:

- `DBInstanceIdentifier` – the name of the db instance.
- `EngineVersion` – the version number of the database engine to upgrade to.
- `AllowMajorVersionUpgrade` – set to `true` to upgrade major version.
- `ApplyImmediately` – whether to apply changes immediately or during the next maintenance window. To apply changes immediately, set the value to `true`. To apply changes during the next

maintenance window, set the value to `false`. For more information, see The Impact of Apply Immediately (p. 114).

You might also need to include the following parameters. For more information, see Option Group Considerations (p. 765) and Parameter Group Considerations (p. 765).

- `OptionGroupName` – the option group for the upgraded db instance.
- `DBParameterGroupName` – the parameter group for the upgraded db instance.

Example

The following code upgrades a DB instance. These changes are applied during the next maintenance window.

```
https://rds.amazonaws.com/
    ?Action=ModifyDBInstance
    &AllowMajorVersionUpgrade=true
    &ApplyImmediately=false
    &DBInstanceIdentifier=mydbinstance
    &DBParameterGroupName=default.sqlserver-se-11.0
    &EngineVersion=11.00.6020.0.v1
    &OptionGroupName=default:sqlserver-se-11-00
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-10-31
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20131016/us-west-1/rds/aws4_request
    &X-Amz-Date=20131016T233051Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=087a8eb41cb1ab5f99e81575f23e73757ffc6a1e42d7d2b30b9cc0be988cff97
```

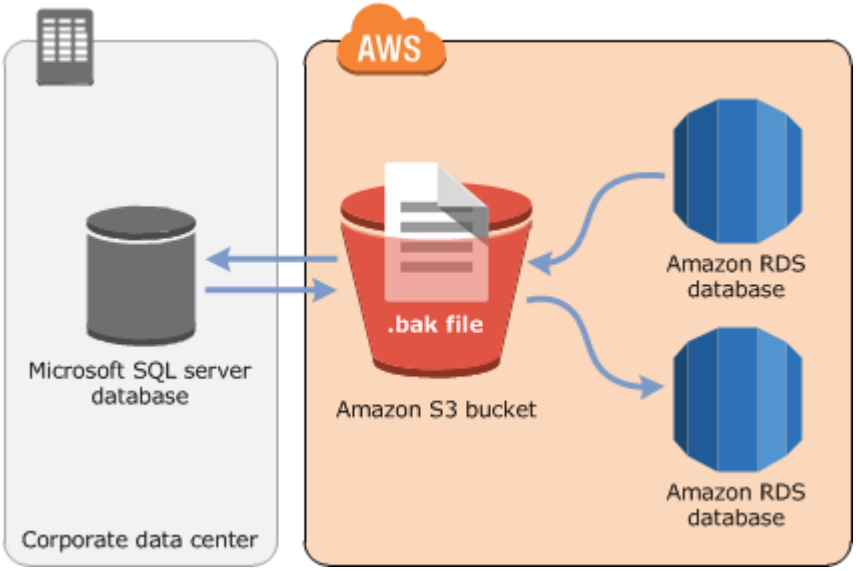
Related Topics

- Amazon RDS Maintenance (p. 102)
- Updating the Operating System for a DB Instance or DB Cluster (p. 108)
- Modifying a DB Instance Running the Microsoft SQL Server Database Engine (p. 756)

Importing and Exporting SQL Server Databases

Amazon Relational Database Service (Amazon RDS) supports native backup and restore for Microsoft SQL Server databases using full backup files (.bak files). You can import and export SQL Server databases in a single, easily portable file. You can create a full backup of your on-premises database, store it on Amazon Simple Storage Service (Amazon S3), and then restore the backup file onto an existing Amazon RDS DB instance running SQL Server. You can back up an Amazon RDS SQL Server database, store it on Amazon S3, and then restore the backup file onto an on-premises server, or a different Amazon RDS DB instance running SQL Server.

The following diagram shows the supported scenarios.



Using .bak files to back up and restore databases is heavily optimized, and is usually the fastest way to backup and restore databases. There are many additional advantages to using native backup and restore. You can do the following:

- Migrate databases to Amazon RDS.
- Move databases between Amazon RDS SQL Server DB instances.
- Import and export data.
- Migrate schemas, stored procedures, triggers and other database code.
- Backup and restore single databases, instead of entire DB instances.
- Create copies of databases for testing, training, and demonstrations.
- Store and transfer backup files into and out of Amazon RDS through Amazon S3, giving you an added layer of protection for disaster recovery.

Native backup and restore is available in all AWS Regions, and for both Single-AZ and Multi-AZ DB instances. Native backup and restore is available for all editions of Microsoft SQL Server supported on Amazon RDS, and for both the License Included and the Bring Your Own License models.

The following are some limitations to using native backup and restore:

- You can't back up to, or restore from, an Amazon S3 bucket in a different AWS Region than your Amazon RDS DB instance.

- We strongly recommend that you don't restore a backup file from one time zone to a different time zone. If you restore a backup file from one time zone to a different time zone, you must audit your queries and applications for the effects of the time zone change.
- You can't restore a backup file to the same DB instance that was used to create the backup file. Instead, restore the backup file to a new DB instance. Renaming the database is not a workaround for this limitation.
- You can't restore the same backup file to a DB instance multiple times. That is, you can't restore a backup file to a DB instance that already contains the database that you are restoring. Renaming the database is not a workaround for this limitation.
- You can't back up databases larger than 1 TB in size.
- You can't restore databases larger than 4 TB in size.
- You can't back up a database during the maintenance window, or any time Amazon RDS is in the process of taking a snapshot of the database.
- When you restore a backup file to a Multi-AZ DB instance, mirroring is terminated and then reestablished. Mirroring is terminated and reestablished for all databases on the DB instance, not just the one you are restoring. While RDS reestablishes mirroring, your DB instance can't failover. It can take 30 minutes or more to reestablish mirroring, depending on the size of the restore. For more information, see Multi-AZ Deployments for Microsoft SQL Server with Database Mirroring (p. 787).

We recommend that you use native backup and restore to migrate your database to Amazon RDS if your database can be offline while the backup file is created, copied, and restored. If your on-premises database can't be offline, we recommend that you use the AWS Database Migration Service to migrate your database to Amazon RDS. For more information, see What Is AWS Database Migration Service?

Native backup and restore is not intended to replace the data recovery capabilities of the cross-region snapshot copy feature. We recommend that you use snapshot copy to copy your database snapshot to another region for cross-region disaster recovery in Amazon RDS. For more information, see Copying a DB Snapshot or DB Cluster Snapshot (p. 213).

Setting Up for Native Backup and Restore

There are three components you'll need to set up for native backup and restore:

- An Amazon S3 bucket to store your backup files.
- An AWS Identity and Access Management (IAM) role to access the bucket.
- The `SQLSERVER_BACKUP_RESTORE` option added to an option group on your DB instance.

If you already have an Amazon S3 bucket, you can use that. If you don't have an Amazon S3 bucket, you can create a new one manually. Alternatively, you can choose to have a new bucket created for you when you add the `SQLSERVER_BACKUP_RESTORE` option by using the AWS Management Console. If you want to create a new bucket manually, see Creating a Bucket.

If you already have an IAM role, you can use that. If you don't have an IAM role, you can create a new one manually. Alternatively, you can choose to have a new IAM role created for you when you add the `SQLSERVER_BACKUP_RESTORE` option by using the AWS Management Console. If you want to create a new IAM role manually, or attach trust and permissions policies to an existing IAM role, take the approach discussed in the next section.

To enable native backup and restore on your DB instance, you add the `SQLSERVER_BACKUP_RESTORE` option to an option group on your DB instance. For more information and instructions, see Microsoft SQL Server Native Backup and Restore Support (p. 795).

Manually Creating an IAM Role for Native Backup and Restore

If you want to manually create a new IAM role to use with native backup and restore, you create a role to delegate permissions from the Amazon RDS service to your Amazon S3 bucket. When you create an IAM role, you attach trust and permissions policies. For the native backup and restore feature, use trust and permissions policies similar to the examples following. For more information about creating the role, see Creating a Role to Delegate Permissions to an AWS Service.

The trust and permissions policies require that you provide an Amazon Resource Name (ARN). For more information about ARN formatting, see Amazon Resource Names (ARNs) and AWS Service Namespaces.

Example Trust Policy for Native Backup and Restore

```
{
    "Version": "2012-10-17",
    "Statement":
    [{
        "Effect": "Allow",
        "Principal": {"Service":  "rds.amazonaws.com"},
        "Action": "sts:AssumeRole"
    }]
}
```

Example Permissions Policy for Native Backup and Restore Without Encryption Support

```
{
    "Version": "2012-10-17",
    "Statement":
    [
        {
        "Effect": "Allow",
        "Action":
            [
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
        "Resource": "arn:aws:s3:::bucket_name"
        },
        {
        "Effect": "Allow",
        "Action":
            [
                "s3:GetObjectMetaData",
                "s3:GetObject",
                "s3:PutObject",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload"
            ],
        "Resource": "arn:aws:s3:::bucket_name/*"
        }
    ]
}
```

Example Permissions Policy for Native Backup and Restore with Encryption Support

If you want to encrypt your backup files, include an encryption key in your permissions policy. For more information about encryption keys, see Getting Started in the AWS Key Management Service (AWS KMS) documentation.

```
{
```

```
    "Version": "2012-10-17",
    "Statement":
    [
        {
        "Effect": "Allow",
        "Action":
            [
                "kms:DescribeKey",
                "kms:GenerateDataKey",
                "kms:Encrypt",
                "kms:Decrypt"
            ],
        "Resource": "arn:aws:kms:region:account-id:key/key-id"
        },
        {
        "Effect": "Allow",
        "Action":
            [
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
        "Resource": "arn:aws:s3:::bucket_name"
        },
        {
        "Effect": "Allow",
        "Action":
            [
                "s3:GetObjectMetaData",
                "s3:GetObject",
                "s3:PutObject",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload"
            ],
        "Resource": "arn:aws:s3:::bucket_name/*"
        }
    ]
}
```

Using Native Backup and Restore

After you have enabled and configured native backup and restore, you can start using it. First you connect to your Microsoft SQL Server database, and then call an Amazon RDS stored procedure to do the work. For instructions on connecting to your database, see Connecting to a DB Instance Running the Microsoft SQL Server Database Engine (p. 749).

Some of the stored procedures require that you provide an Amazon Resource Name (ARN) to your Amazon S3 bucket and file. The format for your ARN is `arn:aws:s3:::bucket_name/file_name`. Amazon S3 doesn't require an account number or region in ARNs. If you also provide an optional AWS KMS encryption key, the format your ARN is `arn:aws:kms:region:account-id:key/key-id`. For more information, see Amazon Resource Names (ARNs) and AWS Service Namespaces.

There are stored procedures for backing up your database, restoring your database, canceling tasks that are in progress, and tracking the status of the backup and restore tasks. For instructions on how to call each stored procedure, see the following subsections:

- Backing Up a Database (p. 773)
- Restoring a Database (p. 773)
- Canceling a Task (p. 774)
- Tracking the Status of Tasks (p. 774)

Backing Up a Database

To back up your database, you call the `rds_backup_database` stored procedure.

> **Note**
> You can't back up a database during the maintenance window, or any time Amazon RDS is in the process of taking a snapshot of the database.

The following parameters are required:

- **`@source_db_name`** – The name of the database to create a backup of.
- **`@s3_arn_to_backup_to`** – The Amazon S3 bucket to save the backup file in, and the name of the file. The file can have the extension `.bak`, or any extension you want.

The following parameters are optional:

- **`@kms_master_key_arn`** – If you want to encrypt the backup file, the key to use to encrypt the file. For more information about encryption keys, see Getting Started in the AWS Key Management Service (AWS KMS) documentation.
- **`@overwrite_S3_backup_file`** – Whether or not to overwrite the backup file if it already exists in the Amazon S3 bucket. Specify `1` to overwrite the existing file. This overwrites any file in the bucket with the specified name, whether it is a backup file or another type of file. Specify `0` to not overwrite the existing file, and return an error instead if the file already exists. The default is `0`.

Example Without Encryption

```
exec msdb.dbo.rds_backup_database
        @source_db_name='database_name',
        @s3_arn_to_backup_to='arn:aws:s3:::bucket_name/file_name_and_extension',
        @overwrite_S3_backup_file=1;
```

Example With Encryption

```
exec msdb.dbo.rds_backup_database
        @source_db_name='database_name',
        @s3_arn_to_backup_to='arn:aws:s3:::bucket_name/file_name_and_extension',
        @kms_master_key_arn='arn:aws:kms:region:account-id:key/key-id',
        @overwrite_S3_backup_file=1;
```

Restoring a Database

To restore your database, you call the `rds_restore_database` stored procedure.

The following parameters are required:

- **`@restore_db_name`** – The name of the database to restore.
- **`@s3_arn_to_restore_from`** – The Amazon S3 bucket that contains the backup file, and the name of the file.

The following parameters are optional:

- **`@kms_master_key_arn`** – If you encrypted the backup file, the key to use to decrypt the file.

Example Without Encryption

```
exec msdb.dbo.rds_restore_database
        @restore_db_name='database_name',
        @s3_arn_to_restore_from='arn:aws:s3:::bucket_name/file_name_and_extension';
```

Example With Encryption

```
exec msdb.dbo.rds_restore_database
        @restore_db_name='database_name',
        @s3_arn_to_restore_from='arn:aws:s3:::bucket_name/file_name_and_extension',
        @kms_master_key_arn='arn:aws:kms:region:account-id:key/key-id';
```

Canceling a Task

To cancel a backup or restore task, you call the `rds_cancel_task` stored procedure.

The following parameters are optional:

- **`@db_name`** – The name of the database to cancel the task for.
- **`@task_id`** – The ID of the task to cancel. You can get the task ID by calling `rds_task_status`.

Example

```
exec msdb.dbo.rds_cancel_task @task_id=1234;
```

Tracking the Status of Tasks

To track the status of your backup and restore tasks, you call the `rds_task_status` stored procedure. If you don't provide any parameters, the stored procedure returns the status of all tasks. The status for tasks is updated approximately every 2 minutes.

The following parameters are optional:

- **`@db_name`** – The name of the database to show the task status for.
- **`@task_id`** – The ID of the task to show the task status for.

Example

```
exec msdb.dbo.rds_task_status @db_name='database_name';
```

The `rds_task_status` stored procedure returns the following columns.

Column	Description
`task_id`	The ID of the task.
`task_type`	Either `BACKUP_DB` for a back up task, or `RESTORE_DB` for a restore task.
`database_name`	The name of the database that the task is associated with.

Column	Description
`% complete`	The progress of the task as a percentage.
`duration (mins)`	The amount of time spent on the task, in minutes.
`lifecycle`	The status of the task. The possible statuses for a task are the following: • `CREATED` – As soon as you call `rds_backup_database` or `rds_restore_database`, a task is created and the status is set to `CREATED`. • `IN_PROGRESS` – After a backup or restore task starts, the status is set to `IN_PROGRESS`. It can take up to 5 minutes for the status to change from `CREATED` to `IN_PROGRESS`. • `SUCCESS` – After a backup or restore task completes, the status is set to `SUCCESS`. • `ERROR` – If a backup or restore task fails, the status is set to `ERROR`. Read the `task_info` column for more information about the error. • `CANCEL_REQUESTED` – As soon as you call `rds_cancel_task`, the status of the task is set to `CANCEL_REQUESTED`. • `CANCELLED` – After a task is successfully canceled, the status of the task is set to `CANCELLED`.
`task_info`	Additional information about the task. If an error occurs while backing up or restoring a database, this column contains information about the error. For a list of possible errors, and mitigation strategies, see Troubleshooting (p. 776).
`last_updated`	The date and time that the task status was last updated. The status is updated after every 5% of progress.
`created_at`	The date and time that the task was created.
`overwrite_S3_backup_file`	The value of the `@overwrite_S3_backup_file` parameter specified when calling a backup task. For more information, see Backing Up a Database (p. 773).

Compressing Backup Files

To save space in your Amazon S3 bucket, you can compress your backup files. For more information about compressing backup files, see Backup Compression in the Microsoft documentation.

Compressing your backup files is supported for the following database editions:

- Microsoft SQL Server Enterprise Edition
- Microsoft SQL Server Standard Edition

To turn on compression for your backup files, run the following code:

```
exec rdsadmin..rds_set_configuration 'S3 backup compression', 'true';
```

To turn off compression for your backup files, run the following code:

```
exec rdsadmin..rds_set_configuration 'S3 backup compression', 'false';
```

Migrating to Amazon RDS by Using Native Backup and Restore

To migrate your database from your corporate data center to Amazon RDS, you follow the procedures in this topic. However, you can perform the following steps to prepare:

1. Create an Amazon S3 bucket. For more information, see Creating a Bucket.
2. Upload your database backup file to your Amazon S3 bucket. For more information, see Uploading Objects into Amazon S3.

Troubleshooting

The following are issues you might encounter when you use native backup and restore.

Issue	Troubleshooting Suggestions
`Access Denied`	Verify that you have provided a correct bucket, in the correct format. The ARN must include the file name. For more information, see Using Native Backup and Restore (p. 772).
`BACKUP DATABASE WITH COMPRESSION is not supported on <edition_name> Edition`	Compressing your backup files is only supported for Microsoft SQL Server Enterprise Edition and Standard Edition. For more information, see Compressing Backup Files (p. 775).
`Database <database_name> cannot be restored because there is already an existing database with the same family_guid on the instance`	You can't restore a backup file to the same DB instance that was used to create the backup file. Instead, restore the backup file to a new DB instance. You also can't restore the same backup file to a DB instance multiple times. That is, you can't restore a backup file to a DB instance that already contains the database that you are restoring. Instead, restore the backup file to a new DB instance.
`Key <ARN> does not exist`	You attempted to restore an encrypted backup, but didn't provide a valid encryption key. Check your encryption key and retry. For more information, see Restoring a Database (p. 773).
`Please reissue task with correct type and overwrite property`	If you attempt to back up your database and provide the name of a file that already exists, but set the overwrite property to false, the save operation fails. To fix this error, either provide the name of a file that doesn't already exist, or set the overwrite property to true. For more information, see Backing Up a Database (p. 773). It's also possible that you intended to restore your database, but called the `rds_backup_database` stored procedure accidentally. In that case, call the `rds_restore_database` stored procedure instead. For more information, see Restoring a Database (p. 773). If you intended to restore your database and called the `rds_restore_database` stored procedure, make sure that you provided the name of a valid backup file.

Issue	Troubleshooting Suggestions
	For more information, see Using Native Backup and Restore (p. 772).
`Please specify a bucket that is in the same region as RDS instance`	You can't back up to, or restore from, an Amazon S3 bucket in a different AWS Region than your Amazon RDS DB instance. You can use Amazon S3 replication to copy the backup file to the correct region. For more information, see Cross-Region Replication in the Amazon S3 documentation.
`The specified bucket does not exist`	Verify that you have provided the correct ARN for your bucket and file, in the correct format. For more information, see Using Native Backup and Restore (p. 772).
`User <ARN> is not authorized to perform <kms action> on resource <ARN>`	You requested an encrypted operation, but didn't provide correct AWS KMS permissions. Verify that you have the correct permissions, or add them. For more information, see Setting Up for Native Backup and Restore (p. 770).

Related Topics

- Importing and Exporting SQL Server Data Using Other Methods (p. 778)
- Backing Up and Restoring Amazon RDS DB Instances (p. 200)

Importing and Exporting SQL Server Data Using Other Methods

Following, you can find information about importing your Microsoft SQL Server data to Amazon RDS, and exporting your data from an Amazon RDS DB instance running SQL Server, by using snapshots.

If your scenario supports it, it is easier to move data in and out of Amazon RDS by using the native backup and restore functionality. For more information, see Importing and Exporting SQL Server Databases (p. 769).

Note
Amazon RDS for Microsoft SQL Server does not support importing data into the `msdb` database.

Importing Data into SQL Server on Amazon RDS by Using a Snapshot

To import data into a SQL Server DB instance by using a snapshot

1. Create a DB instance. For more information, see Creating a DB Instance Running the Microsoft SQL Server Database Engine (p. 738).
2. Stop applications from accessing the destination DB instance.

   If you prevent access to your DB instance while you are importing data, data transfer is faster. Additionally, you won't need to worry about conflicts while data is being loaded if other applications cannot write to the DB instance at the same time. If something goes wrong and you have to roll back to a prior database snapshot, the only changes that you lose are the imported data, which you can import again after you resolve the issue.

   For information about controlling access to your DB instance, see Working with DB Security Groups (EC2-Classic Platform) (p. 380).
3. Create a snapshot of the target database.

   If the target database is already populated with data, we recommend that you take a snapshot of the database before you import the data. If something goes wrong with the data import or you want to discard the changes, you can restore the database to its previous state by using the snapshot. For information about database snapshots, see Creating a DB Snapshot (p. 207).

   Note
   When you take a database snapshot, I/O operations to the database are suspended for about 10 seconds while the backup is in progress.
4. Disable automated backups on the target database.

   Disabling automated backups on the target DB instance will improve performance while you are importing your data because Amazon RDS doesn't log transactions when automatic backups are disabled. However, there are some things to consider. Because automated backups are required to perform a point-in-time recovery, you won't be able to restore the database to a specific point in time while you are importing data. Additionally, any automated backups that were created on the DB instance are erased. You can still use previous snapshots to recover the database, and any snapshots that you have taken will remain available. For information about automated backups, see Working With Backups (p. 201).
5. Disable foreign key constraints, if applicable.

   If you need to disable foreign key constraints, you can do so with the following script.

```
--Disable foreign keys on all tables
```

```
    DECLARE @table_name SYSNAME;
    DECLARE @cmd NVARCHAR(MAX);
    DECLARE table_cursor CURSOR FOR SELECT name FROM sys.tables;

    OPEN table_cursor;
    FETCH NEXT FROM table_cursor INTO @table_name;

    WHILE @@FETCH_STATUS = 0 BEGIN
      SELECT @cmd = 'ALTER TABLE '+QUOTENAME(@table_name)+' NOCHECK CONSTRAINT ALL';
      EXEC (@cmd);
      FETCH NEXT FROM table_cursor INTO @table_name;
    END

    CLOSE table_cursor;
    DEALLOCATE table_cursor;

    GO
```

6. Drop indexes, if applicable.
7. Disable triggers, if applicable.

   If you need to disable triggers, you can do so with the following script.

```
--Disable triggers on all tables
    DECLARE @enable BIT = 0;
    DECLARE @trigger SYSNAME;
    DECLARE @table SYSNAME;
    DECLARE @cmd NVARCHAR(MAX);
    DECLARE trigger_cursor CURSOR FOR SELECT trigger_object.name trigger_name,
     table_object.name table_name
    FROM sysobjects trigger_object
    JOIN sysobjects table_object ON trigger_object.parent_obj = table_object.id
    WHERE trigger_object.type = 'TR';

    OPEN trigger_cursor;
    FETCH NEXT FROM trigger_cursor INTO @trigger, @table;

    WHILE @@FETCH_STATUS = 0 BEGIN
      IF @enable = 1
         SET @cmd = 'ENABLE ';
      ELSE
         SET @cmd = 'DISABLE ';

      SET @cmd = @cmd + ' TRIGGER dbo.'+QUOTENAME(@trigger)+' ON
 dbo.'+QUOTENAME(@table)+' ';
      EXEC (@cmd);
      FETCH NEXT FROM trigger_cursor INTO @trigger, @table;
    END

    CLOSE trigger_cursor;
    DEALLOCATE trigger_cursor;

    GO
```

8. Query the source SQL Server instance for any logins that you want to import to the destination DB instance.

   SQL Server stores logins and passwords in the `master` database. Because Amazon RDS doesn't grant access to the `master` database, you cannot directly import logins and passwords into your destination DB instance. Instead, you must query the `master` database on the source SQL Server instance to generate a data definition language (DDL) file that includes all logins and passwords that you want to add to the destination DB instance, and also role memberships and permissions that you want to transfer.

For information about querying the `master` database, see How to Transfer the Logins and the Passwords Between Instances of SQL Server 2005 and SQL Server 2008 in the Microsoft Knowledge Base.

The output of the script is another script that you can run on the destination DB instance. The script in the Knowledge Base article has the following code:

```
p.type IN
```

Every place `p.type` appears, use the following code instead:

```
p.type = 'S'
```

9. Import the data using the method in Import the Data (p. 781).
10. Grant applications access to the target DB instance.

    When your data import is complete, you can grant access to the DB instance to those applications that you blocked during the import. For information about controlling access to your DB instance, see Working with DB Security Groups (EC2-Classic Platform) (p. 380).
11. Enable automated backups on the target DB instance.

    For information about automated backups, see Working With Backups (p. 201).
12. Enable foreign key constraints.

    If you disabled foreign key constraints earlier, you can now enable them with the following script.

```
--Enable foreign keys on all tables
    DECLARE @table_name SYSNAME;
    DECLARE @cmd NVARCHAR(MAX);
    DECLARE table_cursor CURSOR FOR SELECT name FROM sys.tables;

    OPEN table_cursor;
    FETCH NEXT FROM table_cursor INTO @table_name;

    WHILE @@FETCH_STATUS = 0 BEGIN
      SELECT @cmd = 'ALTER TABLE '+QUOTENAME(@table_name)+' CHECK CONSTRAINT ALL';
      EXEC (@cmd);
      FETCH NEXT FROM table_cursor INTO @table_name;
    END

    CLOSE table_cursor;
    DEALLOCATE table_cursor;
```

13. Enable indexes, if applicable.
14. Enable triggers, if applicable.

    If you disabled triggers earlier, you can now enable them with the following script.

```
--Enable triggers on all tables
    DECLARE @enable BIT = 1;
    DECLARE @trigger SYSNAME;
    DECLARE @table SYSNAME;
    DECLARE @cmd NVARCHAR(MAX);
    DECLARE trigger_cursor CURSOR FOR SELECT trigger_object.name trigger_name,
     table_object.name table_name
    FROM sysobjects trigger_object
    JOIN sysobjects table_object ON trigger_object.parent_obj = table_object.id
    WHERE trigger_object.type = 'TR';
```

```
   OPEN trigger_cursor;
   FETCH NEXT FROM trigger_cursor INTO @trigger, @table;

   WHILE @@FETCH_STATUS = 0 BEGIN
     IF @enable = 1
        SET @cmd = 'ENABLE ';
     ELSE
        SET @cmd = 'DISABLE ';

     SET @cmd = @cmd + ' TRIGGER dbo.'+QUOTENAME(@trigger)+' ON
dbo.'+QUOTENAME(@table)+' ';
     EXEC (@cmd);
     FETCH NEXT FROM trigger_cursor INTO @trigger, @table;
   END

   CLOSE trigger_cursor;
   DEALLOCATE trigger_cursor;
```

Import the Data

Microsoft SQL Server Management Studio is a graphical SQL Server client that is included in all Microsoft SQL Server editions except the Express Edition. SQL Server Management Studio Express is available from Microsoft as a free download. To find this download, see the Microsoft website.

Note
SQL Server Management Studio is available only as a Windows-based application.

SQL Server Management Studio includes the following tools, which are useful in importing data to a SQL Server DB instance:

- Generate and Publish Scripts Wizard
- Import and Export Wizard
- Bulk copy

Generate and Publish Scripts Wizard

The Generate and Publish Scripts Wizard creates a script that contains the schema of a database, the data itself, or both. If you generate a script for a database in your local SQL Server deployment, you can then run the script to transfer the information that it contains to an Amazon RDS DB instance.

Note
For databases of 1 GB or larger, it is more efficient to script only the database schema and then use the Import and Export Wizard or the bulk copy feature of SQL Server to transfer the data.

For detailed information about the Generate and Publish Scripts Wizard, see the Microsoft SQL Server documentation.

In the wizard, pay particular attention to the advanced options on the **Set Scripting Options** page to ensure that everything you want your script to include is selected. For example, by default, database triggers are not included in the script.

When the script is generated and saved, you can use SQL Server Management Studio to connect to your DB instance and then run the script.

Import and Export Wizard

The Import and Export Wizard creates a special Integration Services package, which you can use to copy data from your local SQL Server database to the destination DB instance. The wizard can filter which tables and even which tuples within a table are copied to the destination DB instance.

Note
The Import and Export Wizard works well for large datasets, but it might not be the fastest way to remotely export data from your local deployment. For an even faster way, consider the SQL Server bulk copy feature.

For detailed information about the Import and Export Wizard, see the Microsoft SQL Server documentation.

In the wizard, on the **Choose a Destination** page, do the following:

- For **Server Name**, type the name of the endpoint for your DB instance.
- For the server authentication mode, choose **Use SQL Server Authentication**.
- For **User name** and **Password**, type the credentials for the master user that you created for the DB instance.

Bulk Copy

The SQL Server bulk copy feature is an efficient means of copying data from a source database to your DB instance. Bulk copy writes the data that you specify to a data file, such as an ASCII file. You can then run bulk copy again to write the contents of the file to the destination DB instance.

This section uses the **bcp** utility, which is included with all editions of SQL Server. For detailed information about bulk import and export operations, see the Microsoft SQL Server documentation.

Note
Before you use bulk copy, you must first import your database schema to the destination DB instance. The Generate and Publish Scripts Wizard, described earlier in this topic, is an excellent tool for this purpose.

The following command connects to the local SQL Server instance to generate a tab-delimited file of a specified table in the C:\ root directory of your existing SQL Server deployment. The table is specified by its fully qualified name, and the text file has the same name as the table that is being copied.

```
bcp dbname.schema_name.table_name out C:\table_name.txt -n -S localhost -U username -
P password -b 10000
```

The preceding code includes the following options:

- `-n` specifies that the bulk copy will use the native data types of the data to be copied.
- `-S` specifies the SQL Server instance that the *bcp* utility will connect to.
- `-U` specifies the user name of the account that will log in to the SQL Server instance.
- `-P` specifies the password for the user specified by `-U`.
- `-b` specifies the number of rows per batch of imported data.

Note
There might be other parameters that are important to your import situation. For example, you might need the `-E` parameter that pertains to identity values. For more information; see the full description of the command line syntax for the **bcp** utility in the Microsoft SQL Server documentation.

For example, suppose a database named `store` that uses the default schema, `dbo`, contains a table named `customers`. The user account `admin`, with the password `insecure`, will copy 10,000 rows of the `customers` table to a file named `customers.txt`.

```
bcp store.dbo.customers out C:\customers.txt -n -S localhost -U admin -P insecure -b 10000
```

After you generate the data file, if you have created the database and schema on the target DB instance, you can upload the data to your DB instance by using a similar command. In this case, you will use the `in` argument to specify an input file instead of `out` to specify an output file. Instead of using localhost to specify the local SQL Server instance, you will specify the endpoint of your DB instance. If you use a port other than 1433, you will specify that, too. The user name and password are the master user and password for your DB instance. The syntax is as follows.

```
bcp dbname.schema_name.table_name in C:\table_name.txt -n -S endpoint,port -
U master_user_name -P master_user_password -b 10000
```

To continue the previous example, suppose the master user name is `admin`, and the password is `insecure`. The endpoint for the DB instance is `rds.ckz2kqd4qsn1.us-east-1.rds.amazonaws.com`, and you use port 4080. The command is as follows.

```
bcp store.dbo.customers in C:\customers.txt -n -S rds.ckz2kqd4qsn1.us-
east-1.rds.amazonaws.com,4080 -U admin -P insecure -b 10000
```

Exporting Data from SQL Server on Amazon RDS

You can choose one of the following options to export data from an Amazon RDS SQL DB instance :

- **Native database backup using a full backup file (.bak)** – Using .bak files to backup databases is heavily optimized, and is usually the fastest way to export data. For more information, see Importing and Exporting SQL Server Databases (p. 769).
- **SQL Server Import and Export Wizard** – For more information, see SQL Server Import and Export Wizard (p. 783).
- **SQL Server Generate and Publish Scripts Wizard and bcp utility** – For more information, see SQL Server Generate and Publish Scripts Wizard and bcp Utility (p. 784).

SQL Server Import and Export Wizard

You can use the SQL Server Import and Export Wizard to copy one or more tables, views, or queries from your Amazon RDS SQL DB instance to another data store. This choice is best if the target data store is not SQL Server. For more information, see SQL Server Import and Export Wizard in the SQL Server documentation.

The SQL Server Import and Export Wizard is available as part of Microsoft SQL Server Management Studio, a graphical SQL Server client that is included in all Microsoft SQL Server editions except the Express Edition. SQL Server Management Studio is available only as a Windows-based application. SQL Server Management Studio Express is available from Microsoft as a free download. To find this download, see the Microsoft website.

To use the SQL Server Import and Export Wizard to export data

1. In SQL Server Management Studio, connect to your Amazon RDS SQL DB instance. For details on how to do this, see Connecting to a DB Instance Running the Microsoft SQL Server Database Engine (p. 749).
2. In **Object Explorer**, expand **Databases**, open the context (right-click) menu for the source database, choose **Tasks**, and then choose **Export Data**. The wizard appears.
3. On the **Choose a Data Source** page, do the following:
    1. For **Data source**, choose **`SQL Server Native Client 11.0`**.
    2. Verify that the **Server name** box shows the endpoint of your Amazon RDS SQL DB instance.

3. Select **Use SQL Server Authentication**. For **User name** and **Password**, type the master user name and password of your Amazon RDS SQL DB.
4. Verify that the **Database** box shows the database from which you want to export data.
5. Choose **Next**.

4. On the **Choose a Destination** page, do the following:

   1. For **Destination**, choose **`SQL Server Native Client 11.0`**.

      Note
      Other target data sources are available, include .NET Framework data providers, OLE DB providers, SQL Server Native Client providers, ADO.NET providers, Microsoft Office Excel, Microsoft Office Access, and the Flat File source. If you choose to target one of these data sources, skip the remainder of step 4 and see Choose a Destination in the SQL Server documentation for details on the connection information to provide.
   2. For **Server name**, type the server name of the target SQL Server DB instance.
   3. Choose the appropriate authentication type. Type a user name and password if necessary.
   4. For **Database**, choose the name of the target database, or choose **New** to create a new database to contain the exported data.

      If you choose **New**, see Create Database in the SQL Server documentation for details on the database information to provide.
   5. Choose **Next**.
5. On the **Table Copy or Query** page, choose **Copy data from one or more tables or views** or **Write a query to specify the data to transfer**. Choose **Next**.
6. If you chose **Write a query to specify the data to transfer**, you see the **Provide a Source Query** page. Type or paste in a SQL query, and then choose **Parse** to verify it. Once the query validates, choose **Next**.
7. On the **Select Source Tables and Views** page, do the following:

   1. Select the tables and views that you want to export, or verify that the query you provided is selected.
   2. Choose **Edit Mappings** and specify database and column mapping information. For more information, see Column Mappings in the SQL Server documentation.
   3. (Optional) To see a preview of data to be exported, select the table, view, or query, and then choose **Preview**.
   4. Choose **Next**.
8. On the **Run Package** page, verify that **Run immediately** is selected. Choose **Next**.
9. On the **Complete the Wizard** page, verify that the data export details are as you expect. Choose **Finish**.
10. On the **The execution was successful** page, choose **Close**.

SQL Server Generate and Publish Scripts Wizard and bcp Utility

You can use the SQL Server Generate and Publish Scripts Wizard to create scripts for an entire database or just selected objects. You can run these scripts on a target SQL Server DB instance to recreate the scripted objects. You can then use the bcp utility to bulk export the data for the selected objects to the target DB instance. This choice is best if you want to move a whole database (including objects other than tables) or large quantities of data between two SQL Server DB instances. For a full description of the bcp command line syntax, see bcp Utility in the Microsoft SQL Server documentation.

The SQL Server Generate and Publish Scripts Wizard is available as part of Microsoft SQL Server Management Studio, a graphical SQL Server client that is included in all Microsoft SQL Server editions

except the Express Edition. SQL Server Management Studio is available only as a Windows-based application. SQL Server Management Studio Express is available from Microsoft as a free download.

To use the SQL Server Generate and Publish Scripts Wizard and the bcp utility to export data

1. In SQL Server Management Studio, connect to your Amazon RDS SQL DB instance. For details on how to do this, see Connecting to a DB Instance Running the Microsoft SQL Server Database Engine (p. 749).
2. In **Object Explorer**, expand the **Databases** node and select the database you want to script.
3. Follow the instructions in Generate and Publish Scripts Wizard in the SQL Server documentation to create a script file.
4. In SQL Server Management Studio, connect to your target SQL Server DB instance.
5. With the target SQL Server DB instance selected in **Object Explorer**, choose **Open** on the **File** menu, choose **File**, and then open the script file.
6. If you have scripted the entire database, review the CREATE DATABASE statement in the script to make sure the database is being created in the location and with the parameters that you want. For more information, see CREATE DATABASE in the SQL Server documentation.
7. If you are creating database users in the script, check to see if server logins exist on the target DB instance for those users. If not, create logins for those users; the scripted commands to create the database users will fail otherwise. For more information, see Create a Login in the SQL Server documentation.
8. Choose **!Execute** on the SQL Editor menu to execute the script file and create the database objects. When the script finishes, verify that all database objects exist as expected.
9. Use the bcp utility to export data from the Amazon RDS SQL DB instance into files. Open a command prompt and type the following command.

```
bcp database_name.schema_name.table_name out data_file -n -S aws_rds_sql_endpoint -U
 username -P password
```

   The preceding code includes the following options:

   - *table_name* is the name of one of the tables that you've recreated in the target database and now want to populate with data.
   - *data_file* is the full path and name of the data file to be created.
   - `-n` specifies that the bulk copy will use the native data types of the data to be copied.
   - `-S` specifies the SQL Server DB instance to export from.
   - `-U` specifies the user name to use when connecting to the SQL Server DB instance.
   - `-P` specifies the password for the user specified by `-U`.

   The following shows an example command.

```
bcp world.dbo.city out C:\Users\JohnDoe\city.dat -n -S sql-jdoe.1234abcd.us-
west-2.rds.amazonaws.com,1433 -U JohnDoe -P ClearTextPassword
```

   Repeat this step until you have data files for all of the tables you want to export.
10. Prepare your target DB instance for bulk import of data by following the instructions at Basic Guidelines for Bulk Importing Data in the SQL Server documentation.
11. Decide on a bulk import method to use after considering performance and other concerns discussed in About Bulk Import and Bulk Export Operations in the SQL Server documentation.
12. Bulk import the data from the data files you created using the bcp utility, following the instructions at either Import and Export Bulk Data by Using the bcp Utility or Import Bulk Data by Using BULK

INSERT or OPENROWSET(BULK...) in the SQL Server documentation, depending on what you decided in step 11.

Related Topics

- Importing and Exporting SQL Server Databases (p. 769)

Multi-AZ Deployments for Microsoft SQL Server with Database Mirroring

Amazon RDS supports Multi-AZ deployments for DB instances running Microsoft SQL Server by using SQL Server Database Mirroring. Multi-AZ deployments provide increased availability, data durability, and fault tolerance for DB instances. In the event of planned database maintenance or unplanned service disruption, Amazon RDS automatically fails over to the up-to-date standby so database operations can resume quickly without manual intervention. The primary and standby instances use the same endpoint, whose physical network address transitions to the mirror as part of the failover process. You don't have to reconfigure your application when a failover occurs.

Amazon RDS manages failover by actively monitoring your Multi-AZ deployment and initiating a failover when a problem with your primary occurs. Failover doesn't occur unless the standby and primary are fully in sync. Amazon RDS actively maintains your Multi-AZ deployment by automatically repairing unhealthy DB instances and reestablishing synchronous replication. You don't have to manage anything; Amazon RDS handles the primary, the Mirroring witness, and the standby instance for you. When you set up SQL Server Multi-AZ, all databases on the instance are mirrored automatically.

Amazon RDS supports Multi-AZ with Mirroring for the following SQL Server versions and editions:

- SQL Server 2017: Standard and Enterprise Editions
- SQL Server 2016: Standard and Enterprise Editions
- SQL Server 2014: Standard and Enterprise Editions
- SQL Server 2012: Standard and Enterprise Editions
- SQL Server 2008 R2: Standard and Enterprise Editions

Amazon RDS supports Multi-AZ with Mirroring for SQL Server in all AWS Regions, with the following exceptions:

- Not supported
  - US West (N. California)
  - Asia Pacific (Singapore)
  - AWS GovCloud (US)

- Supported in most cases
  - Asia Pacific (Sydney) – Supported for DB instances in VPCs.
  - Asia Pacific (Tokyo) – Supported for DB instances in VPCs.
  - South America (São Paulo) – Supported on all DB instance classes except m1 or m2.

Adding Multi-AZ with Mirroring to a Microsoft SQL Server DB Instance

When creating a new SQL Server DB instance using the AWS Management Console, you can simply select **Yes (Mirroring)** from the **Multi-AZ Deployment** list on the **Specify DB Details** page to add Multi-AZ with Mirroring. For more information, see Creating a DB Instance Running the Microsoft SQL Server Database Engine (p. 738).

When modifying an existing SQL Server DB instance using the AWS Management Console, you can simply select **Yes (Mirroring)** from the **Multi-AZ Deployment** list on the **Modify DB Instance** page to

add Multi-AZ with Mirroring. For more information, see Modifying a DB Instance Running the Microsoft SQL Server Database Engine (p. 756).

Microsoft SQL Server Multi-AZ Deployment Notes and Recommendations

The following are some restrictions when working with Multi-AZ deployments for Microsoft SQL Server DB instances:

- Cross-region Multi-AZ is not currently supported.
- You can't configure the standby to accept database read activity.
- Multi-AZ with Mirroring is not supported for DB instances with dedicated tenancy.
- Multi-AZ with Mirroring is not supported for DB instances with in-memory optimization enabled. For more information, see Unsupported SQL Server Features for In-Memory OLTP in the Microsoft documentation.
- You can't rename a database on a SQL Server DB instance that is in a SQL Server Multi-AZ with Mirroring deployment. If you need to rename a database on such an instance, first turn off Multi-AZ for the DB instance, then rename the database, and finally turn Multi-AZ back on for the DB instance.

The following are some notes about working with Multi-AZ deployments for Microsoft SQL Server DB instances:

- To use SQL Server Multi-AZ with Mirroring with a SQL Server DB instance in a VPC, you first create a DB subnet group that has subnets in at least two distinct Availability Zones. You then assign the DB subnet group to the SQL Server DB instance that is being mirrored.
- When a DB instance is modified to be a Multi-AZ deployment, during the modification it has a status of **modifying**. Amazon RDS creates the standby mirror, and makes a backup of the primary DB instance. Once the process is complete, the status of the primary DB instance becomes **available**.
- Multi-AZ deployments maintain all databases on the same node. If a database on the primary host fails over, all your SQL Server databases fail over as one atomic unit to your standby host. Amazon RDS provisions a new healthy host, and replace the unhealthy host.
- Multi-AZ with Mirroring supports one standby mirror.
- Users, logins, and permissions are automatically replicated for you on the standby mirror. You don’t need to recreate them. User-defined server roles (a SQL Server 2012 feature) are not replicated in Multi-AZ instances.
- If you have SQL Server Agent jobs, you need to recreate them in the secondary, as these jobs are stored in the msdb database, and this database can't be replicated via Mirroring. Create the jobs first in the original primary, then fail over, and create the same jobs in the new primary.
- You might observe elevated latencies compared to a standard DB instance deployment (in a single Availability Zone) as a result of the synchronous data replication performed on your behalf.
- Failover times are affected by the time it takes to complete the recovery process. Large transactions increase the failover time.
- When you restore a backup file to a Multi-AZ DB instance, mirroring is terminated and then reestablished. Mirroring is terminated and reestablished for all databases on the DB instance, not just the one you are restoring. While RDS reestablishes mirroring, your DB instance can't failover. It can take 30 minutes or more to reestablish mirroring, depending on the size of the restore. For more information, see Importing and Exporting SQL Server Databases (p. 769).

The following are some recommendations for working with Multi-AZ deployments for Microsoft SQL Server DB instances:

- For databases used in production or preproduction, we recommend Multi-AZ deployments for high availability, Provisioned IOPS for fast, consistent performance, and instance classes (m3.large and larger, m4.large and larger) that are optimized for Provisioned IOPS.
- You can't select the Availability Zone (AZ) for the standby instance, so when you deploy application hosts, take this into account. Your database could fail over to another AZ, and the application hosts might not be in the same AZ as the database. For this reason, it is a best practice to balance your application hosts across all AZs in the region.
- For best performance, do not enable mirroring during a large data load operation. If you want your data load to be as fast as possible, complete the loading before you convert your DB instance to a Multi-AZ deployment.
- Applications that access the SQL Server databases should have exception handling that catches connection errors. The following code sample shows a try/catch block that catches a communication error.

```
for (int iRetryCount = 0; (iRetryCount < RetryMaxAttempts && keepInserting); iRetryCount+
+)
{
   using (SqlConnection connection = new SqlConnection(DatabaseConnString))
   {
      using (SqlCommand command = connection.CreateCommand())
      {
         command.CommandText = "INSERT INTO SOME_TABLE VALUES ('SomeValue');";

         try
         {
            connection.Open();

            while (keepInserting)
            {
               command.ExecuteNonQuery();
               intervalCount++;
            }
                  connection.Close();
         }

         catch (Exception ex)
         {
                  Logger(ex.Message);
         }
      }
   }

   if (iRetryCount < RetryMaxAttempts && keepInserting)
   {
            Thread.Sleep(RetryIntervalPeriodInSeconds * 1000);
   }
}
```

- You should not use the `Set Partner Off` command when working with Multi-AZ instances. For example, *do not* do the following:

```
ALTER DATABASE db1 SET PARTNER off
```

- You should not set the recovery mode to `simple`. For example, *do not* do the following:

```
ALTER DATABASE db1 SET RECOVERY simple
```

- You should not use the `DEFAULT_DATABASE` parameter when creating new logins on Multi-AZ DB instances, as these settings can't be applied to the standby mirror. For example, *do not* do the following:

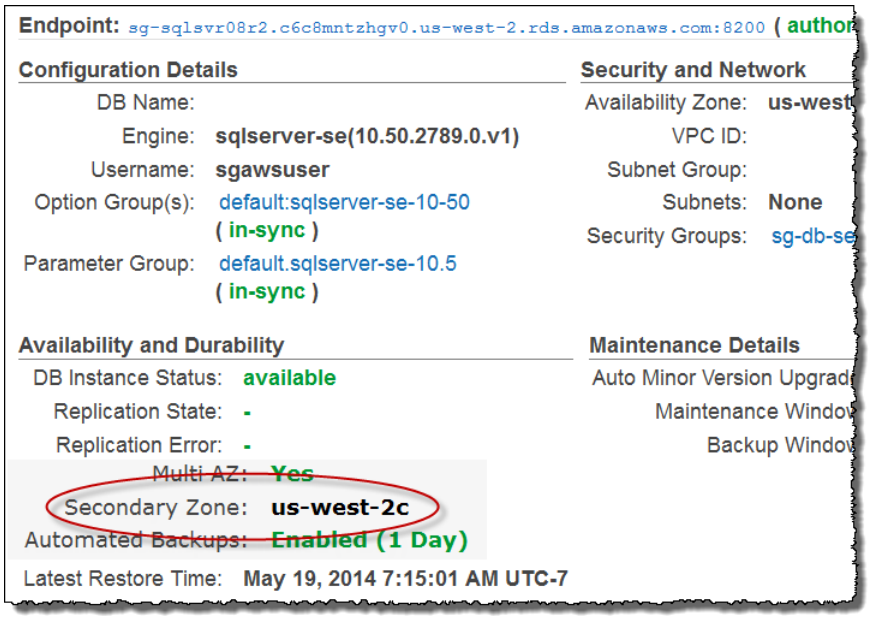
```
CREATE LOGIN [test_dba] WITH PASSWORD=foo, DEFAULT_DATABASE=[db2]
```

and *do not* do the following:

```
ALTER LOGIN [test_dba] SET DEFAULT_DATABASE=[db3]
```

Determining the Location of the Standby Mirror

You can determine the location of the standby mirror by using the AWS Management Console. You need to know the location of the standby mirror if you are setting up your primary DB instance in a VPC.



You can also view the Availability Zone of the standby mirror using the AWS CLI command `describe-db-instances` or RDS API action `DescribeDBInstances`. The output will show the secondary AZ where the standby mirror is located.

Related Topics

- Licensing Microsoft SQL Server Multi-AZ Deployments (p. 735)
- High Availability (Multi-AZ) (p. 99)

Using SSL with a Microsoft SQL Server DB Instance

You can use Secure Sockets Layer (SSL) to encrypt connections between your client applications and your Amazon RDS DB instances running Microsoft SQL Server. SSL support is available in all AWS regions for all supported SQL Server editions.

When you create a SQL Server DB instance, Amazon RDS creates an SSL certificate for it. The SSL certificate includes the DB instance endpoint as the Common Name (CN) for the SSL certificate to guard against spoofing attacks.

There are 2 ways to use SSL to connect to your SQL Server DB instance:

- Force SSL for all connections — this happens transparently to the client, and the client doesn't have to do any work to use SSL.
- Encrypt specific connections — this sets up an SSL connection from a specific client computer, and you must do work on the client to encrypt connections.

Forcing Connections to Your DB Instance to Use SSL

You can force all connections to your DB instance to use SSL. If you force connections to use SSL, it happens transparently to the client, and the client doesn't have to do any work to use SSL.

If you want to force SSL, use the `rds.force_ssl` parameter. By default, the `rds.force_ssl` parameter is set to `false`. Set the `rds.force_ssl` parameter to `true` to force connections to use SSL. The `rds.force_ssl` parameter is static, so after you change the value, you must reboot your DB instance for the change to take effect.

To force all connections to your DB instance to use SSL

1. Determine the parameter group that is attached to your DB instance.
   a. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
   b. In the top right corner of the Amazon RDS console, select the region of your DB instance.
   c. In the navigation pane, choose **DB Instances**, and then select your DB instance.
   d. Choose the **Details** tab. Find the **Parameter Group** field in the **Configuration Details** section.
2. If necessary, create a new parameter group. If your DB instance uses the default parameter group, you must create a new parameter group. If your DB instance uses a nondefault parameter group, you can choose to edit the existing parameter group or to create a new parameter group. If you edit an existing parameter group, the change affects all DB instances that use that parameter group.

   To create a new parameter group, follow the instructions in Creating a DB Parameter Group (p. 171).
3. Edit your new or existing parameter group to set the `rds.force_ssl` parameter to `true`. To edit the parameter group, follow the instructions in Modifying Parameters in a DB Parameter Group (p. 172).
4. If you created a new parameter group, modify your DB instance to attach the new parameter group. Modify the **DB Parameter Group** setting of the DB instance. For more information, see Modifying a DB Instance Running the Microsoft SQL Server Database Engine (p. 756).
5. Reboot your DB instance. For more information, see Rebooting a DB Instance (p. 119).

Encrypting Specific Connections

You can force all connections to your DB instance to use SSL, or you can encrypting connections from specific client computers only. To use SSL from a specific client, you must obtain certificates for the client computer, import certificates on the client computer, and then encrypt the connections from the client computer.

> **Note**
> All SQL Server instances created after August 5, 2014, use the DB instance endpoint in the Common Name (CN) field of the SSL certificate. Prior to August 5, 2014, SSL certificate verification was not available for VPC-based SQL Server instances. If you have a VPC-based SQL Server DB instance that was created before August 5, 2014, and you want to use SSL certificate verification and ensure that the instance endpoint is included as the CN for the SSL certificate for that DB instance, then rename the instance. When you rename a DB instance, a new certificate is deployed and the instance is rebooted to enable the new certificate.

Obtaining Certificates for Client Computers

To encrypt connections from a client computer to an Amazon RDS DB instance running Microsoft SQL Server, you need a certificate on your client computer.

To obtain that certificate, download the certificate to your client computer. You can download a root certificate that works for all regions from https://s3.amazonaws.com/rds-downloads/rds-ca-2015-root.pem. You can download a certificate bundle that contains both the old and new root certificates from https://s3.amazonaws.com/rds-downloads/rds-combined-ca-bundle.pem. For region-specific intermediate certificates, and more information, see Using SSL to Encrypt a Connection to a DB Instance (p. 358).

After you have downloaded the appropriate certificate, import the certificate into your Microsoft Windows operating system by following the procedure in the section following.

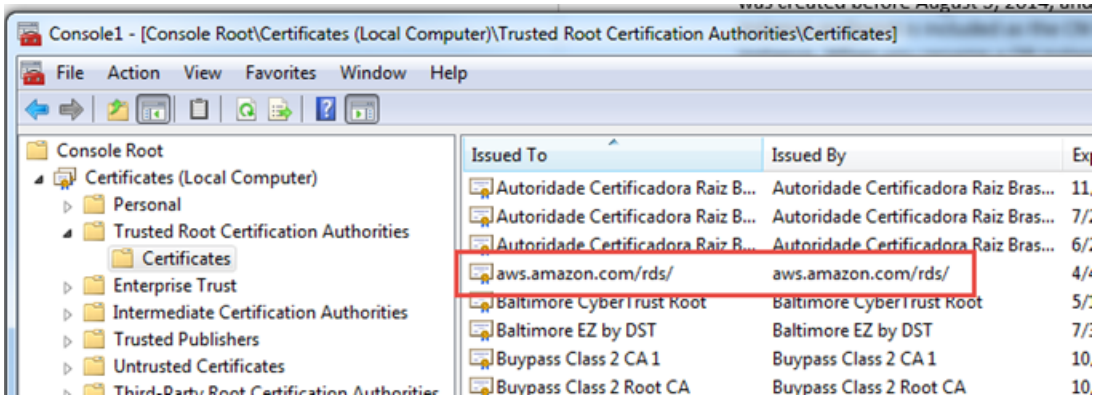
Importing Certificates on Client Computers

You can use the following procedure to import your certificate into the Microsoft Windows operating system on your client computer.

To import the certificate into your Windows operating system:

1. On the **Start** menu, type **`Run`** in the search box and press **Enter**.
2. In the **Open** box, type **`MMC`** and then choose **OK**.
3. In the MMC console, on the **File** menu, choose **Add/Remove Snap-in**.
4. In the **Add or Remove Snap-ins** dialog box, for **Available snap-ins**, select **`Certificates`**, and then choose **Add**.
5. In the MMC console, on the **File** menu, choose **Add/Remove Snap-in**.
6. In the **Certificates snap-in** dialog box, choose **Computer account**, and then choose **Next**.
7. In the **Select computer** dialog box, choose **Finish**.
8. In the **Add or Remove Snap-ins** dialog box, choose **OK**.
9. In the MMC console, expand **Certificates**, open the context (right-click) menu for **Trusted Root Certification Authorities**, choose **All Tasks**, and then choose **Import**.
10. On the first page of the Certificate Import Wizard, choose **Next**.
11. On the second page of the Certificate Import Wizard, choose **Browse**. In the browse window, change the file type to **All files (*.*)** because .pem is not a standard certificate extension. Locate the .pem file that you downloaded previously.
12. Choose **Open** to select the certificate file, and then choose **Next**.

13. On the third page of the Certificate Import Wizard, choose **Next**.
14. On the fourth page of the Certificate Import Wizard, choose **Finish**. A dialog box appears indicating that the import was successful.
15. In the MMC console, expand **Certificates**, expand **Trusted Root Certification Authorities**, and then choose **Certificates**. Locate the certificate to confirm it exists, as shown following.



16. Restart your computer.

Encrypting Connections to an Amazon RDS DB Instance Running Microsoft SQL Server

After you have imported a certificate into your client computer, you can encrypt connections from the client computer to an Amazon RDS DB instance running Microsoft SQL Server.

For SQL Server Management Studio, use the following procedure. For more information about SQL Server Management Studio, see Use SQL Server Management Studio.

To encrypt connections from SQL Server Management Studio

1. Launch SQL Server Management Studio.
2. For **Connect to server**, type the server information, login user name, and password.
3. Choose **Options**.
4. Select **Encrypt connection**.
5. Choose **Connect**.
6. Confirm that your connection is encrypted by running the following query. Verify that the query returns `true` for `encrypt_option`.

```
select ENCRYPT_OPTION from SYS.DM_EXEC_CONNECTIONS where SESSION_ID = @@SPID
```

For any other SQL client, use the following procedure.

To encrypt connections from other SQL clients

1. Append `encrypt=true` to your connection string. This string might be available as an option, or as a property on the connection page in GUI tools.

   Note
   To enable SSL encryption for clients that connect using JDBC, you might need to add the Amazon RDS SQL certificate to the Java CA certificate (cacerts) store. You can do this by using the keytool utility.

2. Confirm that your connection is encrypted by running the following query. Verify that the query returns `true` for `encrypt_option`.

```
select ENCRYPT_OPTION from SYS.DM_EXEC_CONNECTIONS where SESSION_ID = @@SPID
```

Related Topics

- Microsoft SQL Server on Amazon RDS (p. 720)
- Using SSL to Encrypt a Connection to a DB Instance (p. 358)

Options for the Microsoft SQL Server Database Engine

This section describes options, or additional features, that are available for Amazon RDS instances running the Microsoft SQL Server DB engine. To enable these options, you add them to an option group, and then associate the option group with your DB instance. For more information, see Working with Option Groups (p. 153).

Amazon RDS supports the following options for Microsoft SQL Server DB instances.

Option	Option ID	Engine Editions
Native Backup and Restore (p. 795)	SQLSERVER_BACKUP_RESTORE	SQL Server Enterprise Edition SQL Server Standard Edition SQL Server Web Edition SQL Server Express Edition
Transparent Data Encryption (p. 797)	TRANSPARENT_DATA_ENCRYPTION	SQL Server Enterprise Edition

Microsoft SQL Server Native Backup and Restore Support

Amazon RDS supports native backup and restore for Microsoft SQL Server databases using full backup files (.bak files). You can import and export SQL Server databases in a single, easily portable file. You can create a full backup of your on-premises database, store it on Amazon Simple Storage Service (Amazon S3), and then restore the backup file onto an existing Amazon RDS DB instance running SQL Server. You can back up an Amazon RDS SQL Server database, store it on Amazon S3, and then restore the backup file onto an on-premises server, or a different Amazon RDS DB instance running SQL Server. For more information, see Importing and Exporting SQL Server Databases (p. 769).

Native Backup and Restore Option Settings

Amazon RDS supports the following settings for the Native Backup and Restore option.

Option Setting	Valid Values	Description
IAM_ROLE_ARN	A valid Amazon Resource Name (ARN) in the format arn:aws:iam::*account-id*:role/*role-name*.	The ARN for an AWS Identity and Access Management (IAM) role to access the Amazon S3 bucket that contains your backup files. For more information, see AWS Identity and Access Management (IAM) .

Adding the Native Backup and Restore Option

The general process for adding the Native Backup and Restore option to a DB instance is the following:

1. Create a new option group, or copy or modify an existing option group.
2. Add the option to the option group.
3. Associate the option group with the DB instance.

After you add the Native Backup and Restore option, you don't need to restart your DB instance. As soon as the option group is active, you can begin backing up and restoring immediately.

To add the Native Backup and Restore option

1. Determine the option group you want to use. You can create a new option group or use an existing option group. If you want to use an existing option group, skip to the next step. Otherwise, create a custom DB option group. For more information, see Creating an Option Group (p. 154).
2. Add the **SQLSERVER_BACKUP_RESTORE** option to the option group, and configure the option settings. For more information about adding options, see Adding an Option to an Option Group (p. 157).

   a. For **IAM Role**, select an existing IAM role. Alternatively, you can choose to have a new IAM role created for you by choosing **Create a New Role**.
   b. For **Select S3 Bucket**, select an existing bucket. Alternatively, you can choose to have a new Amazon S3 bucket created for you by choosing **Create a New S3 Bucket**.
   c. For **Enable Encryption**, choose **Yes** to encrypt the backup file. If you choose **Yes**, for **Master Key** you must also choose an encryption key. For more information about encryption keys, see Getting Started in the AWS Key Management Service (AWS KMS) documentation.
3. Apply the option group to a new or existing DB instance.

   - For a new DB instance, you apply the option group when you launch the instance. For more information, see Creating a DB Instance Running the Microsoft SQL Server Database Engine (p. 738).
   - For an existing DB instance, you apply the option group by modifying the instance and attaching the new option group. For more information, see Modifying a DB Instance Running the Microsoft SQL Server Database Engine (p. 756).

Modifying Native Backup and Restore Option Settings

After you enable the Native Backup and Restore option, you can modify the settings for the option. For more information about how to modify option settings, see Modifying an Option Setting (p. 163). For more information about each setting, see Native Backup and Restore Option Settings (p. 795).

Removing the Native Backup and Restore Option

You can turn off the native backup and restore feature by removing the option from your DB instance. After you remove the Native Backup and Restore option, you don't need to restart your DB instance.

To remove the Native Backup and Restore option from a DB instance, do one of the following:

- Remove the Native Backup and Restore option from the option group it belongs to. This change affects all DB instances that use the option group. For more information, see Removing an Option from an Option Group (p. 167)
- Modify the DB instance and specify a different option group that doesn't include the Native Backup and Restore option. This change affects a single DB instance. You can specify the default (empty)

option group, or a different custom option group. For more information, see Modifying a DB Instance Running the Microsoft SQL Server Database Engine (p. 756).

Microsoft SQL Server Transparent Data Encryption Support

Amazon RDS supports using Transparent Data Encryption (TDE) to encrypt stored data on your DB instances running Microsoft SQL Server. TDE automatically encrypts data before it is written to storage, and automatically decrypts data when the data is read from storage.

Amazon RDS supports TDE for the following SQL Server versions and editions:

- SQL Server 2017 Enterprise Edition
- SQL Server 2016 Enterprise Edition
- SQL Server 2014 Enterprise Edition
- SQL Server 2012 Enterprise Edition
- SQL Server 2008 R2 Enterprise Edition

To enable transparent data encryption for a DB instance that is running SQL Server, specify the **TDE** option in an Amazon RDS option group that is associated with that DB instance.

Transparent data encryption for SQL Server provides encryption key management by using a two-tier key architecture. A certificate, which is generated from the database master key, is used to protect the data encryption keys. The database encryption key performs the actual encryption and decryption of data on the user database. Amazon RDS backs up and manages the database master key and the TDE certificate. To comply with several security standards, Amazon RDS is working to implement automatic periodic master key rotation.

Transparent data encryption is used in scenarios where you need to encrypt sensitive data in case data files and backups are obtained by a third party or when you need to address security-related regulatory compliance issues. Note that you cannot encrypt the system databases for SQL Server, such as the Model or Master databases.

A detailed discussion of transparent data encryption is beyond the scope of this guide, but you should understand the security strengths and weaknesses of each encryption algorithm and key. For information about transparent data encryption for SQL Server, see Transparent Data Encryption (TDE) on the Microsoft website.

You should determine if your DB instance is already associated with an option group that has the **TDE** option. To view the option group that a DB instance is associated with, you can use the RDS console, the describe-db-instance AWS CLI command, or the API action DescribeDBInstances.

The process for enabling transparent data encryption on a SQL Server DB instance is as follows:

1. If the DB instance is not associated with an option group that has the **TDE** option enabled, you must either create an option group and add the **TDE** option or modify the associated option group to add the **TDE** option. For information about creating or modifying an option group, see Working with Option Groups (p. 153). For information about adding an option to an option group, see Adding an Option to an Option Group (p. 157).
2. Associate the DB instance with the option group with the **TDE** option. For information about associating a DB instance with an option group, see Modifying a DB Instance Running the Microsoft SQL Server Database Engine (p. 756).

When the **TDE** option is added to an option group, Amazon RDS generates a certificate that is used in the encryption process. You can then use the certificate to run SQL statements that will encrypt data in a database on the DB instance. The following example uses the RDS-created certificate called `RDSTDECertificateName` to encrypt a database called `customerDatabase`.

```
---------- Enabling TDE -------------

-- Find a RDSTDECertificate to use
USE [master]
GO
SELECT name FROM sys.certificates WHERE name LIKE 'RDSTDECertificate%'
GO

USE [customerDatabase]
GO
-- Create DEK using one of the certificates from the previous step
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_128
ENCRYPTION BY SERVER CERTIFICATE [RDSTDECertificateName]
GO

-- Enable encryption on the database
ALTER DATABASE [customerDatabase]
SET ENCRYPTION ON
GO

-- Verify that the database is encrypted
USE [master]
GO
SELECT name FROM sys.databases WHERE is_encrypted = 1
GO
SELECT db_name(database_id) as DatabaseName, * FROM sys.dm_database_encryption_keys
GO
```

The time it takes to encrypt a SQL Server database using TDE depends on several factors, including the size of the DB instance, whether PIOPS is enabled for the instance, the amount of data, and other factors.

The **TDE** option is a persistent option than cannot be removed from an option group unless all DB instances and backups are disassociated from the option group. Once you add the **TDE** option to an option group, the option group can only be associated with DB instances that use TDE. For more information about persistent options in an option group, see Option Groups Overview (p. 153).

Because the **TDE** option is a persistent option, you can have a conflict between the option group and an associated DB instance. You can have a conflict between the option group and an associated DB instance in the following situations:

- The current option group has the **TDE** option, and you replace it with an option group that does not have the **TDE** option.
- You restore from a DB snapshot to a new DB instance that does not have an option group that contains the TDE option. For more information about this scenario, see Option Group Considerations (p. 214).

To disable TDE for a DB instance, first ensure that there are no encrypted objects left on the DB instance by either unencrypting the objects or by dropping them. If any encrypted objects exist on the DB instance, you will not be allowed to disable TDE for the DB instance. When you use the AWS Management Console to remove the **TDE** option from an option group, the console indicates that it is processing, and an event is created indicating an error if the option group is associated with an encrypted DB instance or DB snapshot.

The following example removes the TDE encryption from a database called `customerDatabase`.

```
------------- Removing TDE ----------------

USE [customerDatabase]
GO

-- Disable encryption on the database
ALTER DATABASE [customerDatabase]
SET ENCRYPTION OFF
GO

-- Wait until the encryption state of the database becomes 1. The state is 5 (Decryption in
 progress) for a while
SELECT db_name(database_id) as DatabaseName, * FROM sys.dm_database_encryption_keys
GO

-- Drop the DEK used for encryption
DROP DATABASE ENCRYPTION KEY
GO

-- Alter to SIMPLE Recovery mode so that your encrypted log gets truncated
USE [master]
GO
ALTER DATABASE [customerDatabase] SET RECOVERY SIMPLE
GO
```

When all objects are unencrypted, you can modify the DB instance to be associated with an option group without the **TDE** option or you can remove the **TDE** option from the option group.

Performance Considerations

The performance of a SQL Server DB instance can be impacted by using transparent data encryption.

Performance for unencrypted databases can also be degraded if the databases are on a DB instance that has at least one encrypted database. As a result, we recommend that you keep encrypted and unencrypted databases on separate DB instances.

Because of the nature of encryption, the database size and the size of the transaction log is larger than for an unencrypted database. You could run over your allocation of free backup space. The nature of TDE will cause an unavoidable performance hit. If you need high performance and TDE, measure the impact and make sure it meets your needs. There is less of an impact on performance if you use Provisioned IOPS and at least an M3.Large DB instance class.

Common DBA Tasks for Microsoft SQL Server

This section describes the Amazon RDS-specific implementations of some common DBA tasks for DB instances that are running the Microsoft SQL Server database engine. In order to deliver a managed service experience, Amazon RDS does not provide shell access to DB instances, and it restricts access to certain system procedures and tables that require advanced privileges.

> **Note**
> When working with a SQL Server DB instance, you can run scripts to modify a newly created database, but you cannot modify the [model] database, the database used as the model for new databases.

Topics

Accessing the tempdb Database on Microsoft SQL Server DB Instances on Amazon RDS

You can access the tempdb database on your Microsoft SQL Server DB instances on Amazon RDS. You can run code on tempdb by using Transact-SQL through Microsoft SQL Server Management Studio (SSMS), or any other standard SQL client application. For more information about connecting to your DB instance, see Connecting to a DB Instance Running the Microsoft SQL Server Database Engine (p. 749).

The master user for your DB instance is granted `CONTROL` access to tempdb so that this user can modify the tempdb database options. The master user isn't the database owner of the tempdb database. If necessary, the master user can grant `CONTROL` access to other users so that they can also modify the tempdb database options.

Note
You can't run Database Console Commands (DBCC) on the tempdb database.

Modifying tempdb Database Options

You can modify the database options on the tempdb database on your Amazon RDS DB instances. For more information about which options can be modified, see tempdb Database in the Microsoft documentation.

Database options such as the maximum file size options are persistent after you restart your DB instance. You can modify the database options to optimize performance when importing data, and to prevent running out of storage.

Optimizing Performance when Importing Data

To optimize performance when importing large amounts of data into your DB instance, set the `SIZE` and `FILEGROWTH` properties of the tempdb database to large numbers. For more information about how to optimize tempdb, see Optimizing tempdb Performance in the Microsoft documentation.

The following example demonstrates setting the size to 100 GB and file growth to 10 percent.

```
alter database[tempdb] modify file (NAME = N'templog', SIZE=100GB, FILEGROWTH = 10%)
```

Preventing Storage Problems

To prevent the tempdb database from using all available disk space, set the `MAXSIZE` property. The following example demonstrates setting the property to 2048 MB.

```
alter database [tempdb] modify file (NAME = N'templog', MAXSIZE = 2048MB)
```

Shrinking the tempdb Database

There are two ways to shrink the tempdb database on your Amazon RDS DB instance. You can use the `rds_shrink_tempdbfile` procedure, or you can set the `SIZE` property,

Using the rds_shrink_tempdbfile Procedure

You can use the Amazon RDS procedure `msdb.dbo.rds_shrink_tempdbfile` to shrink the tempdb database. You can only call `rds_shrink_tempdbfile` if you have `CONTROL` access to tempdb. When you call `rds_shrink_tempdbfile`, there is no down time for your DB instance.

The `rds_shrink_tempdbfile` procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
`@temp_filename`	SYSNAME	—	required	The logical name of the file to shrink.
`@target_size`	int	null	optional	The new size for the file, in megabytes.

The following example gets the names of the files for the tempdb database.

```
use tempdb;
GO

select name, * from sys.sysfiles;
GO
```

The following example shrinks a tempdb database file named `test_file`, and requests a new size of `10` megabytes:

```
exec msdb.dbo.rds_shrink_tempdbfile @temp_filename = N'test_file', @target_size = 10;
```

Setting the SIZE Property

You can also shrink the tempdb database by setting the `SIZE` property and then restarting your DB instance. For more information about restarting your DB instance, see Rebooting a DB Instance (p. 119).

The following example demonstrates setting the `SIZE` property to 1024 MB.

```
alter database [tempdb] modify file (NAME = N'templog', SIZE = 1024MB)
```

Considerations for Multi-AZ Deployments

If your Amazon RDS DB instance is in a Multi-AZ Deployment for Microsoft SQL Server with Database Mirroring, there are some things to consider.

The tempdb database can't be replicated. No data that you store on your primary instance is replicated to your secondary instance.

If you modify any database options on the tempdb database, you can capture those changes on the secondary by using one of the following methods:

- First modify your DB instance and turn Multi-AZ off, then modify tempdb, and finally turn Multi-AZ back on. This method doesn't involve any downtime.

  For more information, see Modifying a DB Instance Running the Microsoft SQL Server Database Engine (p. 756).
- First modify tempdb in the original primary instance, then fail over manually, and finally modify tempdb in the new primary instance. This method involves downtime.

  For more information, see Rebooting a DB Instance (p. 119).

Analyzing Your Database Workload on an Amazon RDS DB Instance with SQL Server Tuning Advisor

The Database Engine Tuning Advisor is a client application provided by Microsoft that analyzes database workload and recommends an optimal set of indexes for your Microsoft SQL Server databases based on the kinds of queries you run. Like SQL Server Management Studio, you run Tuning Advisor from a client computer that connects to your Amazon RDS DB instance that is running SQL Server. The client computer can be a local computer that you run on premises within your own network or it can be an Amazon EC2 Windows instance that is running in the same region as your Amazon RDS DB instance.

This section shows how to capture a workload for Tuning Advisor to analyze. This is the preferred process for capturing a workload because Amazon RDS restricts host access to the SQL Server instance. The full documentation on Tuning Advisor can be found on MSDN.

To use Tuning Advisor, you must provide what is called a workload to the advisor. A workload is a set of Transact-SQL statements that execute against a database or databases that you want to tune. Database Engine Tuning Advisor uses trace files, trace tables, Transact-SQL scripts, or XML files as workload input when tuning databases. When working with Amazon RDS, a workload can be a file on a client computer or a database table on an Amazon RDS SQL Server DB accessible to your client computer. The file or the table must contain queries against the databases you want to tune in a format suitable for replay.

For Tuning Advisor to be most effective, a workload should be as realistic as possible. You can generate a workload file or table by performing a trace against your DB instance. While a trace is running, you can either simulate a load on your DB instance or run your applications with a normal load.

There are two types of traces: client-side and server-side. A client-side trace is easier to set up and you can watch trace events being captured in real-time in SQL Server Profiler. A server-side trace is more complex to set up and requires some Transact-SQL scripting. In addition, because the trace is written to a file on the Amazon RDS DB instance, storage space is consumed by the trace. It is important to track of how much storage space a running server-side trace uses because the DB instance could enter a storage-full state and would no longer be available if it runs out of storage space.

For a client-side trace, when a sufficient amount of trace data has been captured in the SQL Server Profiler, you can then generate the workload file by saving the trace to either a file on your local computer or in a database table on an DB instance that is available to your client computer. The main disadvantage of using a client-side trace is that the trace may not capture all queries when under heavy loads. This could weaken the effectiveness of the analysis performed by the Database Engine Tuning Advisor. If you need to run a trace under heavy loads and you want to ensure that it captures every query during a trace session, you should use a server-side trace.

For a server-side trace, you must get the trace files on the DB instance into a suitable workload file or you can save the trace to a table on the DB instance after the trace completes. You can use the SQL Server Profiler to save the trace to a file on your local computer or have the Tuning Advisor read from the trace table on the DB instance.

Running a Client-Side Trace on a SQL Server DB Instance

To run a client-side trace on a SQL Server DB instance

1. Start SQL Server Profiler. It is installed in the Performance Tools folder of your SQL Server instance folder. You must load or define a trace definition template to start a client-side trace.
2. In the SQL Server Profiler File menu, click **New Trace**. In the **Connect to Server** dialog box, enter the DB instance endpoint, port, master user name, and password of the database you would like to run a trace on.
3. In the **Trace Properties** dialog box, enter a trace name and choose a trace definition template. A default template, TSQL_Replay, ships with the application. You can edit this template to define your

trace. Edit events and event information under the **Events Selection** tab of the **Trace Properties** dialog box. For more information about trace definition templates and using the SQL Server Profiler to specify a client-side trace see the documentation in MSDN.

4. Start the client-side trace and watch SQL queries in real-time as they execute against your DB instance.
5. Select **Stop Trace** from the File menu when you have completed the trace. Save the results as a file or as a trace table on you DB instance.

Running a Server-Side Trace on a SQL Server DB Instance

Writing scripts to create a server-side trace can be complex and is beyond the scope of this document. This section contains sample scripts that you can use as examples. As with a client-side trace, the goal is to create a workload file or trace table that you can open using the Database Engine Tuning Advisor.

The following is an abridged example script that starts a server-side trace and captures details to a workload file. The trace initially saves to the file RDSTrace.trc in the D:\RDSDBDATA\Log directory and rolls-over every 100 MB so subsequent trace files are named RDSTrace_1.trc, RDSTrace_2.trc, etc.

```
DECLARE @file_name NVARCHAR(245) = 'D:\RDSDBDATA\Log\RDSTrace';
DECLARE @max_file_size BIGINT = 100;
DECLARE @on BIT = 1
DECLARE @rc INT
DECLARE @traceid INT

EXEC @rc = sp_trace_create @traceid OUTPUT, 2, @file_name, @max_file_size
IF (@rc != 0) BEGIN
   EXEC sp_trace_setevent @traceid, 10, 1, @on
   EXEC sp_trace_setevent @traceid, 10, 2, @on
   EXEC sp_trace_setevent @traceid, 10, 3, @on
  …
   EXEC sp_trace_setfilter @traceid, 10, 0, 7, N'SQL Profiler'
   EXEC sp_trace_setstatus @traceid, 1
   END
```

The following example is a script that stops a trace. Note that a trace created by the previous script continues to run until you explicitly stop the trace or the process runs out of disk space.

```
DECLARE @traceid INT
SELECT @traceid = traceid FROM ::fn_trace_getinfo(default)
WHERE property = 5 AND value = 1 AND traceid <> 1

IF @traceid IS NOT NULL BEGIN
   EXEC sp_trace_setstatus @traceid, 0
   EXEC sp_trace_setstatus @traceid, 2
END
```

You can save server-side trace results to a database table and use the database table as the workload for the Tuning Advisor by using the fn_trace_gettable function. The following commands load the results of all files named RDSTrace.trc in the D:\rdsdbdata\Log directory, including all rollover files like RDSTrace_1.trc, into a table named RDSTrace in the current database:

```
SELECT * INTO RDSTrace
FROM fn_trace_gettable('D:\rdsdbdata\Log\RDSTrace.trc', default);
```

To save a specific rollover file to a table, for example the RDSTrace_1.trc file, specify the name of the rollover file and substitute 1 instead of default as the last parameter to fn_trace_gettable.

```
SELECT * INTO RDSTrace_1
FROM fn_trace_gettable('D:\rdsdbdata\Log\RDSTrace_1.trc', 1);
```

Running Tuning Advisor with a Trace

Once you create a trace, either as a local file or as a database table, you can then run Tuning Advisor against your DB instance. Microsoft includes documentation on using the Database Engine Tuning Advisor in MSDN. Using Tuning Advisor with Amazon RDS is the same process as when working with a standalone, remote SQL Server instance. You can either use the Tuning Advisor UI on your client machine or use the dta.exe utility from the command line. In both cases, you must connect to the Amazon RDS DB instance using the endpoint for the DB instance and provide your master user name and master user password when using Tuning Advisor.

The following code example demonstrates using the dta.exe command line utility against an Amazon RDS DB instance with an endpoint of **`dta.cnazcmklsdei.us-east-1.rds.amazonaws.com`**. The example includes the master user name **`admin`** and the master user password **`test`**, the example database to tune is named **`RDSDTA`** and the input workload is a trace file on the local machine named **`C:\RDSTrace.trc`**. The example command line code also specifies a trace session named **`RDSTrace1`** and specifies output files to the local machine named **`RDSTrace.sql`** for the SQL output script, **`RDSTrace.txt`** for a result file, and **`RDSTrace.xml`** for an XML file of the analysis. There is also an error table specified on the RDSDTA database named **`RDSTraceErrors`**.

```
dta -S dta.cnazcmklsdei.us-east-1.rds.amazonaws.com -U admin -P test -D RDSDTA -if C:
\RDSTrace.trc -s RDSTrace1 -of C:\ RDSTrace.sql -or C:\ RDSTrace.txt -ox C:\ RDSTrace.xml -
e RDSDTA.dbo.RDSTraceErrors
```

Here is the same example command line code except the input workload is a table on the remote Amazon RDS instance named **`RDSTrace`** which is on the **`RDSDTA`** database.

```
dta -S dta.cnazcmklsdei.us-east-1.rds.amazonaws.com -U admin -P test -D RDSDTA -it
 RDSDTA.dbo.RDSTrace -s RDSTrace1 -of C:\ RDSTrace.sql -or C:\ RDSTrace.txt -ox C:\
 RDSTrace.xml -e RDSDTA.dbo.RDSTraceErrors
```

A full list of dta utility command-line parameters can be found in MSDN.

Collations and Character Sets for Microsoft SQL Server

Amazon RDS creates a default server collation for character sets when a Microsoft SQL Server DB instance is created. This default server collation is currently English (United States), or more precisely, SQL_Latin1_General_CP1_CI_AS. You can change the default collation at the database, table, or column level by overriding the collation when creating a new database or database object. For example, you can change from the default collation SQL_Latin1_General_CP1_CI_AS to Japanese_CI_AS for Japanese collation support. Even arguments in a query can be type-cast to use a different collation if necessary.

For example, the following query would change the default collation for the AccountName column to Japanese_CI_AS:

```
CREATE TABLE [dbo].[Account]
(
    [AccountID] [nvarchar](10) NOT NULL,
    [AccountName] [nvarchar](100) COLLATE Japanese_CI_AS NOT NULL
) ON [PRIMARY];
```

The Microsoft SQL Server DB engine supports Unicode by the built-in NCHAR, NVARCHAR, and NTEXT data types. For example, if you need CJK support, use these Unicode data types for character storage and override the default server collation when creating your databases and tables. Here are several links from Microsoft covering collation and Unicode support for SQL Server:

- Working with Collations
- Collation and International Terminology
- Using SQL Server Collations
- International Considerations for Databases and Database Engine Applications

Determining a Recovery Model for Your Microsoft SQL Server Database

In Amazon RDS, the recovery model, retention period, and database status are linked. Changes to one can impact the other settings. For example:

- Changing a database's recovery model to "Simple" while backup retention is enabled will result in Amazon RDS setting the recovery model to "Full" within five minutes of the setting change. This will also result in Amazon RDS taking a snapshot of the DB instance.
- Setting the backup retention to "0" days results in Amazon RDS setting the recovery mode to "Simple."
- Changing a database's recovery model from "Simple" to any other option while backup retention is set to "0" days results in Amazon RDS setting the recovery model back to "Simple."

Dropping a Microsoft SQL Server Database in a Multi-AZ with Mirroring Deployment

You can drop a database on an Amazon RDS DB instance running Microsoft SQL Server in a Multi-AZ deployment using Mirroring. You can use the following commands:

```
ALTER DATABASE <database_name> SET PARTNER OFF;
GO
DROP DATABASE <database_name>;
GO
```

Renaming a Microsoft SQL Server Database in a Multi-AZ with Mirroring Deployment

You can't rename a database on a Microsoft SQL Server DB instance that is in a SQL Server Multi-AZ with Mirroring deployment. If you need to rename a database on such an instance, first turn off Multi-AZ with Mirroring for the DB instance, then rename the database, and finally turn Multi-AZ with Mirroring back on for the DB instance. For more information, see Modifying a DB Instance Running the Microsoft SQL Server Database Engine (p. 756).

You can rename a database using the following procedure, which renames a database from MOO to ZAR. The procedure is analogous to the command `DDL ALTER DATABASE [MOO] MODIFY NAME = [ZAR]`.

```
EXEC rdsadmin.dbo.rds_modify_db_name N'MOO', N'ZAR'
```

```
GO
```

Resetting the db_owner Role Password

If you lock yourself out of the db_owner role on your Microsoft SQL Server database, you can reset the db_owner role password by modifying the DB instance master password. By changing the DB instance master password, you can regain access to the DB instance, access databases using the modified password for the db_owner, and restore privileges for the db_owner role that may have been accidentally revoked. You can change the DB instance password by using the Amazon RDS console, the AWS CLI command modify-db-instance, or by using the ModifyDBInstance action. For more information about modifying a SQL Server DB instance, see Modifying a DB Instance Running the Microsoft SQL Server Database Engine (p. 756).

Restoring License-Terminated DB Instances

Microsoft has requested that some Amazon RDS customers who did not report their Microsoft License Mobility information terminate their DB instance. Amazon RDS takes snapshots of these DB instances, and you can restore from the snapshot to a new DB instance that has the License Included model.

You can restore from a snapshot of Standard Edition to either Standard Edition or Enterprise Edition.

You can restore from a snapshot of Enterprise Edition to either Standard Edition or Enterprise Edition.

To restore from a SQL Server snapshot after Amazon RDS has created a final snapshot of your instance:

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Snapshots**.
3. Choose the snapshot of your SQL Server DB instance. Amazon RDS created a final snapshot of your DB instance; the name of the terminated instance snapshot is in the format: '<name of instance>-final-snapshot'. For example, if your DB instance name was **mytest.cdxgahslksma.us-east-1.rds.com**, the final snapshot would be called **mytest-final-snapshot** and would be located in the same region as the original DB instance.
4. Choose **Restore Snapshot**.

   The **Restore DB Instance** window appears.
5. For **License Model** choose **license-included**.
6. Choose the SQL Server DB engine you want to use.
7. In the **DB Instance Identifier** text box type the name for the restored DB instance.
8. Choose **Restore DB Instance**.

For more information about restoring from a snapshot, see Restoring from a DB Snapshot (p. 209).

Transitioning a Microsoft SQL Server Database from OFFLINE to ONLINE

You can transition your Microsoft SQL Server database on an Amazon RDS DB instance from OFFLINE to ONLINE.

SQL Server method	Amazon RDS method
ALTER DATABASE *name* SET ONLINE;	EXEC rdsadmin.dbo.rds_set_database_online *name*

Using SQL Server Agent

With Amazon RDS, you can use SQL Server Agent on a DB instance running Microsoft SQL Server Standard, Web Edition, or Enterprise Edition. SQL Server Agent is a Microsoft Windows service that executes scheduled administrative tasks, which are called jobs. You can use SQL Server Agent to run T-SQL jobs to rebuild indexes, run corruption checks, and aggregate data in a SQL Server DB instance.

SQL Server Agent can run a job on a schedule, in response to a specific event, or on demand. For more information, see SQL Server Agent in the SQL Server documentation. You should avoid scheduling jobs to run during the maintenance and backup windows for your DB instance because these maintenance and backup processes that are launched by AWS could interrupt the job or cause it to be cancelled. Because Amazon RDS backs up your DB instance, you do not use SQL Server Agent to create backups.

To view the history of an individual SQL Server Agent job in the SQL Server Management Studio, you open Object Explorer, right-click the job, and then click **View History**.

Because SQL Server Agent is running on a managed host in a DB instance, there are some actions that are not supported. Running replication jobs and running command-line scripts by using ActiveX, Windows command shell, or Windows PowerShell are not supported. In addition, you cannot manually start, stop, or restart SQL Server Agent because its operation is managed by the host. Email notifications through SQL Server Agent are not available from a DB instance.

When you create a SQL Server DB instance, the master user name is enrolled in the SQLAgentUserRole role. To allow an additional login/user to use SQL Server Agent, you must log in as the master user and do the following.

1. Create another server-level login by using the `CREATE LOGIN` command.
2. Create a user in msdb using `CREATE USER` command, and then link this user to the login that you created in the previous step.
3. Add the user to the SQLAgentUserRole using the `sp_addrolemember` system stored procedure.

For example, suppose your master user name is **`myawsmaster`** and you want to give access to SQL Server Agent to a user named **`theirname`** with a password **`theirpassword`**. You would log in using the master user name and run the following commands.

```
--Initially set context to master database
USE [master];
GO
--Create a server-level login named theirname with password theirpassword
CREATE LOGIN [theirname] WITH PASSWORD = 'theirpassword';
GO
--Set context to msdb database
USE [msdb];
GO
--Create a database user named theirname and link it to server-level login theirname
CREATE USER [theirname] FOR LOGIN [theirname];
GO
--Added database user theirname in msdb to SQLAgentUserRole in msdb
EXEC sp_addrolemember [SQLAgentUserRole], [theirname];
```

You cannot use the UI in SQL Server Management Console to delete a SQL Server Agent job. To delete a SQL Server Agent job, run the following T-SQL statement.

```
EXEC msdb..sp_delete_job @job_name = '<job-name>';
```

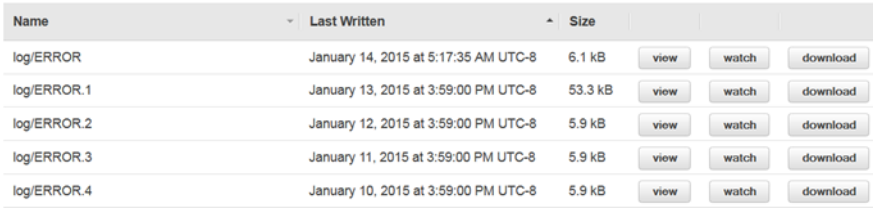
Working with Microsoft SQL Server Logs

You can use the Amazon RDS console to view, watch, and download SQL Server Agent logs and Microsoft SQL Server error logs.

Watching Log Files

If you view a log in the Amazon RDS console, you can see its contents as they exist at that moment. Watching a log in the console opens it in a dynamic state so that you can see updates to it in near real time.

Only the latest log is active for watching. For example, suppose you have the logs shown following:



Only log/ERROR, as the most recent log, is being actively updated. You can choose to watch others, but they are static and will not update.

Archiving Log Files

The Amazon RDS console shows logs for the past week through the current day. You can download and archive logs to keep them for reference past that time. One way to archive logs is to load them into an Amazon S3 instance. For instructions on how to set up an Amazon S3 instance and upload a file, see Amazon S3 Basics in the *Amazon Simple Storage Service Getting Started Guide* and click **Get Started**.

Using the rds_read_error_log Procedure

To view Microsoft SQL server error and agent logs, use the Amazon RDS stored procedure `rds_read_error_log` with the following parameters:

- **`@index`** – the version of the log to retrieve. The default value is 0, which retrieves the current error log. Specify 1 to retrieve the previous log, specify 2 to retrieve the one before that, and so on.
- **`@type`** – the type of log to retrieve. Specify 1 to retrieve an error log. Specify 2 to retrieve an agent log.

Example

The following example requests the current error log.

```
EXEC rdsadmin.dbo.rds_read_error_log @index = 0, @type = 1;
```

Related Topics

- Amazon RDS Database Log Files (p. 303)

- Microsoft SQL Server Database Log Files (p. 312)
- Working with Trace and Dump Files (p. 810)

Working with Trace and Dump Files

This section describes working with trace files and dump files for your Amazon RDS DB instances running Microsoft SQL Server.

Generating a Trace SQL Query

```
declare @rc int
declare @TraceID int
declare @maxfilesize bigint

set @maxfilesize = 5

exec @rc = sp_trace_create @TraceID output,  0, N'D:\rdsdbdata\log\rdstest', @maxfilesize,
 NULL
```

Viewing an Open Trace

```
select * from ::fn_trace_getinfo(default)
```

Viewing Trace Contents

```
select * from ::fn_trace_gettable('D:\rdsdbdata\log\rdstest.trc', default)
```

Setting the Retention Period for Trace and Dump Files

Trace and dump files can accumulate and consume disk space. By default, Amazon RDS purges trace and dump files that are older than seven days.

To view the current trace and dump file retention period, use the `rds_show_configuration` procedure, as shown in the following example.

```
exec rdsadmin..rds_show_configuration;
```

To modify the retention period for trace files, use the `rds_set_configuration` procedure and set the `tracefile retention` in minutes. The following example sets the trace file retention period to 24 hours.

```
exec rdsadmin..rds_set_configuration 'tracefile retention', 1440;
```

To modify the retention period for dump files, use the `rds_set_configuration` procedure and set the `dumpfile retention` in minutes. The following example sets the dump file retention period to 3 days.

```
exec rdsadmin..rds_set_configuration 'dumpfile retention', 4320;
```

For security reasons, you cannot delete a specific trace or dump file on a SQL Server DB instance. To delete all unused trace or dump files, set the retention period for the files to 0.

Related Topics

- Amazon RDS Database Log Files (p. 303)
- Microsoft SQL Server Database Log Files (p. 312)
- Working with Microsoft SQL Server Logs (p. 809)

Related Topics

- Local Time Zone for Microsoft SQL Server DB Instances (p. 731)

Advanced Administrative Tasks and Concepts for Microsoft SQL Server DB Instances

This section provides information about advanced administrative tasks and concepts for Microsoft SQL Server DB instances on Amazon RDS.

Topics

- Using Windows Authentication with a Microsoft SQL Server DB Instance (p. 812)

Using Windows Authentication with a Microsoft SQL Server DB Instance

You can use Windows Authentication to authenticate users when they connect to your Amazon RDS DB instance running Microsoft SQL Server. The DB instance works with AWS Directory Service for Microsoft Active Directory (Enterprise Edition), also called **Microsoft AD**, to enable Windows Authentication. When users authenticate with a SQL Server DB instance joined to the trusting domain, authentication requests are forwarded to the domain directory that you create with AWS Directory Service.

Amazon RDS supports Windows Authentication for SQL Server in all AWS Regions except the following:

- US West (N. California)
- Asia Pacific (Mumbai)
- South America (São Paulo)

Amazon RDS uses Mixed Mode for Windows Authentication. This approach means that the *master user* (the name and password used to create your SQL Server DB instance) uses SQL Authentication. Because the master user account is a privileged credential, you should restrict access to this account.

To get Windows Authentication using an on-premises or self-hosted Microsoft Active Directory, you need to create a forest trust. For more information on setting up forest trusts using AWS Directory Service, see Create a Trust Relationship (Microsoft AD).

To set up Windows authentication for a SQL Server DB instance, do the following steps (explained in greater detail in this section):

1. Use the AWS Directory Service for Microsoft Active Directory (Enterprise Edition), also called Microsoft AD, either from the AWS console or AWS Directory Service API to create a Microsoft AD directory.
2. If you use the AWS CLI or Amazon RDS API to create your SQL Server DB instance, you need to create an IAM role that uses the managed IAM policy AmazonRDSDirectoryServiceAccess. The role allows Amazon RDS to make calls to your directory. If you use the AWS console to create your SQL Server DB instance, AWS creates the IAM role for you.
3. Create and configure users and groups in the *Microsoft AD* directory using the Microsoft Active Directory tools. For more information about creating users and groups in your Active Directory, see **Add Users and Groups (Simple AD and Microsoft AD)** in the AWS Directory Service documentation. Add Users and Groups (Simple AD and Microsoft AD).
4. Use Amazon RDS to create a new SQL Server DB instance either from the AWS console, AWS CLI, or Amazon RDS API. In the create request, you provide the domain identifier ("d-*" identifier) that was generated when you created your directory and the name of the role you created. You can also modify an existing SQL Server DB instance to use Windows Authentication by setting the *domain* and *IAM role* parameters for the DB instance, and locating the DB instance in the same VPC as the domain directory.
5. Use the Amazon RDS *master user* credentials to connect to the SQL Server DB instance as you would any other DB instance. Because the DB instance is joined to the *Microsoft AD* domain, you can provision SQL Server logins and users from the Active Directory users and groups in their domain (known as SQL Server "Windows" logins). Database permissions are managed through standard SQL Server permissions granted and revoked to these windows logins.

Creating the Endpoint for Kerberos Authentication

Kerberos-based authentication requires that the endpoint be the customer-specified host name, a period, and then the fully qualified domain name (FQDN). For example, the following is an example of an endpoint you would use with Kerberos-based authentication. In this example, the SQL Server DB instance host name is `ad-test` and the domain name is `corp-ad.company.com`:

```
ad-test.corp-ad.company.com
```

If you want to check to make sure your connection is using Kerberos, you can run the following query:

```
SELECT net_transport, auth_scheme
  FROM sys.dm_exec_connections
 WHERE session_id = @@SPID;
```

Setting Up Windows Authentication for SQL Server DB Instances

You use AWS Directory Service for Microsoft Active Directory (Enterprise Edition), also called **Microsoft AD**, to set up Windows Authentication for a SQL Server DB instance. To set up Windows Authentication, you take the following steps:

Step 1: Create a Directory Using the AWS Directory Service for Microsoft Active Directory (Enterprise Edition)

AWS Directory Service creates a fully managed, Microsoft Active Directory in the AWS cloud. When you create a Microsoft AD directory, AWS Directory Service creates two domain controllers and DNS servers on your behalf. The directory servers are created in different subnets in a VPC; this redundancy helps ensure that your directory remains accessible even if a failure occurs.

When you create a *Microsoft AD* directory, AWS Directory Service performs the following tasks on your behalf:

- Sets up a Microsoft Active Directory within the VPC.
- Creates a directory administrator account with the user name Admin and the specified password. You use this account to manage your directory.

  Note
  Be sure to save this password. AWS Directory Service does not store this password and it cannot be retrieved or reset.
- Creates a security group for the directory controllers.

When you launch an AWS Directory Service for Microsoft Active Directory (Enterprise Edition), AWS creates an Organizational Unit (OU) that contains all your directory's objects. This OU, which has the NetBIOS name that you typed when you created your directory, is located in the domain root. The domain root is owned and managed by AWS.

The *admin* account that was created with your *Microsoft AD* directory has permissions for the most common administrative activities for your OU:

- Create update, or delete users, groups, and computers
- Add resources to your domain such as file or print servers, and then assign permissions for those resources to users and groups in your OU
- Create additional OUs and containers
- Delegate authority
- Create and link group policies
- Restore deleted objects from the Active Directory Recycle Bin
- Run AD and DNS Windows PowerShell modules on the Active Directory Web Service

The *admin* account also has rights to perform the following domain-wide activities:

- Manage DNS configurations (Add, remove, or update records, zones, and forwarders)
- View DNS event logs
- View security event logs

To create a directory with AWS Directory Service for Microsoft Active Directory (Enterprise Edition)

1. In the AWS Directory Service console navigation pane, select **Directories** and choose **Set up Directory**.
2. Choose **Create Microsoft AD**. Microsoft AD is the only option currently supported for use with Amazon RDS.
3. Provide the following information:

   Directory DNS

   The fully qualified name for the directory, such as corp.example.com.

   NetBIOS name

   The short name for the directory, such as CORP.

   Administrator password

   The password for the directory administrator. The directory creation process creates an administrator account with the user name Admin and this password.

   The directory administrator password and cannot include the word "admin." The password is case-sensitive and must be between 8 and 64 characters in length, inclusive. It must also contain at least one character from three of the following four categories:

   - Lowercase letters (a-z)
   - Uppercase letters (A-Z)
   - Numbers (0-9)
   - Non-alphanumeric characters (~!@#$%^&*_-+=`|\(){}[]:;"'<>,.?/)

   Confirm password

   Retype the administrator password.

   Description

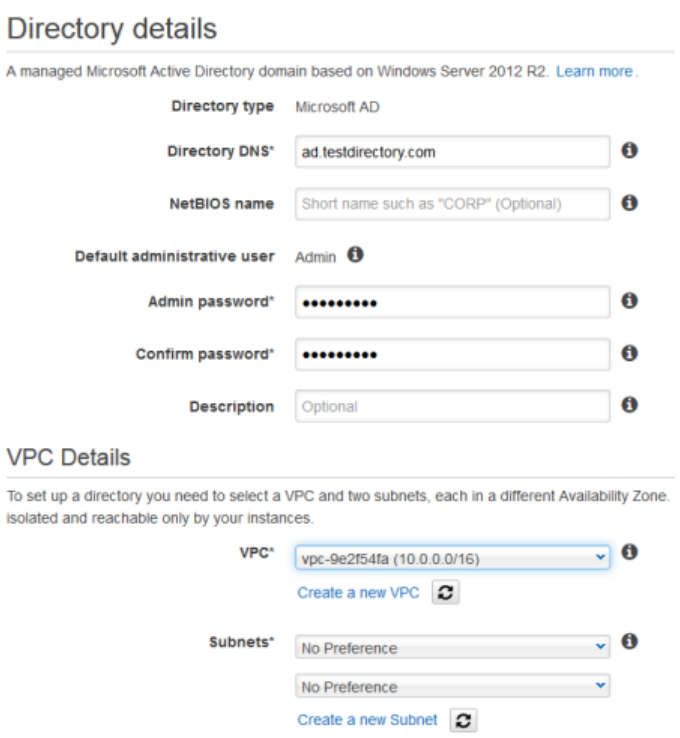
   An optional description for the directory.
4. Provide the following information in the **VPC Details** section and choose **Next Step**.

   VPC

   The VPC for the directory. Note that the SQL Server DB instance must be created in this same VPC.

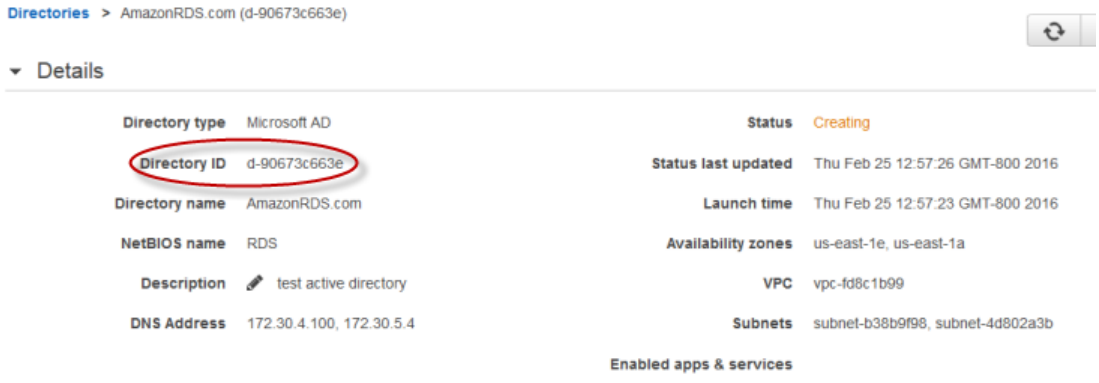
   Subnets

   Select the subnets for the directory servers. The two subnets must be in different Availability Zones.
5. Review the directory information and make any necessary changes. When the information is correct, choose **Create Microsoft AD**.



It takes several minutes for the directory to be created. When it has been successfully created, the **Status** value changes to **Active**.

To see information about your directory, select the directory in the directory listing. Note the Directory ID; you will need this value when you create or modify your SQL Server DB instance.



Step 2: Create the IAM role for Use by Amazon RDS

If you use the AWS console to create your SQL Server DB instance, you can skip this step. If you used the AWS CLI or Amazon RDS API to create your SQL Server DB instance, you must create an IAM role that uses the managed IAM policy **AmazonRDSDirectoryServiceAccess**. This role allows Amazon RDS to make calls to the AWS Directory Service for you.

The following IAM policy, **AmazonRDSDirectoryServiceAccess,** provides access to AWS Directory Service:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
            "ds:DescribeDirectories",
            "ds:AuthorizeApplication",
            "ds:UnauthorizeApplication"
        ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Create an IAM role using this policy. For more information about creating IAM roles, see Creating Customer Managed Policies.

Step 3: Create and Configure Users and Groups

You can create users and groups with the Active Directory Users and Computers tool, which is part of the Active Directory Domain Services and Active Directory Lightweight Directory Services tools. Users represent individual people or entities that have access to your directory. Groups are very useful for giving or denying privileges to groups of users, rather than having to apply those privileges to each individual user.

To create users and groups in an AWS Directory Service directory, you must be connected to a Windows EC2 instance that is a member of the AWS Directory Service directory, and be logged in as a user that has privileges to create users and groups. For more information, see Add Users and Groups (Simple AD and Microsoft AD).

Step 4: Create or Modify a SQL Server DB Instance

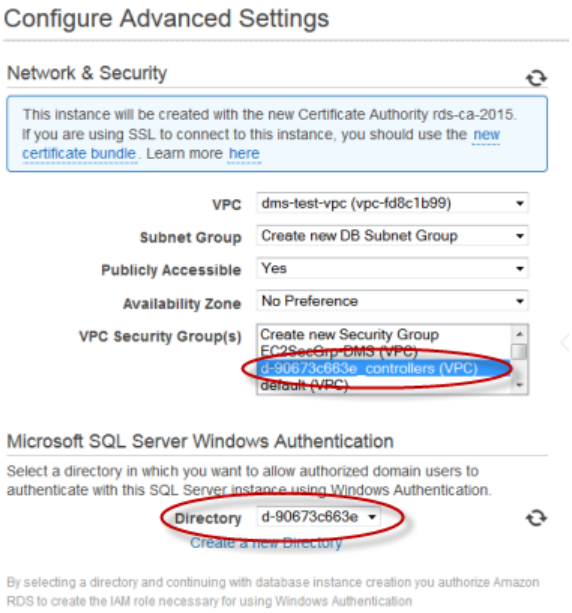
Next, you create or modify a Microsoft SQL Server DB instance for use with the directory. You can do this in one of the following ways:

- Create a new SQL Server DB instance
- Modify an existing SQL Server DB instance
- Restore a SQL Server DB instance from a DB Snapshot
- Restore a SQL Server DB instance from a Point-in-Time Restore

Windows Authentication is only supported for SQL Server DB instances in a VPC, and the DB instance must be in the same VPC as the directory.

Several parameters are required for the DB instance to be able to use the domain directory you created:

- For the **domain** parameter, you must enter the domain identifier ("d-*" identifier) generated when you created the directory.
- Use the same VPC that was used when you created the directory.
- Use a security group that allows egress within the VPC so the DB instance can communicate with the directory.



Step 5: Create Windows Authentication SQL Server Logins

Use the Amazon RDS *master user* credentials to connect to the SQL Server DB instance as you would any other DB instance. Because the DB instance is joined to the *Microsoft AD* domain, you can provision SQL Server logins and users from the Active Directory users and groups in your domain. Database permissions are managed through standard SQL Server permissions granted and revoked to these windows logins.

To allow an Active Directory user to authenticate with SQL Server, a SQL Server Windows login must exist for the user or a group that the user is a member of. Fine-grained access control is handled through granting and revoking permissions on these SQL Server logins. If a user does not have a corresponding SQL Server login and is not a member of a group with a corresponding SQL Server login, that user cannot access the SQL Server DB instance.

The ALTER ANY LOGIN permission is required to create an Active Directory SQL Server login. If you have not yet created any logins with this permission, connect as the DB instance's *master user* using SQL Server Authentication. Run the following data definition language (DDL) command to create a SQL Server login for an Active Directory user or group:

```
CREATE LOGIN [<user or group>] FROM WINDOWS WITH DEFAULT_DATABASE = [master],
   DEFAULT_LANGUAGE = [us_english];
```

Users or groups must be specified using the pre–Windows 2000 login name in the format *domainName**login_name*. You cannot use a User Principle Name (UPN) in the format *login_name*@*DomainName*. For more information about CREATE LOGIN, go to https://msdn.microsoft.com/en-us/library/ms189751.aspx in the Microsoft Developer Network documentation.

Users (both humans and applications) from your domain can now connect to the RDS SQL Server instance from a domain joined client machine using Windows authentication.

Managing a DB Instance in a Domain

You can use the AWS console, AWS CLI, or the Amazon RDS API to manage your DB instance and its relationship with your domain, such as moving the DB instance into, out of, or between domains.

For example, using the Amazon RDS API, you can do the following:

- To re-attempt a domain join for a failed membership, use the *ModifyDBInstance* API action and specify the current membership's directory ID.
- To update the IAM role name for membership, use the *ModifyDBInstance* API action and specify the current membership's directory ID and the new IAM role.
- To remove a DB instance from a domain, use the *ModifyDBInstance* API action and specify 'none' as the domain parameter.
- To move a DB instance from one domain to another, use the *ModifyDBInstance* API action and specify the domain identifier of the new domain as the domain parameter.
- To list membership for each DB instance, use the *DescribeDBInstances* API action.

Understanding Domain Membership

After you create or modify your DB instance, the instance becomes a member of the domain. The AWS console indicates the status of the domain membership for the DB instance. The status of the DB instance can be one of the following:

- joined - The instance is a member of the domain.
- joining - The instance is in the process of becoming a member of the domain.
- pending-join - The instance membership is pending .
- pending-maintenance-join - AWS will attempt to make the instance a member of the domain during the next scheduled maintenance window.
- pending-removal - The removal of the instance from the domain is pending.
- pending-maintenance-removal - AWS will attempt to remove the instance from the domain during the next scheduled maintenance window.
- failed - A configuration problem has prevented the instance from joining the domain. Check and fix your configuration before re-issuing the instance modify command.
- removing - The instance is being removed from the domain.

A request to become a member of a domain can fail because of a network connectivity issue or an incorrect IAM role. If you create a DB instance or modify an existing instance and the attempt to become a member of a domain fails, you should re-issue the modify command or modify the newly created instance to join the domain.

Connecting to SQL Server with Windows Authentication

To connect to SQL Server with Windows Authentication, you must be logged into a domain-joined computer as a domain user. After launching SQL Server Management Studio, choose **Windows Authentication** as the authentication type, as shown following.

Restoring a SQL Server DB Instance and then Adding It to a Domain

You can restore a DB snapshot or do a point-in-time restore for a SQL Server DB instance and then add it to a domain. Once the DB instance is restored, modify the instance using the process explained in the section Step 4: Create or Modify a SQL Server DB Instance (p. 816) to add the DB instance to a domain.

Related Topics

- Microsoft SQL Server on Amazon RDS (p. 720)
- Security in Amazon RDS (p. 326)

MySQL on Amazon RDS

Amazon RDS supports DB instances running several versions of MySQL. You can use the following major versions:

- MySQL 5.7
- MySQL 5.6
- MySQL 5.5

For more information about minor version suport, see MySQL on Amazon RDS Versions (p. 822).

You first use the Amazon RDS management tools or interfaces to create an Amazon RDS MySQL DB instance. You can then use the resizing the DB instance, authorizing connections to the DB instance, creating and restoring from backups or snapshots, creating Multi-AZ secondaries, creating Read Replicas, and monitoring the performance of the DB instance. You use standard MySQL utilities and applications to store and access the data in the DB instance.

Amazon RDS for MySQL is compliant with many industry standards. For example, you can use Amazon RDS for MySQL databases to build HIPAA-compliant applications and to store healthcare related information, including protected health information (PHI) under an executed Business Associate Agreement (BAA) with AWS. Amazon RDS for MySQL also meets Federal Risk and Authorization Management Program (FedRAMP) security requirements and has received a FedRAMP Joint Authorization Board (JAB) Provisional Authority to Operate (P-ATO) at the FedRAMP HIGH Baseline within the AWS GovCloud (US) region. For more information on supported compliance standards, see AWS Cloud Compliance.

Common Management Tasks for MySQL on Amazon RDS

The following are the common management tasks you perform with an Amazon RDS MySQL DB instance, with links to relevant documentation for each task.

Task Area	Relevant Documentation
Understanding Amazon Relational Database Service (Amazon RDS) Understand key Amazon RDS components, including DB instances, regions, Availability Zones, security groups, parameter groups, and option groups.	What Is Amazon Relational Database Service (Amazon RDS)? (p. 1)
Setting up Amazon RDS for first time use Set up Amazon RDS so that you can create MySQL DB instances in Amazon Web Services (AWS).	Setting Up for Amazon RDS (p. 5)
Understanding Amazon RDS DB instances	Amazon RDS DB Instances (p. 90)

Task Area	Relevant Documentation
Create virtual MySQL server instances that run in AWS. Because DB instances are the building blocks of Amazon RDS, we recommend that you understand their principles.	
Creating a DB instance for production Create a DB instance for production purposes. Creating an instance includes choosing a DB instance class with appropriate processing power and memory capacity and choosing a storage type that supports the way you expect to use your database.	DB Instance Class (p. 92) Amazon RDS Storage Types (p. 410) Creating a DB Instance Running the MySQL Database Engine (p. 830)
Managing security for your DB instance By default, DB instances are created with a firewall that prevents access to them. You must create a security group with the correct IP addresses and network configuration to access the DB instance. You can also use AWS Identity and Access Management (IAM) policies to assign permissions that determine who is allowed to manage RDS resources.	Security in Amazon RDS (p. 326) Overview of Managing Access Permissions to Your Amazon RDS Resources (p. 328) Amazon RDS Security Groups (p. 375) Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform (p. 391)
Connecting to your DB instance Connect to your DB instance using a standard SQL client application such as the MySQL command line utility or MySQL Workbench.	Connecting to a DB Instance Running the MySQL Database Engine (p. 840)
Configuring high availability for a production DB instance Provide high availability with synchronous standby replication in a different Availability Zone, automatic failover, fault tolerance for DB instances using Multi-AZ deployments, and Read Replicas.	High Availability (Multi-AZ) (p. 99)
Configuring a DB instance in an Amazon Virtual Private Cloud Configure a virtual private cloud (VPC) in the Amazon VPC service. An Amazon VPC is a virtual network logically isolated from other virtual networks in AWS.	Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform (p. 391) Working with an Amazon RDS DB Instance in a VPC (p. 399)
Configuring specific MySQL database parameters and features Configure specific MySQL database parameters with a parameter group that can be associated with many DB instances. You can also configure specific MySQL database features with an option group that can be associated with many DB instances.	Working with DB Parameter Groups (p. 170) Working with Option Groups (p. 153) Options for MySQL DB Instances (p. 897)

Task Area	Relevant Documentation
Modifying a DB instance running the MySQL database engine Change the settings of a DB instance to accomplish tasks such as adding additional storage or changing the DB instance class.	Modifying a DB Instance Running the MySQL Database Engine (p. 843) Modifying an Amazon RDS DB Instance and Using the Apply Immediately Parameter (p. 114)
Configuring database backup and restore Configure your DB instance to take automated backups. You can also back up and restore your databases manually by using full backup files.	Working With Backups (p. 201) Backing Up and Restoring Amazon RDS DB Instances (p. 200)
Importing and exporting data Import data from other RDS MySQL DB instances, MySQL instances running external to Amazon RDS, and other types of data sources, and export data to MySQL instances running external to Amazon RDS.	Importing Data into an Amazon RDS MySQL DB Instance (p. 860)
Monitoring a MySQL DB instance Monitor your RDS MySQL DB instance by using Amazon CloudWatch RDS metrics, events, and Enhanced Monitoring. View log files for your RDS MySQL DB instance.	Monitoring Amazon RDS (p. 245) Viewing DB Instance Metrics (p. 254) Viewing Amazon RDS Events (p. 301) Amazon RDS Database Log Files (p. 303) MySQL Database Log Files (p. 313)
Replicating your data Create a MySQL Read Replica—optionally, in a different AWS Region—for load balancing, disaster recovery, and processing read-heavy database workloads, such as for analysis and reporting.	Working with PostgreSQL, MySQL, and MariaDB Read Replicas (p. 134) Replication with a MySQL or MariaDB Instance Running External to Amazon RDS (p. 890)

There are also several appendices with useful information about working with Amazon RDS MySQL DB instances:

- Common DBA Tasks for MySQL DB Instances (p. 905)
- Options for MySQL DB Instances (p. 897)
- Appendix: MySQL on Amazon RDS SQL Reference (p. 913)

MySQL on Amazon RDS Versions

For MySQL, version numbers are organized as version = X.Y.Z. In Amazon RDS terminology, X.Y denotes the major version, and Z is the minor version number. For Amazon RDS implementations, a version change is considered major if the major version number changes—for example, going from version 5.6 to 5.7. A version change is considered minor if only the minor version number changes—for example, going from version 5.7.16 to 5.7.19.

Amazon RDS currently supports the following versions of MySQL:

Major Version	Minor Version
MySQL 5.7	• 5.7.19 (supported in all AWS Regions) • 5.7.17 (supported in all AWS Regions) • 5.7.16 (supported in all AWS Regions)
MySQL 5.6	• 5.6.37 (supported in all AWS Regions) • 5.6.35 (supported in all AWS Regions) • 5.6.34 (supported in all AWS Regions) • 5.6.29 (supported in all AWS Regions) • 5.6.27 (supported in all AWS Regions except us-east-2, ca-central-1, eu-west-2)
MySQL 5.5	• 5.5.57 (supported in all AWS Regions) • 5.5.54 (supported in all AWS Regions) • 5.5.53 (supported in all AWS Regions) • 5.5.46 (supported in all AWS Regions)

You can specify any currently supported MySQL version when creating a new DB instance. You can specify the MySQL 5.7, 5.6, or 5.5 major versions, and any supported minor version for the specified major version. If no version is specified, Amazon RDS will default to a supported version, typically the most recent version. If a major version (for example, MySQL 5.7) is specified but a minor version is not, Amazon RDS will default to a recent release of the major version you have specified. To see a list of supported versions, as well as defaults for newly created DB instances, use the DescribeDBEngineVersions API action.

With Amazon RDS, you control when to upgrade your MySQL instance to a new version supported by Amazon RDS. You can maintain compatibility with specific MySQL versions, test new versions with your application before deploying in production, and perform version upgrades at times that best fit your schedule.

Unless you specify otherwise, your DB instance will automatically be upgraded to new MySQL minor versions as they are supported by Amazon RDS. This patching occurs during your scheduled maintenance window, and it is announced on the Amazon RDS Community Forum in advance. To turn off automatic version upgrades, set the AutoMinorVersionUpgrade parameter to “false.”

If you opt out of automatically scheduled upgrades, you can manually upgrade to a supported minor version release by following the same procedure as you would for a major version update. For information, see Upgrading the MySQL DB Engine (p. 851).

Amazon RDS currently supports the major version upgrades from MySQL version 5.5 to version 5.6 and MySQL version 5.6 to version 5.7. Because major version upgrades involve some compatibility risk, they do not occur automatically; you must make a request to modify the DB instance. You should thoroughly test any upgrade before upgrading your production instances. For information about upgrading a DB instance, see Upgrading the MySQL DB Engine (p. 851).

You can test a DB instance against a new version before upgrading by creating a DB snapshot of your existing DB instance, restoring from the DB snapshot to create a new DB instance, and then initiating a version upgrade for the new DB instance. You can then experiment safely on the upgraded clone of your DB instance before deciding whether or not to upgrade your original DB instance.

The Amazon RDS deprecation policy for MySQL includes the following:

- We intend to support major MySQL version releases, including MySQL 5.5, for 3 years after they are initially supported by Amazon RDS.
- We intend to support minor MySQL version releases (for example, MySQL 5.5.46) for at least 1 year after they are initially supported by Amazon RDS.
- After a MySQL major or minor version has been "deprecated," we expect to provide a three month grace period for you to initiate an upgrade to a supported version prior to an automatic upgrade being applied during your scheduled maintenance window.

MySQL Features Not Supported By Amazon RDS

Amazon RDS does not currently support the following MySQL features:

- Global Transaction IDs
- Transportable Table Space
- Authentication Plugin
- Password Strength Plugin
- Replication Filters
- Semi-synchronous Replication

In order to deliver a managed service experience, Amazon RDS does not provide shell access to DB instances, and it restricts access to certain system procedures and tables that require advanced privileges. Amazon RDS supports access to databases on a DB instance using any standard SQL client application. Amazon RDS does not allow direct host access to a DB instance via Telnet, Secure Shell (SSH), or Windows Remote Desktop Connection. When you create a DB instance, you are assigned to the *db_owner* role for all databases on that instance, and you have all database-level permissions except for those used for backups. Amazon RDS manages backups for you.

Supported Storage Engines for MySQL on Amazon RDS

While MySQL supports multiple storage engines with varying capabilities, not all of them are optimized for recovery and data durability. Amazon RDS fully supports the InnoDB storage engine for MySQL DB instances. Amazon RDS features such as Point-In-Time restore and snapshot restore require a recoverable storage engine and are supported for the InnoDB storage engine only. You must be running an instance of MySQL 5.6 or later to use the InnoDB `memcached` interface. For more information, see MySQL `MEMCACHED` Support (p. 901).

The Federated Storage Engine is currently not supported by Amazon RDS for MySQL.

The MyISAM storage engine does not support reliable recovery and can result in lost or corrupt data when MySQL is restarted after a recovery, preventing Point-In-Time restore or snapshot restore from working as intended. However, if you still choose to use MyISAM with Amazon RDS, snapshots can be helpful under some conditions.

If you want to convert existing MyISAM tables to InnoDB tables, you can use the alter table command (for example, alter table TABLE_NAME engine=innodb;). Bear in mind that MyISAM and InnoDB have different strengths and weaknesses, so you should fully evaluate the impact of making this switch on your applications before doing so.

MySQL 5.1 is no longer supported in Amazon RDS. However, you can restore existing MySQL 5.1 snapshots. When you restore a MySQL 5.1 snapshot, the instance is automatically upgraded to MySQL 5.5.

MySQL Security on Amazon RDS

Security for Amazon RDS MySQL DB instances is managed at three levels:

- AWS Identity and Access Management controls who can perform Amazon RDS management actions on DB instances. When you connect to AWS using IAM credentials, your IAM account must have IAM policies that grant the permissions required to perform Amazon RDS management operations. For more information, see Authentication and Access Control for Amazon RDS (p. 327).
- When you create a DB instance, you use either a VPC security group or a DB security group to control which devices and Amazon EC2 instances can open connections to the endpoint and port of the DB instance. These connections can be made using SSL. In addition, firewall rules at your company can control whether devices running at your company can open connections to the DB instance.
- To authenticate login and permissions for a MySQL DB instance, you can take either of the following approaches, or a combination of them.

  You can take the same approach as with a stand-alone instance of MySQL. Commands such as `CREATE USER`, `RENAME USER`, `GRANT`, `REVOKE`, and `SET PASSWORD` work just as they do in on-premises databases, as does directly modifying database schema tables. For information, see MySQL User Account Management in the MySQL documentation.

  You can also use IAM database authentication. With IAM database authentication, you authenticate to your DB instance by using an IAM user or IAM role and an authentication token. An *authentication token* is a unique value that is generated using the Signature Version 4 signing process. By using IAM database authentication, you can use the same credentials to control access to your AWS resources and your databases. For more information, see IAM Database Authentication for MySQL and Amazon Aurora (p. 360).

When you create an Amazon RDS DB instance, the master user has the following default privileges:

- `alter`
- `alter routine`
- `create`
- `create routine`
- `create temporary tables`
- `create user`
- `create view`
- `delete`
- `drop`
- `event`
- `execute`
- `grant option`
- `index`
- `insert`
- `lock tables`
- `process`
- `references`

- `replication client`
- `replication slave (MySQL 5.6 and later)`
- `select`
- `show databases`
- `show view`
- `trigger`
- `update`

> **Note**
> Although it is possible to delete the master user on the DB instance, it is not recommended. To recreate the master user, use the ModifyDBInstance RDS API action or the modify-db-instance AWS CLI command and specify a new master user password with the appropriate parameter. If the master user does not exist in the instance, the master user is created with the specified password.

To provide management services for each DB instance, the `rdsadmin` user is created when the DB instance is created. Attempting to drop, rename, change the password, or change privileges for the `rdsadmin` account will result in an error.

To allow management of the DB instance, the standard `kill` and `kill_query` commands have been restricted. The Amazon RDS commands `rds_kill` and `rds_kill_query` are provided to allow you to terminate user sessions or queries on DB instances.

SSL Support for MySQL DB Instances

Amazon RDS supports SSL connections with DB instances running the MySQL database engine.

> **Note**
> Amazon Aurora is compatible with MySQL. However, you use a different SSL certificate to connect to an Amazon Aurora DB cluster. For information on connecting to Amazon Aurora using SSL, see Securing Aurora Data with SSL (p. 434).

Amazon RDS creates an SSL certificate and installs the certificate on the DB instance when Amazon RDS provisions the instance. These certificates are signed by a certificate authority. The SSL certificate includes the DB instance endpoint as the Common Name (CN) for the SSL certificate to guard against spoofing attacks. The public key is stored at https://s3.amazonaws.com/rds-downloads/rds-combined-ca-bundle.pem.

An SSL certificate created by Amazon RDS is the trusted root entity and should work in most cases but might fail if your application does not accept certificate chains. If your application does not accept certificate chains, you might need to use an intermediate certificate to connect to your region. For example, you must use an intermediate certificate to connect to the AWS GovCloud (US) region using SSL. For a list of regional intermediate certificates that you can download, see Intermediate Certificates (p. 359).

To encrypt connections using the default `mysql` client, launch the mysql client using the `--ssl-ca parameter` to reference the public key, for example:

```
mysql -h myinstance.c9akciq32.rds-us-east-1.amazonaws.com
--ssl-ca=[full path]rds-combined-ca-bundle.pem --ssl-verify-server-cert
```

You can use the GRANT statement to require SSL connections for specific users accounts. For example, you can use the following statement to require SSL connections on the user account encrypted_user:

```
GRANT USAGE ON *.* TO 'encrypted_user'@'%' REQUIRE SSL
```

For more information on SSL connections with MySQL, go to the MySQL documentation.

Using memcached and Other Options with MySQL

Most Amazon RDS DB engines support option groups that allow you to select additional features for your DB instance. DB instances on MySQL version 5.6 and later support the `memcached` option, a simple, key-based cache. For more information about `memcached` and other options, see Options for MySQL DB Instances (p. 897). For more information about working with option groups, see Working with Option Groups (p. 153).

InnoDB Cache Warming

InnoDB cache warming can provide performance gains for your MySQL DB instance by saving the current state of the buffer pool when the DB instance is shut down, and then reloading the buffer pool from the saved information when the DB instance starts up. This bypasses the need for the buffer pool to "warm up" from normal database use and instead preloads the buffer pool with the pages for known common queries. The file that stores the saved buffer pool information only stores metadata for the pages that are in the buffer pool, and not the pages themselves. As a result, the file does not require much storage space. The file size is about 0.2 percent of the cache size. For example, for a 64 GB cache, the cache warming file size is 128 MB. For more information on InnoDB cache warming, go to Saving and Restoring the Buffer Pool State in the MySQL documentation.

MySQL on Amazon RDS supports InnoDB cache warming for MySQL version 5.6 and later. To enable InnoDB cache warming, set the `innodb_buffer_pool_dump_at_shutdown` and `innodb_buffer_pool_load_at_startup` parameters to 1 in the parameter group for your DB instance. Changing these parameter values in a parameter group will affect all MySQL DB instances that use that parameter group. To enable InnoDB cache warming for specific MySQL DB instances, you might need to create a new parameter group for those instances. For information on parameter groups, see Working with DB Parameter Groups (p. 170).

InnoDB cache warming primarily provides a performance benefit for DB instances that use standard storage. If you use PIOPS storage, you do not commonly see a significant performance benefit.

Important
If your MySQL DB instance does not shut down normally, such as during a failover, then the buffer pool state will not be saved to disk. In this case, MySQL loads whatever buffer pool file is available when the DB instance is restarted. No harm is done, but the restored buffer pool might not reflect the most recent state of the buffer pool prior to the restart. To ensure that you have a recent state of the buffer pool available to warm the InnoDB cache on startup, we recommend that you periodically dump the buffer pool "on demand." You can dump or load the buffer pool on demand if your DB instance is running MySQL version 5.6.19 or later.
You can create an event to dump the buffer pool automatically and on a regular interval. For example, the following statement creates an event named `periodic_buffer_pool_dump` that dumps the buffer pool every hour.

```
CREATE EVENT periodic_buffer_pool_dump
ON SCHEDULE EVERY 1 HOUR
DO CALL mysql.rds_innodb_buffer_pool_dump_now();
```

For more information on MySQL events, see Event Syntax in the MySQL documentation.

Dumping and Loading the Buffer Pool on Demand

For MySQL version 5.6.19 and later, you can save and load the InnoDB cache "on demand."

- To dump the current state of the buffer pool to disk, call the mysql.rds_innodb_buffer_pool_dump_now (p. 921) stored procedure.
- To load the saved state of the buffer pool from disk, call the mysql.rds_innodb_buffer_pool_load_now (p. 922) stored procedure.
- To cancel a load operation in progress, call the mysql.rds_innodb_buffer_pool_load_abort (p. 922) stored procedure.

Local Time Zone for MySQL DB Instances

By default, the time zone for an RDS MySQL DB instance is Universal Time Coordinated (UTC). You can set the time zone for your DB instance to the local time zone for your application instead.

Local time zone is supported for MySQL versions 5.5, 5.6, and 5.7 only.

To set the local time zone for a DB instance, set the `time_zone` parameter in the parameter group for your DB instance to one of the supported values listed later in this section. When you set the `time_zone` parameter for a parameter group, all DB instances and Read Replicas that are using that parameter group change to use the new local time zone. For information on setting parameters in a parameter group, see Working with DB Parameter Groups (p. 170).

After you set the local time zone, all new connections to the database reflect the change. If you have any open connections to your database when you change the local time zone, you won't see the local time zone update until after you close the connection and open a new connection.

You can set a different local time zone for a DB instance and one or more of its Read Replicas. To do this, use a different parameter group for the DB instance and the replica or replicas and set the `time_zone` parameter in each parameter group to a different local time zone.

If you are replicating across regions, then the replication master DB instance and the Read Replica use different parameter groups (parameter groups are unique to a region). To use the same local time zone for each instance, you must set the `time_zone` parameter in the instance's and Read Replica's parameter groups.

When you restore a DB instance from a DB snapshot, the local time zone is set to UTC. You can update the time zone to your local time zone after the restore is complete. If you restore a DB instance to a point in time, then the local time zone for the restored DB instance is the time zone setting from the parameter group of the restored DB instance.

You can set your local time zone to one of the following values.

`Africa/Cairo`	`Asia/Bangkok`	`Australia/Darwin`
`Africa/Casablanca`	`Asia/Beirut`	`Australia/Hobart`
`Africa/Harare`	`Asia/Calcutta`	`Australia/Perth`
`Africa/Monrovia`	`Asia/Damascus`	`Australia/Sydney`
`Africa/Nairobi`	`Asia/Dhaka`	`Brazil/East`
`Africa/Tripoli`	`Asia/Irkutsk`	`Canada/Newfoundland`

Africa/Windhoek	Asia/Jerusalem	Canada/Saskatchewan
America/Araguaina	Asia/Kabul	Europe/Amsterdam
America/Asuncion	Asia/Karachi	Europe/Athens
America/Bogota	Asia/Kathmandu	Europe/Dublin
America/Caracas	Asia/Krasnoyarsk	Europe/Helsinki
America/Chihuahua	Asia/Magadan	Europe/Istanbul
America/Cuiaba	Asia/Muscat	Europe/Kaliningrad
America/Denver	Asia/Novosibirsk	Europe/Moscow
America/Fortaleza	Asia/Riyadh	Europe/Paris
America/Guatemala	Asia/Seoul	Europe/Prague
America/Halifax	Asia/Shanghai	Europe/Sarajevo
America/Manaus	Asia/Singapore	Pacific/Auckland
America/Matamoros	Asia/Taipei	Pacific/Fiji
America/Monterrey	Asia/Tehran	Pacific/Guam
America/Montevideo	Asia/Tokyo	Pacific/Honolulu
America/Phoenix	Asia/Ulaanbaatar	Pacific/Samoa
America/Santiago	Asia/Vladivostok	US/Alaska
America/Tijuana	Asia/Yakutsk	US/Central
Asia/Amman	Asia/Yerevan	US/Eastern
Asia/Ashgabat	Atlantic/Azores	US/East-Indiana
Asia/Baghdad	Australia/Adelaide	US/Pacific
Asia/Baku	Australia/Brisbane	UTC

Known Issues and Limitations for MySQL on Amazon RDS

There are some known issues and limitations for working with MySQL on Amazon RDS. For more information, see Known Issues and Limitations for MySQL on Amazon RDS (p. 909).

Creating a DB Instance Running the MySQL Database Engine

The basic building block of Amazon RDS is the DB instance. The DB instance is where you create your MySQL databases.

> **Important**
> You must complete the tasks in the Setting Up for Amazon RDS (p. 5) section before you can create or connect to a DB instance.

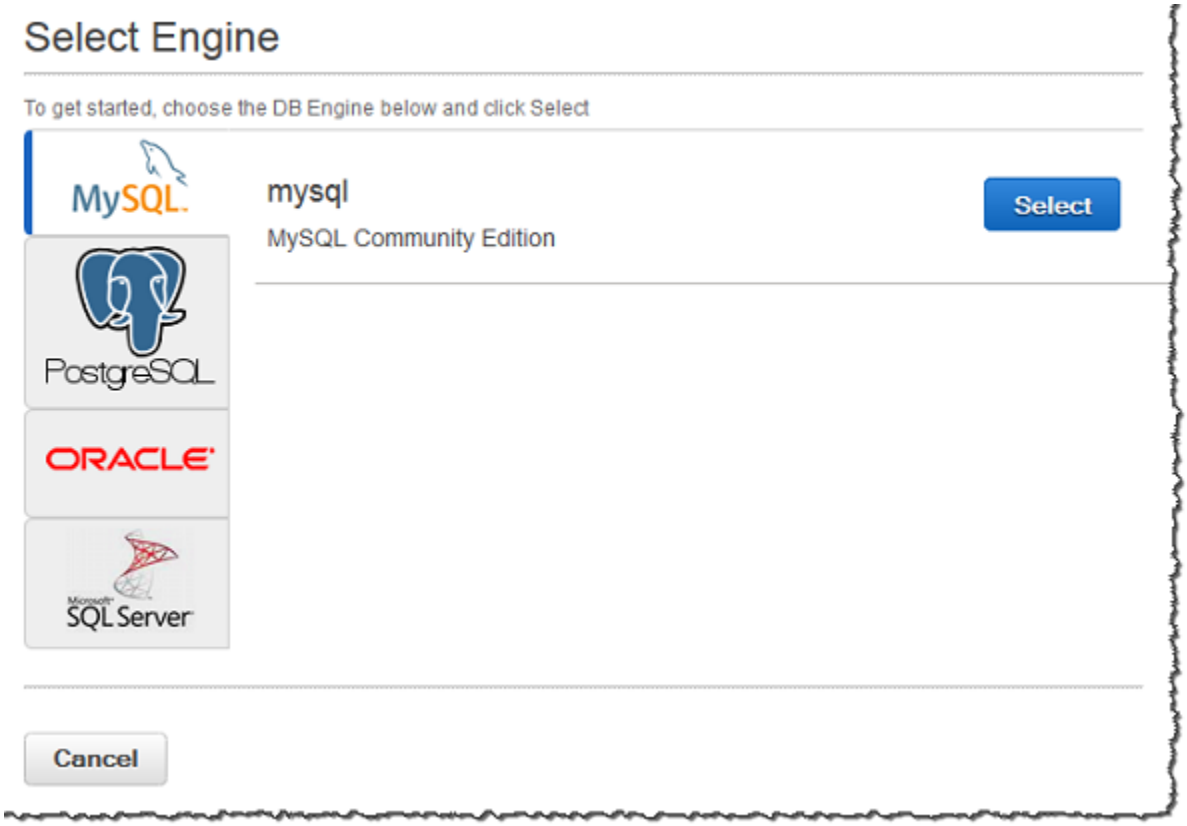
For an example that walks you through the process of creating and connecting to a sample DB instance, see Creating a MySQL DB Instance and Connecting to a Database on a MySQL DB Instance (p. 35).

AWS Management Console

To launch a MySQL DB instance

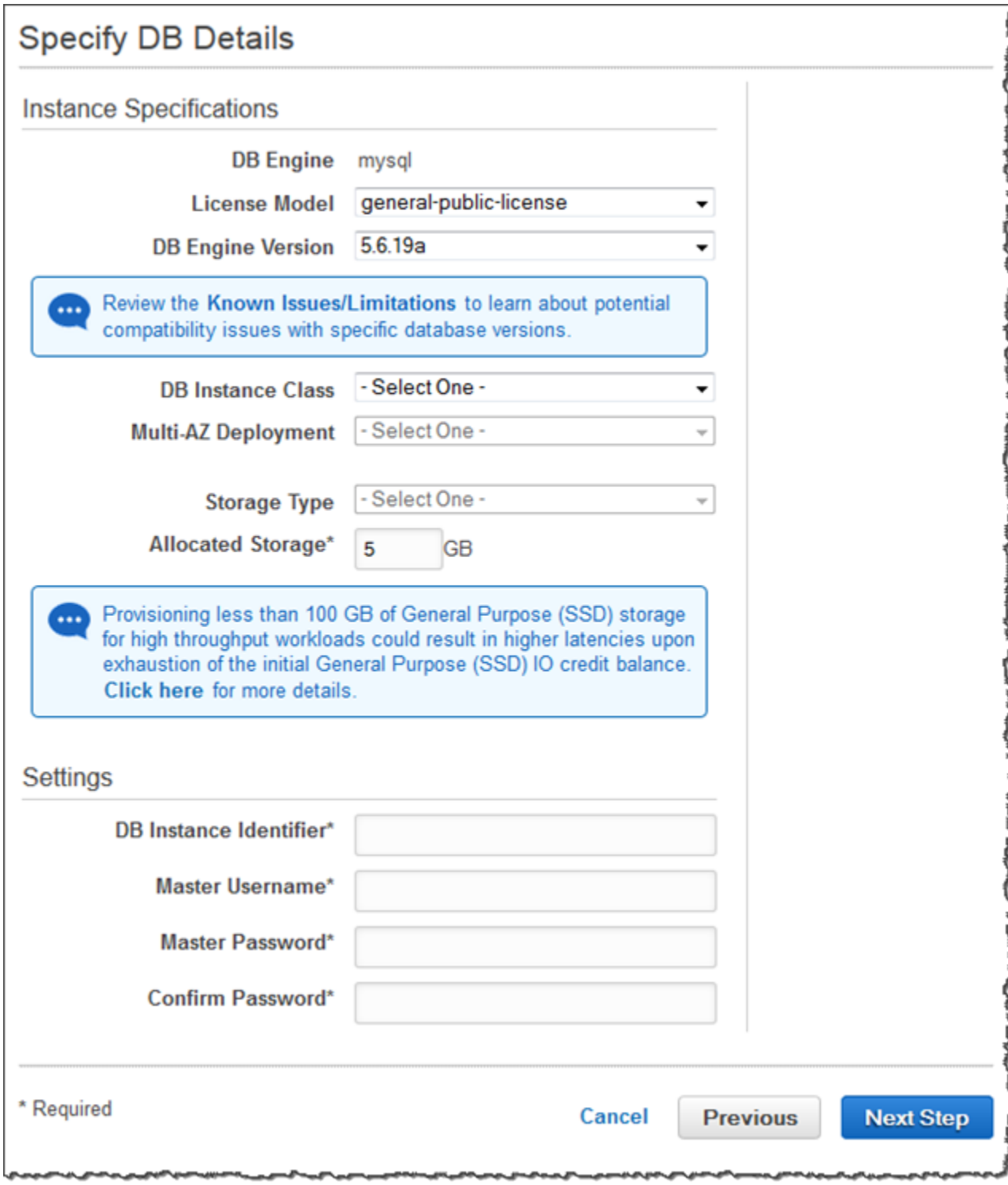
1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the top right corner of the AWS Management Console, choose the region in which you want to create the DB instance.
3. In the navigation pane, choose **Instances**.
4. Choose **Launch DB Instance** to start the **Launch DB Instance Wizard**.

   The wizard opens on the **Select Engine** page.



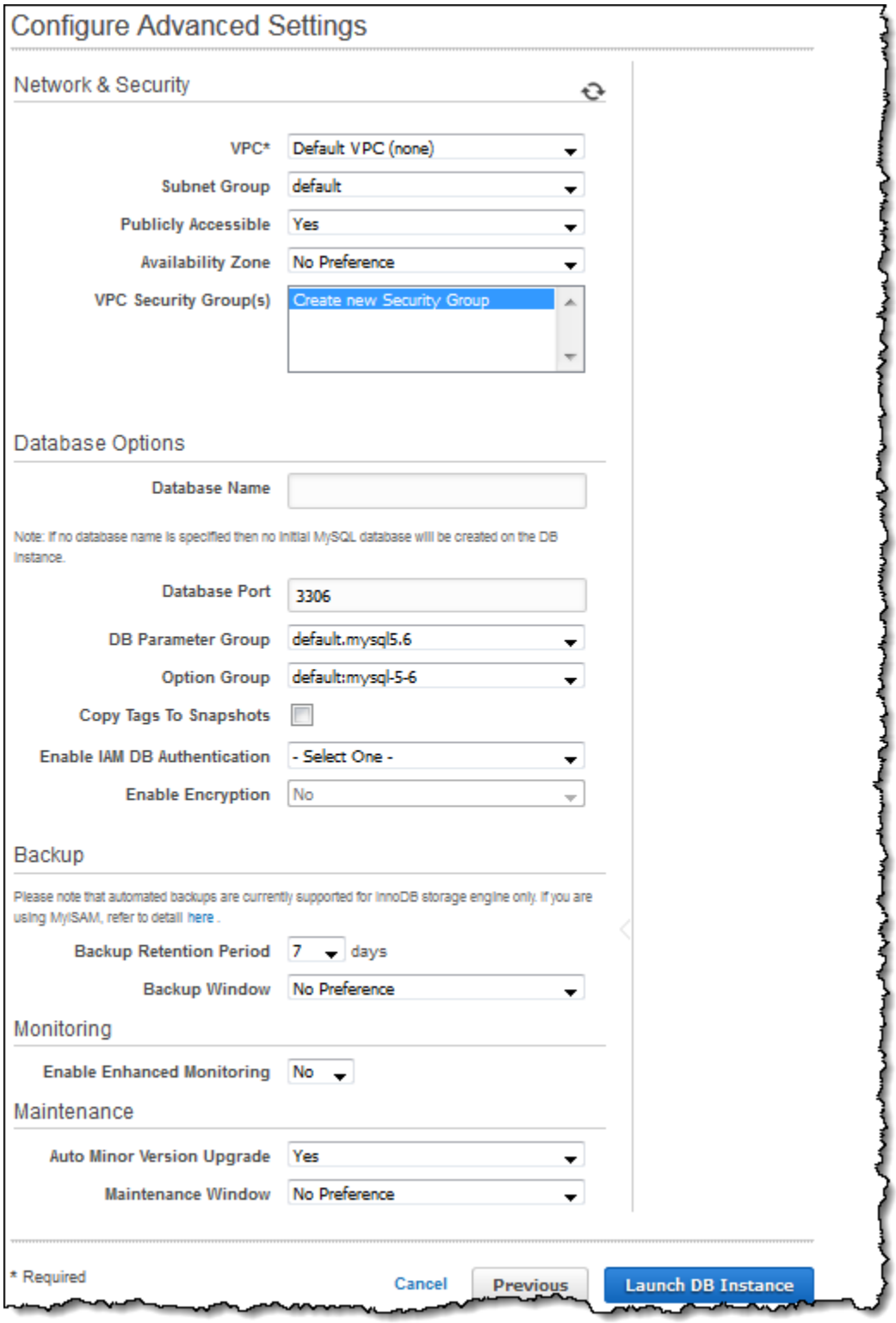
5. In the **Select Engine** window, click the **Select** button for the MySQL DB engine.
6. The **Production?** step asks if you are planning to use the DB instance you are creating for production. If you are, choose **Yes**. If you choose **Yes**, the failover option **Multi-AZ** and the **Provisioned IOPS** storage option are preselected in the following step. We recommend these features for any production environment.
7. Choose **Next** to continue. The **Specify DB Details** page appears.

   On the **Specify DB Details** page, specify your DB instance information. For information about each setting, see Settings for MySQL DB Instances (p. 835).



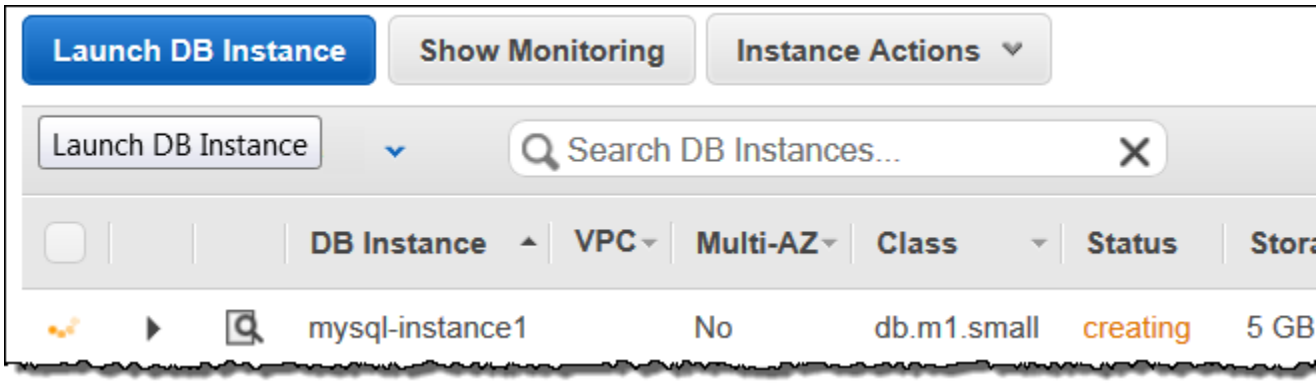
8. Choose **Next** to continue. The **Configure Advanced Settings** page appears.

   On the **Configure Advanced Settings** page, provide additional information that Amazon RDS needs to launch the DB instance. For information about each setting, see Settings for MySQL DB Instances (p. 835).



9. Choose **Launch DB Instance**.
10. On the final page of the wizard, choose **Close**.

On the RDS console, the new DB instance appears in the list of DB instances. The DB instance has a status of **creating** until the DB instance is ready to use. When the state changes to **available**, you can connect to the DB instance. Depending on the DB instance class and the amount of storage, it can take up to 20 minutes before the new instance is available.



CLI

To create a MySQL DB instance by using the AWS CLI, call the create-db-instance command with the parameters below. For information about each setting, see Settings for MySQL DB Instances (p. 835).

- `--db-instance-identifier`
- `--db-instance-class`
- `--db-security-groups`
- `--db-subnet-group`
- `--engine`
- `--master-user-name`
- `--master-user-password`
- `--allocated-storage`
- `--backup-retention-period`

Example

The following example creates a MySQL db instance named mydbinstance.

For Linux, OS X, or Unix:

```
aws rds create-db-instance \
    --db-instance-identifier mydbinstance \
    --db-instance-class db.m1.small \
    --engine MySQL \
    --allocated-storage 20 \
    --master-username masterawsuser \
    --master-user-password masteruserpassword \
    --backup-retention-period 3
```

For Windows:

```
aws rds create-db-instance ^
    --db-instance-identifier mydbinstance ^
    --db-instance-class db.m3.medium ^
```

```
    --engine MySQL ^
    --allocated-storage 20 ^
    --master-username masterawsuser ^
    --master-user-password masteruserpassword ^
    --backup-retention-period 3
```

This command should produce output similar to the following:

```
DBINSTANCE  mydbinstance  db.m3.medium  mysql  20  sa  creating  3  ****  n  5.6.27
SECGROUP  default  active
PARAMGRP  default.mysql5.6  in-sync
```

API

To create a MySQL DB instance by using the Amazon RDS API, call the CreateDBInstance action with the parameters below. For information about each setting, see Settings for MySQL DB Instances (p. 835).

- `AllocatedStorage`
- `BackupRetentionPeriod`
- `DBInstanceClass`
- `DBInstanceIdentifier`
- `DBSecurityGroups`
- `DBSubnetGroup`
- `Engine`
- `MasterUsername`
- `MasterUserPassword`

Example

The following example creates a MySQL db instance named mydbinstance.

```
https://rds.us-west-2.amazonaws.com/
    ?Action=CreateDBInstance
    &AllocatedStorage=20
    &BackupRetentionPeriod=3
    &DBInstanceClass=db.m3.medium
    &DBInstanceIdentifier=mydbinstance
    &DBName=mydatabase
    &DBSecurityGroups.member.1=mysecuritygroup
    &DBSubnetGroup=mydbsubnetgroup
    &Engine=mysql
    &MasterUserPassword=masteruserpassword
    &MasterUsername=masterawsuser
    &Version=2014-10-31
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20140213/us-west-2/rds/aws4_request
    &X-Amz-Date=20140213T162136Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=8052a76dfb18469393c5f0182cdab0ebc224a9c7c5c949155376c1c250fc7ec3
```

Settings for MySQL DB Instances

The following table contains details about settings that you choose when you create a MySQL DB instance.

Setting	Setting Description
Allocated Storage	The amount of storage to allocate for your DB instance (in gigabytes). In some cases, allocating a higher amount of storage for your DB instance than the size of your database can improve I/O performance. For more information, see Storage for Amazon RDS (p. 410).
Auto Minor Version Upgrade	**Yes** to enable your DB instance to receive minor DB engine version upgrades automatically when they become available.
Availability Zone	The availability zone for your DB instance. Use the default value of **No Preference** unless you want to specify an Availability Zone. For more information, see Regions and Availability Zones (p. 97).
Backup Retention Period	The number of days that you want automatic backups of your DB instance to be retained. For any non-trivial DB instance, you should set this value to `1` or greater. For more information, see Working With Backups (p. 201).
Backup Window	The time period during which Amazon RDS automatically takes a backup of your DB instance. Unless you have a specific time that you want to have your database backup, use the default of **No Preference**. For more information, see Working With Backups (p. 201).
Copy Tags To Snapshots	Select this option to copy any DB instance tags to a DB snapshot when you create a snapshot. For more information, see Tagging Amazon RDS Resources (p. 129).
Database Name	The name for the database on your DB instance. The name must contain 1 to 64 alpha-numeric characters. If you do not provide a name, Amazon RDS does not create a database on the DB instance you are creating. To create additional databases on your DB instance, connect to your DB instance and use the SQL command CREATE DATABASE. For more information, see Connecting to a DB Instance Running the MySQL Database Engine (p. 840).
Database Port	The port that you want to access the DB instance through. MySQL installations default to port 3306. If you use a DB security group with your DB instance, this must be the same port value you provided when creating the DB security group. The firewalls at some companies block connections to the default MySQL port. If your company firewall blocks the default port, choose another port for your DB instance.
DB Engine Version	The version of MySQL that you want to use.

Setting	Setting Description
DB Instance Class	The configuration for your DB instance. For example, a **db.m1.small** instance class equates to 1.7 GB memory, 1 ECU (1 virtual core with 1 ECU), 64-bit platform, and moderate I/O capacity. If possible, choose an instance class large enough that a typical query working set can be held in memory. When working sets are held in memory the system can avoid writing to disk, and this improves performance. For more information, see DB Instance Class (p. 92).
DB Instance Identifier	The name for your DB instance. Your DB instance identifier can contain up to 63 alphanumeric characters, and must be unique for your account in the region you chose. You can add some intelligence to the name, such as including the region you chose, for example **`mysql-instance1`**.
DB Parameter Group	A parameter group for your DB instance. You can choose the default parameter group or you can create a custom parameter group. For more information, see Working with DB Parameter Groups (p. 170).
Enable Encryption	**Yes** to enable encryption at rest for this DB instance. For more information, see Encrypting Amazon RDS Resources (p. 355).
Enable Enhanced Monitoring	**Yes** to gather metrics in real time for the operating system that your DB instance runs on. For more information, see Enhanced Monitoring (p. 258).
Enable IAM DB Authentication	**Yes** to enable IAM database authentication for this DB instance. For more information, see IAM Database Authentication for MySQL and Amazon Aurora (p. 360).
License Model	MySQL has only one license model, **`General-Public-License`** the general license agreement for MySQL.
Maintenance Window	The 30 minute window in which pending modifications to your DB instance are applied. If the time period doesn't matter, choose **No Preference**. For more information, see The Amazon RDS Maintenance Window (p. 103).
Master Username	The name that you use as the master user name to log on to your DB Instance. For more information, and a list of the default privileges for the master user, see MySQL Security on Amazon RDS (p. 825).

Setting	Setting Description
Master User Password	The password for your master user account. The password must contain from 8 to 16 printable ASCII characters (excluding /,", a space, and @).
Multi-AZ Deployment	**Yes** to create a standby mirror of your DB instance in another Availability Zone for failover support. We recommend Multi-AZ for production workloads to maintain high availability. For development and testing, you can choose **No**. For more information, see High Availability (Multi-AZ) (p. 99).
Option Group	An option group for your DB instance. You can choose the default option group or you can create a custom option group. For more information, see Working with Option Groups (p. 153).
Publicly Accessible	**Yes** to give your DB instance a public IP address. This means that it is accessible outside the VPC (the DB instance also needs to be in a public subnet in the VPC). Choose **No** if you want the DB instance to only be accessible from inside the VPC. For more information, see Hiding a DB Instance in a VPC from the Internet (p. 401).
Storage Type	The storage type for your DB instance. For more information, see Amazon RDS Storage Types (p. 410).
Subnet Group	This setting depends on the platform you are on. If you are a new customer to AWS, choose **default**, which is the default DB subnet group that was created for your account. If you are creating a DB instance on the previous E2-Classic platform and you want your DB instance in a specific VPC, choose the DB subnet group you created for that VPC.
VPC	This setting depends on the platform you are on. If you are a new customer to AWS, choose the default VPC shown. If you are creating a DB instance on the previous E2-Classic platform that does not use a VPC, choose **Not in VPC**. For more information, see Amazon Virtual Private Cloud (VPCs) and Amazon RDS (p. 390).
VPC Security Group	If you are a new customer to AWS, choose the default VPC. Otherwise, choose the VPC security group you previously created. For more information, see Working with DB Security Groups (EC2-Classic Platform) (p. 380).

Related Topics

- Tutorial: Create an Amazon VPC for Use with an Amazon RDS DB Instance (p. 406)
- Connecting to a DB Instance Running the MySQL Database Engine (p. 840)
- Modifying a DB Instance Running the MySQL Database Engine (p. 843)
- Deleting a DB Instance (p. 126)

Connecting to a DB Instance Running the MySQL Database Engine

Before you can connect to a DB instance running the MySQL database engine, you must create a DB instance. For information, see Creating a DB Instance Running the MySQL Database Engine (p. 830). Once Amazon RDS provisions your DB instance, you can use any standard MySQL client application or utility to connect to the instance. In the connection string, you specify the DNS address from the DB instance endpoint as the host parameter, and specify the port number from the DB instance endpoint as the port parameter.

To authenticate to your RDS DB instance, you can use one of the authentication methods for MySQL and IAM database authentication.

- To learn how to authenticate to MySQL using one of the authentication methods for MySQL, see Authentication Method in the MySQL documentation.
- To learn how to authenticate to MySQL using IAM database authentication, see IAM Database Authentication for MySQL and Amazon Aurora (p. 360).

You can use the AWS Management Console, the AWS CLI describe-db-instances command, or the Amazon RDS API DescribeDBInstances action to list the details of an Amazon RDS DB instance, including its endpoint. If an endpoint value is `myinstance.123456789012.us-east-1.rds.amazonaws.com:3306`, then you would specify the following values in a MySQL connection string:

- For host or host name, specify `myinstance.123456789012.us-east-1.rds.amazonaws.com`
- For port, specify `3306`

You can connect to an Amazon RDS MySQL DB instance by using tools like the MySQL command line utility. For more information on using the MySQL utility, go to mysql - The MySQL Command Line Tool in the MySQL documentation. One GUI-based application you can use to connect is MySQL Workbench. For more information, go to the Download MySQL Workbench page.

Two common causes of connection failures to a new DB instance are:

- The DB instance was created using a security group that does not authorize connections from the device or Amazon EC2 instance where the MySQL application or utility is running. If the DB instance was created in a VPC, it must have a VPC security group that authorizes the connections. If the DB instance was created outside of a VPC, it must have a DB security group that authorizes the connections.
- The DB instance was created using the default port of 3306, and your company has firewall rules blocking connections to that port from devices in your company network. To fix this failure, recreate the instance with a different port.

You can use SSL encryption on connections to an Amazon RDS MySQL DB instance. For information, see SSL Support for MySQL DB Instances (p. 826). If you are using IAM database authentication, you must use an SSL connection. For information, see IAM Database Authentication for MySQL and Amazon Aurora (p. 360).

For information on connecting to an Amazon Aurora DB cluster, see Connecting to an Amazon Aurora DB Cluster (p. 457).

For information on connecting to a MariaDB DB instance, see Connecting to a DB Instance Running the MariaDB Database Engine (p. 688).

Connecting from the MySQL Utility

To connect to a DB instance using the MySQL utility, type the following command at a command prompt to connect to a DB instance using the MySQL utility. For the -h parameter, substitute the DNS name for your DB instance. For the -P parameter, substitute the port for your DB instance. Enter the master user password when prompted.

```
mysql -h myinstance.123456789012.us-east-1.rds.amazonaws.com -P 3306 -u mymasteruser -p
```

You will see output similar to the following.

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 350
Server version: 5.6.27-log MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Connecting with SSL

Amazon RDS creates an SSL certificate for your DB instance when the instance is created. If you enable SSL certificate verification, then the SSL certificate includes the DB instance endpoint as the Common Name (CN) for the SSL certificate to guard against spoofing attacks. To connect to your DB instance using SSL, you can use native password authentication or IAM database authentication. To connect to your DB instance using IAM database authentication, see IAM Database Authentication for MySQL and Amazon Aurora (p. 360). To connect to your DB instance using native password authentication, you can follow these steps:

To connect to a DB instance with SSL using the MySQL utility

1. A root certificate that works for all regions can be downloaded here.
2. Type the following command at a command prompt to connect to a DB instance with SSL using the MySQL utility. For the -h parameter, substitute the DNS name for your DB instance. For the --ssl-ca parameter, substitute the SSL certificate file name as appropriate.

   ```
   mysql -h myinstance.123456789012.us-east-1.rds.amazonaws.com --ssl-ca=rds-ca-2015-
   root.pem
   ```

3. Include the `--ssl-verify-server-cert` parameter so that the SSL connection verifies the DB instance endpoint against the endpoint in the SSL certificate. For example:

   For Linux, OS X, or Unix:

   ```
   mysql \
       -h myinstance.123456789012.us-east-1.rds.amazonaws.com \
       --ssl-ca=rds-ca-2015-root.pem \
       --ssl-verify-server-cert
   ```

   For Windows:

   ```
   mysql ^
       -h myinstance.123456789012.us-east-1.rds.amazonaws.com ^
       --ssl-ca=rds-ca-2015-root.pem ^
       --ssl-verify-server-cert
   ```

4. Enter the master user password when prompted.

You will see output similar to the following.

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 350
Server version: 5.6.27-log MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Maximum MySQL connections

The maximum number of connections allowed to an Amazon RDS MySQL DB instance is based on the amount of memory available for the DB instance class of the DB instance. A DB instance class with more memory available will result in a larger amount of connections available. For more information on DB instance classes, see DB Instance Class (p. 92).

The connection limit for a DB instance is set by default to the maximum for the DB instance class for the DB instance. You can limit the number of concurrent connections to any value up to the maximum number of connections allowed using the `max_connections` parameter in the parameter group for the DB instance. For more information, see Working with DB Parameter Groups (p. 170).

You can retrieve the maximum number of connections allowed for an Amazon RDS MySQL DB instance by executing the following query on your DB instance:

```
SELECT @@max_connections;
```

You can retrieve the number of active connections to an Amazon RDS MySQL DB instance by executing the following query on your DB instance:

```
SHOW STATUS WHERE `variable_name` = 'Threads_connected';
```

Related Topics

- Amazon RDS DB Instances (p. 90)
- Creating a DB Instance Running the MySQL Database Engine (p. 830)
- Amazon RDS Security Groups (p. 375)
- Deleting a DB Instance (p. 126)
- IAM Database Authentication for MySQL and Amazon Aurora (p. 360)

Modifying a DB Instance Running the MySQL Database Engine

You can change the settings of a DB instance to accomplish tasks such as adding additional storage or changing the DB instance class. This topic guides you through modifying an Amazon RDS MySQL DB instance, and describes the settings for MySQL instances.

We recommend that you test any changes on a test instance before modifying a production instance, so that you fully understand the impact of each change. This is especially important when upgrading database versions.

After you modify your DB instance settings, you can apply the changes immediately, or apply them during the next maintenance window for the DB instance. Some modifications cause an interruption by restarting the DB instance.

AWS Management Console

To modify a MySQL DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **DB Instances**, and then select the DB instance that you want to modify.
3. Choose **Instance Actions**, and then choose **Modify**. The **Modify DB Instance** page appears.
4. Change any of the settings that you want. For information about each setting, see Settings for MySQL DB Instances (p. 844).
5. To apply the changes immediately, select **Apply Immediately**. Selecting this option can cause an outage in some cases. For more information, see The Impact of Apply Immediately (p. 114).
6. When all the changes are as you want them, choose **Continue**.
7. On the confirmation page, review your changes. If they are correct, choose **Modify DB Instance** to save your changes.

   Alternatively, choose **Back** to edit your changes, or choose **Cancel** to cancel your changes.

CLI

To modify a MySQL DB instance by using the AWS CLI, call the modify-db-instance command. Specify the DB instance identifier, and the parameters for the settings that you want to modify. For information about each parameter, see Settings for MySQL DB Instances (p. 844).

Example

The following code modifies `mydbinstance` by setting the backup retention period to 1 week (7 days). The code disables automatic minor version upgrades by using `--no-auto-minor-version-upgrade`. To allow automatic minor version upgrades, use `--auto-minor-version-upgrade`. The changes are applied during the next maintenance window by using `--no-apply-immediately`. Use `--apply-immediately` to apply the changes immediately. For more information, see The Impact of Apply Immediately (p. 114).

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --backup-retention-period 7 \
```

```
    --no-auto-minor-version-upgrade \
    --no-apply-immediately
```

For Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --backup-retention-period 7 ^
    --no-auto-minor-version-upgrade ^
    --no-apply-immediately
```

API

To modify a MySQL instance by using the Amazon RDS API, call the ModifyDBInstance action. Specify the DB instance identifier, and the parameters for the settings that you want to modify. For information about each parameter, see Settings for MySQL DB Instances (p. 844).

Example

The following code modifies `mydbinstance` by setting the backup retention period to 1 week (7 days) and disabling automatic minor version upgrades. These changes are applied during the next maintenance window.

```
https://rds.amazonaws.com/
    ?Action=ModifyDBInstance
    &ApplyImmediately=false
    &AutoMinorVersionUpgrade=false
    &BackupRetentionPeriod=7
    &DBInstanceIdentifier=mydbinstance
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-10-31
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20131016/us-west-1/rds/aws4_request
    &X-Amz-Date=20131016T233051Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=087a8eb41cb1ab0fc9ec1575f23e73757ffc6a1e42d7d2b30b9cc0be988cff97
```

Settings for MySQL DB Instances

The following table contains details about which settings you can modify, which settings you can't modify, when the changes can be applied, and whether the changes cause downtime for the DB instance.

Setting	Setting Description	When the Change Occurs	Downtime Notes
Allocated Storage	The storage, in gigabytes, that you want to allocate for your DB instance. For more information, see Storage for Amazon RDS (p. 410).	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	No downtime. Performance may be degraded during the change.
Auto Minor	**Yes** if you want your DB instance to receive minor engine version	–	–

Setting	Setting Description	When the Change Occurs	Downtime Notes
Version Upgrade	upgrades automatically when they become available. Upgrades are installed only during your scheduled maintenance window.		
Backup Retention Period	The number of days that automatic backups are retained. To disable automatic backups, set the backup retention period to 0. For more information, see Working With Backups (p. 201).	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false and you change the setting from a non-zero value to another non-zero value, the change is applied asynchronously, as soon as possible. Otherwise, the change occurs during the next maintenance window.	An outage occurs if you change from 0 to a non-zero value, or from a non-zero value to 0.
Backup Window	The time range during which automated backups of your databases occur. The backup window is a start time in Universal Coordinated Time (UTC), and a duration in hours. For more information, see Working With Backups (p. 201).	The change is applied asynchronously, as soon as possible.	–
Certificate Authority	The certificate that you want to use.	–	–
Copy Tags to Snapshots	If you have any DB instance tags, this option copies them when you create a DB snapshot. For more information, see Tagging Amazon RDS Resources (p. 129).	–	–
Database Port	The port that you want to use to access the database. The port value must not match any of the port values specified for options in the option group for the DB instance.	The change occurs immediately. This setting ignores the **Apply Immediately** setting.	The DB instance is rebooted immediately.

Setting	Setting Description	When the Change Occurs	Downtime Notes
DB Engine Version	The version of the MySQL database engine that you want to use. Before you upgrade your production DB instances, we recommend that you test the upgrade process on a test instance to verify its duration and to validate your applications. For more information, see Upgrading the MySQL DB Engine (p. 851).	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	An outage occurs during this change.
DB Instance Class	The DB instance class that you want to use. For more information, see DB Instance Class (p. 92).	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	An outage occurs during this change.
DB Instance Identifier	The DB instance identifier. This value is stored as a lowercase string. For more information about the effects of renaming a DB instance, see Renaming a DB Instance (p. 116).	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	An outage occurs during this change. The DB instance is rebooted.
DB Parameter Group	The parameter group that you want associated with the DB instance. For more information, see Working with DB Parameter Groups (p. 170).	The parameter group change occurs immediately. However, parameter changes only occur when you reboot the DB instance manually without failover. For more information, see Rebooting a DB Instance (p. 119).	An outage doesn't occur during this change. However, parameter changes only occur when you reboot the DB instance manually without failover.
Enable Enhanced Monitoring	**Yes** to enable gathering metrics in real time for the operating system that your DB instance runs on. For more information, see Enhanced Monitoring (p. 258).	–	–
Enable IAM DB Authentication	**Yes** to enable IAM database authentication for this DB instance. For more information, see IAM Database Authentication for MySQL and Amazon Aurora (p. 360).	–	–

Setting	Setting Description	When the Change Occurs	Downtime Notes
Maintenance Window	The time range during which system maintenance occurs. System maintenance includes upgrades, if applicable. The maintenance window is a start time in Universal Coordinated Time (UTC), and a duration in hours. If you set the window to the current time, there must be at least 30 minutes between the current time and end of the window to ensure any pending changes are applied. For more information, see The Amazon RDS Maintenance Window (p. 103).	The change occurs immediately. This setting ignores the **Apply Immediately** setting.	If there are one or more pending actions that cause an outage, and the maintenance window is changed to include the current time, then those pending actions are applied immediately, and an outage occurs.
Multi-AZ Deployment	**Yes** to deploy your DB instance in multiple Availability Zones; otherwise, **No**. For more information, see Regions and Availability Zones (p. 97).	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	–
New Master Password	The password for your master user. The password must contain from 8 to 41 alphanumeric characters.	The change is applied asynchronously, as soon as possible. This setting ignores the **Apply Immediately** setting.	–
Option Group	The option group that you want associated with the DB instance. For more information, see Working with Option Groups (p. 153).	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	–
Publicly Accessible	**Yes** to give the DB instance a public IP address, meaning that it is accessible outside the VPC. To be publicly accessible, the DB instance also has to be in a public subnet in the VPC. **No** to make the DB instance accessible only from inside the VPC. For more information, see Hiding a DB Instance in a VPC from the Internet (p. 401).	The change occurs immediately. This setting ignores the **Apply Immediately** setting.	–

Setting	Setting Description	When the Change Occurs	Downtime Notes
Security Group	The security group you want associated with the DB instance. For more information, see Working with DB Security Groups (EC2-Classic Platform) (p. 380).	The change is applied asynchronously, as soon as possible. This setting ignores the **Apply Immediately** setting.	–

Setting	Setting Description	When the Change Occurs	Downtime Notes
Storage Type	The storage type that you want to use. For more information, see Amazon RDS Storage Types (p. 410).	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	The following changes all result in a brief outage while the process starts. After that, you can use your database normally while the change takes place. • From **General Purpose (SSD)** to **Magnetic**. • From **General Purpose (SSD)** to **Provisioned IOPS (SSD)**, if the DB instance is single-AZ or if you are using a custom parameter group and the DB instance is a read replica. There is no outage for a multi-AZ DB instance or for the source DB instance of a read replica. • From **Magnetic** to **General Purpose (SSD)**. • From **Magnetic** to **Provisioned IOPS (SSD)**. • From **Provisioned IOPS (SSD)** to **Magnetic**. • From **Provisioned IOPS (SSD)** to **General Purpose (SSD)**, if the DB instance is single-AZ or if you are using a custom parameter group and the DB instance is a read replica. There is no outage for a multi-AZ

Setting	Setting Description	When the Change Occurs	Downtime Notes
			DB instance or for the source DB instance of a read replica.
Subnet Group	The subnet group for the DB instance. You can use this setting to move your DB instance to a different VPC. If your DB instance is not in a VPC, you can use this setting to move your DB instance into a VPC. For more information, see Moving a DB Instance Not in a VPC into a VPC (p. 405).	–	–

Related Topics

- Rebooting a DB Instance (p. 119)
- Connecting to a DB Instance Running the MySQL Database Engine (p. 840)
- Upgrading the MySQL DB Engine (p. 851)
- Deleting a DB Instance (p. 126)

Upgrading the MySQL DB Engine

When Amazon Relational Database Service (Amazon RDS) supports a new version of a database engine, you can upgrade your DB instances to the new version. There are two kinds of upgrades: major version upgrades and minor version upgrades.

Overview of Upgrading

Amazon RDS takes two DB snapshots during the upgrade process. The first DB snapshot is of the DB instance before any upgrade changes have been made. If the upgrade doesn't work for your databases, you can restore this snapshot to create a DB instance running the old version. The second DB snapshot is taken when the upgrade completes.

After the upgrade is complete, you can't revert to the previous version of the database engine. If you want to return to the previous version, restore the first DB snapshot taken to create a new DB instance.

You control when to upgrade your DB instance to a new version supported by Amazon RDS. This level of control helps you maintain compatibility with specific database versions and test new versions with your application before deploying in production. When you are ready, you can perform version upgrades at the times that best fit your schedule.

If your DB instance is using read replication, you must upgrade all of the Read Replicas before upgrading the source instance.

If your DB instance is in a Multi-AZ deployment, both the primary and standby replicas are upgraded. The primary and standby DB instances are upgraded at the same time and you will experience an outage until the upgrade is complete. The time for the outage varies based on the size of your DB instance.

Major Version Upgrades for MySQL

Amazon RDS supports the following in-place upgrades for major versions of the MySQL database engine:

- MySQL 5.5 to MySQL 5.6
- MySQL 5.6 to MySQL 5.7

> **Note**
> You can only create MySQL version 5.7 DB instances with current generation DB instance classes and the M3 previous generation DB instance class. If you want to upgrade a MySQL version 5.6 DB instance running on a previous generation DB instance class (other than M3) to a MySQL version 5.7 DB instance, you must first modify the DB instance to use a current generation DB instance class. After the DB instance has been modified to use a current generation DB instance class, you can then modify the DB instance to use the MySQL version 5.7 database engine. For information on Amazon RDS DB instance classes, see DB Instance Class (p. 92).

Major version upgrades can contain database changes that are not backward-compatible with existing applications. As a result, Amazon Relational Database Service (Amazon RDS) doesn't apply major version upgrades automatically; you must manually modify your DB instance. You should thoroughly test any upgrade before applying it to your production instances.

To perform a major version upgrade for a MySQL version 5.5 DB instance on Amazon RDS to MySQL version 5.6 or later, you should first perform any available OS updates. After OS updates are complete, you must upgrade to each major version: 5.5 to 5.6, and then 5.6 to 5.7. MySQL DB instances created before April 24, 2014, show an available OS update until the update has been applied. For more information on OS updates, see Updating the Operating System for a DB Instance or DB Cluster (p. 108).

During a major version upgrade of MySQL, Amazon RDS runs the MySQL binary `mysql_upgrade` to upgrade tables, if required. Also, Amazon RDS empties the `slow_log` and `general_log` tables during a major version upgrade. To preserve log information, save the log contents before the major version upgrade.

MySQL major version upgrades typically complete in about 10 minutes. Some upgrades might take longer because of the DB instance class size or because the instance doesn't follow certain operational guidelines in Best Practices for Amazon RDS (p. 80). If you upgrade a DB instance from the Amazon RDS console, the status of the DB instance indicates when the upgrade is complete. If you upgrade using the AWS Command Line Interface (AWS CLI), use the describe-db-instances command and check the `Status` value.

Upgrades to MySQL Version 5.7 Might Be Slow

MySQL version 5.6.4 introduced a new date and time format for the `datetime`, `time`, and `timestamp` columns that allows fractional components in date and time values. When upgrading a DB instance to MySQL version 5.7, MySQL will force the conversion of all date and time column types to the new format. Because this conversion rebuilds your tables, it might take a considerable amount of time to complete the DB instance upgrade. The forced conversion will occur for any DB instances that are running a version prior to MySQL version 5.6.4, and also any DB instances that were upgraded from a version prior to MySQL version 5.6.4 to a version other than 5.7.

If your DB instance is running a version prior to MySQL version 5.6.4, or was upgraded from a version prior to MySQL version 5.6.4, then we recommend that you convert the `datetime`, `time`, and `timestamp` columns in your database before upgrading your DB instance to MySQL version 5.7. This conversion can significantly reduce the amount of time required to upgrade the DB instance to MySQL version 5.7. To upgrade your date and time columns to the new format, issue the `ALTER TABLE <table_name> FORCE;` command for each table that contains date or time columns. Because altering a table locks the table as read-only, we recommend that you perform this update during a maintenance window.

You can use the following query to find all tables in your database that have columns of type datetime, time, or timestamp and to create an `ALTER TABLE <table_name> FORCE;` command for each table:

```
SELECT DISTINCT CONCAT('ALTER TABLE `',
      REPLACE(is_tables.TABLE_SCHEMA, '`', '``'), '`.`',
      REPLACE(is_tables.TABLE_NAME, '`', '``'), '` FORCE;')
    FROM information_schema.TABLES is_tables
      INNER JOIN information_schema.COLUMNS col ON col.TABLE_SCHEMA =
 is_tables.TABLE_SCHEMA
        AND col.TABLE_NAME = is_tables.TABLE_NAME
      LEFT OUTER JOIN information_schema.INNODB_SYS_TABLES systables ON
        SUBSTRING_INDEX(systables.NAME, '#', 1) =
 CONCAT(is_tables.TABLE_SCHEMA,'/',is_tables.TABLE_NAME)
      LEFT OUTER JOIN information_schema.INNODB_SYS_COLUMNS syscolumns ON
        syscolumns.TABLE_ID = systables.TABLE_ID AND syscolumns.NAME = col.COLUMN_NAME
    WHERE col.COLUMN_TYPE IN ('time','timestamp','datetime')
      AND is_tables.TABLE_TYPE = 'BASE TABLE'
      AND is_tables.TABLE_SCHEMA NOT IN ('mysql','information_schema','performance_schema')
      AND (is_tables.ENGINE = 'InnoDB' AND syscolumns.MTYPE = 6);
```

Minor Version Upgrades for MySQL

Minor version upgrades only occur automatically if a minor upgrade replaces an unsafe version, such as a minor upgrade that contains bug fixes for a previous version. In all other cases, you must modify the DB instance manually to perform a minor version upgrade.

We don't automatically upgrade an Amazon RDS DB instance until we post an announcement to the forums announcement page and send a customer e-mail notification. Even though upgrades take place

during the instance maintenance window, we still schedule them at specific times through the year. We schedule them so you can plan around them, because downtime is required to upgrade a DB engine version, even for Multi-AZ instances.

Testing an Upgrade

Before you perform a major version upgrade on your DB instance, you should thoroughly test your database, and all applications that access the database, for compatibility with the new version. We recommend that you use the following procedure.

To test a major version upgrade

1. Review the upgrade documentation for the new version of the database engine to see if there are compatibility issues that might affect your database or applications:
   - MySQL 5.5 Upgrade Documentation
   - MySQL 5.6 Upgrade Documentation
2. If your DB instance is a member of a custom DB parameter group, you need to create a new DB parameter group with your existing settings that is compatible with the new major version. Specify the new DB parameter group when you upgrade your test instance, so that your upgrade testing ensures that it works correctly. For more information about creating a DB parameter group, see Working with DB Parameter Groups (p. 170).
3. Create a DB snapshot of the DB instance to be upgraded. For more information, see Creating a DB Snapshot (p. 207).
4. Restore the DB snapshot to create a new test DB instance. For more information, see Restoring from a DB Snapshot (p. 209).
5. Modify this new test DB instance to upgrade it to the new version, using one of the methods detailed following. If you created a new parameter group in step 2, specify that parameter group.
6. Evaluate the storage used by the upgraded instance to determine if the upgrade requires additional storage.
7. Run as many of your quality assurance tests against the upgraded DB instance as needed to ensure that your database and application work correctly with the new version. Implement any new tests needed to evaluate the impact of any compatibility issues you identified in step 1. Test all stored procedures and functions. Direct test versions of your applications to the upgraded DB instance.
8. If all tests pass, then perform the upgrade on your production DB instance. We recommend that you do not allow write operations to the DB instance until you confirm that everything is working correctly.

Upgrading a MySQL Database with Reduced Downtime

If your MySQL DB instance is currently in use with a production application, you can use the following procedure to upgrade the database version for your DB instance and reduce the amount of downtime for your application. This procedure shows an example of upgrading from MySQL version 5.5 to MySQL version 5.6.

To upgrade an MySQL database while a DB instance is in use

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. Create a Read Replica of your MySQL 5.5 DB instance. This process creates an upgradable copy of your database.

a. On the console, choose **Instances**, and then choose the DB instance that you want to upgrade.
b. Choose **Instance Actions**, and then choose **Create Read Replica**.
c. Provide a value for **DB Instance Identifier** for your Read Replica and ensure that the DB instance **Class** and other settings match your MySQL 5.5 DB instance.
d. Choose **Yes, Create Read Replica**.

3. When the Read Replica has been created and **Status** shows **available**, upgrade the Read Replica to MySQL 5.6.

   a. On the console, choose **Instances**, and then choose the Read Replica that you just created.
   b. Choose **Instance Actions**, and then choose **Modify**.
   c. For **DB Engine Version**, choose the MySQL 5.6 version to upgrade to, and then choose **Apply Immediately**. Choose **Continue**.
   d. Choose **Modify DB Instance** to start the upgrade.

4. When the upgrade is complete and **Status** shows `available`, verify that the upgraded Read Replica is up to date with the master MySQL 5.5 DB instance. You can do this by connecting to the Read Replica and issuing the `SHOW SLAVE STATUS` command. If the `Seconds_Behind_Master` field is `0`, then replication is up to date.
5. Make your MySQL 5.6 Read Replica a master DB instance.

   Important
   When you promote your MySQL 5.6 Read Replica to a standalone, single-AZ DB instance, it will no longer be a replication slave to your MySQL 5.5 DB instance. We recommend that you promote your MySQL 5.6 Read Replica during a maintenance window when your source MySQL 5.5 DB instance is in read-only mode and all write operations are suspended. When the promotion is completed, you can direct your write operations to the upgraded MySQL 5.6 DB instance to ensure that no write operations are lost.
   In addition, we recommend that before promoting your MySQL 5.6 Read Replica you perform all necessary data definition language (DDL) operations, such as creating indexes, on the MySQL 5.6 Read Replica. This approach avoids negative effects on the performance of the MySQL 5.6 Read Replica after it has been promoted. To promote a Read Replica, use this procedure:

   a. On the console, choose **Instances**, and then choose the Read Replica that you just upgraded.
   b. Choose **Instance Actions**, and then choose **Promote Read Replica**.
   c. Enable automated backups for the Read Replica instance. For more information, see Working With Backups (p. 201).

      Choose **Continue**.
   d. Choose **Yes, Promote Read Replica**.

6. You now have an upgraded version of your MySQL database. At this point, you can direct your applications to the new MySQL 5.6 DB instance, add Read Replicas, set up Multi-AZ support, and so on.

AWS Management Console

To upgrade the engine version of a DB instance by using the AWS Management Console

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Instances**.
3. Choose the check box for the DB instance that you want to upgrade.
4. Choose **Instance Actions**, and then choose **Modify**.

5. For **DB Engine Version**, choose the new version.
6. To upgrade immediately, select **Apply Immediately**. To delay the upgrade to the next maintenance window, clear **Apply Immediately**.
7. Choose **Continue**.
8. Review the modification summary information. To proceed with the upgrade, choose **Modify DB Instance**. To cancel the upgrade, choose **Cancel** or **Back**.

CLI

To upgrade the engine version of a DB instance, use the AWS CLI modify-db-instance command. Specify the following parameters:

- `--db-instance-identifier` – the name of the db instance.
- `--engine-version` – the version number of the database engine to upgrade to.
- `--allow-major-version-upgrade` – to to upgrade major version.
- `--no-apply-immediately` – apply changes during the next maintenance window. To apply changes immediately, use `--apply-immediately`.

Example

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier <mydbinstance> \
    --engine-version <new_version> \
    --allow-major-version-upgrade \
    --apply-immediately
```

For Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier <mydbinstance> ^
    --engine-version <new_version> ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

API

To upgrade the engine version of a DB instance, use the ModifyDBInstance action. Specify the following parameters:

- `DBInstanceIdentifier` – the name of the db instance, for example *`mydbinstance`*.
- `EngineVersion` – the version number of the database engine to upgrade to.
- `AllowMajorVersionUpgrade` – set to `true` to upgrade major version.
- `ApplyImmediately` – whether to apply changes immediately or during the next maintenance window. To apply changes immediately, set the value to *`true`*. To apply changes during the next maintenance window, set the value to *`false`*.

Example

```
https://rds.us-east-1.amazonaws.com/
```

```
?Action=ModifyDBInstance
&ApplyImmediately=false
&DBInstanceIdentifier=mydbinstance
&EngineVersion=new_version
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&Version=2013-09-09
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20131016/us-east-1/rds/aws4_request
&X-Amz-Date=20131016T233051Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=087a8eb41cb1ab5f99e81575f23e73757ffc6a1e42d7d2b30b9cc0be988cff97
```

Related Topics

- Amazon RDS Maintenance (p. 102)
- Updating the Operating System for a DB Instance or DB Cluster (p. 108)

Upgrading a MySQL DB Snapshot

With Amazon RDS, you can create a storage volume DB snapshot of your MySQL DB instance. When you create a DB snapshot, the snapshot is based on the engine version used by your Amazon RDS instance. In addition to upgrading the DB engine version of your DB instance, you can also upgrade the engine version for your DB snapshots. For example, you can upgrade DB snapshots created from the MySQL 5.1 engine to DB snapshots for the MySQL 5.5 engine. After restoring a DB snapshot upgraded to a new engine version, you should test that the upgrade was successful. To learn how to test a major version upgrade, see Testing an Upgrade (p. 853). To learn how to restore a DB snapshot, see Restoring from a DB Snapshot (p. 209).

Amazon RDS supports upgrading a MySQL DB snapshot from MySQL 5.1 to MySQL 5.5.

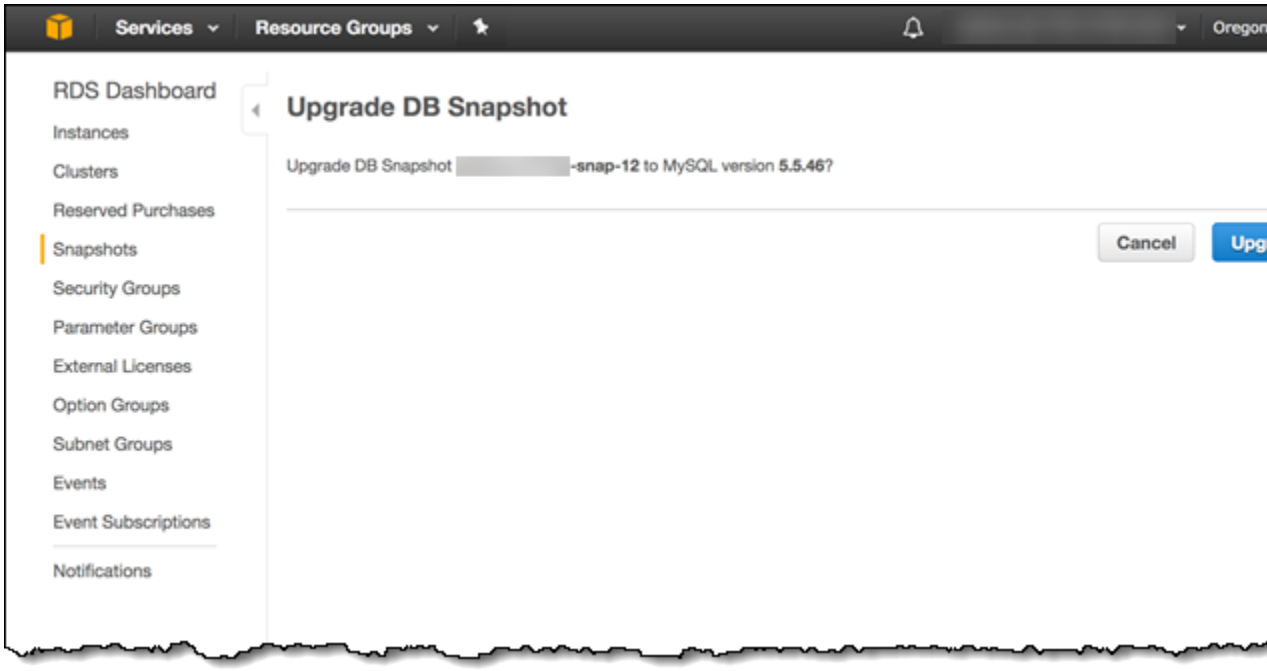
Upgrading a MySQL DB Snapshot

You can upgrade manual DB snapshots, which can be encrypted or not encrypted, from MySQL 5.1 to MySQL 5.5 within the same region. You can't upgrade automated DB snapshots that are created during the automated backup process.

AWS Management Console

To upgrade a DB snapshot

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Snapshots**.
3. Choose **Upgrade Snapshot**. During the upgrade process, all snapshot actions except **Upgrade Snapshot** are disabled. Also, the DB snapshot status changes from **available** to **upgrading**, and then changes to **active** upon completion. If the DB snapshot can't be upgraded because of snapshot corruption issues, the status changes to **unavailable**. You can't recover the snapshot from this state.



AWS CLI

To upgrade a DB snapshot to a new database engine version, use the AWS CLI modify-db-snapshot command.

Parameters

- `--db-snapshot-identifier` – The identifier of the DB snapshot to upgrade. The identifier must be a unique Amazon Resource Name (ARN). For more information, see Working with Amazon Resource Names (ARNs) in Amazon RDS (p. 184).
- `--engine-version` – The engine version to upgrade the DB snapshot to.

Example

For Linux, OS X, or Unix:

```
aws rds modify-db-snapshot \
    --db-snapshot-identifier <mydbsnapshot> \
    --engine-version <new_version>
```

For Windows:

```
aws rds modify-db-snapshot ^
    --db-snapshot-identifier <mydbsnapshot> ^
    --engine-version <new_version>
```

API

To upgrade a DB snapshot to a new database engine version, call the Amazon RDS API ModifyDBSnapshot action.

- `DBSnapshotIdentifier` – The identifier of the DB snapshot to upgrade. The identifier must be a unique Amazon Resource Name (ARN). For more information, see Working with Amazon Resource Names (ARNs) in Amazon RDS (p. 184).
- `EngineVersion` – The engine version to upgrade the DB snapshot to.

Example

```
https://rds.us-west-2.amazonaws.com/
   ?Action=ModifyDBSnapshot
   &DBSnapshotIdentifier=mydbsnapshot
   &EngineVersion=newversion
   &SignatureVersion=4
   &Version=2014-10-31
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20161222/us-west-1/rds/aws4_request
   &X-Amz-Date=20161222T233051Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
   &X-Amz-Signature=8052a76dfb18469393c5f0182cdab0ebc224a9c7c5c949155376c1c250fc7ec3
```

Related Topics

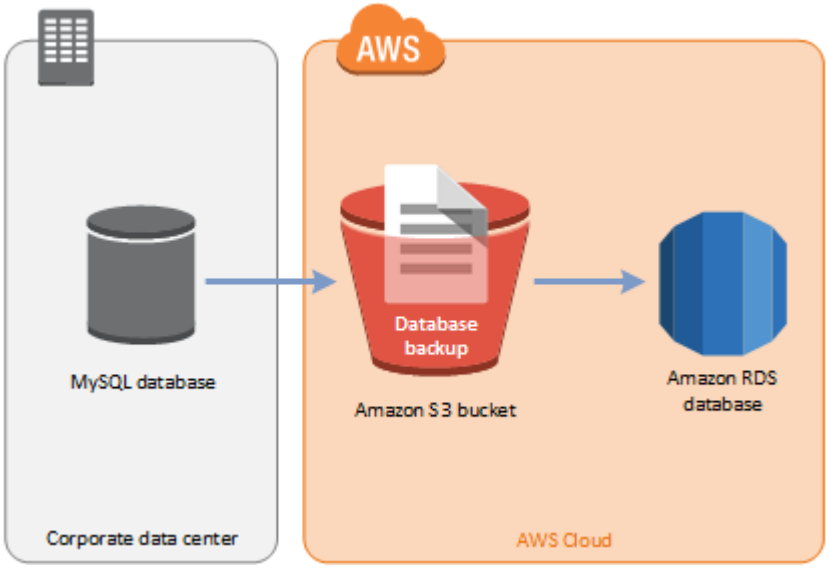
- Testing an Upgrade (p. 853)

- Restoring from a DB Snapshot (p. 209)

Importing Data into an Amazon RDS MySQL DB Instance

Amazon Relational Database Service (Amazon RDS) supports importing MySQL databases by using backup files. You can create a backup of your on-premises database, store it on Amazon Simple Storage Service (Amazon S3), and then restore the backup file onto a new Amazon RDS DB instance running MySQL.

The following diagram shows the supported scenario.



Importing backup files from Amazon S3 is supported for MySQL version 5.6. Importing backup files from Amazon S3 is available in all AWS Regions.

We recommend that you import your database to Amazon RDS by using backup files if your database can be offline while the backup file is created, copied, and restored. If your on-premises database can't be offline, you can use binlog replication to update your database after you have migrated to Amazon RDS through Amazon S3 as explained in this topic. For more information, see Replication with a MySQL or MariaDB Instance Running External to Amazon RDS (p. 890). You can also use the AWS Database Migration Service to migrate your database to Amazon RDS. For more information, see What Is AWS Database Migration Service?

Limitations and Recommendations for Importing Backup Files from Amazon S3 to Amazon RDS

The following are some limitations and recommendations for importing backup files from Amazon S3:

- You can only import your data to a new DB instance, not an existing DB instance.
- You must use Percona XtraBackup to create the backup of your on-premises database.
- You can't migrate from a source database that has tables defined outside of the default MySQL data directory.
- You can't import a MySQL 5.5 database.
- You can't import an on-premises MySQL 5.6 database to an Amazon RDS MySQL 5.7 database. You can upgrade your DB instance after you complete the import.
- You can't restore databases larger than 6 TB in size.

- You can't restore from an encrypted source database, but you can restore to an encrypted Amazon RDS DB instance.
- Your Amazon S3 bucket can't be encrypted.
- You can't restore from an Amazon S3 bucket in a different AWS Region than your Amazon RDS DB instance.
- Importing from Amazon S3 is not supported on the db.t2.micro DB instance class. However, you can restore to a different DB instance class, and then change the instance class later. For more information about instance classes, see Specifications for All Available DB Instance Classes (p. 92).
- Amazon S3 limits the size of a file uploaded to an Amazon S3 bucket to 5 TB. If a backup file exceeds 5 TB, then you must split the backup file into smaller files.
- Amazon RDS limits the number of files uploaded to an Amazon S3 bucket to 1 million. If the backup data for your database, including all full and incremental backups, exceeds 1 million files, use a tarball (.tar.gz) file to store full and incremental backup files in the Amazon S3 bucket.
- User accounts are not imported automatically. Save your user accounts from your source database and add them to your new DB instance later.
- Functions are not imported automatically. Save your functions from your source database and add them to your new DB instance later.
- Stored procedures are not imported automatically. Save your stored procedures from your source database and add them to your new DB instance later.
- Time zone information is not imported automatically. Record the time zone information for your source database, and set the time zone of your new DB instance later. For more information, see Local Time Zone for MySQL DB Instances (p. 828).

Overview of Setting Up to Import Backup Files from Amazon S3 to Amazon RDS

These are the components you need to set up to import backup files from Amazon S3 to Amazon RDS:

- An Amazon S3 bucket to store your backup files.
- A backup of your on-premises database created by Percona XtraBackup.
- An AWS Identity and Access Management (IAM) role to allow Amazon RDS to access the bucket.

If you already have an Amazon S3 bucket, you can use that. If you don't have an Amazon S3 bucket, you can create a new one. Your Amazon S3 bucket can't be encrypted. If you want to create a new bucket, see Creating a Bucket.

Use the Percona XtraBackup tool to create your backup. For more information, see Creating Your Database Backup (p. 861).

If you already have an IAM role, you can use that. If you don't have an IAM role, you can create a new one manually. Alternatively, you can choose to have a new IAM role created for you in your account by the wizard when you restore the database by using the AWS Management Console. If you want to create a new IAM role manually, or attach trust and permissions policies to an existing IAM role, see Creating an IAM Role Manually (p. 863). If you want to have a new IAM role created for you, follow the procedure in AWS Management Console (p. 864)

Creating Your Database Backup

Use the Percona XtraBackup software to create your backup. Amazon RDS supports backup files created with the following versions of the Percona XtraBackup software:

- For MySQL 5.6, use Percona XtraBackup version 2.3.

We recommend that if you don't already have Percona XtraBackup installed, you use the latest version of the software available. You can download Percona XtraBackup from the Percona website.

You can create a full backup of your MySQL database files using Percona XtraBackup. Alternatively, if you already use Percona XtraBackup to back up your MySQL database files, you can upload your existing full and incremental backup directories and files.

For more information about backing up your database with Percona XtraBackup, see Percona XtraBackup - Documentation and The innobackupex Script on the Percona website.

Creating a Full Backup With Percona XtraBackup

To create a full backup of your MySQL database files that can be restored from Amazon S3, use the Percona XtraBackup utility (`innobackupex`) to back up your database.

For example, the following command creates a backup of a MySQL database and stores the files in the folder `/on-premises/s3-restore/backup` folder.

```
innobackupex --user=<myuser> --password=<password> --no-timestamp /on-premises/s3-restore/
backup
```

If you want to compress your backup into a single file (which can be split later, if needed), you can save your backup in one of the following formats:

- Gzip (.gz)
- tar (.tar)
- Percona xbstream (.xbstream)

The following command creates a backup of your MySQL database split into multiple Gzip files.

```
innobackupex --user=<myuser> --password=<password> --stream=tar \
   /on-premises/s3-restore/backup | gzip - | split -d --bytes=500MB \
   - /on-premises/s3-restore/backup/backup.tar.gz
```

The following command creates a backup of your MySQL database split into multiple tar files.

```
innobackupex --user=<myuser> --password=<password> --stream=tar \
   /on-premises/s3-restore/backup | split -d --bytes=500MB \
   - /on-premises/s3-restore/backup/backup.tar
```

The following command creates a backup of your MySQL database split into multiple xbstream files.

```
innobackupex --stream=xbstream  \
   /on-premises/s3-restore/backup | split -d --bytes=500MB \
   - /on-premises/s3-restore/backup/backup.xbstream
```

Using Incremental Backups With Percona XtraBackup

If you already use Percona XtraBackup to perform full and incremental backups of your MySQL database files, you don't need to create a full backup and upload the backup files to Amazon S3. Instead, you can save a significant amount of time by copying your existing backup directories and files to your Amazon S3 bucket. For more information about creating incremental backups using Percona XtraBackup, see Incremental Backups with innobackupex.

When copying your existing full and incremental backup files to an Amazon S3 bucket, you must recursively copy the contents of the base directory. Those contents include the full backup and also all

incremental backup directories and files. This copy must preserve the directory structure in the Amazon S3 bucket. Amazon RDS iterates through all files and directories. Amazon RDS uses the `xtrabackup_checkpoints` file that is included with each incremental backup to identify the base directory, and to order incremental backups by log sequence number (LSN) range.

Backup Considerations for Percona XtraBackup

Amazon RDS consumes your backup files based on the file name. Name your backup files with the appropriate file extension based on the file format—for example, `.xbstream` for files stored using the Percona xbstream format.

Amazon RDS consumes your backup files in alphabetical order and also in natural number order. Use the `split` option when you issue the `innobackupex` command to ensure that your backup files are written and named in the proper order.

Amazon RDS doesn't support partial backups created using Percona XtraBackup. You can't use the `--include`, `--tables-file`, or `--databases` options to create a partial backup when you backup the source files for your database.

Amazon RDS supports incremental backups created using Percona XtraBackup with or without the `--no-timestamp` option. We recommend that you use the `--no-timestamp` option to reduce the depth of the directory structure for your incremental backup.

Creating an IAM Role Manually

If you don't have an IAM role, you can create a new one manually. Alternatively, you can choose to have a new IAM role created for you by the wizard when you restore the database by using the AWS Management Console. If you want to have a new IAM role created for you, follow the procedure in AWS Management Console (p. 864)

If you want to manually create a new IAM role to use to import your database from Amazon S3, you create a role to delegate permissions from the Amazon RDS service to your Amazon S3 bucket. When you create an IAM role, you attach trust and permissions policies. To import your backup files from Amazon S3, use trust and permissions policies similar to the examples following. For more information about creating the role, see Creating a Role to Delegate Permissions to an AWS Service.

Alternatively, you can choose to have a new IAM role created for you by the wizard when you restore the database by using the AWS Management Console. If you want to have a new IAM role created for you, follow the procedure in AWS Management Console (p. 864)

The trust and permissions policies require that you provide an Amazon Resource Name (ARN). For more information about ARN formatting, see Amazon Resource Names (ARNs) and AWS Service Namespaces.

Example Trust Policy for Importing from Amazon S3

```
{
    "Version": "2012-10-17",
    "Statement":
    [{
        "Effect": "Allow",
        "Principal": {"Service":  "rds.amazonaws.com"},
        "Action": "sts:AssumeRole"
    }]
}
```

Example Permissions Policy for Importing from Amazon S3 — IAM User Permissions

```
{
```

```
    "Version":"2012-10-17",
    "Statement":
    [
        {
            "Sid":"AllowS3AccessRole",
            "Effect":"Allow",
            "Action":"iam:PassRole",
            "Resource":"arn:aws:iam::IAM User ID:role/S3Access"
        }
    ]
}
```

Example Permissions Policy for Importing from Amazon S3 — Role Permissions

```
{
    "Version": "2012-10-17",
    "Statement":
    [
        {
        "Effect": "Allow",
        "Action":
            [
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
        "Resource": "arn:aws:s3:::bucket_name"
        },
        {
        "Effect": "Allow",
        "Action":
            [
                "s3:GetObject"
            ],
        "Resource": "arn:aws:s3:::bucket_name/prefix*"
        }
    ]
}
```

Note
If you include a file name prefix, include the asterisk (*) after the prefix. If you don't want to specify a prefix, specify only an asterisk.

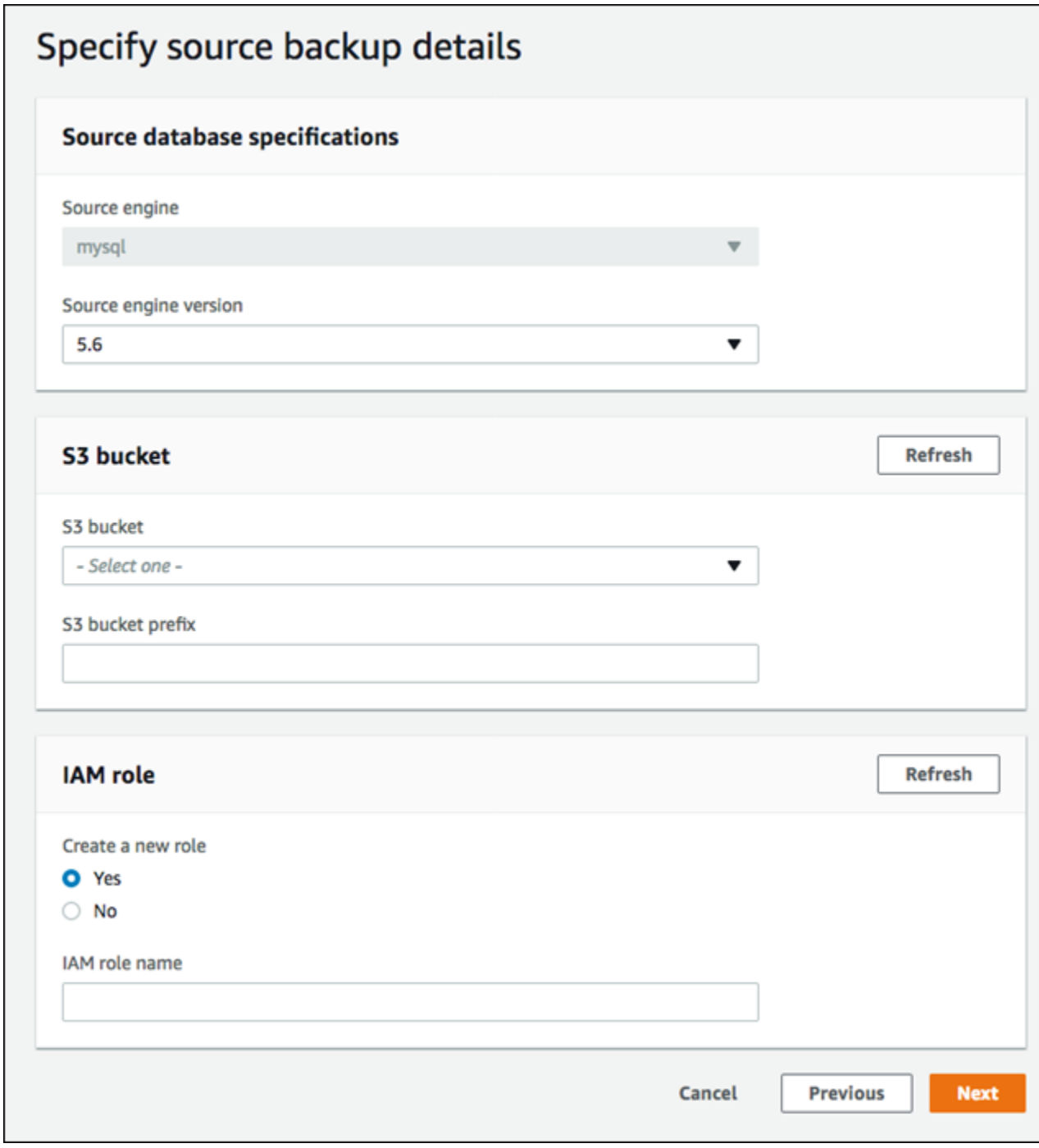
AWS Management Console

To import data from Amazon S3 to a new MySQL DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the top right corner of the Amazon RDS console, choose the AWS Region in which to create your DB instance. Choose the same AWS Region as the Amazon S3 bucket that contains your database backup.
3. In the navigation pane, choose **Instances**.
4. Choose **Restore from S3** to launch the wizard.

   The wizard opens on the **Select Engine** page.
5. On the **Select Engine** page, choose the MySQL icon, and then choose the **Select** button.

   The **Specify source backup details** page appears.



6. On the **Specify source backup details** page, specify your backup information.

   a. For **Source engine**, choose **mysql**.
   b. For **Source engine version**, choose the MySQL version of your source database.
   c. For **S3 bucket**, choose your Amazon S3 bucket.
   d. (Optional) For **S3 bucket prefix**, type a file path prefix for the files stored in your Amazon S3 bucket. If you don't specify a prefix, then RDS creates your DB instance using all of the files and folders in the root folder of the S3 bucket. If you do specify a prefix, then RDS creates your DB instance using the files and folders in the S3 bucket where the path for the file

begins with the specified prefix. For example, suppose that you store your backup files on S3 in a sub-folder named backups, and you have multiple sets of backup files, each in its own directory (gzip_backup1, gzip_backup2, and so on.) In this case, you specify a prefix of backups/gzip_backup1 to restore from the files in the gzip_backup1 folder.

e. For **Create a new role**, choose **Yes** to have a new IAM role created for you in your account, or choose **No** to select an existing IAM role.

f. For **IAM Role**, select an existing IAM role, or specify the name for a new IAM Role. You can choose to have a new IAM role created for you by choosing **Yes** for **Create a New Role**.

7. Choose **Next** to continue. The **Specify DB Details** page appears.

   On the **Specify DB Details** page, specify your DB instance information. For information about each setting, see Settings for MySQL DB Instances (p. 835).

   Note
   Be sure to allocate enough memory for your new DB instance so that the restore can succeed. You can also allocate additional memory for future growth.

8. Choose **Next** to continue. The **Configure Advanced Settings** page appears.

   On the **Configure Advanced Settings** page, provide additional information that Amazon RDS needs to launch the DB instance. For information about each setting, see Settings for MySQL DB Instances (p. 835).

9. Choose **Launch DB Instance**.

CLI

To import data from Amazon S3 to a new MySQL DB instance by using the AWS CLI, call the restore-db-instance-from-s3 command with the parameters following. For information about each setting, see Settings for MySQL DB Instances (p. 835).

Note
Be sure to allocate enough memory for your new DB instance so that the restore can succeed. You can also allocate additional memory for future growth.

- `--allocated-storage`
- `--db-instance-identifier`
- `--db-instance-class`
- `--engine`
- `--master-user-name`
- `--master-user-password`
- `--s3-bucket-name`
- `--s3-ingestion-role-arn`
- `--s3-prefix`
- `--source-engine`
- `--source-engine-version`

Example

For Linux, OS X, or Unix:

```
aws rds restore-db-instance-from-s3 \
--allocated-storage 250 \
--db-instance-identifier myidentifier \
--db-instance-class db.m4.large \
```

```
--engine mysql \
--master-user-name masterawsuser \
--master-user-password masteruserpassword \
--s3-bucket-name mybucket \
--s3-ingestion-role-arn arn:aws:iam::account-number:role/rolename \
--s3-prefix bucketprefix \
--source-engine mysql \
--source-engine-version 5.6.27
```

For Windows:

```
aws rds restore-db-instance-from-s3 ^
--allocated-storage 250 ^
--db-instance-identifier myidentifier ^
--db-instance-class db.m4.large ^
--engine mysql ^
--master-user-name masterawsuser ^
--master-user-password masteruserpassword ^
--s3-bucket-name mybucket ^
--s3-ingestion-role-arn arn:aws:iam::account-number:role/rolename ^
--s3-prefix bucketprefix ^
--source-engine mysql ^
--source-engine-version 5.6.27
```

API

To import data from Amazon S3 to a new MySQL DB instance by using the Amazon RDS API, call the RestoreDBInstanceFromS3 action.

Related Topics

- Importing Data into a MySQL DB Instance by Using Other Methods (p. 868)
- Backing Up and Restoring Amazon RDS DB Instances (p. 200)

Importing Data into a MySQL DB Instance by Using Other Methods

If your scenario supports it, it's easier to move data in and out of Amazon RDS by using backup files and Amazon S3. For more information, see Importing Data into an Amazon RDS MySQL DB Instance (p. 860).

Following, you can find information about alternate methods to import your MySQL data to an Amazon RDS DB instance running MySQL.

We recommend using the procedures in this section to import data into or export it from a MySQL DB instance. You can use these procedures to import data from other MySQL DB instances, MySQL instances running external to Amazon RDS, and other types of data sources. To use replication to export data to an instance of MySQL that is running external to Amazon RDS, we recommend using the procedure discussed in Exporting Data from a MySQL DB Instance by Using Replication (p. 893)

Overview

We recommend the following procedures for importing data into a MySQL DB instance in the situations described:

- You might be able to use the AWS Database Migration Service to migrate your data in the most efficient way. AWS DMS can migrate databases with minimal downtime and, for many database engines, continue ongoing replication until you are ready to switch over to your MySQL DB instance. You can use AWS DMS to migrate from a non-MySQL database engine to an Amazon RDS MySQL DB instance, or to do a partial migration of a MySQL database. If you are migrating to MySQL from a different database engine, you can use the AWS Schema Conversion Tool to migrate schema objects that are not migrated by AWS DMS. For more information about AWS DMS, see What is AWS Database Migration Service.

  We recommend that you do not use AWS DMS and instead use the MySQL database migration tools if all of following conditions are met:
  - You have a homogeneous migration, where you are migrating from a MySQL database to an Amazon RDS MySQL DB instance.
  - You are migrating an entire database.

    AWS DMS is a good option if you are migrating a subset of the data from your MySQL database to Amazon RDS. However, when migrating an entire database, AWS DMS creates tables, primary keys, and in some cases unique indexes, but it doesn't create any other objects that are not required to efficiently migrate the data from the source. For example, it doesn't create secondary indexes, non-primary key constraints, or data defaults. If you are migrating your full database, you can copy your schema to your RDS MySQL DB instance and then use AWS DMS to migrate your data, or use the native MySQL migration tools discussed later in this topic.
  - Using the MySQL database migration tools reduces the amount of downtime required to migrate your database. For example, see Importing Data to an Amazon RDS MySQL or MariaDB DB Instance with Reduced Downtime (p. 873).
- To import data from an existing database in a MySQL DB instance, you can create a Read Replica, and then promote the Read Replica. For more information, see Working with PostgreSQL, MySQL, and MariaDB Read Replicas (p. 134).
- To move small amounts of MySQL data, or where service interruption on the source MySQL database isn't an issue, you can use a simple procedure to copy the data directly to your Amazon RDS MySQL DB instance using a command-line utility. For more information, see Importing Data from a MySQL or MariaDB DB to an Amazon RDS MySQL or MariaDB DB Instance (p. 872).
- To move large amounts of MySQL data, or when you want to minimize service interruption for live sites or applications that use an external MySQL instance, you can back up the data, copy it to Amazon Elastic Compute Cloud (Amazon EC2), and import it into an Amazon RDS MySQL DB instance. You

can then use replication to bring the two instances into sync for any data that has been added to the source system since the copy to Amazon EC2. For more information Importing Data to an Amazon RDS MySQL or MariaDB DB Instance with Reduced Downtime (p. 873).

- For data in sources other than an existing MySQL database, you can create flat files and import them using the `mysqlimport` utility. For more information, see Importing Data From Any Source to a MySQL or MariaDB DB Instance (p. 886).
- To set up replication using an existing MySQL DB instance as the replication master, see Replication with a MySQL or MariaDB Instance Running External to Amazon RDS (p. 890).

Note
The 'mysql' system database contains authentication and authorization information required to log in to your DB instance and access your data. Dropping, altering, renaming, or truncating tables, data, or other contents of the 'mysql' database in your DB instance can result in error and may render the DB instance and your data inaccessible. If this occurs, the DB instance can be restored from a snapshot using the AWS CLI `restore-db-instance-from-db-snapshot` command, or recovered using the AWS CLI `restore-db-instance-to-point-in-time` command.

Importing Data Considerations

This section contains additional technical information related to loading data into MySQL. It is intended for advanced users who are familiar with the MySQL server architecture. Note that all comments related to LOAD DATA LOCAL INFILE apply to `mysqlimport` as well.

Binary Log

Data loads incur a performance penalty and require additional free disk space (up to 4X more) when binary logging is enabled versus loading the same data with binary logging turned off. The severity of the performance penalty and the amount of free disk space required is directly proportional to the size of the transactions used to load the data.

Transaction Size

Transaction size plays an important role in MySQL data loads. It has a major influence on resource consumption, disk space utilization, resume process, time to recover, and input format (flat files or SQL). This section describes how transaction size affects binary logging and makes the case for disabling binary logging during large data loads. As noted earlier, binary logging is enabled and disabled by setting the Amazon RDS automated backup retention period. Non-zero values enable binary logging, and zero disables it. We also describe the impact of large transactions on InnoDB and why it's important to keep transaction sizes small.

Small Transactions

For small transactions, binary logging doubles the number of disk writes required to load the data. Depending upon the upload rate, other database activity taking place during the load, and the capacity of your Amazon RDS DB instance, this can severely degrade performance for other database sessions and increase the time required to load the data.

The binary logs also consume disk space roughly equal to the amount of data loaded until they are backed up and removed. Fortunately, Amazon RDS minimizes this by backing up and removing binary logs on a frequent basis.

Large Transactions

Large transactions incur a 3X penalty for IOPS and disk consumption with binary logging enabled. This is due to the binary log cache spilling to disk, consuming disk space and incurring additional IO for each write. The cache cannot be written to the binlog until the transaction commits or rolls back, so it

consumes disk space in proportion to the amount of data loaded. When the transaction commits, the cache must be copied to the binlog, creating a third copy of the data on disk.

Because of this, there must be at least three times as much free disk space available to load the data compared to loading with binary logging disabled. For example, 10GB of data loaded as a single transaction will consume at least 30GB disk space during the load: 10GB for the table + 10GB for the binary log cache + 10GB for the binary log itself. The cache file remains on disk until the session that created it terminates or the session fills its binary log cache again during another transaction. The binary log must remain on disk until backed up, so it may be some time before the extra 20GB is freed.

If the data was loaded using LOAD DATA LOCAL INFILE, yet another copy of the data is created if the database has to be recovered from a backup made prior to the load. During recovery, MySQL extracts the data from the binary log into a flat file and then executes LOAD DATA LOCAL INFILE, just as the original transaction, only this time the input file is local to the database server. Continuing with the example above, recovery will fail unless there is at least 40GB free disk space available.

Disable Binary Logging

Whenever possible, disable binary logging during large data loads to avoid the resource overhead and addition disk space requirements. In Amazon RDS, disabling binary logging is as simple as setting the backup retention period to zero. If you do this, it's recommended that you take a DB snapshot of the database instance immediately before the load so that you can quickly and easily undo changes made during loading if the need arises.

After the load, set the backup retention period back to an appropriate (no zero) value.

You cannot set the backup retention period to zero if the DB instance is a source DB instance for Read Replicas.

InnoDB

The information in this section provides a strong argument for keeping transaction sizes small when using InnoDB.

Undo

InnoDB generates undo to support features such as transaction rollback and MVCC. Undo is stored in the InnoDB system tablespace (usually ibdata1) and is retained until removed by the purge thread. The purge thread cannot advance beyond the undo of the oldest active transaction, so it is effectively blocked until the transaction commits or completes a rollback. If the database is processing other transactions during the load, their undo also accumulates in the system tablespace and cannot be removed even if they commit and no other transaction needs the undo for MVCC. In this situation, all transactions (including read-only transactions) that access any of the rows changed by any transaction (not just the load transaction) slow down as they scan through undo that could have been purged if not for the long running load transaction.

Since undo is stored in the system tablespace and since the system tablespace never shrinks in size, large data load transactions can cause the system tablespace to become quite large, consuming disk space that cannot be reclaimed without recreating the database from scratch.

Rollback

InnoDB is optimized for commits. Rolling back a large transaction can take a very, very long time. In some cases, it may be faster to perform a point-in-time recovery or restore a DB snapshot.

Input Data Format

MySQL can accept incoming data in one of two forms: flat files and SQL. This section points out some key advantages and disadvantages of each.

Flat Files

Loading flat files with LOAD DATA LOCAL INFILE can be the fastest and least costly method of loading data as long as transactions are kept relatively small. Compared to loading the same data with SQL, flat files usually require less network traffic, lowering transmission costs and load much faster due to the reduced overhead in the database.

One Big Transaction

LOAD DATA LOCAL INFILE loads the entire flat file as one transaction. This isn't necessarily a bad thing. If the size of the individual files can be kept small, this has a number of advantages:

- Resume Capability - Keeping track of which files have been loaded is easy. If a problem arises during the load, you can pick up where you left off with little effort. Some data may have to be retransmitted to Amazon RDS, but with small files, the amount retransmitted is minimal.
- Load data in parallel - If you've got IOPs and network bandwidth to spare with a single file load, loading in parallel may save time.
- Throttle the load rate - Data load impacting other processes? Throttle the load by increasing the interval between files.

Be Careful

The advantages of LOAD DATA LOCAL INFILE diminish rapidly as transaction size increases. If breaking up a large set of data into smaller ones isn't an option, SQL may be the better choice.

SQL

SQL has one main advantage over flat files: it's easy to keep transaction sizes small. However, SQL can take significantly longer to load than flat files and it can be difficult to determine where to resume the load after a failure. For example, mysqldump files are not restartable. If a failure occurs while loading a mysqldump file, the file will require modification or replacement before the load can resume. The alternative is to restore to the point in time prior to the load and replay the file once the cause of the failure has been corrected.

Take Checkpoints Using Amazon RDS Snapshots

If you have a load that's going to take several hours or even days, loading without binary logging isn't a very attractive prospect unless you can take periodic checkpoints. This is where the Amazon RDS DB snapshot feature comes in very handy. A DB snapshot creates a point-in-time consistent copy of your database instance which can be used restore the database to that point in time after a crash or other mishap.

To create a checkpoint, simply take a DB snapshot. Any previous DB snapshots taken for checkpoints can be removed without affecting durability or restore time.

Snapshots are fast too, so frequent checkpointing doesn't add significantly to load time.

Decreasing Load Time

Here are some additional tips to reduce load times:

- Create all secondary indexes prior to loading. This is counter-intuitive for those familiar with other databases. Adding or modifying a secondary index causes MySQL to create a new table with the index changes, copy the data from the existing table to the new table, and drop the original table.
- Load data in PK order. This is particularly helpful for InnoDB tables where load times can be reduced by 75-80% and data file size cut in half.

- Disable foreign key constraints foreign_key_checks=0 For flat files loaded with LOAD DATA LOCAL INFILE, this is required in many cases. For any load, disabling FK checks will provide significant performance gains. Just be sure to enable the constraints and verify the data after the load.
- Load in parallel unless already near a resource limit. Use partitioned tables when appropriate.
- Use multi-value inserts when loading with SQL to minimize statement execution overhead. When using mysqldump, this is done automatically.
- Reduce InnoDB log IO innodb_flush_log_at_trx_commit=0

Note
Using innodb_flush_log_at_trx_commit=0 causes InnoDB to flush its logs every second instead of at each commit. This provides a significant speed advantage, but can lead to data loss during a crash. Use with caution.

Importing Data from a MySQL or MariaDB DB to an Amazon RDS MySQL or MariaDB DB Instance

If your scenario supports it, it is easier to move data in and out of Amazon RDS by using backup files and Amazon S3. For more information, see Importing Data into an Amazon RDS MySQL DB Instance (p. 860).

You can also import data from an existing MySQL or MariaDB database to an Amazon RDS MySQL or MariaDB DB instance by copying the database with mysqldump and piping it directly into the Amazon RDS MySQL or MariaDB DB instance. The `mysqldump` command-line utility is commonly used to make backups and transfer data from one MySQL or MariaDB server to another. It is included with MySQL and MariaDB client software.

A typical `mysqldump` command to move data from an external database to an Amazon RDS DB instance looks similar to the following:

```
mysqldump -u <local_user> \
    --databases <database_name> \
    --single-transaction \
    --compress \
    --order-by-primary  \
    -p<local_password> | mysql -u <RDS_user> \
        --port=<port_number> \
        --host=<host_name> \
        -p<RDS_password>
```

Important
Make sure not to leave a space between the `-p` option and the entered password.

The parameters used are as follows:

- `-u <local_user>` – Use to specify a user name. In the first usage of this parameter, you specify the name of a user account on the local MySQL or MariaDB database identified by the `--databases` parameter.
- `--databases <database_name>` – Use to specify the name of the database on the local MySQL or MariaDB instance that you want to import into Amazon RDS.
- `--single-transaction` – Use to ensure that all of the data loaded from the local database is consistent with a single point in time. If there are other processes changing the data while `mysqldump` is reading it, using this option helps maintain data integrity.
- `--compress` – Use to reduce network bandwidth consumption by compressing the data from the local database before sending it to Amazon RDS.
- `--order-by-primary` – Use to reduce load time by sorting each table's data by its primary key.

- `-p<local_password>` – Use to specify a password. In the first usage of this parameter, you specify the password for the user account identified by the first `-u` parameter.
- `-u <RDS_user>` – Use to specify a user name. In the second usage of this parameter, you specify the name of a user account on the default database for the Amazon RDS MySQL or MariaDB DB instance identified by the `--host` parameter.
- `--port <port_number>` – Use to specify the port for your Amazon RDS MySQL or MariaDB DB instance. By default, this is 3306 unless you changed the value when creating the instance.
- `--host <host_name>` – Use to specify the DNS name from the Amazon RDS DB instance endpoint, for example, `myinstance.123456789012.us-east-1.rds.amazonaws.com`. You can find the endpoint value in the instance details in the Amazon RDS Management Console.
- `-p<RDS_password>` – Use to specify a password. In the second usage of this parameter, you specify the password for the user account identified by the second `-u` parameter.

You must create any stored procedures, triggers, functions, or events manually in your Amazon RDS database. If you have any of these objects in the database that you are copying, then exclude them when you run `mysqldump` by including the following parameters with your `mysqldump` command: `--routines=0 --triggers=0 --events=0`.

The following example copies the `world` sample database on the local host to an Amazon RDS MySQL DB instance.

For Linux, OS X, or Unix:

```
sudo mysqldump -u localuser \
    --databases world \
    --single-transaction \
    --compress \
    --order-by-primary  \
    -plocalpassword | mysql -u rdsuser
        --port=3306 \
        --host=myinstance.123456789012.us-east-1.rds.amazonaws.com \
        -prdspassword
```

For Windows, the following command needs to be run in a command prompt that has been opened by right-clicking **Command Prompt** on the Windows programs menu and choosing **Run as administrator**:

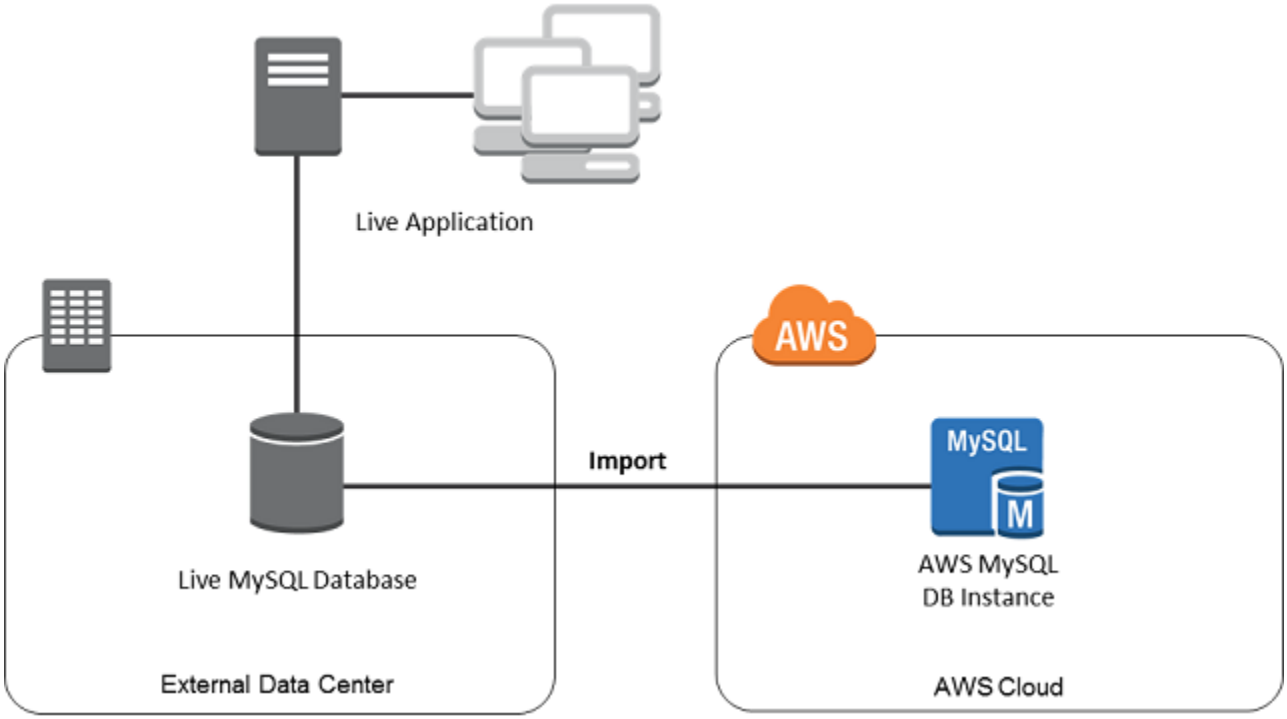
```
mysqldump -u localuser ^
    --databases world ^
    --single-transaction ^
    --compress ^
    --order-by-primary  ^
    -plocalpassword | mysql -u rdsuser ^
        --port=3306 ^
        --host=myinstance.123456789012.us-east-1.rds.amazonaws.com ^
        -prdspassword
```

Importing Data to an Amazon RDS MySQL or MariaDB DB Instance with Reduced Downtime

When importing data from an external MySQL or MariaDB database that supports a live application to an Amazon RDS MySQL or MariaDB DB instance, you can use the following procedure to minimize the impact on application availability. This procedure can also help if you are working with a very large database, because you can reduce the cost of the import by reducing the amount of data that is passed across the network to AWS.

In this procedure, you transfer a copy of your database data to an Amazon EC2 instance and import the data into a new Amazon RDS DB instance. You then use replication to bring the Amazon RDS DB instance

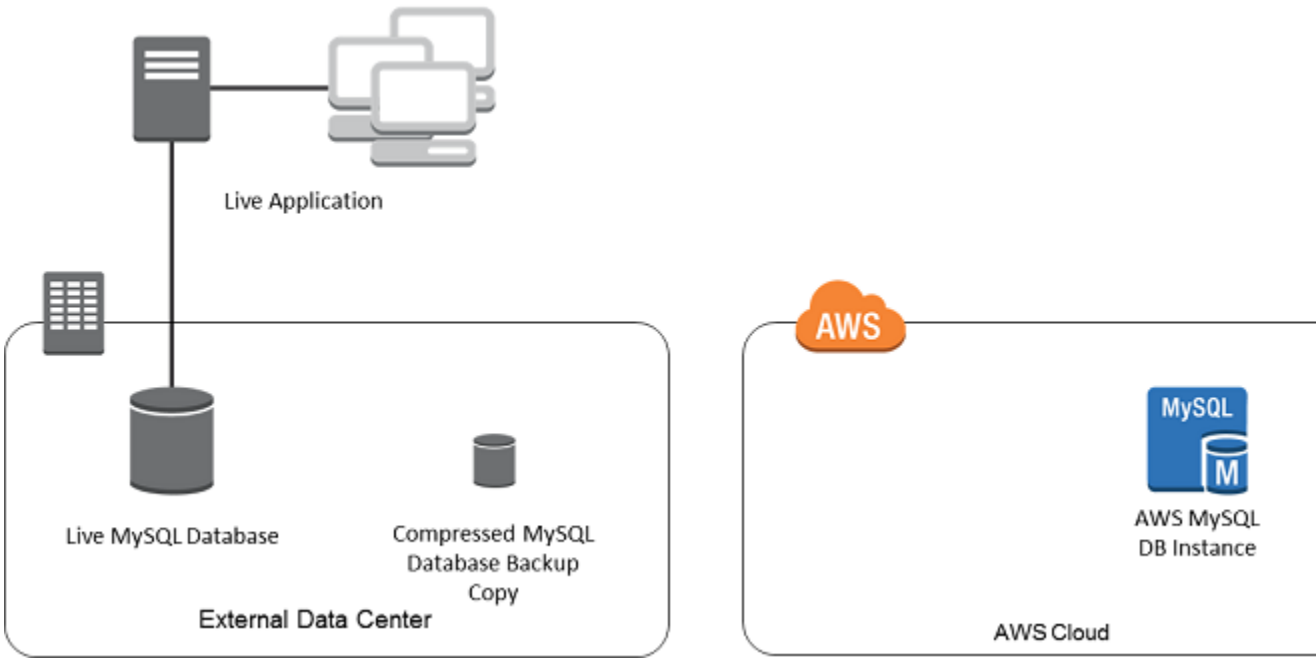
up-to-date with your live external instance, before redirecting your application to the Amazon RDS DB instance. You configure MariaDB replication based on global transaction identifiers (GTIDs) if the external instance is MariaDB 10.0.2 or greater and the target instance is Amazon RDS MariaDB; otherwise, you configure replication based on binary log coordinates. We recommend GTID-based replication if your external database supports it due to its enhanced crash-safety features. For more information, see Global Transaction ID in the MariaDB documentation.



Note
We don't recommend that you use this procedure with source MySQL databases from MySQL versions earlier than version 5.1, due to potential replication issues. For more information, see Replication Compatibility Between MySQL Versions in the MySQL documentation.

Create a Copy of Your Existing Database

The first step in the process of migrating a large amount of data to an Amazon RDS MySQL or MariaDB DB instance with minimal downtime is to create a copy of the source data.



You can use the `mysqldump` utility to create a database backup in either SQL or delimited-text format. You should do a test run with each format in a nonproduction environment to see which method minimizes the amount of time that `mysqldump` runs.

You should also weigh `mysqldump` performance against the benefit offered by using the delimited-text format for loading. A backup using delimited-text format creates a tab-separated text file for each table being dumped. You can load these files in parallel using the `LOAD DATA LOCAL INFILE` command to reduce the amount of time required to import your database. For more information about choosing a `mysqldump` format and then loading the data, see Using mysqldump For Backups in the MySQL documentation.

Before you start the backup operation, you must set the replication options on the MySQL or MariaDB database that you are copying to Amazon RDS. The replication options include enabling binary logging and setting a unique server ID. Setting these options will cause your server to start logging database transactions and prepare it to be a replication master later in this process.

> **Note**
> Your database needs to be stopped to set the replication options and be in read-only mode while the backup copy is created, so you need to schedule a maintenance window for these operations.

To Set Replication Options

1. Edit the my.cnf file (this file is usually under `/etc`):

```
sudo vi /etc/my.cnf
```

   Add the `log_bin` and `server_id` options to the `[mysqld]` section. The `log_bin` option provides a file name identifier for binary log files. The `server_id` option provides a unique identifier for the server in master-replica relationships.

   The following example shows the updated `[mysqld]` section of a my.cnf file:

```
[mysqld]
```

```
log-bin=mysql-bin
server-id=1
```

For more information, see Setting the Replication Master Configuration in the MySQL documentation.

2. Restart the `mysql` service:

```
sudo service mysqld restart
```

To Create a Backup Copy of Your Existing Database

1. Create a backup of your data using the `mysqldump` utility, specifying either SQL or delimited-text format.

   You must specify `--master-data=2` in order to create a backup file that can be used to start replication between servers. For more information, see the mysqldump documentation.

   To improve performance and ensure data integrity, use the `--order-by-primary` and `--single-transaction` options of `mysqldump`.

   To avoid including the MySQL system database in the backup, do not use the `--all-databases` option with `mysqldump`. For more information, see Creating a Dump Snapshot Using mysqldump in the MySQL documentation.

   Use `chmod` if necessary to make sure that the directory where the backup file is being created is writeable.

   Important
   On Windows, run the command window as an administrator.

   - To produce SQL output, use the following command:

     For Linux, OS X, or Unix:

     ```
     sudo mysqldump \
         --databases <database_name> \
         --master-data=2  \
         --single-transaction \
         --order-by-primary \
         -r backup.sql \
         -u <local_user> \
         -p <password>
     ```

     For Windows:

     ```
     mysqldump ^
         --databases <database_name> ^
         --master-data=2  ^
         --single-transaction ^
         --order-by-primary ^
         -r backup.sql ^
         -u <local_user> ^
         -p <password>
     ```

   - To produce delimited-text output, use the following command:

     For Linux, OS X, or Unix:

     ```
     sudo mysqldump \
         --tab=<target_directory> \
     ```

```
    --fields-terminated-by ',' \
    --fields-enclosed-by '"' \
    --lines-terminated-by 0x0d0a \
    <database_name> \
    --master-data=2 \
    --single-transaction \
    --order-by-primary \
    -p <password>
```

For Windows:

```
mysqldump ^
    --tab=<target_directory> ^
    --fields-terminated-by ',' ^
    --fields-enclosed-by '"' ^
    --lines-terminated-by 0x0d0a ^
    <database_name> ^
    --master-data=2 ^
    --single-transaction ^
    --order-by-primary ^
    -p <password>
```

Note
You must create any stored procedures, triggers, functions, or events manually in your Amazon RDS database. If you have any of these objects in the database that you are copying, exclude them when you run `mysqldump` by including the following arguments with your `mysqldump` command: `--routines=0 --triggers=0 --events=0`.

When using the delimited-text format, a CHANGE MASTER TO comment is returned when you run `mysqldump`. This comment contains the master log file name and position. If the external instance is other than MariaDB version 10.0.2 or greater, note the values for MASTER_LOG_FILE and MASTER_LOG_POS; you need these values when setting up replication.

```
-- Position to start replication or point-in-time recovery from
--
-- CHANGE MASTER TO MASTER_LOG_FILE='mysql-bin-changelog.000031', MASTER_LOG_POS=107;
```

If you are using SQL format, you can get the master log file name and position in step 4 of the procedure at Replicate Between Your External Database and New Amazon RDS DB Instance (p. 882). If the external instance is MariaDB version 10.0.2 or greater, you can get the GTID in the next step.

2. If the external instance you are using is MariaDB version 10.0.2 or greater, you use GTID-based replication. Run `SHOW MASTER STATUS` on the external MariaDB instance to get the binary log file name and position, then convert them to a GTID by running `BINLOG_GTID_POS` on the external MariaDB instance:

```
SELECT BINLOG_GTID_POS('<binary log file name>', <binary log file position>);
```

Note the GTID returned; you need it to configure replication.

3. Compress the copied data to reduce the amount of network resources needed to copy your data to the Amazon RDS DB instance. Take note of the size of the backup file; you need this information when determining how large an Amazon EC2 instance to create. When you are done, compress the backup file using GZIP or your preferred compression utility.
   - To compress SQL output, use the following command:

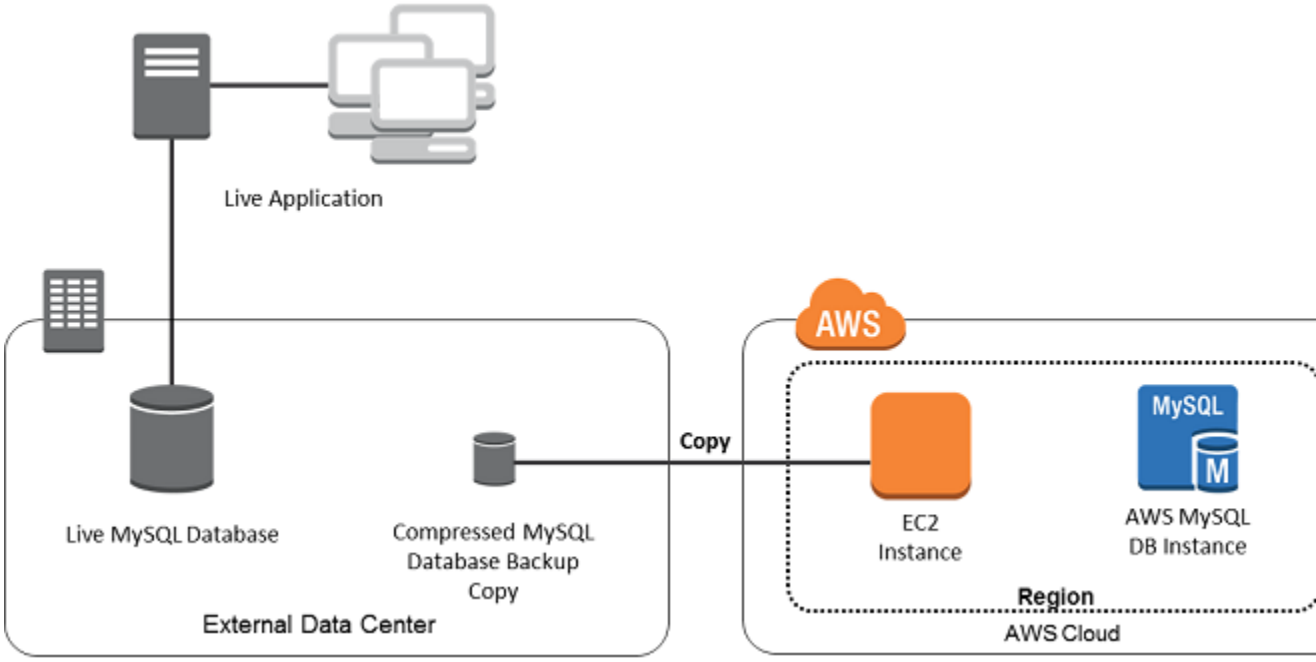
```
gzip backup.sql
```

- To compress delimited-text output, use the following command:

```
tar -zcvf backup.tar.gz <target_directory>
```

Create an Amazon EC2 Instance and Copy the Compressed Database

Copying your compressed database backup file to an Amazon EC2 instance takes fewer network resources than doing a direct copy of uncompressed data between database instances. Once your data is in Amazon EC2, you can copy it from there directly to your Amazon RDS MySQL or MariaDB DB instance. Note that for you to save on the cost of network resources, your Amazon EC2 instance must be in the same region as your Amazon RDS DB instance. Having the Amazon EC2 instance in the same region as your Amazon RDS DB instance also reduces network latency during the import.



To Create an Amazon EC2 Instance and Copy Your Data

1. In the region where you will create the RDS DB instance to run your MySQL database engine, create a VPC, a VPC security group, and a VPC subnet. Ensure that the inbound rules for your VPC security group allow the IP addresses required for your application to connect to AWS. This can be a range of IP addresses (for example, `203.0.113.0/24`), or another VPC security group. You can use the Amazon VPC Management Console to create and manage VPCs, subnets, and security groups. For more information, see Getting Started with Amazon VPC in the *Amazon Virtual Private Cloud Getting Started Guide*.

   Note
   Older AWS accounts can also launch instances in Amazon EC2-Classic mode. In this case, make sure that the inbound rules in the DB security group for your Amazon RDS instance allow access for your EC2-Classic instance using the Amazon EC2 private IP address. For more information, see Working with DB Security Groups (EC2-Classic Platform) (p. 380).

2. Open the Amazon EC2 Management Console and select the region to contain both your Amazon EC2 instance and your Amazon RDS DB instance. Launch an Amazon EC2 instance using the VPC, subnet, and security group that you created in Step 1. Ensure that you select an instance type with enough storage for your database backup file when it is uncompressed. For details on Amazon EC2 instances,

see Getting Started with Amazon EC2 Linux Instances in the *Amazon Elastic Compute Cloud User Guide for Linux*.

3. To connect to your Amazon RDS DB instance from your Amazon EC2 instance, you need to edit your VPC security group, and add an inbound rule specifying the private IP address of your EC2 instance. You can find the private IP address on the **Details** tab of the **Instance** pane in the EC2 console window. To edit the VPC security group and add an inbound rule, choose **Security Groups** in the EC2 console navigation pane, choose your security group, and then add an inbound rule for MySQL/Aurora specifying the private IP address of your EC2 instance. To learn how to add an inbound rule to a VPC security group, see Adding and Removing Rules.
4. Copy your compressed database backup file from your local system to your Amazon EC2 instance. Use `chmod` if necessary to make sure you have write permission for the target directory of the Amazon EC2 instance. You can use `scp` or an SSH client to copy the file. The following is an example:

```
$ scp -r -i <key pair>.pem backup.sql.gz ec2-user@<EC2 DNS>:/<target_directory>/
backup.sql.gz
```

   Important
   Be sure to copy sensitive data using a secure network transfer protocol.

5. Connect to your Amazon EC2 instance and install the latest updates and the MySQL client tools using the following commands:

```
sudo yum update -y
sudo yum install mysql-server -y
```

   For more information, see Connect to Your Instance in the *Amazon Elastic Compute Cloud User Guide for Linux*.

6. While connected to your Amazon EC2 instance, decompress your database backup file. For example:
   - To decompress SQL output, use the following command:

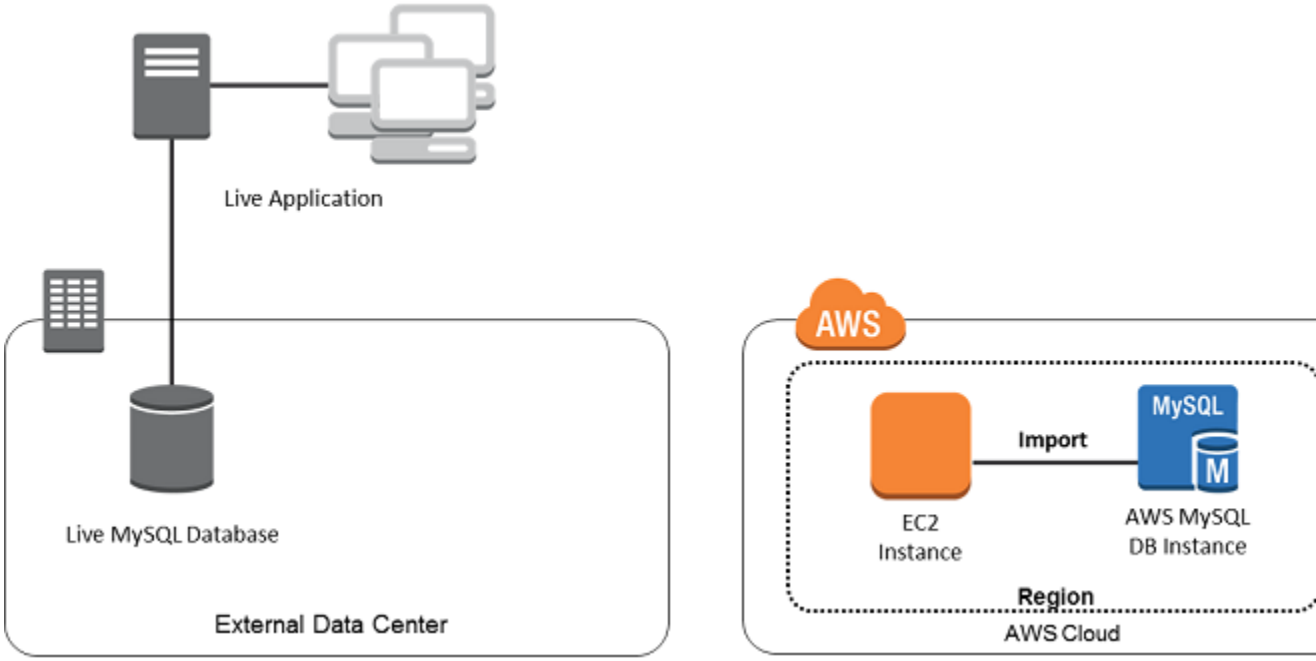
```
gzip backup.sql.gz -d
```

   - To decompress delimited-text output, use the following command:

```
tar xzvf backup.tar.gz
```

Create an Amazon RDS MySQL or MariaDB DB instance and Import Data from Your Amazon EC2 Instance

By creating an Amazon RDS MySQL or MariaDB DB instance in the same region as your Amazon EC2 instance, you can import the database backup file from Amazon EC2 faster than you can import it over the Internet.



To Create an Amazon RDS MySQL or MariaDB DB Instance and Import Your Data

1. Determine which DB instance class and what amount of storage space is required to support the expected workload for this Amazon RDS DB instance. This process should include deciding what is sufficient space and processing capacity for your data load procedures, and also what is required to handle the production workload. You can estimate this based on the size and resources of the source MySQL or MariaDB database. For more information, see DB Instance Class (p. 92).
2. Determine if Amazon RDS provisioned input/output operations per second (IOPS) is required to support the workloads. Provisioned IOPS storage delivers fast throughput for online transaction processing (OLTP) workloads, which are I/O intensive. For more information, see Provisioned IOPS Storage (p. 413).
3. Open the Amazon RDS Console. In the upper-right corner, select the region that contains your Amazon EC2 instance.
4. Choose **Launch a DB Instance**, and then go through the steps to select options for your DB instance:
   a. On the **Select Engine** page, choose **MySQL** or **MariaDB**, as appropriate.
   b. On the **Do you plan to use this database for production purposes?** page, choose **No** to skip configuring Multi-AZ deployment and provisioned IOPS storage.
   c. In the **Instance Specifications** section of the **Specify DB Details** page, specify the DB instance class and allocated storage size that you have determined are appropriate. Choose **No** for **Multi-AZ Deployment**. Specify whether or not to use Provisioned IOPS as you determined in Step 2. For **DB Engine Version**, choose the version that is compatible with your source MySQL instance, as follows:
      - If your source instance is MySQL 5.1.x, the Amazon RDS DB instance must be MySQL 5.5.x.
      - If your source instance is MySQL 5.5.x, the Amazon RDS DB instance must be MySQL 5.5.x or greater.
      - If your source instance is MySQL 5.6.x, the Amazon RDS DB instance must be MySQL 5.6.x or MariaDB.
      - If your source instance is MySQL 5.7.x, the Amazon RDS DB instance must be MySQL 5.7.x, 5.6.x, or MariaDB.
      - If your source instance is MariaDB 5.1, 5.2, or 5.3, the Amazon RDS DB instance must be MySQL 5.1.x.

- If your source instance is MariaDB 5.5 or greater, the Amazon RDS DB instance must be MariaDB.

  Important
  If your source MySQL 5.6.x instance runs a version prior to version 5.6.4, or if the source MySQL 5.6.x instance was upgraded from a version prior to version 5.6.4, then you must create an Amazon RDS MySQL DB instance running version 5.6.27 or later.

  Accept the default values for all other boxes in this section.

  In the **Settings** section, specify the requested database and user information. Choose **Next** when you are done.

d. In the **Network & Security** section of the **Configure Advanced Settings** page, select the same VPC and VPC security group as for your Amazon EC2 instance. This approach ensures that your Amazon EC2 instance and your Amazon RDS instance are visible to each other over the network. Set **Publicly Accessible** to `Yes`. Your DB instance must be publicly accessible to set up replication with your source database as described later in this topic. Accept the default values for all other boxes in this section.

   In the **Database Options** section, specify a database name. Accept the default values for all other boxes in this section.

   In the **Backup** section, set the backup retention period to 0. Accept the default values for all other boxes in this section.

   In the **Maintenance** section, accept the default values for all of the boxes. Choose **Launch Instance** when you are done.

Do not configure multiple Availability Zones, backup retention, or Read Replicas until after you have imported the database backup. When that import is done, you can set Multi-AZ and backup retention the way you want them for the production instance. For a detailed walkthrough of creating an Amazon RDS MySQL DB instance, see Creating a DB Instance Running the MySQL Database Engine (p. 830). For a detailed walkthrough of creating an Amazon RDS MariaDB DB instance, see Creating a DB Instance Running the MariaDB Database Engine (p. 678).

5. Review the default configuration options for the Amazon RDS DB instance. In the left navigation pane of the Amazon RDS Management Console, choose **Parameter Groups** , and then choose the magnifying glass icon next to the **default.mysqlx.x** or **default.mariadbx.x** parameter group. If this parameter group does not have the configuration options that you want, find a different one that does, or create a new parameter group. For more information on creating a parameter group, see Working with DB Parameter Groups (p. 170). If you decide to use a different parameter group than the default, associate it with your Amazon RDS DB instance. For more information, see Modifying a DB Instance Running the MySQL Database Engine (p. 843) or Modifying a DB Instance Running the MariaDB Database Engine (p. 691).
6. Connect to the new Amazon RDS DB instance as the master user, and create the users required to support the administrators, applications, and services that need to access the instance. The host name for the Amazon RDS DB instance is the **Endpoint** value for this instance without including the port number, for example `mysampledb.claxc2oy9ak1.us-west-2.rds.amazonaws.com`. You can find the endpoint value in the instance details in the Amazon RDS Management Console.
7. Connect to your Amazon EC2 instance. For more information, see Connect to Your Instance in the *Amazon Elastic Compute Cloud User Guide for Linux*.
8. Connect to your Amazon RDS DB instance as a remote host from your Amazon EC2 instance using the `mysql` command. The following is an example:

```
mysql -h <host_name> -P 3306 -u <db_master_user> -p
```

   The host name is the DNS name from the Amazon RDS DB instance endpoint.

9. At the `mysql` prompt, run the `source` command and pass it the name of your database dump file to load the data into the Amazon RDS DB instance.
   - For SQL format, use the following command:

     ```
     mysql> source backup.sql;
     ```

   - For delimited-text format, first create the database (if it isn't the default database you created when setting up the Amazon RDS DB instance):

     ```
     $ mysql> create database <database_name>;
     $ mysql> use <database_name>;
     ```

     Then create the tables:

     ```
     $ mysql> source <table1>.sql
     $ mysql> source <table2>.sql
     etc…
     ```

     Then import the data:

     ```
     $ mysql> LOAD DATA LOCAL INFILE 'table1.txt' INTO TABLE table1 FIELDS TERMINATED BY ','
      ENCLOSED BY '"' LINES TERMINATED BY '0x0d0a';
     $ mysql> LOAD DATA LOCAL INFILE 'table2.txt' INTO TABLE table2 FIELDS TERMINATED BY ','
      ENCLOSED BY '"' LINES TERMINATED BY '0x0d0a';
     etc…
     ```

     To improve performance, you can perform these operations in parallel from multiple connections so that all of your tables get created and then loaded at the same time.

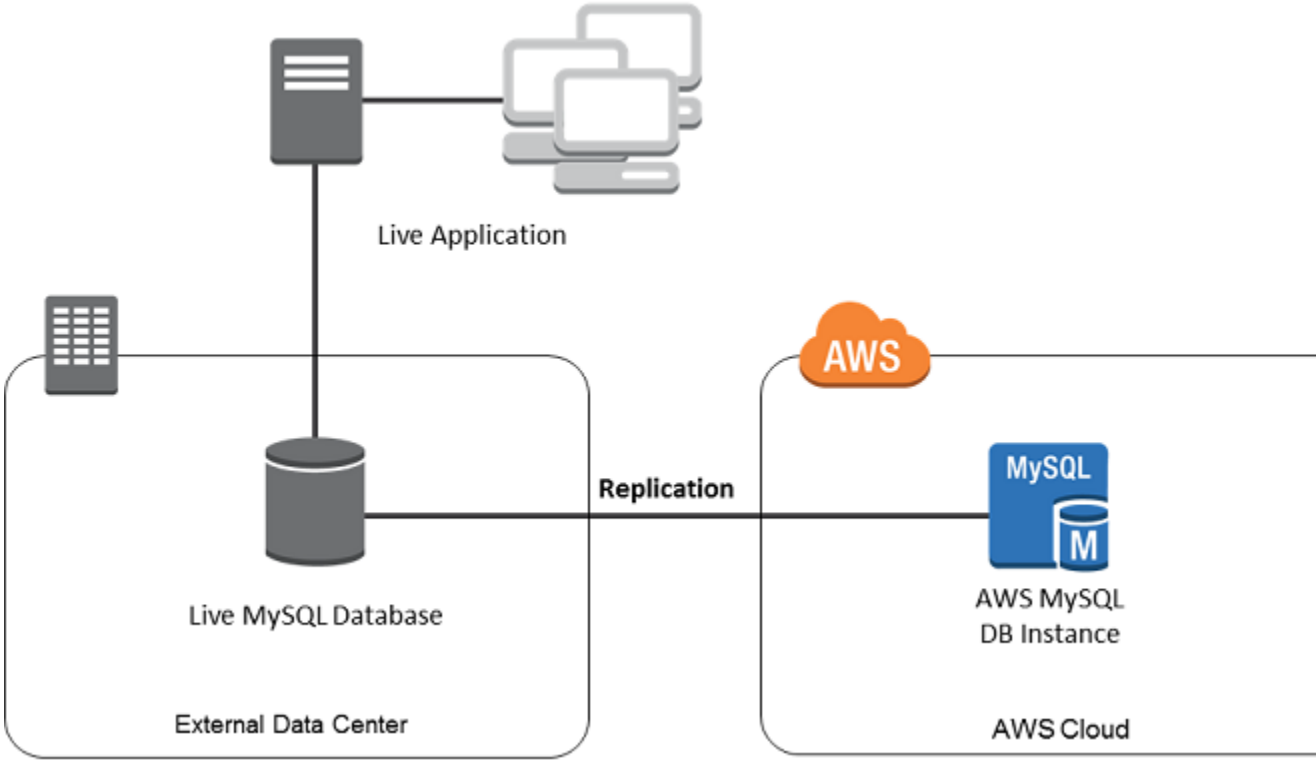
     Note
     If you used any data-formatting options with `mysqldump` when you initially dumped the table, you must use the same options with `mysqlimport` or LOAD DATA LOCAL INFILE to ensure proper interpretation of the data file contents.

10. Run a simple SELECT query against one or two of the tables in the imported database to verify that the import was successful.

Note
If you no longer need the Amazon EC2 instance used in this procedure, you should terminate the EC2 instance to reduce your Amazon AWS resource usage. To terminate an EC2 instance, see Terminating an Instance.

Replicate Between Your External Database and New Amazon RDS DB Instance

Because your source database was likely updated during the time that it took to copy and transfer the data to the Amazon RDS MySQL or MariaDB DB instance, you can use replication to bring the copied database up-to-date with the source database.



Note
The permissions required to start replication on an Amazon RDS DB instance are restricted and not available to your Amazon RDS master user. Because of this, you must use either the Amazon RDS mysql.rds_set_external_master (p. 914) command or the mysql.rds_set_external_master_gtid (p. 717) command to configure replication, and the mysql.rds_start_replication (p. 917) command to start replication between your live database and your Amazon RDS database.

To Start Replication

Earlier, you enabled binary logging and set a unique server ID for your source database. Now you can set up your Amazon RDS DB instance as a replica with your live database as the replication master.

1. In the Amazon RDS Management Console, add the IP address of the server that hosts the source database to the VPC security group for the Amazon RDS DB instance. For more information on modifying a VPC security group, see Security Groups for Your VPC in the *Amazon Virtual Private Cloud User Guide*.

   You might also need to configure your local network to permit connections from the IP address of your Amazon RDS DB instance, so that it can communicate with your source instance. To find the IP address of the Amazon RDS DB instance, use the `host` command:

   ```
   host <RDS_MySQL_DB_host_name>
   ```

   The host name is the DNS name from the Amazon RDS DB instance endpoint, for example `myinstance.123456789012.us-east-1.rds.amazonaws.com`. You can find the endpoint value in the instance details in the Amazon RDS Management Console.
2. Using the client of your choice, connect to the source instance and create a user to be used for replication. This account is used solely for replication and must be restricted to your domain to improve security. The following is an example:

```
CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY '<password>';
```

3. For the source instance, grant `REPLICATION CLIENT` and `REPLICATION SLAVE` privileges to your replication user. For example, to grant the `REPLICATION CLIENT` and `REPLICATION SLAVE` privileges on all databases for the '`repl_user`' user for your domain, issue the following command:

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com'
 IDENTIFIED BY '<password>';
```

4. If you used SQL format to create your backup file and the external instance is not MariaDB 10.0.2 or greater, look at the contents of that file:

```
cat backup.sql
```

   The file includes a CHANGE MASTER TO comment that contains the master log file name and position. This comment is included in the backup file when you use the `--master-data` option with `mysqldump`. Note the values for MASTER_LOG_FILE and MASTER_LOG_POS.

```
--
-- Position to start replication or point-in-time recovery from
--

-- CHANGE MASTER TO MASTER_LOG_FILE='mysql-bin-changelog.000031', MASTER_LOG_POS=107;
```

   If you used delimited text format to create your backup file and the external instance is not MariaDB 10.0.2 or greater, you should already have binary log coordinates from step 1 of the procedure at To Create a Backup Copy of Your Existing Database (p. 876).

   If the external instance is MariaDB 10.0.2 or greater, you should already have the GTID from which to start replication from step 2 of the procedure at To Create a Backup Copy of Your Existing Database (p. 876).

5. Make the Amazon RDS DB instance the replica. If the external instance is not MariaDB 10.0.2 or greater, connect to the Amazon RDS DB instance as the master user and identify the source database as the replication master by using the mysql.rds_set_external_master (p. 914) command. Use the master log file name and master log position that you determined in the previous step if you have a SQL format backup file, or that you determined when creating the backup files if you used delimited-text format. The following is an example:

```
CALL mysql.rds_set_external_master ('mymasterserver.mydomain.com', 3306,
    'repl_user', '<password>', 'mysql-bin-changelog.000031', 107, 0);
```

   If the external instance is MariaDB 10.0.2 or greater, connect to the Amazon RDS DB instance as the master user and identify the source database as the replication master by using the mysql.rds_set_external_master_gtid (p. 717) command. Use the GTID that you determined in step 2 of the procedure at To Create a Backup Copy of Your Existing Database (p. 876). The following is an example:

```
CALL mysql.rds_set_external_master_gtid
 ('Sourcedb.some.com',3306,'ReplicationUser','SomePassW0rd','0-123-456',0);
```

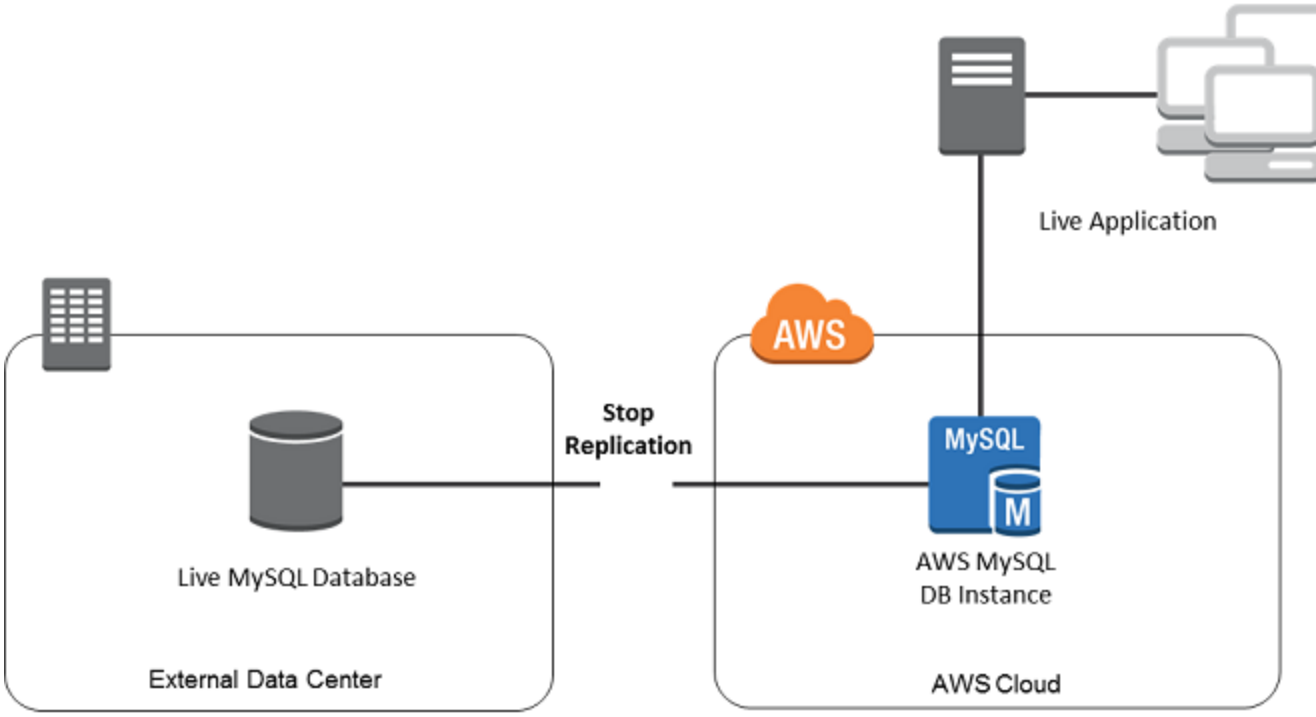
6. On the Amazon RDS DB instance, issue the mysql.rds_start_replication (p. 917) command to start replication:

```
CALL mysql.rds_start_replication;
```

7. On the Amazon RDS DB instance, run the SHOW SLAVE STATUS command to determine when the replica is up-to-date with the replication master. The results of the SHOW SLAVE STATUS command include the Seconds_Behind_Master field. When the Seconds_Behind_Master field returns 0, then the replica is up-to-date with the master.
8. After the Amazon RDS DB instance is up-to-date, enable automated backups so you can restore that database if needed. You can enable or modify automated backups for your Amazon RDS DB instance using the Amazon RDS Management Console. For more information, see Working With Backups (p. 201).

Redirect Your Live Application to Your Amazon RDS Instance

Once the Amazon RDS MySQL or MariaDB DB instance is up-to-date with the replication master, you can now update your live application to use the Amazon RDS instance.



To Redirect Your Live Application to Your Amazon RDS MySQL or MariaDB DB Instance and Stop Replication

1. To add the VPC security group for the Amazon RDS DB instance, add the IP address of the server that hosts the application. For more information on modifying a VPC security group, see Security Groups for Your VPC in the *Amazon Virtual Private Cloud User Guide*.
2. Verify that the Seconds_Behind_Master field in the SHOW SLAVE STATUS command results is 0, which indicates that the replica is up-to-date with the replication master:

```
SHOW SLAVE STATUS;
```

3. Stop replication for the Amazon RDS instance using the mysql.rds_stop_replication (p. 918) command:

```
CALL mysql.rds_stop_replication;
```

4. Update your application to use the Amazon RDS DB instance. This update typically involves changing the connection settings to identify the host name and port of the Amazon RDS DB instance, the user account and password to connect with, and the database to use.
5. Run the mysql.rds_reset_external_master (p. 916) command on your Amazon RDS DB instance to reset the replication configuration so this instance is no longer identified as a replica:

```
CALL mysql.rds_reset_external_master;
```

6. Enable additional Amazon RDS features such as Multi-AZ support and Read Replicas. For more information, see High Availability (Multi-AZ) (p. 99) and Working with PostgreSQL, MySQL, and MariaDB Read Replicas (p. 134).

Note
If you no longer need the Amazon RDS instance used in this procedure, you should delete the RDS instance to reduce your Amazon AWS resource usage. To delete an RDS instance, see Deleting a DB Instance (p. 126).

Importing Data From Any Source to a MySQL or MariaDB DB Instance

If you have more than 1GB of data to load, or if your data is coming from somewhere other than a MySQL or MariaDB database, we recommend creating flat files and loading them with mysqlimport. mysqlimport is another command line utility bundled with the MySQL and MariaDB client software whose purpose is to load flat files into MySQL or MariaDB. For information about mysqlimport, see mysqlimport - A Data Import Program in the MySQL documentation.

We also recommend creating DB snapshots of the target Amazon RDS DB instance before and after the data load. Amazon RDS DB snapshots are complete backups of your DB instance that can be used to restore your DB instance to a known state. When you initiate a DB snapshot, I/O operations to your database instance are momentarily suspended while your database is backed up.

Creating a DB snapshot immediately before the load allows you restore the database to its state prior to the load, should the need arise. A DB snapshot taken immediately after the load protects you from having to load the data again in case of a mishap and can also be used to seed new database instances.

The following list shows the steps to take. Each step is discussed in more detail below.

1. Create flat files containing the data to be loaded.
2. Stop any applications accessing the target DB instance.
3. Create a DB snapshot.
4. Consider disabling Amazon RDS automated backups.
5. Load the data using mysqlimport.
6. Enable automated backups again.

Step 1: Create Flat Files Containing the Data to be Loaded

Use a common format, such as CSV (Comma-Separated Values), to store the data to be loaded. Each table must have its own file; data for multiple tables cannot be combined in the same file. Give each file the same name as the table it corresponds to. The file extension can be anything you like. For example, if the table name is "sales", the file name could be "sales.csv" or "sales.txt", but not "sales_01.csv".

Whenever possible, order the data by the primary key of the table being loaded. This drastically improves load times and minimizes disk storage requirements.

The speed and efficiency of this procedure is dependent upon keeping the size of the files small. If the uncompressed size of any individual file is larger than 1GB, split it into multiple files and load each one separately.

On Unix-like systems (including Linux), use the 'split' command. For example, the following command splits the sales.csv file into multiple files of less than 1GB, splitting only at line breaks (-C 1024m). The new files will be named sales.part_00, sales.part_01, etc.

```
split -C 1024m -d sales.csv sales.part_
```

Similar utilities are available on other operating systems.

Step 2: Stop Any Applications Accessing the Target DB Instance

Before starting a large load, stop all application activity accessing the target DB instance that you will be loading to (particularly if other sessions will be modifying the tables being loaded or tables they reference). This will reduce the risk of constraint violations occurring during the load, improve load performance, and make it possible to restore the database instance to the point just prior to the load without losing changes made by processes not involved in the load.

Of course, this may not be possible or practical. If you are unable to stop applications from accessing the DB instance prior to the load, take steps to ensure the availability and integrity of your data. The specific steps required vary greatly depending upon specific use cases and site requirements.

Step 3: Create a DB Snapshot

If you will be loading data into a new DB instance that contains no data, you may skip this step. Otherwise, creating a DB snapshot of your DB instance will allow you to restore the database Instance to the point just prior to the load, if it becomes necessary. As previously mentioned, when you initiate a DB snapshot, I/O operations to your database instance are suspended for a few minutes while the database is backed up.

In the example below, we use the AWS CLI `create-db-snapshot` command to create a DB Snapshot of our AcmeRDS instance and give the DB snapshot the identifier "preload".

For Linux, OS X, or Unix:

```
aws rds create-db-snapshot \
    --db-instance-identifier AcmeRDS \
    --db-snapshot-identifier preload
```

For Windows:

```
aws rds create-db-snapshot ^
    --db-instance-identifier AcmeRDS ^
    --db-snapshot-identifier preload
```

You can also use the restore from DB snapshot functionality in order to create test database instances for dry runs or to "undo" changes made during the load.

It is important to keep in mind that restoring a database from a DB snapshot creates a new DB instance which, like all DB instances, has a unique identifier and endpoint. If you need to restore the database instance without changing the endpoint, you must first delete the DB instance so that the endpoint can be reused.

For example, to create a DB instance for dry runs or other testing, you would give the DB instance its own identifier. In the example, "AcmeRDS-2" is the identifier and we would connect to the database instance using the endpoint associated with AcmeRDS-2.

For Linux, OS X, or Unix:

```
aws rds restore-db-instance-from-db-snapshot \
    --db-instance-identifier AcmeRDS-2 \
    --db-snapshot-identifier preload
```

For Windows:

```
aws rds restore-db-instance-from-db-snapshot ^
    --db-instance-identifier AcmeRDS-2 ^
    --db-snapshot-identifier preload
```

To reuse the existing endpoint, we must first delete the database instance and then give the restored database the same identifier:

For Linux, OS X, or Unix:

```
aws rds delete-db-instance \
    --db-instance-identifier AcmeRDS \
    --final-db-snapshot-identifier AcmeRDS-Final

aws rds restore-db-instance-from-db-snapshot \
    --db-instance-identifier AcmeRDS \
    --db-snapshot-identifier preload
```

For Windows:

```
aws rds delete-db-instance ^
    --db-instance-identifier AcmeRDS ^
    --final-db-snapshot-identifier AcmeRDS-Final

aws rds restore-db-instance-from-db-snapshot ^
    --db-instance-identifier AcmeRDS ^
    --db-snapshot-identifier preload
```

Note that the example takes a final DB snapshot of the database instance before deleting it. This is optional, but recommended.

Step 4: Consider Disabling Amazon RDS Automated Backups

Warning: DO NOT DISABLE AUTOMATED BACKUPS IF YOU NEED TO RETAIN THE ABILITY TO PERFORM POINT-IN-TIME RECOVERY. Disabling automated backups erases all existing backups, so point-in-time recovery will not be possible after automated backups have been disabled. Disabling automated backups is a performance optimization and is not required for data loads. Note that DB snapshots are not affected by disabling automated backups. All existing DB snapshots are still available for restore.

Disabling automated backups will reduce load time by about 25% and reduce the amount of storage space required during the load. If you will be loading data into a new DB instance that contains no data, disabling backups is an easy way to speed up the load and avoid using the additional storage needed for backups. However, if you will be loading into a DB instance that already contains data; you must weigh the benefits of disabling backups against the impact of losing the ability to perform point-in-time-recovery.

DB instances have automated backups enabled by default (with a one day retention period). In order to disable automated backups, you must set the backup retention period to zero. After the load, you can re-enable backups by setting the backup retention period to a non-zero value. In order to enable or disable backups, Amazon RDS must shut the DB instance down and restart it in order to turn MySQL or MariaDB logging on or off.

Use the AWS CLI `modify-db-instance` command to set the backup retention to zero and apply the change immediately. Setting the retention period to zero requires a DB instance restart, so wait until the restart has completed before proceeding.

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier AcmeRDS \
    --apply-immediately \
    --backup-retention-period 0
```

For Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier AcmeRDS ^
    --apply-immediately ^
    --backup-retention-period 0
```

You can check the status of your DB instance with the AWS CLI `describe-db-instances` command. The example displays the status of the AcmeRDS database instance and includes the --headers option to show column headings.

For Linux, OS X, or Unix:

```
aws rds describe-db-instances \
    --db-instance-identifier AcmeRDS \
    --headers
```

For Windows:

```
aws rds describe-db-instances ^
    --db-instance-identifier AcmeRDS ^
    --headers
```

When the Status column shows that the database is available, you're ready to proceed.

Step 5: Load the Data

Use the mysqlimport utility to load the flat files into Amazon RDS. In the example we tell mysqlimport to load all of the files named "sales" with an extension starting with "part_". This is a convenient way to load all of the files created in the "split" example. Use the --compress option to minimize network traffic. The --fields-terminated-by=',' option is used for CSV files and the --local option specifies that the incoming data is located on the client. Without the --local option, the Amazon RDS DB instance will look for the data on the database host, so always specify the --local option.

For Linux, OS X, or Unix:

```
mysqlimport --local \
    --compress \
    --user=username \
    --password \
    --host=hostname \
    --fields-terminated-by=',' Acme sales.part_*
```

For Windows:

```
mysqlimport --local ^
    --compress ^
    --user=username ^
```

```
    --password ^
    --host=hostname ^
    --fields-terminated-by=',' Acme sales.part_*
```

For very large data loads, take additional DB snapshots periodically between loading files and note which files have been loaded. If a problem occurs, you can easily resume from the point of the last DB snapshot, avoiding lengthy reloads.

Step 6: Enable Amazon RDS Automated Backups

Once the load is finished, re-enable Amazon RDS automated backups by setting the backup retention period back to its pre-load value. As noted earlier, Amazon RDS will restart the DB instance, so be prepared for a brief outage.

In the example, we use the AWS CLI modify-db-instance command to enable automated backups for the AcmeRDS DB instance and set the retention period to 1 day.

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier AcmeRDS \
    --backup-retention-period 1 \
    --apply-immediately
```

For Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier AcmeRDS ^
    --backup-retention-period 1 ^
    --apply-immediately
```

Replication with a MySQL or MariaDB Instance Running External to Amazon RDS

You can set up replication between an Amazon RDS MySQL or MariaDB DB instance and a MySQL or MariaDB instance that is external to Amazon RDS. Use the procedure in this topic to configure replication in all cases except when the external instance is MariaDB version 10.0.2 or greater and the Amazon RDS instance is MariaDB. In that case, use the procedure at Configuring GTID-Based Replication into an Amazon RDS MariaDB DB instance (p. 707) to set up GTID-based replication.

Be sure to follow these guidelines when you set up an external replication master and a replica on Amazon RDS:

- Monitor failover events for the Amazon RDS DB instance that is your replica. If a failover occurs, then the DB instance that is your replica might be recreated on a new host with a different network address. For information on how to monitor failover events, see Using Amazon RDS Event Notification (p. 279).
- Maintain the binlogs on your master instance until you have verified that they have been applied to the replica. This maintenance ensures that you can restore your master instance in the event of a failure.
- Turn on automated backups on your Amazon RDS DB instance. Turning on automated backups ensures that you can restore your replica to a particular point in time if you need to re-synchronize your master and replica. For information on backups and point-in-time restore, see Backing Up and Restoring Amazon RDS DB Instances (p. 200).

Note
The permissions required to start replication on an Amazon RDS DB instance are restricted and not available to your Amazon RDS master user. Because of this, you must use the Amazon RDS

mysql.rds_set_external_master (p. 914) and mysql.rds_start_replication (p. 917) commands to set up replication between your live database and your Amazon RDS database.

Start replication between an external master instance and a DB instance on Amazon RDS

1. Make the source MySQL or MariaDB instance read-only:

```
mysql> FLUSH TABLES WITH READ LOCK;
mysql> SET GLOBAL read_only = ON;
```

2. Run the `SHOW MASTER STATUS` command on the source MySQLor MariaDB instance to determine the binlog location. You will receive output similar to the following example:

```
File                         Position
-------------------------------------
 mysql-bin-changelog.000031      107
-------------------------------------
```

3. Copy the database from the external instance to the Amazon RDS DB instance using `mysqldump`. For very large databases, you might want to use the procedure in Importing Data to an Amazon RDS MySQL or MariaDB DB Instance with Reduced Downtime (p. 873).

   For Linux, OS X, or Unix:

```
mysqldump --databases <database_name> \
    --single-transaction \
    --compress \
    --order-by-primary \
    -u <local_user> \
    -p<local_password> | mysql \
        --host=hostname \
        --port=3306 \
        -u <RDS_user_name> \
        -p<RDS_password>
```

   For Windows:

```
mysqldump --databases <database_name> ^
    --single-transaction ^
    --compress ^
    --order-by-primary ^
    -u <local_user> \
    -p<local_password> | mysql ^
        --host=hostname ^
        --port=3306 ^
        -u <RDS_user_name> ^
        -p<RDS_password>
```

   Note
   Make sure there is not a space between the `-p` option and the entered password.

   Use the `--host`, `--user (-u)`, `--port` and `-p` options in the `mysql` command to specify the hostname, username, port, and password to connect to your Amazon RDS DB instance. The host name is the DNS name from the Amazon RDS DB instance endpoint, for example, `myinstance.123456789012.us-east-1.rds.amazonaws.com`. You can find the endpoint value in the instance details in the Amazon RDS Management Console.

4. Make the source MySQL or MariaDB instance writeable again:

```
mysql> SET GLOBAL read_only = OFF;
mysql> UNLOCK TABLES;
```

For more information on making backups for use with replication, see Backing Up a Master or Slave by Making It Read Only in the MySQL documentation.

5. In the Amazon RDS Management Console, add the IP address of the server that hosts the external database to the VPC security group for the Amazon RDS DB instance. For more information on modifying a VPC security group, see Security Groups for Your VPC in the *Amazon Virtual Private Cloud User Guide*.

   You might also need to configure your local network to permit connections from the IP address of your Amazon RDS DB instance, so that it can communicate with your external MySQL or MariaDB instance. To find the IP address of the Amazon RDS DB instance, use the `host` command:

   ```
   host <RDS_MySQL_DB_host_name>
   ```

   The host name is the DNS name from the Amazon RDS DB instance endpoint.
6. Using the client of your choice, connect to the external instance and create a user that will be used for replication. This account is used solely for replication and must be restricted to your domain to improve security. The following is an example:

   ```
   CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY '<password>';
   ```
7. For the external instance, grant `REPLICATION CLIENT` and `REPLICATION SLAVE` privileges to your replication user. For example, to grant the `REPLICATION CLIENT` and `REPLICATION SLAVE` privileges on all databases for the '`repl_user`' user for your domain, issue the following command:

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com'
    IDENTIFIED BY '<password>';
   ```
8. Make the Amazon RDS DB instance the replica. Connect to the Amazon RDS DB instance as the master user and identify the external MySQL or MariaDB database as the replication master by using the mysql.rds_set_external_master (p. 914) command. Use the master log file name and master log position that you determined in Step 2. The following is an example:

   ```
   CALL mysql.rds_set_external_master ('mymasterserver.mydomain.com', 3306,
       'repl_user', '<password>', 'mysql-bin-changelog.000031', 107, 0);
   ```
9. On the Amazon RDS DB instance, issue the mysql.rds_start_replication (p. 917) command to start replication:

   ```
   CALL mysql.rds_start_replication;
   ```

Exporting Data from a MySQL DB Instance by Using Replication

You can use replication to export data from a MySQL 5.6 or later DB instance to a MySQL instance running external to Amazon RDS. The MySQL instance external to Amazon RDS can be running either on-premises in your data center, or on an Amazon EC2 instance. The MySQL DB instance must be running version 5.6.13 or later. The MySQL instance external to Amazon RDS must be running the same version as the Amazon RDS instance, or a later version.

Replication to an instance of MySQL running external to Amazon RDS is only supported during the time it takes to export a database from a MySQL DB instance. The replication should be terminated when the data has been exported and applications can start accessing the external instance.

The following list shows the steps to take. Each step is discussed in more detail in later sections.

1. Prepare an instance of MySQL running external to Amazon RDS.
2. Configure the MySQL DB instance to be the replication source.
3. Use `mysqldump` to transfer the database from the Amazon RDS instance to the instance external to Amazon RDS.
4. Start replication to the instance running external to Amazon RDS.
5. After the export completes, stop replication.

Prepare an Instance of MySQL External to Amazon RDS

Install an instance of MySQL external to Amazon RDS.

Connect to the instance as the master user, and create the users required to support the administrators, applications, and services that access the instance.

Follow the directions in the MySQL documentation to prepare the instance of MySQL running external to Amazon RDS as a replica. For more information, see Setting the Replication Slave Configuration.

Configure an egress rule for the external instance to operate as a Read Replica during the export. The egress rule will allow the MySQL Read Replica to connect to the MySQL DB instance during replication. Specify an egress rule that allows TCP connections to the port and IP address of the source Amazon RDS MySQL DB instance.

If the Read Replica is running in an Amazon EC2 instance in an Amazon VPC, specify the egress rules in a VPC security group. If the Read Replica is running in an Amazon EC2 instance that is not in a VPC, specify the egress rule in an Amazon EC2 security group. If the Read Replica is installed on-premises, specify the egress rule in a firewall.

If the Read Replica is running in a VPC, configure VPC ACL rules in addition to the security group egress rule. For more information about Amazon VPC network ACLs, see Network ACLs.

- ACL ingress rule allowing TCP traffic to ports 1024-65535 from the IP address of the source MySQL DB instance.
- ACL egress rule: allowing outbound TCP traffic to the port and IP address of the source MySQL DB instance.

Prepare the Replication Source

Prepare the MySQL DB instance as the replication source.

Ensure your client computer has enough disk space available to save the binary logs while setting up replication.

Create a replication account by following the directions in Creating a User For Replication.

Configure ingress rules on the system running the replication source MySQL DB instance that will allow the external MySQL Read Replica to connect during replication. Specify an ingress rule that allows TCP connections to the port used by the Amazon RDS instance from the IP address of the MySQL Read Replica running external to Amazon RDS.

If the Amazon RDS instance is running in a VPC, specify the ingress rules in a VPC security group. If the Amazon RDS instance is not running in an in a VPC, specify the ingress rules in a database security group.

If the Amazon RDS instance is running in a VPC, configure VPC ACL rules in addition to the security group ingress rule. For more information about Amazon VPC network ACLs, see Network ACLs.

- ACL ingress rule: allow TCP connections to the port used by the Amazon RDS instance from the IP address of the external MySQL Read Replica.
- ACL egress rule: allow TCP connections from ports 1024-65535 to the IP address of the external MySQL Read Replica.

Ensure that the backup retention period is set long enough that no binary logs are purged during the export. If any of the logs are purged before the export is complete, you must restart replication from the beginning. For more information about setting the backup retention period, see Working With Backups (p. 201).

Use the `mysql.rds_set_configuration` stored procedure to set the binary log retention period long enough that the binary logs are not purged during the export. For more information, see Accessing MySQL Binary Logs (p. 316).

To further ensure that the binary logs of the source instance are not purged, create an Amazon RDS Read Replica from the source instance. For more information, see Creating a Read Replica (p. 139). After the Amazon RDS Read Replica has been created, call the `mysql.rds_stop_replication` stored procedure to stop the replication process. The source instance will no longer purge its binary log files, so they will be available for the replication process.

Copy the Database

Run the MySQL `SHOW SLAVE STATUS` statement on the RDS read replica, and note the values for the following:

- `master_host`
- `master_port`
- `master_log_file`
- `exec_master_log_pos`

Use the `mysqldump` utility to create a snapshot, which copies the data from Amazon RDS to your local client computer. Then run another utility to load the data into the MySQL instance running external to RDS. Ensure your client computer has enough space to hold the `mysqldump` files from the databases to be replicated. This process can take several hours for very large databases. Follow the directions in Creating a Dump Snapshot Using mysqldump.

The following example shows how to run `mysqldump` on a client, and then pipe the dump into the `mysql` client utility, which loads the data into the external MySQL instance.

For Linux, OS X, or Unix:

```
mysqldump -h RDS instance endpoint \
    -u user \
    -p password \
    --port=3306 \
    --single-transaction \
    --routines \
    --triggers \
    --databases  database database2 \
    --compress  \
    --compact | mysql \
        -h MySQL host \
        -u master user \
        -p password \
        --port 3306
```

For Windows:

```
mysqldump -h RDS instance endpoint ^
    -u user ^
    -p password ^
    --port=3306 ^
    --single-transaction ^
    --routines ^
    --triggers ^
    --databases  database database2 ^
    --compress  ^
    --compact | mysql ^
        -h MySQL host ^
        -u master user ^
        -p password ^
        --port 3306
```

The following example shows how to run `mysqldump` on a client and write the dump to a file.

For Linux, OS X, or Unix:

```
mysqldump -h RDS instance endpoint \
    -u user \
    -p password \
    --port=3306 \
    --single-transaction \
    --routines \
    --triggers \
    --databases  database database2 > path/rds-dump.sql
```

For Windows:

```
mysqldump -h RDS instance endpoint ^
    -u user ^
    -p password ^
    --port=3306 ^
    --single-transaction ^
    --routines ^
    --triggers ^
    --databases  database database2 > path\rds-dump.sql
```

Complete the Export

After you have loaded the `mysqldump` files to create the databases on the MySQL instance running external to Amazon RDS, start replication from the source MySQL DB instance to export all source changes that have occurred after you stopped replication from the Amazon RDS Read Replica.

Use the MySQL `CHANGE MASTER` statement to configure the external MySQL instance. Specify the ID and password of the user granted REPLICATION SLAVE permissions. Specify the `master_host`, `master_port`, `relay_master_log_file` and `exec_master_log_pos` values you got from the Mysql `SHOW SLAVE STATUS` statement you ran on the RDS Read Replica. For more information, see Setting the Master Configuration on the Slave.

Use the MySQL `START SLAVE` command to initiate replication from the source MySQL DB instance and the MySQL replica.

Run the MySQL `SHOW SLAVE STATUS` command on the Amazon RDS instance to verify that it is operating as a Read Replica. For more information about interpreting the results, see SHOW SLAVE STATUS Syntax.

After replication on the MySQL instance has caught up with the Amazon RDS source, use the MySQL `STOP SLAVE` command to terminate replication from the source MySQL DB instance.

On the Amazon RDS Read Replica, call the `mysql.rds_start_replication` stored procedure. This will allow Amazon RDS to start purging the binary log files from the source MySQL DB instance.

Related Topics

- Importing Data into an Amazon RDS MySQL DB Instance (p. 860)
- Backing Up and Restoring Amazon RDS DB Instances (p. 200)

Options for MySQL DB Instances

This appendix describes options, or additional features, that are available for Amazon RDS instances running the MySQL DB engine. To enable these options, you can add them to a custom option group, and then associate the option group with your DB instance. For more information about working with option groups, see Working with Option Groups (p. 153).

Amazon RDS supports the following options for MySQL:

Option	Option ID	Engine Versions
MariaDB Audit Plugin Support (p. 898)	`MARIADB_AUDIT_PLUGIN`	MySQL 5.6.29 and later MySQL 5.7.16 and later
MySQL `MEMCACHED` Support (p. 901)	`MEMCACHED`	MySQL 5.6 and later

MariaDB Audit Plugin Support

Amazon RDS supports using the MariaDB Audit Plugin on MySQL database instances. The MariaDB Audit Plugin records database activity such as users logging on to the database, queries run against the database, and more. The record of database activity is stored in a log file.

Audit Plugin Option Settings

Amazon RDS supports the following settings for the MariaDB Audit Plugin option.

Option Setting	Valid Values	Default Value	Description
SERVER_AUDIT_FILE_PATH	/rdsdbdata/log/audit/	/rdsdbdata/log/audit/	The location of the log file. The log file contains the record of the activity specified in SERVER_AUDIT_EVENTS. For more information, see Viewing and Listing Database Log Files (p. 303) and MySQL Database Log Files (p. 313).
SERVER_AUDIT_FILE_ROTATE_SIZE	1–1000000000	1000000	The size in bytes that when reached, causes the file to rotate. For more information, see Log File Size (p. 314).
SERVER_AUDIT_FILE_ROTATIONS	0–100	9	The number of log rotations to save. For more information, see Log File Size (p. 314) and Downloading a Database Log File (p. 304).
SERVER_AUDIT_EVENTS	CONNECT, QUERY	CONNECT, QUERY	The types of activity to record in the log. Installing the MariaDB Audit Plugin is itself logged. • CONNECT: Log successful and unsuccessful connections to the database, and disconnections from the database. • QUERY: Log the text of all queries run against the database. • TABLE: Log tables affected by queries when the queries are run against the database. For MariaDB, CONNECT, QUERY, and TABLE are supported. For MySQL, CONNECT and QUERY are supported.
SERVER_AUDIT_INCL_USERS	Multiple comma-separated values	None	Include only activity from the specified users. By default, activity is recorded for all users. If a user is specified in both SERVER_AUDIT_EXCL_USERS and SERVER_AUDIT_INCL_USERS, then activity is recorded for the user.
SERVER_AUDIT_EXCL_USERS	Multiple comma-separated values	None	Exclude activity from the specified users. By default, activity is recorded for all users. If a user is specified in both SERVER_AUDIT_EXCL_USERS and SERVER_AUDIT_INCL_USERS, then activity is recorded for the user. The rdsadmin user queries the database every second to check the health of the database.

Option Setting	Valid Values	Default Value	Description
			Depending on your other settings, this activity can possibly cause the size of your log file to grow very large, very quickly. If you don't need to record this activity, add the `rdsadmin` user to the `SERVER_AUDIT_EXCL_USERS` list.
`SERVER_AUDIT_LOGGING`	`ON`	`ON`	Logging is active. The only valid value is `ON`. Amazon RDS does not support deactivating logging. If you want to deactivate logging, remove the MariaDB Audit Plugin. For more information, see Removing the MariaDB Audit Plugin (p. 900).

Adding the MariaDB Audit Plugin

The general process for adding the MariaDB Audit Plugin to a DB instance is the following:

- Create a new option group, or copy or modify an existing option group
- Add the option to the option group
- Associate the option group with the DB instance

After you add the MariaDB Audit Plugin, you don't need to restart your DB instance. As soon as the option group is active, auditing begins immediately.

To add the MariaDB Audit Plugin

1. Determine the option group you want to use. You can create a new option group or use an existing option group. If you want to use an existing option group, skip to the next step. Otherwise, create a custom DB option group. Choose **mysql** for **Engine**, and choose **5.6**, **5.7**, or later for **Major Engine Version**. For more information, see Creating an Option Group (p. 154).
2. Add the **MARIADB_AUDIT_PLUGIN** option to the option group, and configure the option settings. For more information about adding options, see Adding an Option to an Option Group (p. 157). For more information about each setting, see Audit Plugin Option Settings (p. 898).
3. Apply the option group to a new or existing DB instance.
   - For a new DB instance, you apply the option group when you launch the instance. For more information, see Creating a DB Instance Running the MySQL Database Engine (p. 830).
   - For an existing DB instance, you apply the option group by modifying the instance and attaching the new option group. For more information, see Modifying a DB Instance Running the MySQL Database Engine (p. 843).

Viewing and Downloading the MariaDB Audit Plugin Log

After you enable the MariaDB Audit Plugin, you access the results in the log files the same way you access any other text-based log files. The audit log files are located at `/rdsdbdata/log/audit/`. For information about viewing the log file in the console, see Viewing and Listing Database Log Files (p. 303). For information about downloading the log file, see Downloading a Database Log File (p. 304).

Modifying MariaDB Audit Plugin Settings

After you enable the MariaDB Audit Plugin, you can modify the settings. For more information about how to modify option settings, see Modifying an Option Setting (p. 163). For more information about each setting, see Audit Plugin Option Settings (p. 898).

Removing the MariaDB Audit Plugin

Amazon RDS doesn't support turning off logging in the MariaDB Audit Plugin. However, you can remove the plugin from a DB instance. After you remove the MariaDB Audit Plugin, you need to restart your DB instance to stop auditing.

To remove the MariaDB Audit Plugin from a DB instance, do one of the following:

- Remove the MariaDB Audit Plugin option from the option group it belongs to. This change affects all DB instances that use the option group. For more information, see Removing an Option from an Option Group (p. 167)
- Modify the DB instance and specify a different option group that doesn't include the plugin. This change affects a single DB instance. You can specify the default (empty) option group, or a different custom option group. For more information, see Modifying a DB Instance Running the MySQL Database Engine (p. 843).

MySQL MEMCACHED Support

Amazon RDS supports using the memcached interface to InnoDB tables that was introduced in MySQL 5.6. The memcached API enables applications to use InnoDB tables in a manner similar to NoSQL key-value data stores.

memcached is a simple, key-based cache. Applications use memcached to insert, manipulate, and retrieve key-value data pairs from the cache. MySQL 5.6 introduced a plugin that implements a daemon service that exposes data from InnoDB tables through the memcached protocol. For more information about the MySQL memcached plugin, go to InnoDB Integration with memcached.

You enable memcached support for an Amazon RDS MySQL 5.6 or later instance by:

1. Determining the security group to use for controlling access to the memcached interface. If the set of applications already using the SQL interface are the same set that will access the memcached interface, you can use the existing VPC or DB security group used by the SQL interface. If a different set of applications will access the memcached interface, define a new VPC or DB security group. For more information about managing security groups, see Amazon RDS Security Groups (p. 375)
2. Creating a custom DB option group, selecting MySQL as the engine type and a 5.6 or later version. For more information about creating an option group, see Creating an Option Group (p. 154).
3. Adding the MEMCACHED option to the option group. Specify the port that the memcached interface will use, and the security group to use in controlling access to the interface. For more information about adding options, see Adding an Option to an Option Group (p. 157).
4. Modifying the option settings to configure the memcached parameters, if necessary. For more information about how to modify option settings, see Modifying an Option Setting (p. 163).
5. Applying the option group to an instance. Amazon RDS enables memcached support for that instance when the option group is applied:
   - You enable memcached support for a new instance by specifying the custom option group when you launch the instance. For more information about launching a MySQL instance, see Creating a DB Instance Running the MySQL Database Engine (p. 830).
   - You enable memcached support for an existing instance by specifying the custom option group when you modify the instance. For more information about modifying a MySQL instance, see Modifying a DB Instance Running the MySQL Database Engine (p. 843).
6. Specifying which columns in your MySQL tables can be accessed through the memcached interface. The memcached plug-in creates a catalog table named containers in a dedicated database named innodb_memcache. You insert a row into the containers table to map an InnoDB table for access through memcached. You specify a column in the InnoDB table that is used to store the memcached key values, and one or more columns that are used to store the data values associated with the key. You also specify a name that a memcached application uses to refer to that set of columns. For details on inserting rows in the containers table, go to Internals of the InnoDB memcached Plugin. For an example of mapping an InnoDB table and accessing it through memcached, go to Specifying the Table and Column Mappings for an InnoDB + memcached Application.
7. If the applications accessing the memcached interface are on different computers or EC2 instances than the applications using the SQL interface, add the connection information for those computers to the VPC or DB security group associated with the MySQL instance. For more information about managing security groups, see Amazon RDS Security Groups (p. 375).

You turn off the memcached support for an instance by modifying the instance and specifying the default option group for your MySQL version. For more information about modifying a MySQL instance, see Modifying a DB Instance Running the MySQL Database Engine (p. 843).

MySQL memcached Security Considerations

The memcached protocol does not support user authentication. For more information about MySQL memcached security considerations, go to memcached Deployment and Using memcached as a MySQL Caching Layer.

You can take the following actions to help increase the security of the memcached interface:

- Specify a different port than the default of 11211 when adding the MEMCACHED option to the option group.
- Ensure that you associate the memcached interface with either a VPC or DB security group that limits access to known, trusted client addresses or EC2 instances. For more information about managing security groups, see Amazon RDS Security Groups (p. 375).

MySQL memcached Connection Information

To access the memcached interface, an application must specify both the DNS name of the Amazon RDS instance and the memcached port number. For example, if an instance has a DNS name of my-cache-instance.cg034hpkmmjt.region.rds.amazonaws.com and the memcached interface is using port 11212, the connection information specified in PHP would be:

```
<?php

$cache = new Memcache;
$cache->connect('my-cache-instance.cg034hpkmmjt.region.rds.amazonaws.com',11212);
?>
```

To find the DNS name and memcached port of an Amazon RDS MySQL instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the top right corner of the AWS Management Console, select the region that contains the DB instance.
3. In the navigation pane, click **Instances**.
4. Select the arrow to the left of name of the DB Instance running the MySQL database engine. In the description display, note the value of the **endpoint** field. The DNS name is the part of the endpoint up to the semicolon (:). Ignore the semicolon and the port number after the semicolon, that port is not used to access the memcached interface.
5. Note the name listed in the **Option Group(s)** field.
6. In the navigation pane, click **Option Groups**.
7. Select the arrow to the left of the name of the option group used by the MySQL DB instance. In the description display, note the value of the **port** setting in the **MEMCACHED** option.

MySQL memcached Option Settings

Amazon RDS exposes the MySQL memcached parameters as option settings in the Amazon RDS MEMCACHED option.

MySQL memcached Parameters

- DAEMON_MEMCACHED_R_BATCH_SIZE - an integer that specifies how many memcached read operations (get) to perform before doing a COMMIT to start a new transaction. The allowed values are 1 to 4294967295, the default is 1. The option does not take effect until the instance is restarted.

- DAEMON_MEMCACHED_W_BATCH_SIZE - an integer that specifies how many memcached write operations, such as add, set, or incr, to perform before doing a COMMIT to start a new transaction. The allowed values are 1 to 4294967295, the default is 1. The option does not take effect until the instance is restarted.
- INNODB_API_BK_COMMIT_INTERVAL - an integer that specifies how often to auto-commit idle connections that use the InnoDB memcached interface. The allowed values are 1 to 1073741824, the default is 5. The option takes effect immediately, without requiring that you restart the instance.
- INNODB_API_DISABLE_ROWLOCK - a Boolean that disables (1 (true)) or enables (0 (false)) the use of row locks when using the InnoDB memcached interface. The default is 0 (false). The option does not take effect until the instance is restarted.
- INNODB_API_ENABLE_MDL - a Boolean that when set to 0 (false) locks the table used by the InnoDB memcached plugin, so that it cannot be dropped or altered by DDL through the SQL interface. The default is 0 (false). The option does not take effect until the instance is restarted.
- INNODB_API_TRX_LEVEL - an integer that specifies the transaction isolation level for queries processed by the memcached interface. The allowed values are 0 to 3. The default is 0. The option does not take effect until the instance is restarted.

Amazon RDS configures these MySQL memcached parameters, they cannot be modified: DAEMON_MEMCACHED_LIB_NAME, DAEMON_MEMCACHED_LIB_PATH, and INNODB_API_ENABLE_BINLOG. The parameters that MySQL administrators set by using daemon_memcached_options are available as individual MEMCACHED option settings in Amazon RDS.

MySQL daemon_memcached_options Parameters

- BINDING_PROTOCOL - a string that specifies the binding protocol to use. The allowed values are auto, ascii, or binary. The default is auto, which means the server automatically negotiates the protocol with the client. The option does not take effect until the instance is restarted.
- BACKLOG_QUEUE_LIMIT - an integer that specifies how many network connections can be waiting to be processed by memcached. Increasing this limit may reduce errors received by a client that is not able to connect to the memcached instance, but does not improve the performance of the server. The allowed values are 1 to 2048, the default is 1024. The option does not take effect until the instance is restarted.
- CAS_DISABLED - a Boolean that enables (1 (true)) or disables (0 (false)) the use of compare and swap (CAS), which reduces the per-item size by 8 bytes. The default is 0 (false). The option does not take effect until the instance is restarted.
- CHUNK_SIZE - an integer that specifies the minimum chunk size, in bytes, to allocate for the smallest item's key, value, and flags. The allowed values are 1 to 48. The default is 48 and you can significantly improve memory efficiency with a lower value. The option does not take effect until the instance is restarted.
- CHUNCK_SIZE_GROWTH_FACTOR - a float that controls the size of new chunks. The size of a new chunk is the size of the previous chunk times CHUNCK_SIZE_GROWTH_FACTOR. The allowed values are 1 to 2, the default is 1.25. The option does not take effect until the instance is restarted.
- ERROR_ON_MEMORY_EXHAUSTED - a Boolean, when set to 1 (true) it specifies that memcached will return an error rather than evicting items when there is no more memory to store items. If set to 0 (false), memcached will evict items if there is no more memory. The default is 0 (false). The option does not take effect until the instance is restarted.
- MAX_SIMULTANEOUS_CONNECTIONS - an integer that specifies the maximum number of concurrent connections. Setting this value to anything under 10 prevents MySQL from starting. The allowed values are 10 to 1024, the default is 1024. The option does not take effect until the instance is restarted.
- VERBOSITY - an string that specifies the level of information logged in the MySQL error log by the memcached service. The default is v. The option does not take effect until the instance is restarted. The allowed values are:

- v - Logs errors and warnings while executing the main event loop.
- vv - In addition to the information logged by v, also logs each client command and the response.
- vvv - In addition to the information logged by vv, also logs internal state transitions.

Amazon RDS configures these MySQL DAEMON_MEMCACHED_OPTIONS parameters, they cannot be modified: DAEMON_PROCESS, LARGE_MEMORY_PAGES, MAXIMUM_CORE_FILE_LIMIT, MAX_ITEM_SIZE, LOCK_DOWN_PAGE_MEMORY, MASK, IDFILE, REQUESTS_PER_EVENT, SOCKET, and USER.

Common DBA Tasks for MySQL DB Instances

This section describes the Amazon RDS-specific implementations of some common DBA tasks for DB instances running the MySQL database engine. In order to deliver a managed service experience, Amazon RDS does not provide shell access to DB instances, and it restricts access to certain system procedures and tables that require advanced privileges.

For information about working with MySQL log files on Amazon RDS, see MySQL Database Log Files (p. 313)

Topics

Killing a Session or Query

You can terminate user sessions or queries on DB instances by using the `rds_kill` and `rds_kill_query` commands. First connect to your MySQL database instance, then issue the appropriate command as shown following. For more information, see Connecting to a DB Instance Running the MySQL Database Engine (p. 840).

```
CALL mysql.rds_kill(thread-ID)
CALL mysql.rds_kill_query(thread-ID)
```

For example, to kill the session that is running on thread 99, you would type the following:

```
CALL mysql.rds_kill(99);
```

To kill the query that is running on thread 99, you would type the following:

```
CALL mysql.rds_kill_query(99);
```

Skipping the Current Replication Error

Amazon RDS provides a mechanism for you to skip an error on your Read Replicas if the error is causing your Read Replica to hang and the error doesn't affect the integrity of your data. First connect to your MySQL database instance, then issue the appropriate commands as shown following. For more information, see Connecting to a DB Instance Running the MySQL Database Engine (p. 840).

> **Note**
> You should first verify that the error can be safely skipped. In a MySQL utility, connect to the Read Replica and run the following MySQL command:
>
> ```
> SHOW SLAVE STATUS\G
> ```
>
> For information about the values returned, go to SHOW SLAVE STATUS Syntax in the MySQL documentation.

To skip the error, you can issue the following command:

```
CALL mysql.rds_skip_repl_error;
```

This command has no effect if you run it on the source DB instance, or on a Read Replica that has not encountered a replication error.

For more information, such as the versions of MySQL that support `mysql.rds_skip_repl_error`, see mysql.rds_skip_repl_error (p. 918).

Important
If you attempt to call *mysql.rds_skip_repl_error* and encounter the following error: `ERROR 1305 (42000): PROCEDURE mysql.rds_skip_repl_error does not exist`, then upgrade your MySQL DB instance to the latest minor version or one of the minimum minor versions listed in mysql.rds_skip_repl_error (p. 918).

Working with InnoDB Tablespaces to Improve Crash Recovery Times

Every table in MySQL consists of a table definition, data, and indexes. The MySQL storage engine InnoDB stores table data and indexes in a *tablespace*. InnoDB creates a global shared tablespace that contains a data dictionary and other relevant metadata, and it can contain table data and indexes. InnoDB can also create separate tablespaces for each table and partition. These separate tablespaces are stored in files with a .ibd extension and the header of each tablespace contains a number that uniquely identifies it.

Amazon RDS provides a parameter in a MySQL parameter group called `innodb_file_per_table`. This parameters controls whether InnoDB adds new table data and indexes to the shared tablespace (by setting the parameter value to 0) or to individual tablespaces (by setting the parameter value to 1). Amazon RDS sets the default value for `innodb_file_per_table` parameter to 1, which allows you to drop individual InnoDB tables and reclaim storage used by those tables for the DB instance. In most use cases, setting the `innodb_file_per_table` parameter to 1 is the recommended setting.

You should set the `innodb_file_per_table` parameter to 0 when you have a large number of tables, such as over 1000 tables when you use standard (magnetic) or general purpose SSD storage or over 10,000 tables when you use Provisioned IOPS storage. When you set this parameter to 0, individual tablespaces are not created and this can improve the time it takes for database crash recovery.

MySQL processes each metadata file, which includes tablespaces, during the crash recovery cycle. The time it takes MySQL to process the metadata information in the shared tablespace is negligible compared to the time it takes to process thousands of tablespace files when there are multiple tablespaces. Because the tablespace number is stored within the header of each file, the aggregate time to read all the tablespace files can take up to several hours. For example, a million InnoDB tablespaces on standard storage can take from five to eight hours to process during a crash recovery cycle. In some cases, InnoDB can determine that it needs additional cleanup after a crash recovery cycle so it will begin another crash recovery cycle, which will extend the recovery time. Keep in mind that a crash recovery cycle also entails rolling-back transactions, fixing broken pages, and other operations in addition to the processing of tablespace information.

Since the `innodb_file_per_table` parameter resides in a parameter group, you can change the parameter value by editing the parameter group used by your DB instance without having to reboot the DB instance. After the setting is changed, for example, from 1 (create individual tables) to 0 (use shared tablespace), new InnoDB tables will be added to the shared tablespace while existing tables continue to have individual tablespaces. To move an InnoDB table to the shared tablespace, you must use the `ALTER TABLE` command.

Migrating Multiple Tablespaces to the Shared Tablespace

You can move an InnoDB table's metadata from its own tablespace to the shared tablespace, which will rebuild the table metadata according to the `innodb_file_per_table` parameter setting. First connect

to your MySQL database instance, then issue the appropriate commands as shown following. For more information, see Connecting to a DB Instance Running the MySQL Database Engine (p. 840).

```
ALTER TABLE table_name ENGINE = InnoDB, ALGORITHM=COPY;
```

For example, the following query returns an `ALTER TABLE` statement for every InnoDB table.

```
SELECT CONCAT('ALTER TABLE `',
    REPLACE(TABLE_SCHEMA, '`', '``'), '`.`',
    REPLACE(TABLE_NAME, '`', '``'), '` ENGINE=InnoDB, ALGORITHM=COPY;')
    FROM INFORMATION_SCHEMA.TABLES
    WHERE TABLE_TYPE = 'BASE TABLE'
    AND ENGINE = 'InnoDB' AND TABLE_SCHEMA <> 'mysql';
```

Rebuilding a MySQL table to move the table's metadata to the shared tablespace requires additional storage space temporarily to rebuild the table, so the DB instance must have storage space available. During rebuilding, the table is locked and inaccessible to queries. For small tables or tables not frequently accessed, this may not be an issue; for large tables or tables frequently accessed in a heavily concurrent environment, you can rebuild tables on a Read Replica.

You can create a Read Replica and migrate table metadata to the shared tablespace on the Read Replica. While the ALTER TABLE statement blocks access on the Read Replica, the source DB instance is not affected. The source DB instance will continue to generate its binary logs while the Read Replica lags during the table rebuilding process. Because the rebuilding requires additional storage space and the replay log file can become large, you should create a Read Replica with storage allocated that is larger than the source DB instance.

The following steps should be followed to create a Read Replica and rebuild InnoDB tables to use the shared tablespace:

1. Ensure that backup retention is enabled on the source DB instance so that binary logging is enabled
2. Use the AWS Console or AWS CLI to create a Read Replica for the source DB instance. Since the creation of a Read Replica involves many of the same processes as crash recovery, the creation process may take some time if there are a large number of InnoDB tablespaces. Allocate more storage space on the Read Replica than is currently used on the source DB instance.
3. When the Read Replica has been created, create a parameter group with the parameter settings `read_only = 0` and `innodb_file_per_table = 0`, and then associate the parameter group with the Read Replica.
4. Issue ALTER TABLE <name> ENGINE = InnoDB against all tables you want migrated on the replica.
5. When all of your ALTER TABLE statements have completed on the Read Replica, verify that the Read Replica is connected to the source DB instance and that the two instances are in-sync.
6. When ready, use the AWS Console or AWS CLI to promote the Read Replica to be the master instance. Make sure that the parameter group used for the new master has the innodb_file_per_table parameter set to 0. Change the name of the new master, and point any applications to the new master instance.

Managing the Global Status History

MySQL maintains many status variables that provide information about its operation. Their value can help you detect locking or memory issues on a DB instance . The values of these status variables are cumulative since last time the DB instance was started. You can reset most status variables to 0 by using the `FLUSH STATUS` command.

To allow for monitoring of these values over time, Amazon RDS provides a set of procedures that will snapshot the values of these status variables over time and write them to a table, along with any

changes since the last snapshot. This infrastructure, called Global Status History (GoSH), is installed on all MySQL DB instances starting with versions 5.5.23. GoSH is disabled by default.

To enable GoSH, you first enable the event scheduler from a DB parameter group by setting the parameter event_scheduler to ON. For information about creating and modifying a DB parameter group, see Working with DB Parameter Groups (p. 170).

You can then use the procedures in the following table to enable and configure GoSH. First connect to your MySQL database instance, then issue the appropriate commands as shown following. For more information, see Connecting to a DB Instance Running the MySQL Database Engine (p. 840). For each procedure, type the following:

```
CALL procedure-name;
```

Where *procedure-name* is one of the procedures in the table.

Procedure	Description
`rds_enable_gsh_collector`	Enables GoSH to take default snapshots at intervals specified by `rds_set_gsh_collector`.
`rds_set_gsh_collector`	Specifies the interval, in minutes, between snapshots. Default value is 5.
`rds_disable_gsh_collector`	Disables snapshots.
`rds_collect_global_status_history`	Takes a snapshot on demand.
`rds_enable_gsh_rotation`	Enables rotation of the contents of the `mysql.rds_global_status_history` table to `mysql.rds_global_status_history_old` at intervals specified by `rds_set_gsh_rotation`.
`rds_set_gsh_rotation`	Specifies the interval, in days, between table rotations. Default value is 7.
`rds_disable_gsh_rotation`	Disables table rotation.
`rds_rotate_global_status_history`	Rotates the contents of the `mysql.rds_global_status_history` table to `mysql.rds_global_status_history_old` on demand.

When GoSH is running, you can query the tables that it writes to. For example, to query the hit ratio of the Innodb buffer pool, you would issue the following query:

```
select a.collection_end, a.collection_start, (( a.variable_Delta-b.variable_delta)/
a.variable_delta)*100 as "HitRatio"
    from mysql.rds_global_status_history as a join mysql.rds_global_status_history as b on
 a.collection_end = b.collection_end
    where a. variable_name = 'Innodb_buffer_pool_read_requests' and b.variable_name =
 'Innodb_buffer_pool_reads'
```

Known Issues and Limitations for MySQL on Amazon RDS

Known issues and limitations for working with MySQL on Amazon RDS are as follows.

Inconsistent InnoDB Buffer Pool Size

For MySQL 5.7, there is currently a bug in the way that the InnoDB buffer pool size is managed. MySQL 5.7 might adjust the value of the `innodb_buffer_pool_size` parameter to a large value that can result in the InnoDB buffer pool growing too large and using up too much memory. This effect can cause the MySQL database engine to stop running or can prevent the MySQL database engine from starting. This issue is more common for DB instance classes that have less memory available.

To resolve this issue, set the value of the `innodb_buffer_pool_size` parameter to a multiple of the product of the `innodb_buffer_pool_instances` parameter value and the `innodb_buffer_pool_chunk_size` parameter value. For example, you might set the `innodb_buffer_pool_size` parameter value to a multiple of eight times the product of the `innodb_buffer_pool_instances` and `innodb_buffer_pool_chunk_size` parameter values, as shown in the following example.

```
innodb_buffer_pool_chunk_size = 536870912
innodb_buffer_pool_instances = 4
innodb_buffer_pool_size = (536870912 * 4) * 8 = 17179869184
```

For details on this MySQL 5.7 bug, go to https://bugs.mysql.com/bug.php?id=79379 in the MySQL documentation.

MySQL Version 5.5.40 Asynchronous I/O Is Disabled

You might observe reduced I/O performance if you have a MySQL DB instance that was created before April 23, 2014, and then upgraded to MySQL version 5.5.40 after October 17, 2014. This reduced performance can be caused by an error that disables the `innodb_use_native_aio` parameter even if the corresponding DB parameter group enables the `innodb_use_native_aio` parameter.

To resolve this error, we recommend that you upgrade your MySQL DB instance running version 5.5.40 to version 5.5.40a, which corrects this behavior. For information on minor version upgrades, see Upgrading the MySQL DB Engine (p. 851).

For more information on MySQL asynchronous I/O, go to Asynchronous I/O on Linux in the MySQL documentation.

Index Merge Optimization Returns Wrong Results

Queries that use index merge optimization might return wrong results due to a bug in the MySQL query optimizer that was introduced in MySQL 5.5.37. When you issue a query against a table with multiple indexes the optimizer scans ranges of rows based on the multiple indexes, but does not merge the results together correctly. For more information on the query optimizer bug, go to http://bugs.mysql.com/bug.php?id=72745 and http://bugs.mysql.com/bug.php?id=68194 in the MySQL bug database.

For example, consider a query on a table with two indexes where the search arguments reference the indexed columns.

```
SELECT * FROM table1
```

```
WHERE indexed_col1 = 'value1' AND indexed_col2 = 'value2';
```

In this case, the search engine will search both indexes. However, due to the bug, the merged results are incorrect.

To resolve this issue, you can do one of the following:

- Set the `optimizer_switch` parameter to `index_merge=off` in the DB parameter group for your MySQL DB instance. For information on setting DB parameter group parameters, see Working with DB Parameter Groups (p. 170).
- Upgrade your MySQL DB instance to MySQL version 5.6 or 5.7. For more information, see Upgrading a MySQL DB Snapshot (p. 857).
- If you cannot upgrade your instance or change the `optimizer_switch` parameter, you can work around the bug by explicitly identifying an index for the query, for example:

```
SELECT * FROM table1
USE INDEX covering_index
WHERE indexed_col1 = 'value1' AND indexed_col2 = 'value2';
```

For more information, go to Index Merge Optimization.

Log File Size

For MySQL version 5.6.20 and later, there is a size limit on BLOBs written to the redo log. To account for this limit, ensure that the `innodb_log_file_size` parameter for your MySQL DB instance is 10 times larger than the largest BLOB data size found in your tables, plus the length of other variable length fields (`VARCHAR`, `VARBINARY`, `TEXT`) in the same tables. For information on how to set parameter values, see Working with DB Parameter Groups (p. 170). For information on the redo log BLOB size limit, go to Changes in MySQL 5.6.20.

MySQL Parameter Exceptions for Amazon RDS DB Instances

Some MySQL parameters require special considerations when used with an Amazon RDS DB instance.

lower_case_table_names

Because Amazon RDS uses a case-sensitive file system, setting the value of the `lower_case_table_names` server parameter to 2 ("names stored as given but compared in lowercase") is not supported. Supported values for Amazon RDS DB instances are 0 ("names stored as given and comparisons are case-sensitive"), which is the default, or 1 ("names stored in lowercase and comparisons are not case-sensitive").

The `lower_case_table_names` parameter should be set as part of a custom DB parameter group before creating a DB instance. You should avoid changing the `lower_case_table_names` parameter for existing database instances because doing so could cause inconsistencies with point-in-time recovery backups and Read Replica DB instances.

Read Replicas should always use the same `lower_case_table_names` parameter value as the master DB instance.

long_query_time

You can set the `long_query_time` parameter to a floating point value which allows you to log slow queries to the MySQL slow query log with microsecond resolution. You can set a value such as 0.1

seconds, which would be 100 milliseconds, to help when debugging slow transactions that take less than one second.

MySQL File Size Limits

For Amazon RDS MySQL DB instances, the maximum provisioned storage limit constrains the size of a table to a maximum size of 16 TB when using InnoDB file-per-table tablespaces. This limit also constrains the system tablespace to a maximum size of 16 TB. InnoDB file-per-table tablespaces (with tables each in their own tablespace) is set by default for Amazon RDS MySQL DB instances.

> **Note**
> Some existing DB instances have a lower limit. For example, MySQL DB instances created prior to April 2014 have a file and table size limit of 2 TB. This 2 TB file size limit also applies to DB instances or Read Replicas created from DB snapshots taken prior to April 2014, regardless of when the DB instance was created.

There are advantages and disadvantages to using InnoDB file-per-table tablespaces, depending on your application. To determine the best approach for your application, go to InnoDB File-Per-Table Mode in the MySQL documentation.

We don't recommend allowing tables to grow to the maximum file size. In general, a better practice is to partition data into smaller tables, which can improve performance and recovery times.

One option that you can use for breaking a large table up into smaller tables is partitioning. Partitioning distributes portions of your large table into separate files based on rules that you specify. For example, if you store transactions by date, you can create partitioning rules that distribute older transactions into separate files using partitioning. Then periodically, you can archive the historical transaction data that doesn't need to be readily available to your application. For more information, go to https://dev.mysql.com/doc/refman/5.6/en/partitioning.html in the MySQL documentation.

To determine the file size of a table

- Use the following SQL command to determine if any of your tables are too large and are candidates for partitioning.

```
SELECT TABLE_SCHEMA, TABLE_NAME,
round(((DATA_LENGTH + INDEX_LENGTH) / 1024 / 1024), 2) As "Approximate size (MB)"
FROM information_schema.TABLES
WHERE TABLE_SCHEMA NOT IN ('mysql', 'information_schema', 'performance_schema');
```

To enable InnoDB file-per-table tablespaces

- To enable InnoDB file-per-table tablespaces, set the *innodb_file_per_table* parameter to `1` in the parameter group for the DB instance.

To disable InnoDB file-per-table tablespaces

- To disable InnoDB file-per-table tablespaces, set the *innodb_file_per_table* parameter to `0` in the parameter group for the DB instance.

For information on updating a parameter group, see Working with DB Parameter Groups (p. 170).

When you have enabled or disabled InnoDB file-per-table tablespaces, you can issue an `ALTER TABLE` command to move a table from the global tablespace to its own tablespace, or from its own tablespace to the global tablespace as shown in the following example:

```
ALTER TABLE table_name ENGINE=InnoDB;
```

Appendix: MySQL on Amazon RDS SQL Reference

This appendix describes system stored procedures that are available for Amazon RDS instances running the MySQL DB engine.

Overview

The following system stored procedures are supported for Amazon RDS DB instances running MySQL.

Replication

- mysql.rds_set_external_master (p. 914)
- mysql.rds_reset_external_master (p. 916)
- mysql.rds_start_replication (p. 917)
- mysql.rds_stop_replication (p. 918)
- mysql.rds_skip_repl_error (p. 918)
- mysql.rds_next_master_log (p. 919)

InnoDB cache warming

- mysql.rds_innodb_buffer_pool_dump_now (p. 921)
- mysql.rds_innodb_buffer_pool_load_now (p. 922)
- mysql.rds_innodb_buffer_pool_load_abort (p. 922)

Managing additional configuration (for example, binlog file retention)

- mysql.rds_set_configuration (p. 923)
- mysql.rds_show_configuration (p. 923)

Terminating a session or query

- mysql.rds_kill (p. 924)
- mysql.rds_kill_query (p. 925)

Logging

- mysql.rds_rotate_general_log (p. 926)
- mysql.rds_rotate_slow_log (p. 926)

Managing the global status history

- mysql.rds_enable_gsh_collector (p. 927)
- mysql.rds_set_gsh_collector (p. 927)
- mysql.rds_disable_gsh_collector (p. 928)
- mysql.rds_collect_global_status_history (p. 928)
- mysql.rds_enable_gsh_rotation (p. 928)
- mysql.rds_set_gsh_rotation (p. 929)
- mysql.rds_disable_gsh_rotation (p. 929)
- mysql.rds_rotate_global_status_history (p. 930)

SQL Reference Conventions

This section explains the conventions that are used to describe the syntax of the system stored procedures and tables described in the SQL reference section.

Character	Description
UPPERCASE	Words in uppercase are keywords.
[ ]	Square brackets indicate optional arguments.
{ }	Braces indicate that you are required to choose one of the arguments inside the braces.
|	Pipes separate arguments that you can choose.
italics	Words in italics indicate placeholders. You must insert the appropriate value in place of the word in italics.
...	An ellipsis indicates that you can repeat the preceding element.
'	Words in single quotes indicate that you must type the quotes.

mysql.rds_set_external_master

Configures a MySQL DB instance to be a Read Replica of an instance of MySQL running external to Amazon RDS.

Syntax

```
CALL mysql.rds_set_external_master (
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , mysql_binary_log_file_name
  , mysql_binary_log_file_location
  , ssl_encryption
);
```

Parameters

host_name

The host name or IP address of the MySQL instance running external to Amazon RDS that will become the replication master.

host_port

The port used by the MySQL instance running external to Amazon RDS to be configured as the replication master. If your network configuration includes SSH port replication that converts the port number, specify the port number that is exposed by SSH.

replication_user_name

The ID of a user with REPLICATION CLIENT and REPLICATION SLAVE permissions on the MySQL instance running external to Amazon RDS. We recommend that you provide an account that is used solely for replication with the external instance.

replication_user_password

The password of the user ID specified in `replication_user_name`.

mysql_binary_log_file_name

The name of the binary log on the replication master contains the replication information.

mysql_binary_log_file_location

The location in the `mysql_binary_log_file_name` binary log at which replication will start reading the replication information.

ssl_encryption

This option is not currently implemented. The default is 0.

Usage Notes

The `mysql.rds_set_external_master` procedure must be run by the master user. It must be run on the MySQL DB instance to be configured as the Read Replica of a MySQL instance running external to Amazon RDS.

Before you run `mysql.rds_set_external_master`, you must configure the instance of MySQL running external to Amazon RDS to be a replication master. To connect to the MySQL instance running external to Amazon RDS, you must specify `replication_user_name` and `replication_user_password` values that indicate a replication user that has `REPLICATION CLIENT` and `REPLICATION SLAVE` permissions on the external instance of MySQL.

To configure an external instance of MySQL as a replication master

1. Using the MySQL client of your choice, connect to the external instance of MySQL and create a user account to be used for replication. The following is an example:

```
CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'SomePassW0rd'
```

2. On the external instance of MySQL, grant `REPLICATION CLIENT` and `REPLICATION SLAVE` privileges to your replication user. The following example grants `REPLICATION CLIENT` and `REPLICATION SLAVE` privileges on all databases for the 'repl_user' user for your domain:

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com'
IDENTIFIED BY 'SomePassW0rd'
```

For more information, see Replication with a MySQL or MariaDB Instance Running External to Amazon RDS (p. 890).

Note
We recommend that you use Read Replicas to manage replication between two Amazon RDS DB instances when possible, and only use this and other replication-related stored procedures to enable more complex replication topologies between Amazon RDS DB instances. These stored procedures are primarily offered to enable replication with MySQL instances running external to Amazon RDS. For information about managing replication between Amazon RDS DB instances, see Working with PostgreSQL, MySQL, and MariaDB Read Replicas (p. 134).

After calling `mysql.rds_set_external_master` to configure an Amazon RDS DB instance as a Read Replica, you can call mysql.rds_start_replication (p. 917) on the replica to start the replication process. You can call mysql.rds_reset_external_master (p. 916) to remove the Read Replica configuration.

When `mysql.rds_set_external_master` is called, Amazon RDS records the time, user, and an action of "set master" in the `mysql.rds_history` and `mysql.rds_replication_status` tables.

The `mysql.rds_set_external_master` procedure is available in these versions of Amazon RDS MySQL:

- MySQL 5.5
- MySQL 5.6
- MySQL 5.7

Examples

When run on a MySQL DB instance, the following example configures the DB instance to be a Read Replica of an instance of MySQL running external to Amazon RDS.

```
call mysql.rds_set_external_master(
  'Externaldb.some.com',
  3306,
  'repl_user'@'mydomain.com',
  'SomePassW0rd',
  'mysql-bin-changelog.0777',
  120,
  0);
```

Related Topics

- mysql.rds_reset_external_master (p. 916)
- mysql.rds_start_replication (p. 917)
- mysql.rds_stop_replication (p. 918)

mysql.rds_reset_external_master

Reconfigures a MySQL DB instance to no longer be a Read Replica of an instance of MySQL running external to Amazon RDS.

Syntax

```
CALL mysql.rds_reset_external_master;
```

Usage Notes

The `mysql.rds_reset_external_master` procedure must be run by the master user. It must be run on the MySQL DB instance to be removed as a Read Replica of a MySQL instance running external to Amazon RDS.

> **Note**
> We recommend that you use Read Replicas to manage replication between two Amazon RDS DB instances when possible, and only use this and other replication-related stored procedures to enable more complex replication topologies between Amazon RDS DB instances. These stored procedures are primarily offered to enable replication with MySQL instances running external to Amazon RDS. For information about managing replication between Amazon RDS DB instances, see Working with PostgreSQL, MySQL, and MariaDB Read Replicas (p. 134).

For more information about using replication to import data from an instance of MySQL running external to Amazon RDS, see Importing Data into an Amazon RDS MySQL DB Instance (p. 860).

The `mysql.rds_reset_external_master` procedure is available in these versions of Amazon RDS MySQL:

- MySQL 5.5
- MySQL 5.6
- MySQL 5.7

Related Topics

- mysql.rds_set_external_master (p. 914)
- mysql.rds_start_replication (p. 917)
- mysql.rds_stop_replication (p. 918)

mysql.rds_start_replication

Initiates replication from a MySQL DB instance.

Syntax

```
CALL mysql.rds_start_replication;
```

Usage Notes

The `mysql.rds_start_replication` procedure must be run by the master user.

If you are configuring replication to import data from an instance of MySQL running external to Amazon RDS, you call `mysql.rds_start_replication` on the replica to start the replication process after you have called mysql.rds_set_external_master (p. 914) to build the replication configuration. For more information, see Importing Data into an Amazon RDS MySQL DB Instance (p. 860).

If you are configuring replication to export data to an instance of MySQL external to Amazon RDS, you call `mysql.rds_start_replication` and `mysql.rds_stop_replication` on the replica to control some replication actions, such as purging binary logs. For more information, see Exporting Data from a MySQL DB Instance by Using Replication (p. 893).

You can also call `mysql.rds_start_replication` on the replica to restart any replication process that you previously stopped by calling mysql.rds_stop_replication (p. 918). For more information, see Working with PostgreSQL, MySQL, and MariaDB Read Replicas (p. 134).

The `mysql.rds_start_replication` procedure is available in these versions of Amazon RDS MySQL:

- MySQL 5.5
- MySQL 5.6
- MySQL 5.7

Related Topics

- mysql.rds_set_external_master (p. 914)

- mysql.rds_reset_external_master (p. 916)
- mysql.rds_stop_replication (p. 918)

mysql.rds_stop_replication

Terminates replication from a MySQL DB instance.

Syntax

```
CALL mysql.rds_stop_replication;
```

Usage Notes

The `mysql.rds_stop_replication` procedure must be run by the master user.

If you are configuring replication to import data from an instance of MySQL running external to Amazon RDS, you call `mysql.rds_stop_replication` on the replica to stop the replication process after the import has completed. For more information, see Importing Data into an Amazon RDS MySQL DB Instance (p. 860).

If you are configuring replication to export data to an instance of MySQL external to Amazon RDS, you call `mysql.rds_start_replication` and `mysql.rds_stop_replication` on the replica to control some replication actions, such as purging binary logs. For more information, see Exporting Data from a MySQL DB Instance by Using Replication (p. 893).

You can also use `mysql.rds_stop_replication` to stop replication between two Amazon RDS DB instances. You typically stop replication to perform a long running operation on the replica, such as creating a large index on the replica. You can restart any replication process that you stopped by calling mysql.rds_start_replication (p. 917) on the replica. For more information, see Working with PostgreSQL, MySQL, and MariaDB Read Replicas (p. 134).

The `mysql.rds_stop_replication` procedure is available in these versions of Amazon RDS MySQL:

- MySQL 5.5
- MySQL 5.6
- MySQL 5.7

Related Topics

- mysql.rds_set_external_master (p. 914)
- mysql.rds_reset_external_master (p. 916)
- mysql.rds_start_replication (p. 917)

mysql.rds_skip_repl_error

Skips and deletes a replication error on a MySQL DB instance.

Syntax

```
CALL mysql.rds_skip_repl_error;
```

Usage Notes

The `mysql.rds_skip_repl_error` must be run by the master user.

Run the MySQL `show slave status\G` command to determine if there are errors. If a replication error is not critical, you can elect to use `mysql.rds_skip_repl_error` to skip the error. If there are multiple errors, `mysql.rds_skip_repl_error` deletes the first error, then warns that others are present. You can then use `show slave status\G` to determine the correct course of action for the next error. For information about the values returned, go to SHOW SLAVE STATUS Syntax in the MySQL documentation.

For more information about addressing replication errors with Amazon RDS, see Troubleshooting a MySQL or MariaDB Read Replica Problem (p. 150).

The `mysql.rds_skip_repl_error` procedure is available in these versions of Amazon RDS MySQL:

- MySQL 5.5
- MySQL 5.6
- MySQL 5.7

> **Important**
> If you attempt to call `mysql.rds_skip_repl_error` and encounter the following error: `ERROR 1305 (42000): PROCEDURE mysql.rds_skip_repl_error does not exist`, then upgrade your MySQL DB instance to the latest minor version or one of the minimum minor versions listed in this topic.

Slave Down or Disabled Error

When you call the `mysql.rds_skip_repl_error` command, you might receive the following error message: `Slave is down or disabled.`

This error message appears because replication has stopped and could not be restarted.

If you need to skip a large number of errors, the replication lag can increase beyond the default retention period for binary log files. In this case, you might encounter a fatal error due to binary log files being purged before they have been replayed on the replica. This purge causes replication to stop, and you can no longer call the `mysql.rds_skip_repl_error` command to skip replication errors.

You can mitigate this issue by increasing the number of hours that binary log files are retained on your replication master. After you have increased the binlog retention time, you can restart replication and call the `mysql.rds_skip_repl_error` command as needed.

To set the binlog retention time, use the mysql.rds_set_configuration (p. 923) procedure and specify a configuration parameter of 'binlog retention hours' along with the number of hours to retain binlog files on the DB cluster, up to 720 (30 days). The following example sets the retention period for binlog files to 48 hours:

```
CALL mysql.rds_set_configuration('binlog retention hours', 48);
```

mysql.rds_next_master_log

Changes the replication master log position to the start of the next binary log on the master. Use this procedure only if you are receiving replication I/O error 1236 on a Read Replica.

Syntax

```
CALL mysql.rds_next_master_log(
```

```
curr_master_log
);
```

Parameters

curr_master_log

The index of the current master log file. For example, if the current file is named `mysql-bin-changelog.012345`, then the index is 12345. To determine the current master log file name, run the `SHOW SLAVE STATUS` command and view the `Master_Log_File` field.

Usage Notes

The `mysql.rds_next_master_log` procedure must be run by the master user.

Warning
Call `mysql.rds_next_master_log` only if replication fails after a failover of a Multi-AZ DB instance that is the replication source, and the `Last_IO_Errno` field of `SHOW SLAVE STATUS` reports I/O error 1236.
Calling `mysql.rds_next_master_log` may result in data loss in the Read Replica if transactions in the source instance were not written to the binary log on disk before the failover event occurred. You can reduce the chance of this happening by configuring the source instance parameters sync_binlog = 1 and innodb_support_xa = 1, although this may reduce performance. For more information, see Working with PostgreSQL, MySQL, and MariaDB Read Replicas (p. 134).

The `mysql.rds_next_master_log` procedure is available in these versions of Amazon RDS MySQL:

- MySQL 5.5
- MySQL 5.6
- MySQL 5.7

Examples

Assume replication fails on an Amazon RDS Read Replica. Running `SHOW SLAVE STATUS\G` on the replica returns the following result:

```
*************************** 1. row ***************************
               Slave_IO_State:
                  Master_Host: myhost.XXXXXXXXXXXXXX.rr-rrrr-1.rds.amazonaws.com
                  Master_User: MasterUser
                  Master_Port: 3306
                Connect_Retry: 10
              Master_Log_File: mysql-bin-changelog.012345
          Read_Master_Log_Pos: 1219393
               Relay_Log_File: relaylog.012340
                Relay_Log_Pos: 30223388
        Relay_Master_Log_File: mysql-bin-changelog.012345
             Slave_IO_Running: No
            Slave_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
```

```
                   Last_Error:
                 Skip_Counter: 0
          Exec_Master_Log_Pos: 30223232
              Relay_Log_Space: 5248928866
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Master_SSL_Allowed: No
           Master_SSL_CA_File:
           Master_SSL_CA_Path:
              Master_SSL_Cert:
            Master_SSL_Cipher:
               Master_SSL_Key:
        Seconds_Behind_Master: NULL
Master_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 1236
                Last_IO_Error: Got fatal error 1236 from master when reading data from
 binary log: 'Client requested master to start replication from impossible position; the
 first event 'mysql-bin-changelog.013406' at 1219393, the last event read from '/rdsdbdata/
log/binlog/mysql-bin-changelog.012345' at 4, the last byte read from '/rdsdbdata/log/
binlog/mysql-bin-changelog.012345' at 4.'
               Last_SQL_Errno: 0
               Last_SQL_Error:
  Replicate_Ignore_Server_Ids:
             Master_Server_Id: 67285976
```

The `Last_IO_Errno` field shows that the instance is receiving I/O error 1236. The `Master_Log_File` field shows that the file name is `mysql-bin-changelog.012345`, which means that the log file index is `12345`. To resolve the error, you can call `mysql.rds_next_master_log` with the following parameter:

```
CALL mysql.rds_next_master_log(12345);
```

mysql.rds_innodb_buffer_pool_dump_now

Dumps the current state of the buffer pool to disk. For more information, see InnoDB Cache Warming (p. 827).

Syntax

```
CALL mysql.rds_innodb_buffer_pool_dump_now();
```

Usage Notes

The `mysql.rds_innodb_buffer_pool_dump_now` procedure must be run by the master user.

The `mysql.rds_innodb_buffer_pool_dump_now` procedure is available in these versions of Amazon RDS MySQL:

- MySQL 5.6
- MySQL 5.7

Related Topics

- mysql.rds_innodb_buffer_pool_load_now (p. 922)

- mysql.rds_innodb_buffer_pool_load_abort (p. 922)

mysql.rds_innodb_buffer_pool_load_now

Loads the saved state of the buffer pool from disk. For more information, see InnoDB Cache Warming (p. 827).

Syntax

```
CALL mysql.rds_innodb_buffer_pool_load_now();
```

Usage Notes

The `mysql.rds_innodb_buffer_pool_load_now` procedure must be run by the master user.

The `mysql.rds_innodb_buffer_pool_load_now` procedure is available in these versions of Amazon RDS MySQL:

- MySQL 5.6
- MySQL 5.7

Related Topics

- mysql.rds_innodb_buffer_pool_dump_now (p. 921)
- mysql.rds_innodb_buffer_pool_load_abort (p. 922)

mysql.rds_innodb_buffer_pool_load_abort

Cancels a load of the saved buffer pool state while in progress. For more information, see InnoDB Cache Warming (p. 827).

Syntax

```
CALL mysql.rds_innodb_buffer_pool_load_abort();
```

Usage Notes

The `mysql.rds_innodb_buffer_pool_load_abort` procedure must be run by the master user.

The `mysql.rds_innodb_buffer_pool_load_abort` procedure is available in these versions of Amazon RDS MySQL:

- MySQL 5.6
- MySQL 5.7

Related Topics

- mysql.rds_innodb_buffer_pool_dump_now (p. 921)

- mysql.rds_innodb_buffer_pool_load_now (p. 922)

mysql.rds_set_configuration

Specifies the number of hours to retain binary logs.

Syntax

```
CALL mysql.rds_set_configuration(name,value);
```

Parameters

name

The name of the configuration parameter to set.

value

The value of the configuration parameter.

Usage Notes

The `mysql.rds_set_configuration` procedure currently supports only the `binlog retention hours` configuration parameter. The `binlog retention hours` parameter is used to specify the number of hours to retain binary log files. Amazon RDS normally purges a binary log as soon as possible, but the binary log might still be required for replication with a MySQL database external to Amazon RDS. The default value of `binlog retention hours` is NULL (do not retain binary logs).

To specify the number of hours for Amazon RDS to retain binary logs on a DB instance, use the `mysql.rds_set_configuration` stored procedure and specify a period with enough time for replication to occur, as shown in the following example.

```
call mysql.rds_set_configuration('binlog retention hours', 24);
```

For MySQL DB instances, the maximum `binlog retention hours` value is 168 (7 days). For Amazon Aurora DB instances, the maximum is 720 (30 days).

After you set the retention period, monitor storage usage for the DB instance to ensure that the retained binary logs don't take up too much storage.

The `mysql.rds_set_configuration` is available in these versions of Amazon RDS MySQL:

- MySQL 5.6
- MySQL 5.7

Related Topics

- mysql.rds_show_configuration (p. 923)

mysql.rds_show_configuration

The number of hours that binary logs are retained.

Syntax

```
CALL mysql.rds_show_configuration;
```

Usage Notes

To verify the number of hours Amazon RDS will retain binary logs, use the `mysql.rds_show_configuration` stored procedure.

The `mysql.rds_show_configuration` procedure is available in these versions of Amazon RDS MySQL:

- MySQL 5.6
- MySQL 5.7

Related Topics

- mysql.rds_set_configuration (p. 923)

Examples

The following example displays the retention period:

```
call mysql.rds_show_configuration;
                name                          value     description
                binlog retention hours        24        binlog retention hours specifies the
 duration in hours before binary logs are automatically deleted.
```

mysql.rds_kill

Terminates a connection to the MySQL server.

Syntax

```
CALL mysql.rds_kill(processID);
```

Parameters

processID

The identity of the connection thread that will be terminated.

Usage Notes

Each connection to the MySQL server runs in a separate thread. To terminate a connection, use the `mysql.rds_kill` procedure and pass in the thread ID of that connection. To obtain the thread ID, use the MySQL SHOW PROCESSLIST command.

The `mysql.rds_kill` procedure is available in these versions of Amazon RDS MySQL:

- MySQL 5.5

- MySQL 5.6
- MySQL 5.7

Related Topics

- mysql.rds_kill_query (p. 925)

Examples

The following example terminates a connection with a thread ID of 4243:

```
call mysql.rds_kill(4243);
```

mysql.rds_kill_query

Terminates a query running against the MySQL server.

Syntax

```
CALL mysql.rds_kill_query(queryID);
```

Parameters

queryID

The identity of the query that will be terminated.

Usage Notes

To terminate a query running against the MySQL server, use the `mysql_rds_kill_query` procedure and pass in the ID of that query. To obtain the query ID, use the MySQL INFORMATION_SCHEMA PROCESSLIST command. The connection to the MySQL server is retained.

The `mysql_rds_kill_query` procedure is available in these versions of Amazon RDS MySQL:

- MySQL 5.5
- MySQL 5.6
- MySQL 5.7

Related Topics

- mysql.rds_kill (p. 924)

Examples

The following example terminates a query with a thread ID of 230040:

```
call mysql.rds_kill_query(230040);
```

mysql.rds_rotate_general_log

Rotates the `mysql.general_log` table to a backup table. For more information, see MySQL Database Log Files (p. 313).

Syntax

```
CALL mysql.rds_rotate_general_log;
```

Usage Notes

You can rotate the `mysql.general_log` table to a backup table by calling the `mysql.rds_rotate_general_log` procedure. When log tables are rotated, the current log table is copied to a backup log table and the entries in the current log table are removed. If a backup log table already exists, then it is deleted before the current log table is copied to the backup. You can query the backup log table if needed. The backup log table for the `mysql.general_log` table is named `mysql.general_log_backup`.

The `mysql.rds_rotate_general_log` procedure is available in these versions of Amazon RDS MySQL:

- MySQL 5.5
- MySQL 5.6
- MySQL 5.7

Related Topics

- mysql.rds_rotate_slow_log (p. 926)

mysql.rds_rotate_slow_log

Rotates the `mysql.slow_log` table to a backup table. For more information, see MySQL Database Log Files (p. 313).

Syntax

```
CALL mysql.rds_rotate_slow_log;
```

Usage Notes

You can rotate the `mysql.slow_log` table to a backup table by calling the `mysql.rds_rotate_slow_log` procedure. When log tables are rotated, the current log table is copied to a backup log table and the entries in the current log table are removed. If a backup log table already exists, then it is deleted before the current log table is copied to the backup.

You can query the backup log table if needed. The backup log table for the `mysql.slow_log` table is named `mysql.slow_log_backup`.

The `mysql.rds_rotate_slow_log` procedure is available in these versions of Amazon RDS MySQL:

- MySQL 5.5
- MySQL 5.6
- MySQL 5.7

Related Topics

- mysql.rds_rotate_general_log (p. 926)

mysql.rds_enable_gsh_collector

Enables the Global Status History (GoSH) to take default snapshots at intervals specified by `rds_set_gsh_collector`. For more information, see Managing the Global Status History (p. 907).

Syntax

```
CALL mysql.rds_enable_gsh_collector;
```

Related Topics

- mysql.rds_set_gsh_collector (p. 927)
- mysql.rds_disable_gsh_collector (p. 928)
- mysql.rds_collect_global_status_history (p. 928)
- mysql.rds_enable_gsh_rotation (p. 928)
- mysql.rds_set_gsh_rotation (p. 929)
- mysql.rds_disable_gsh_rotation (p. 929)
- mysql.rds_rotate_global_status_history (p. 930)

mysql.rds_set_gsh_collector

Specifies the interval, in minutes, between snapshots taken by the Global Status History (GoSH). Default value is 5. For more information, see Managing the Global Status History (p. 907).

Syntax

```
CALL mysql.rds_set_gsh_collector(intervalPeriod);
```

Parameters

intervalPeriod

The interval, in minutes, between snapshots. Default value is 5.

Related Topics

- mysql.rds_enable_gsh_collector (p. 927)
- mysql.rds_disable_gsh_collector (p. 928)
- mysql.rds_collect_global_status_history (p. 928)
- mysql.rds_enable_gsh_rotation (p. 928)
- mysql.rds_set_gsh_rotation (p. 929)
- mysql.rds_disable_gsh_rotation (p. 929)
- mysql.rds_rotate_global_status_history (p. 930)

mysql.rds_disable_gsh_collector

Disables snapshots taken by the Global Status History (GoSH). For more information, see Managing the Global Status History (p. 907).

Syntax

```
CALL mysql.rds_disable_gsh_collector;
```

Related Topics

- mysql.rds_enable_gsh_collector (p. 927)
- mysql.rds_set_gsh_collector (p. 927)
- mysql.rds_collect_global_status_history (p. 928)
- mysql.rds_enable_gsh_rotation (p. 928)
- mysql.rds_set_gsh_rotation (p. 929)
- mysql.rds_disable_gsh_rotation (p. 929)
- mysql.rds_rotate_global_status_history (p. 930)

mysql.rds_collect_global_status_history

Takes a snapshot on demand for the Global Status History (GoSH). For more information, see Managing the Global Status History (p. 907).

Syntax

```
CALL mysql.rds_collect_global_status_history;
```

Related Topics

- mysql.rds_enable_gsh_collector (p. 927)
- mysql.rds_set_gsh_collector (p. 927)
- mysql.rds_disable_gsh_collector (p. 928)
- mysql.rds_enable_gsh_rotation (p. 928)
- mysql.rds_set_gsh_rotation (p. 929)
- mysql.rds_disable_gsh_rotation (p. 929)
- mysql.rds_rotate_global_status_history (p. 930)

mysql.rds_enable_gsh_rotation

Enables rotation of the contents of the `mysql.global_status_history` table to `mysql.global_status_history_old` at intervals specified by `rds_set_gsh_rotation`. For more information, see Managing the Global Status History (p. 907).

Syntax

```
CALL mysql.rds_enable_gsh_rotation;
```

Related Topics

- mysql.rds_enable_gsh_collector (p. 927)
- mysql.rds_set_gsh_collector (p. 927)
- mysql.rds_disable_gsh_collector (p. 928)
- mysql.rds_collect_global_status_history (p. 928)
- mysql.rds_set_gsh_rotation (p. 929)
- mysql.rds_disable_gsh_rotation (p. 929)
- mysql.rds_rotate_global_status_history (p. 930)

mysql.rds_set_gsh_rotation

Specifies the interval, in days, between rotations of the `mysql.global_status_history` table. Default value is 7. For more information, see Managing the Global Status History (p. 907).

Syntax

```
CALL mysql.rds_set_gsh_rotation(intervalPeriod);
```

Parameters

intervalPeriod

The interval, in days, between table rotations. Default value is 7.

Related Topics

- mysql.rds_enable_gsh_collector (p. 927)
- mysql.rds_set_gsh_collector (p. 927)
- mysql.rds_disable_gsh_collector (p. 928)
- mysql.rds_collect_global_status_history (p. 928)
- mysql.rds_enable_gsh_rotation (p. 928)
- mysql.rds_disable_gsh_rotation (p. 929)
- mysql.rds_rotate_global_status_history (p. 930)

mysql.rds_disable_gsh_rotation

Disables rotation of the `mysql.global_status_history` table. For more information, see Managing the Global Status History (p. 907).

Syntax

```
CALL mysql.rds_disable_gsh_rotation;
```

Related Topics

- mysql.rds_enable_gsh_collector (p. 927)

mysql.rds_rotate_global_status_history

Rotates the contents of the `mysql.global_status_history` table to `mysql.global_status_history_old` on demand. For more information, see Managing the Global Status History (p. 907).

Syntax

```
CALL mysql.rds_rotate_global_status_history;
```

Related Topics

- mysql.rds_enable_gsh_collector (p. 927)
- mysql.rds_set_gsh_collector (p. 927)
- mysql.rds_disable_gsh_collector (p. 928)
- mysql.rds_collect_global_status_history (p. 928)
- mysql.rds_enable_gsh_rotation (p. 928)
- mysql.rds_set_gsh_rotation (p. 929)
- mysql.rds_disable_gsh_rotation (p. 929)

Oracle on Amazon RDS

Amazon RDS supports DB instances running several versions and editions of Oracle Database. You can use the following versions and editions:

- Oracle 12c, Version 12.1.0.2
- Oracle 11g, Version 11.2.0.4

Amazon RDS also currently supports the following versions and editions that are on deprecation paths, because Oracle no longer provides patches for them:

- Oracle 12c, Version 12.1.0.1 (Deprecation of Oracle 12.1.0.1 (p. 945))
- Oracle 11g, Version 11.2.0.3 (Deprecation of Oracle 11.2.0.3 (p. 944))
- Oracle 11g, Version 11.2.0.2 (Deprecation of Oracle 11.2.0.2 (p. 944))

You can create DB instances and DB snapshots, point-in-time restores and automated or manual backups. DB instances running Oracle can be used inside a VPC. You can also enable various options to add additional features to your Oracle DB instance. Amazon RDS supports Multi-AZ deployments for Oracle as a high-availability, failover solution.

In order to deliver a managed service experience, Amazon RDS does not provide shell access to DB instances, and it restricts access to certain system procedures and tables that require advanced privileges. Amazon RDS supports access to databases on a DB instance using any standard SQL client application such as Oracle SQL Plus. Amazon RDS does not allow direct host access to a DB instance via Telnet or Secure Shell (SSH).

When you create a DB instance, the master account that you use to create the instance gets DBA user privileges (with some limitations). Use this account for any administrative tasks such as creating additional user accounts in the database. The SYS user, SYSTEM user, and other administrative accounts are locked and cannot be used.

Before creating a DB instance, you should complete the steps in the Setting Up for Amazon RDS (p. 5) section of this guide.

Common Management Tasks for Oracle on Amazon RDS

The following are the common management tasks you perform with an Amazon RDS Oracle DB instance, with links to relevant documentation for each task.

Task Area	Relevant Documentation
Instance Classes, Storage, and PIOPS If you are creating a DB instance for production purposes, you should understand how instance classes, storage types, and Provisioned IOPS work in Amazon RDS.	DB Instance Class Support for Oracle (p. 934) Amazon RDS Storage Types (p. 410)
Multi-AZ Deployments	High Availability (Multi-AZ) (p. 99)

Task Area	Relevant Documentation
A production DB instance should use Multi-AZ deployments. Multi-AZ deployments provide increased availability, data durability, and fault tolerance for DB instances.	
Amazon Virtual Private Cloud (VPC) If your AWS account has a default VPC, then your DB instance is automatically created inside the default VPC. If your account does not have a default VPC, and you want the DB instance in a VPC, you must create the VPC and subnet groups before you create the DB instance.	Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform (p. 391) Working with an Amazon RDS DB Instance in a VPC (p. 399)
Security Groups By default, DB instances are created with a firewall that prevents access to them. You therefore must create a security group with the correct IP addresses and network configuration to access the DB instance. The security group you create depends on what Amazon EC2 platform your DB instance is on, and whether you will access your DB instance from an Amazon EC2 instance. In general, if your DB instance is on the *EC2-Classic* platform, you will need to create a DB security group; if your DB instance is on the *EC2-VPC* platform, you will need to create a VPC security group.	Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform (p. 391) Amazon RDS Security Groups (p. 375)
Parameter Groups If your DB instance is going to require specific database parameters, you should create a parameter group before you create the DB instance.	Working with DB Parameter Groups (p. 170)
Option Groups If your DB instance is going to require specific database options, you should create an option group before you create the DB instance.	Options for Oracle DB Instances (p. 993)
Connecting to Your DB Instance After creating a security group and associating it to a DB instance, you can connect to the DB instance using any standard SQL client application such as Oracle SQL Plus.	Connecting to a DB Instance Running the Oracle Database Engine (p. 959)
Backup and Restore You can configure your DB instance to take automated backups, or take manual snapshots, and then restore instances from the backups or snapshots.	Backing Up and Restoring Amazon RDS DB Instances (p. 200)
Monitoring You can monitor an Oracle DB instance by using CloudWatch Amazon RDS metrics, events, and enhanced monitoring.	Viewing DB Instance Metrics (p. 254) Viewing Amazon RDS Events (p. 301)
Log Files You can access the log files for your Oracle DB instance.	Amazon RDS Database Log Files (p. 303)

There are also advanced tasks and optional features for working with Oracle DB instances. For more information, see the following documentation:

- For information on common DBA tasks for Oracle on Amazon RDS, see Common DBA Tasks for Oracle DB Instances (p. 1042).
- For information on Oracle GoldenGate support, see Using Oracle GoldenGate with Amazon RDS (p. 1101).
- For information on Siebel Customer Relationship Management (CRM) support, see Installing a Siebel Database on Oracle on Amazon RDS (p. 1117).

Oracle Licensing

There are two licensing options available for Amazon RDS for Oracle; License Included and Bring Your Own License (BYOL). After you create an Oracle DB instance on Amazon RDS, you can change the licensing model by using the AWS Management Console, the Amazon RDS API ModifyDBInstance action, or the AWS CLI modify-db-instance command.

License Included

In the License Included model, you don't need to purchase Oracle licenses separately; AWS holds the license for the Oracle database software. In this model, if you have an AWS Support account with case support, you contact AWS Support for both Amazon RDS and Oracle Database service requests.

The License Included model is supported on Amazon RDS for the following Oracle database editions:

- Oracle Database Standard Edition One (SE1)
- Oracle Database Standard Edition Two (SE2)

Bring Your Own License (BYOL)

In the Bring Your Own License model, you can use your existing Oracle Database licenses to run Oracle deployments on Amazon RDS. You must have the appropriate Oracle Database license (with Software Update License and Support) for the DB instance class and Oracle Database edition you wish to run. You must also follow Oracle's policies for licensing Oracle Database software in the cloud computing environment. For more information on Oracle's licensing policy for Amazon EC2, see Licensing Oracle Software in the Cloud Computing Environment.

In this model, you continue to use your active Oracle support account, and you contact Oracle directly for Oracle Database service requests. If you have an AWS Support account with case support, you can contact AWS Support for Amazon RDS issues. Amazon Web Services and Oracle have a multi-vendor support process for cases which require assistance from both organizations.

The Bring Your Own License model is supported on Amazon RDS for the following Oracle database editions:

- Oracle Database Enterprise Edition (EE)
- Oracle Database Standard Edition (SE)
- Oracle Database Standard Edition One (SE1)
- Oracle Database Standard Edition Two (SE2)

Licensing Oracle Multi-AZ Deployments

Amazon RDS supports Multi-AZ deployments for Oracle as a high-availability, failover solution. We recommend Multi-AZ for production workloads. For more information, see High Availability (Multi-AZ) (p. 99).

If you use the Bring Your Own License model, you must have a license for both the primary DB instance and the standby DB instance in a Multi-AZ deployment.

DB Instance Class Support for Oracle

The computation and memory capacity of a DB instance is determined by its DB instance class. The DB instance class you need depends on your processing power and memory requirements. For more information, see DB Instance Class (p. 92).

The following are the DB instance classes supported for Oracle.

Oracle Edition	Version 12.1.0.2 Support	Version 11.2.0.4 Support
Enterprise Edition (EE) Bring Your Own License (BYOL)	db.m4.large–db.m4.16xlarge db.m3.medium–db.m3.2xlarge db.m2.xlarge–db.m2.4xlarge db.m1.small–db.m1.xlarge db.r4.large–db.r4.16xlarge db.r3.large–db.r3.8xlarge db.t2.micro–db.t2.2xlarge	db.m4.large–db.m4.16xlarge db.m3.medium–db.m3.2xlarge db.m2.xlarge–db.m2.4xlarge db.m1.small–db.m1.xlarge db.r4.large–db.r4.16xlarge db.r3.large–db.r3.8xlarge db.t2.micro–db.t2.2xlarge
Standard Edition 2 (SE2) Bring Your Own License (BYOL)	db.m4.large–db.m4.4xlarge db.m3.medium–db.m3.2xlarge db.m2.xlarge–db.m2.4xlarge db.m1.small–db.m1.xlarge db.r4.large–db.r4.4xlarge db.r3.large–db.r3.4xlarge db.t2.micro–db.t2.2xlarge	—
Standard Edition 2 (SE2) License Included	db.m4.large–db.m4.4xlarge db.m3.medium–db.m3.2xlarge db.m2.xlarge–db.m2.4xlarge db.m1.small–db.m1.xlarge db.r4.large–db.r4.4xlarge db.r3.large–db.r3.4xlarge	—

Oracle Edition	Version 12.1.0.2 Support	Version 11.2.0.4 Support
	db.t2.micro–db.t2.2xlarge	
Standard Edition 1 (SE1) Bring Your Own License (BYOL)	—	db.m4.large–db.m4.4xlarge db.m3.medium–db.m3.2xlarge db.m2.xlarge–db.m2.4xlarge db.m1.small–db.m1.xlarge db.r4.large–db.r4.4xlarge db.r3.large–db.r3.4xlarge db.t2.micro–db.t2.2xlarge
Standard Edition 1 (SE1) License Included	—	db.m4.large–db.m4.4xlarge db.m3.medium–db.m3.2xlarge db.m2.xlarge–db.m2.4xlarge db.m1.small–db.m1.xlarge db.r3.large–db.r3.4xlarge db.t2.micro–db.t2.large
Standard Edition (SE) Bring Your Own License (BYOL)	—	db.m4.large–db.m4.4xlarge db.m3.medium–db.m3.2xlarge db.m2.xlarge–db.m2.4xlarge db.m1.small–db.m1.xlarge db.r4.large–db.r4.8xlarge db.r3.large–db.r3.8xlarge db.t2.micro–db.t2.2xlarge

Oracle Security

The Oracle database engine uses role-based security. A role is a collection of privileges that can be granted to or revoked from a user. A predefined role, named *DBA*, normally allows all administrative privileges on an Oracle database engine. The following privileges are not available for the DBA role on an Amazon RDS DB instance using the Oracle engine:

- Alter database
- Alter system
- Create any directory
- Drop any directory
- Grant any privilege
- Grant any role

When you create a DB instance, the master account that you use to create the instance gets DBA user privileges (with some limitations). Use this account for any administrative tasks such as creating additional user accounts in the database. The SYS user, SYSTEM user, and other administrative accounts are locked and cannot be used.

Amazon RDS Oracle supports SSL/TLS encrypted connections as well as the Oracle Native Network Encryption (NNE) option to encrypt connections between your application and your Oracle DB instance. For more information about using SSL with Oracle on Amazon RDS, see SSL Support for Oracle DB Instances (p. 936). For more information about the Oracle Native Network Encryption option, see Oracle Native Network Encryption (p. 1003).

SSL Support for Oracle DB Instances

Secure Sockets Layer (SSL) is an industry standard protocol used for securing network connections between client and server. After SSL version 3.0, the name was changed to Transport Layer Security (TLS), but it is still often referred to as SSL and we refer to the protocol as SSL. Amazon RDS supports SSL encryption for Oracle DB instances. Using SSL, you can encrypt a connection between your application client and your Oracle DB instance. SSL support is available in all AWS regions for Oracle.

You enable SSL encryption for an Oracle DB instance by adding the Oracle SSL option to the option group associated with the DB instance. Amazon RDS uses a second port, as required by Oracle, for SSL connections which allows both clear text and SSL-encrypted communication to occur at the same time between a DB instance and an Oracle client. For example, you can use the port with clear text communication to communicate with other resources inside a VPC while using the port with SSL-encrypted communication to communicate with resources outside the VPC.

For more information, see Oracle SSL (p. 1021).

Note
You can't use both SSL and Oracle native network encryption (NNE) on the same DB instance. Before you can use SSL encryption, you must disable any other connection encryption.

Oracle 12c with Amazon RDS

Amazon RDS supports Oracle version 12c, which includes Oracle Enterprise Edition and Oracle Standard Edition Two. Oracle version 12c brings over 500 new features and updates from the previous version. This section covers the features and changes important to using Oracle 12c on Amazon RDS. For a complete list of the changes, see the Oracle 12c documentation. For a complete list of features supported by each Oracle 12c edition, see Feature Availability by Edition.

Oracle 12c includes sixteen new parameters that impact your Amazon RDS DB instance, as well as eighteen new system privileges, several no longer supported packages, and several new option group settings. The following sections provide more information on these changes.

Amazon RDS Parameter Changes for Oracle 12c

Oracle 12c includes sixteen new parameters in addition to several parameters with new ranges and new default values.

The following table shows the new Amazon RDS parameters for Oracle 12c:

Name	Values	Modifi	Description
connection_brokers	CONNECTION_BROKERS = broker_description[,...]	N	Specifies connection broker types, the number of connection brokers of each

Name	Values	Modifi	Description
			type, and the maximum number of connections per broker.
db_index_compression_inheritance	TABLESPACE, TABL, ALL, NONE	Y	Displays the options that are set for table or tablespace level compression inheritance.
db_big_table_cache_percent_target	0-90	Y	Specifies the cache section target size for automatic big table caching, as a percentage of the buffer cache.
heat_map	ON,OFF	Y	Enables the database to track read and write access of all segments, as well as modification of database blocks, due to data manipulation language (DML) and data definition language (DDL) statements.
inmemory_clause_default	INMEMORY,NO INMEMORY	Y	INMEMORY_CLAUSE_DEFAULT enables you to specify a default In-Memory Column Store (IM column store) clause for new tables and materialized views.
inmemory_clause_default_memcompress	NO MEMCOMPRESS,MEMCOMPRESS FOR DML,MEMCOMPRESS FOR QUERY, MEMCOMPRESS FOR QUERY LOW,MEMCOMPRESS FOR QUERY HIGH,MEMCOMPRESS FOR CAPACITY,MEMCOMPRESS FOR CAPACITY LOW,MEMCOMPRESS FOR CAPACITY HIGH	Y	See INMEMORY_CLAUSE_DEFAULT.
inmemory_clause_default_priority	PRIORITY LOW,PRIORITY MEDIUM,PRIORITY HIGH,PRIORITY CRITICAL,PRIORITY NONE	Y	See INMEMORY_CLAUSE_DEFAULT.
inmemory_force	DEFAULT, OFF	Y	INMEMORY_FORCE allows you to specify whether tables and materialized view that are specified as INMEMORY are populated into the In-Memory Column Store (IM column store) or not.

Name	Values	Modifi	Description
inmemory_max_populate_servers	Null	N	INMEMORY_MAX_POPULATE_SERVERS specifies the maximum number of background populate servers to use for In-Memory Column Store (IM column store) population, so that these servers do not overload the rest of the system.
inmemory_query	ENABLE (default), DISABLE	Y	INMEMORY_QUERY is used to enable or disable in-memory queries for the entire database at the session or system level.
inmemory_size	0,104857600-27487790	6944	INMEMORY_SIZE sets the size of the In-Memory Column Store (IM column store) on a database instance.
inmemory_trickle_repopulate_servers_percent	0 to 50	Y	INMEMORY_TRICKLE_REPOPULATE_SERVERS_PERC limits the maximum number of background populate servers used for In-Memory Column Store (IM column store) repopulation, as trickle repopulation is designed to use only a small percentage of the populate servers.
max_string_size	STANDARD (default), EXTENDED	N	Controls the maximum size of VARCHAR2, NVARCHAR2, and RAW.
optimizer_adaptive_features	TRUE (default), FALSE	Y	Enables or disables all of the adaptive optimizer features.
optimizer_adaptive_reporting_only	TRUE,FALSE (default)	Y	Controls reporting-only mode for adaptive optimizations.
pdb_file_name_convert		N	Maps names of existing files to new file names.
pga_aggregate_limit	1-max of memory	Y	Specifies a limit on the aggregate PGA memory consumed by the instance.
processor_group_name		N	Instructs the database instance to run itself within the specified operating system processor group.
spatial_vector_acceleration	TRUE,FALSE	N	Enables or disables the spatial vector acceleration, part of spacial option.
temp_undo_enabled	TRUE,FALSE (default)	Y	Determines whether transactions within a particular session can have a temporary undo log.
threaded_execution	TRUE,FALSE	N	Enables the multithreaded Oracle model, but prevents OS authentication.

Name	Values	Modifi	Description
unified_audit_sga_queue_size	1 MB - 30 MB	Y	Specifies the size of the system global area (SGA) queue for unified auditing.
use_dedicated_broker	TRUE,FALSE	N	Determines how dedicated servers are spawned.

Several parameter have new value ranges for Oracle 12c on Amazon RDS. The following table shows the old and new value ranges:

Parameter Name	12c Range	11g Range
audit_trail	os | db [, extended] | xml [, extended]	os | db [, extended] | xml [, extended] | true | false
compatible	Starts with 11.0.0	Starts with 10.0.0
db_securefile	PERMITTED | PREFERRED | ALWAYS | IGNORE | FORCE	PERMITTED | ALWAYS | IGNORE | FORCE
db_writer_processes	1-100	1-36
optimizer_features_enable	8.0.0 to 12.1.0.2	8.0.0 to 11.2.0.4
parallel_degree_policy	MANUAL,LIMITED,AUTO,ADAPTIVE	MANUAL,LIMITED,AUTO
parallel_min_server	0 to parallel_max_servers	CPU_COUNT * PARALLEL_THREADS_PER_CPU * 2 to parallel_max_servers

One parameters has a new default value for Oracle 12c on Amazon RDS. The following table shows the new default value:

Parameter Name	Oracle 12c Default Value	Oracle 11g Default Value
job_queue_processes	50	1000

Parameters in Amazon RDS are managed using parameter groups. See Working with DB Parameter Groups (p. 170) for more information. To view the supported parameters for a specific Oracle edition and version, you can run the AWS CLI `describe-engine-default-parameters` command.

For example, to view the supported parameters for Oracle Enterprise Edition, version 12c, run the following command:

```
aws rds describe-engine-default-parameters --db-parameter-group-family oracle-ee-12.1
```

Amazon RDS System Privileges for Oracle 12c

Several new system privileges have been granted to the system account for Oracle 12c. These new system privileges include:

- ALTER ANY CUBE BUILD PROCESS
- ALTER ANY MEASURE FOLDER
- ALTER ANY SQL TRANSLATION PROFILE
- CREATE ANY SQL TRANSLATION PROFILE
- CREATE SQL TRANSLATION PROFILE
- DROP ANY SQL TRANSLATION PROFILE
- EM EXPRESS CONNECT
- EXEMPT DDL REDACTION POLICY
- EXEMPT DML REDACTION POLICY
- EXEMPT REDACTION POLICY
- LOGMINING
- REDEFINE ANY TABLE
- SELECT ANY CUBE BUILD PROCESS
- SELECT ANY MEASURE FOLDER
- USE ANY SQL TRANSLATION PROFILE

Amazon RDS Options for Oracle 12c

Several Oracle options changed between Oracle 11g and Oracle 12c, though most of the options remain the same between the two versions. The Oracle 12c changes include the following:

- Oracle Enterprise Manager Database Express 12c replaced Oracle Enterprise Manager 11g Database Control. For more information, see Oracle Enterprise Manager Database Express (p. 1007).
- The option XMLDB is installed by default in Oracle 12c. You no longer need to install this option yourself.

Amazon RDS PL/SQL Packages for Oracle 12c

Oracle 12c includes a number of new built-in PL/SQL packages. The packages included with Amazon RDS Oracle 12c include the following:

Package Name	Description
CTX_ANL	The CTX_ANL package is used with AUTO_LEXER and provides procedures for adding and dropping a custom dictionary from the lexer.
DBMS_APP_CONT	The DBMS_APP_CONT package provides an interface to determine if the in-flight transaction on a now unavailable session committed or not, and if the last call on that session completed or not.
DBMS_AUTO_REPORT	The DBMS_AUTO_REPORT package provides an interface to view SQL Monitoring and Real-time Automatic Database Diagnostic Monitor (ADDM) data that has been captured into Automatic Workload Repository (AWR).
DBMS_GOLDENGATE_AUTH	The DBMS_GOLDENGATE_AUTH package provides subprograms for granting privileges to and revoking privileges from GoldenGate administrators.

Package Name	Description
DBMS_HEAT_MAP	The DBMS_HEAT_MAP package provides an interface to externalize heatmaps at various levels of storage including block, extent, segment, object and tablespace.
DBMS_ILM	The DBMS_ILM package provides an interface for implementing Information Lifecycle Management (ILM) strategies using Automatic Data Optimization (ADO) policies.
DBMS_ILM_ADMIN	The DBMS_ILM_ADMIN package provides an interface to customize Automatic Data Optimization (ADO) policy execution.
DBMS_PART	The DBMS_PART package provides an interface for maintenance and management operations on partitioned objects.
DBMS_PRIVILEGE_CAPTURE	The DBMS_PRIVILEGE_CAPTURE package provides an interface to database privilege analysis.
DBMS_QOPATCH	The DBMS_QOPATCH package provides an interface to view the installed database patches.
DBMS_REDACT	The DBMS_REDACT package provides an interface to Oracle Data Redaction, which enables you to mask (redact) data that is returned from queries issued by low-privileged users or an application.
DBMS_SPD	The DBMS_SPD package provides subprograms for managing SQL plan directives (SPD).
DBMS_SQL_TRANSLATOR	The DBMS_SQL_TRANSLATOR package provides an interface for creating, configuring, and using SQL translation profiles.
DBMS_SQL_MONITOR	The DBMS_SQL_MONITOR package provides information about real-time SQL Monitoring and real-time Database Operation Monitoring.
DBMS_SYNC_REFRESH	The DBMS_SYNC_REFRESH package provides an interface to perform a synchronous refresh of materialized views.
DBMS_TSDP_MANAGE	The DBMS_TSDP_MANAGE package provides an interface to import and manage sensitive columns and sensitive column types in the database, and is used in conjunction with the DBMS_TSDP_PROTECT package with regard to transparent sensitive data protection (TSDP) policies. DBMS_TSDP_MANAGE is available with the Enterprise Edition only.
DBMS_TSDP_PROTECT	The DBMS_TSDP_PROTECT package provides an interface to configure transparent sensitive data protection (TSDP) policies in conjunction with the DBMS_TSDP_MANAGE package. DBMS_TSDP_PROTECT is available with the Enterprise Edition only.
DBMS_XDB_CONFIG	The DBMS_XDB_CONFIG package provides an interface for configuring Oracle XML DB and its repository.
DBMS_XDB_CONSTANTS	The DBMS_XDB_CONSTANTS package provides an interface to commonly used constants. Users should use constants instead of dynamic strings to avoid typographical errors.
DBMS_XDB_REPOS	The DBMS_XDB_REPOS package provides an interface to operate on the Oracle XML database Repository.

Package Name	Description
DBMS_XMLSCHEMA_ANNOTATE	The DBMS_XMLSCHEMA_ANNOTATE package provides an interface to manage and configure the structured storage model, mainly through the use of pre-registration schema annotations.
DBMS_XMLSTORAGE_MANAGE	The DBMS_XMLSTORAGE_MANAGE package provides an interface to manage and modify XML storage after schema registration has been completed.
DBMS_XSTREAM_ADM	The DBMS_XSTREAM_ADM package provides interfaces for streaming database changes between an Oracle database and other systems. XStream enables applications to stream out or stream in database changes.
DBMS_XSTREAM_AUTH	The DBMS_XSTREAM_AUTH package provides subprograms for granting privileges to and revoking privileges from XStream administrators.
UTL_CALL_STACK	The UTL_CALL_STACK package provides an interface to provide information about currently executing subprograms.

Oracle 12c Features Not Supported

The following features are not supported for Oracle 12c on Amazon RDS:

- Automated Storage Management
- Data Guard / Active Data Guard
- Database Vault
- Java Support
- Multitenant Database
- Real Application Clusters (RAC)
- Unified Auditing

Several Oracle 11g PL/SQL packages are not supported in Oracle 12c. These packages include:

- DBMS_AUTO_TASK_IMMEDIATE
- DBMS_CDC_PUBLISH
- DBMS_CDC_SUBSCRIBE
- DBMS_EXPFIL
- DBMS_OBFUSCATION_TOOLKIT
- DBMS_RLMGR
- SDO_NET_MEM

Oracle 11g with Amazon RDS

Oracle 11g Supported Features

The following list shows the Oracle 11g features supported by Amazon RDS. For a complete list of features supported by each Oracle 11g edition, see Oracle Database 11g Editions.

- Total Recall
- Flashback Table, Query and Transaction Query
- Virtual Private Database
- Fine-Grained Auditing
- Comprehensive support for Microsoft .NET, OLE DB, and ODBC
- Automatic Memory Management
- Automatic Undo Management
- Advanced Compression
- Partitioning
- Star Query Optimization
- Summary Management - Materialized View Query Rewrite
- Oracle Data Redaction
- Distributed Queries/Transactions
- Text
- Materialized Views
- Import/Export and sqlldr Support
- Oracle Enterprise Manager Database Control
- Oracle XML DB (without the XML DB Protocol Server)
- Oracle Application Express
- Automatic Workload Repository for Enterprise Edition (AWR). For more information, see Working with Automatic Workload Repository (AWR) (p. 1058)
- Datapump (network only)
- Native network encryption
- Transparent data encryption (Oracle TDE), part of the Oracle Advanced Security feature

Oracle 11g Features Not Supported

The following features are not supported for Oracle 11g on Amazon RDS:

- Real Application Clusters (RAC)
- Real Application Testing
- Data Guard / Active Data Guard
- Oracle Enterprise Manager Grid Control
- Automated Storage Management
- Database Vault
- Streams
- Java Support
- Oracle Label Security
- Oracle XML DB Protocol Server

Amazon RDS Parameters for Oracle 11g

Parameters in Amazon RDS are managed using parameter groups. See Working with DB Parameter Groups (p. 170) for more information. To view the supported parameters for a specific Oracle edition and version, you can run the AWS CLI `describe-engine-default-parameters` command.

For example, to view the supported parameters for Oracle Enterprise Edition, version 11g, run the following command:

```
aws rds describe-engine-default-parameters --db-parameter-group-family oracle-ee-11.2
```

Oracle Engine Version Management

DB Engine Version Management is a feature of Amazon RDS that enables you to control when and how the database engine software running your DB instances is patched and upgraded. This feature gives you the flexibility to maintain compatibility with database engine patch versions, test new patch versions to ensure they work effectively with your application before deploying in production, and perform version upgrades on your own terms and timelines.

Note
Amazon RDS periodically aggregates official Oracle database patches using an Amazon RDS-specific DB Engine version. To see a list of which Oracle patches are contained in an Amazon RDS Oracle-specific engine version, go to Appendix: Oracle Database Engine Release Notes (p. 1120).

Currently, you perform all Oracle database upgrades manually. For more information about upgrading an Oracle DB instance, see Upgrading the Oracle DB Engine (p. 975).

Deprecation of Oracle 11.2.0.2

In 2017, Amazon RDS is deprecating support for Oracle version 11.2.0.2. Oracle is no longer providing patches for this version. Therefore, to provide the best experience for AWS customers, we are deprecating this version.

There are no longer any production DB instances running Oracle version 11.2.0.2. You might still have a snapshot of an 11.2.0.2 DB instance.

Amazon RDS is deprecating support for Oracle version 11.2.0.2 according to the following schedule.

Date	Information
August 4, 2016	You can no longer create DB instances that use Oracle version 11.2.0.2
April 15, 2018	Any 11.2.0.2 snapshots are upgraded to 11.2.0.4. You can upgrade your snapshots yourself prior to this date. For more information, see Upgrading an Oracle DB Snapshot (p. 980).

Deprecation of Oracle 11.2.0.3

In 2017, Amazon RDS is deprecating support for Oracle version 11.2.0.3. Oracle is no longer providing patches for this version. Therefore, to provide the best experience for AWS customers, we are deprecating this version.

There are no longer any production DB instances running Oracle version 11.2.0.3. You might still have a snapshot of an 11.2.0.3 DB instance.

Amazon RDS is deprecating support for Oracle version 11.2.0.3 according to the following schedule.

Date	Information
August 4, 2016	You can no longer create DB instances that use Oracle version 11.2.0.3.
March 15, 2018	Any 11.2.0.3 snapshots are upgraded to 11.2.0.4. You can upgrade your snapshots yourself prior to this date. For more information, see Upgrading an Oracle DB Snapshot (p. 980).

Deprecation of Oracle 12.1.0.1

In 2017, Amazon RDS is deprecating support for Oracle version 12.1.0.1. Oracle is no longer providing patches for this version. Therefore, to provide the best experience for AWS customers, we are deprecating this version.

There are no longer any production DB instances running Oracle version 12.1.0.1. You might still have a snapshot of an 12.1.0.1 DB instance.

Amazon RDS will deprecate support for Oracle version 12.1.0.1 according to the following schedule.

Date	Information
February 15, 2017	You can no longer create DB instances that use Oracle version 12.1.0.1.
June 1, 2018	Any 12.1.0.1 snapshots are upgraded to 12.1.0.2. You can upgrade your snapshots yourself prior to this date. For more information, see Upgrading an Oracle DB Snapshot (p. 980).

Using Huge Pages with an Oracle DB Instance

Amazon RDS for Oracle supports Linux kernel huge pages for increased database scalability. The use of huge pages results in smaller page tables and less CPU time spent on memory management, increasing the performance of large database instances. For more information, see Overview of HugePages in the Oracle documentation.

You can use huge pages with the following versions and editions of Oracle:

- 12.1.0.2, all editions
- 11.2.0.4, all editions

You can use huge pages with any DB instance class that has at least 14 GiB of memory. Huge pages are not supported for the db.m1, db.m2, and db.m3 DB instance classes. For more information, see Specifications for All Available DB Instance Classes (p. 92).

The `use_large_pages` parameter controls whether huge pages are enabled for a DB instance. The possible settings for this parameter are `ONLY`, `FALSE`, and `{DBInstanceClassHugePagesDefault}`. The `use_large_pages` parameter is set to `{DBInstanceClassHugePagesDefault}` in the default DB parameter group for Oracle.

To control whether huge pages are enabled for a DB instance automatically, you can use the `DBInstanceClassHugePagesDefault` formula variable in parameter groups. The value is determined as follows:

- For the current generation DB instance classes (db.t2, db.r3, and db.m4), `DBInstanceClassHugePagesDefault` always evaluates to `FALSE` by default. You can enable huge pages manually if the instance class has at least 14 GiB of memory.
- For next generation instance classes, such as db.r4, if the instance class has less than 100 GiB of memory, `DBInstanceClassHugePagesDefault` evaluates to `ONLY` by default.
- For next generation instance classes, such as db.r4, if the instance class has at least 100 GiB of memory, `DBInstanceClassHugePagesDefault` always evaluates to `ONLY`.

To enable huge pages for new or existing DB instances manually, set the `use_large_pages` parameter to `ONLY`. You can't use huge pages with Oracle Automatic Memory Management (AMM). If you set the parameter `use_large_pages` to `ONLY`, then you must also set both `memory_target` and `memory_max_target` to 0. For more information about setting DB parameters for your DB instance, see Working with DB Parameter Groups (p. 170).

You can also set the `sga_target`, `sga_max_size`, and `pga_aggregate_target` parameters. When you set system global area (SGA) and program global area (PGA) memory parameters, add the values together. Subtract this total from your available instance memory (`DBInstanceClassMemory`) to determine the free memory beyond the huge pages allocation. You must leave free memory of at least 2 GiB, or 10 percent of the total available instance memory, whichever is smaller.

The following is a sample parameter configuration for huge pages that enables huge pages manually. You should set the values to meet your needs.

```
memory_target            = 0
memory_max_target        = 0
pga_aggregate_target     = {DBInstanceClassMemory*1/8}
sga_target               = {DBInstanceClassMemory*3/4}
sga_max_size             = {DBInstanceClassMemory*3/4}
use_large_pages          = ONLY
```

Assume the following parameters values are set in a parameter group.

```
memory_target            = IF({DBInstanceClassHugePagesDefault}, 0,
 {DBInstanceClassMemory*3/4})
memory_max_target        = IF({DBInstanceClassHugePagesDefault}, 0,
 {DBInstanceClassMemory*3/4})
pga_aggregate_target     = IF({DBInstanceClassHugePagesDefault},
 {DBInstanceClassMemory*1/8}, 0)
sga_target               = IF({DBInstanceClassHugePagesDefault},
 {DBInstanceClassMemory*3/4}, 0)
sga_max_size             = IF({DBInstanceClassHugePagesDefault},
 {DBInstanceClassMemory*3/4}, 0)
use_large_pages          = {DBInstanceClassHugePagesDefault}
```

The parameter group is used by a next generation db.r4 instance class with less than 100 GiB of memory and a current generation db.r3 instance with more than 100 GiB memory. With these parameter settings and `use_large_pages` set to `{DBInstanceClassHugePagesDefault}`, huge pages are enabled on the db.r4 instance, but it is disabled on the db.r3 instance.

Consider another example with following parameters values set in a parameter group.

```
memory_target           = IF({DBInstanceClassHugePagesDefault}, 0,
 {DBInstanceClassMemory*3/4})
memory_max_target       = IF({DBInstanceClassHugePagesDefault}, 0,
 {DBInstanceClassMemory*3/4})
pga_aggregate_target    = IF({DBInstanceClassHugePagesDefault},
 {DBInstanceClassMemory*1/8}, 0)
sga_target              = IF({DBInstanceClassHugePagesDefault},
 {DBInstanceClassMemory*3/4}, 0)
sga_max_size            = IF({DBInstanceClassHugePagesDefault},
 {DBInstanceClassMemory*3/4}, 0)
use_large_pages         = FALSE
```

The parameter group is used by a next generation db.r4 instance class with less than 100 GiB of memory and a current generation db.r3 instance with more than 100 GiB memory. With these parameter settings, huge pages are disabled on both the db.r4 instance and the db.r3 instance.

> **Note**
> If this parameter group is used by a next generation db.r4 instance class with at least 100 GiB of memory, the `FALSE` setting for `use_large_pages` is overridden and set to `ONLY`. In this case, a customer notification regarding the override is sent.

After you configure your parameters, you must reboot your DB instance for the changes to take effect. For more information, see Rebooting a DB Instance (p. 119).

After huge pages are active on your DB instance, you can view huge pages information by enabling enhanced monitoring. For more information, see Enhanced Monitoring (p. 258).

Using utl_http, utl_tcp, and utl_smtp with an Oracle DB Instance

Amazon RDS supports outbound network access on your DB instances running Oracle. You can use `utl_http`, `utl_tcp`, and `utl_smtp` to connect from your DB instance to the network.

Note the following about working with outbound network access:

- To use `utl_http` on DB instances running Oracle 11g, you must install the XMLDB option. For more information, see Oracle XML DB (p. 1040).
- Outbound network access with `utl_http`, `utl_tcp`, and `utl_smtp` is supported only for Oracle DB instances in a VPC. To determine whether or not your DB instance is in a VPC, see Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform (p. 391). To move a DB instance not in a VPC into a VPC, see Moving a DB Instance Not in a VPC into a VPC (p. 405).
- To use SMTP with the UTL_MAIL option, see Oracle UTL_MAIL (p. 1038).
- The Domain Name Server (DNS) name of the remote host can be any of the following:
  - Publicly resolvable.
  - The endpoint of an Amazon RDS DB instance.
  - Resolvable through a custom DNS server. For more information, see Setting Up a Custom DNS Server (p. 1053).
  - The private DNS name of an Amazon EC2 instance in the same VPC or a peered VPC. In this case, make sure that the name is resolvable through a custom DNS server. Alternatively, to use the DNS provided by Amazon, you can enable the `enableDnsSupport` attribute in the VPC settings and enable DNS resolution support for the VPC peering connection. For more information, see DNS Support in Your VPC and Modifying Your VPC Peering Connection.

Using OEM, APEX, TDE, and Other Options

Most Amazon RDS DB engines support option groups that allow you to select additional features for your DB instance. Oracle DB instances support several options, including Oracle Enterprise Manager (OEM), Transparent Data Encryption (TDE), Application Express (APEX), and Native Network Encryption. For a complete list of supported Oracle options, see Options for Oracle DB Instances (p. 993). For more information about working with option groups, see Working with Option Groups (p. 153).

Creating a DB Instance Running the Oracle Database Engine

The basic building block of Amazon RDS is the DB instance. This is the environment in which you run your Oracle databases.

> **Important**
> You must complete the tasks in the Setting Up for Amazon RDS (p. 5) section before you can create or connect to a DB instance.

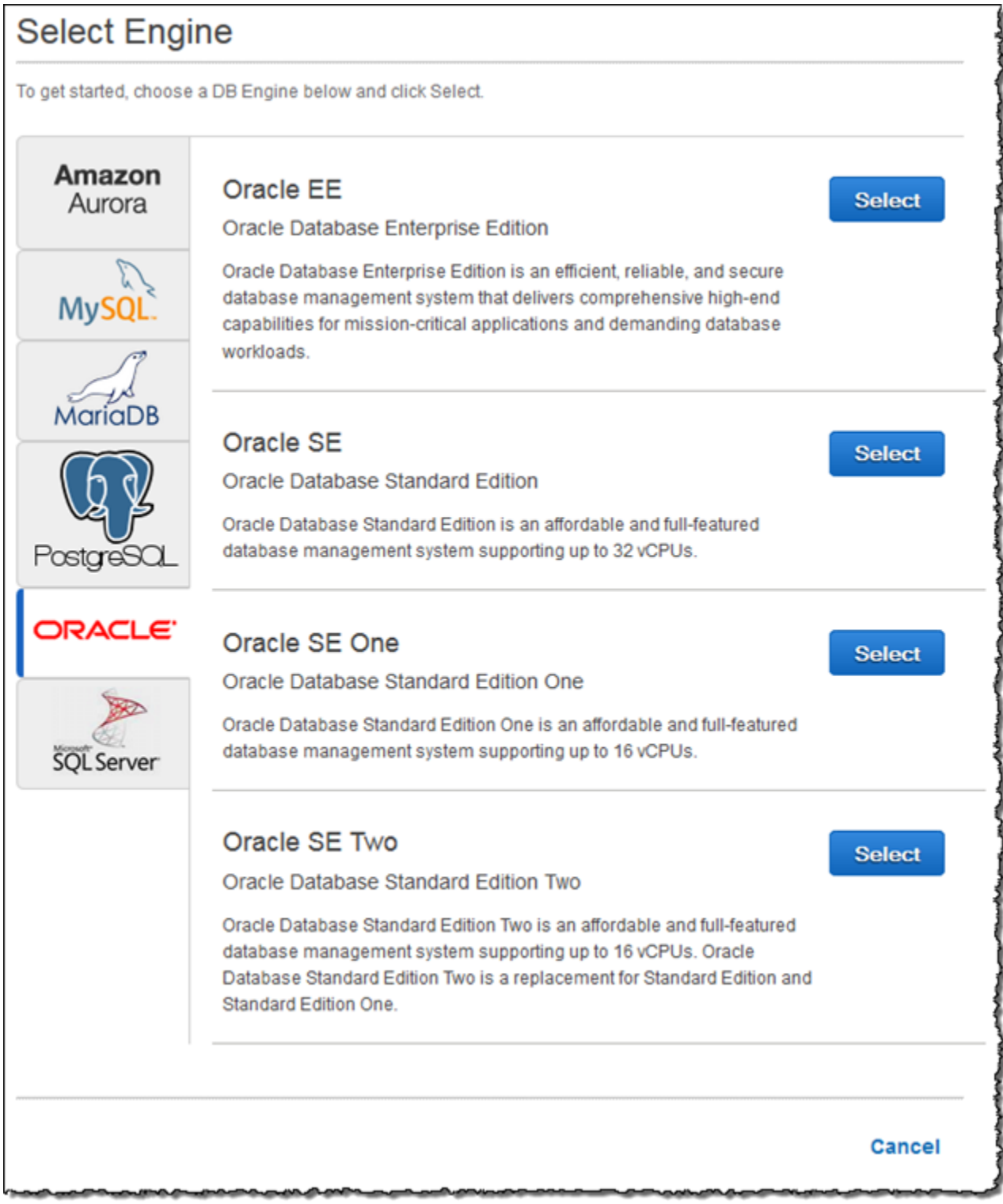
For an example that walks you through the process of creating and connecting to a sample DB instance, see Creating an Oracle DB Instance and Connecting to a Database on an Oracle DB Instance (p. 44).

AWS Management Console

To launch an Oracle DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the top right corner of the AWS Management Console, choose the region in which you want to create the DB instance.
3. In the navigation pane, choose **DB Instances**.
4. Choose **Launch DB Instance** to start the **Launch DB Instance Wizard**.

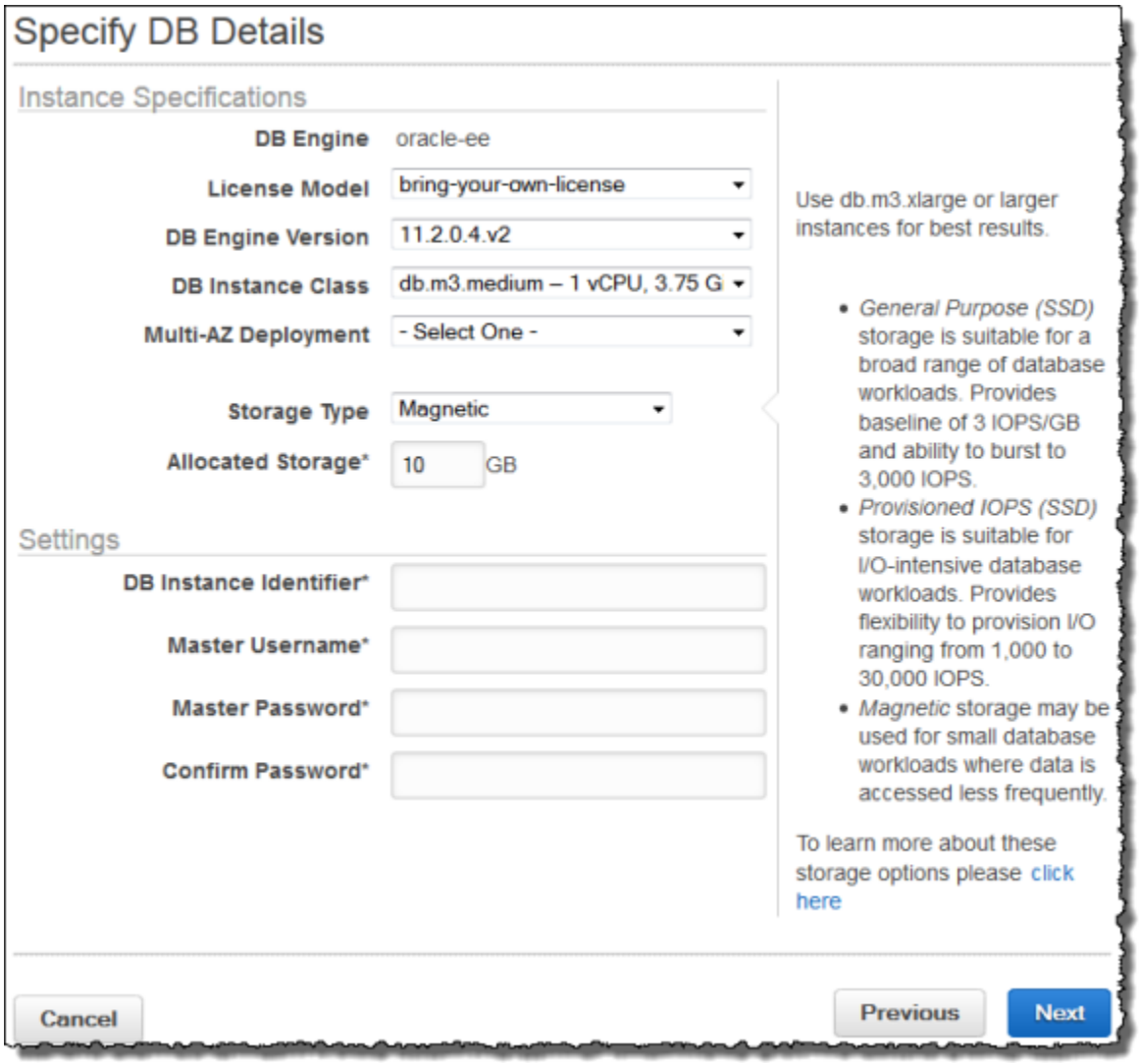
   The wizard opens on the **Select Engine** page. The Oracle editions that are available vary by region.



5. In the **Select Engine** window, choose the **Select** button for the Oracle DB engine you want to use.
6. The next step asks if you are planning to use the DB instance you are creating for production. If you are, choose **Yes**. By choosing **Yes**, the failover option **Multi-AZ** and the **Provisioned IOPS** storage option will be preselected in the following step.

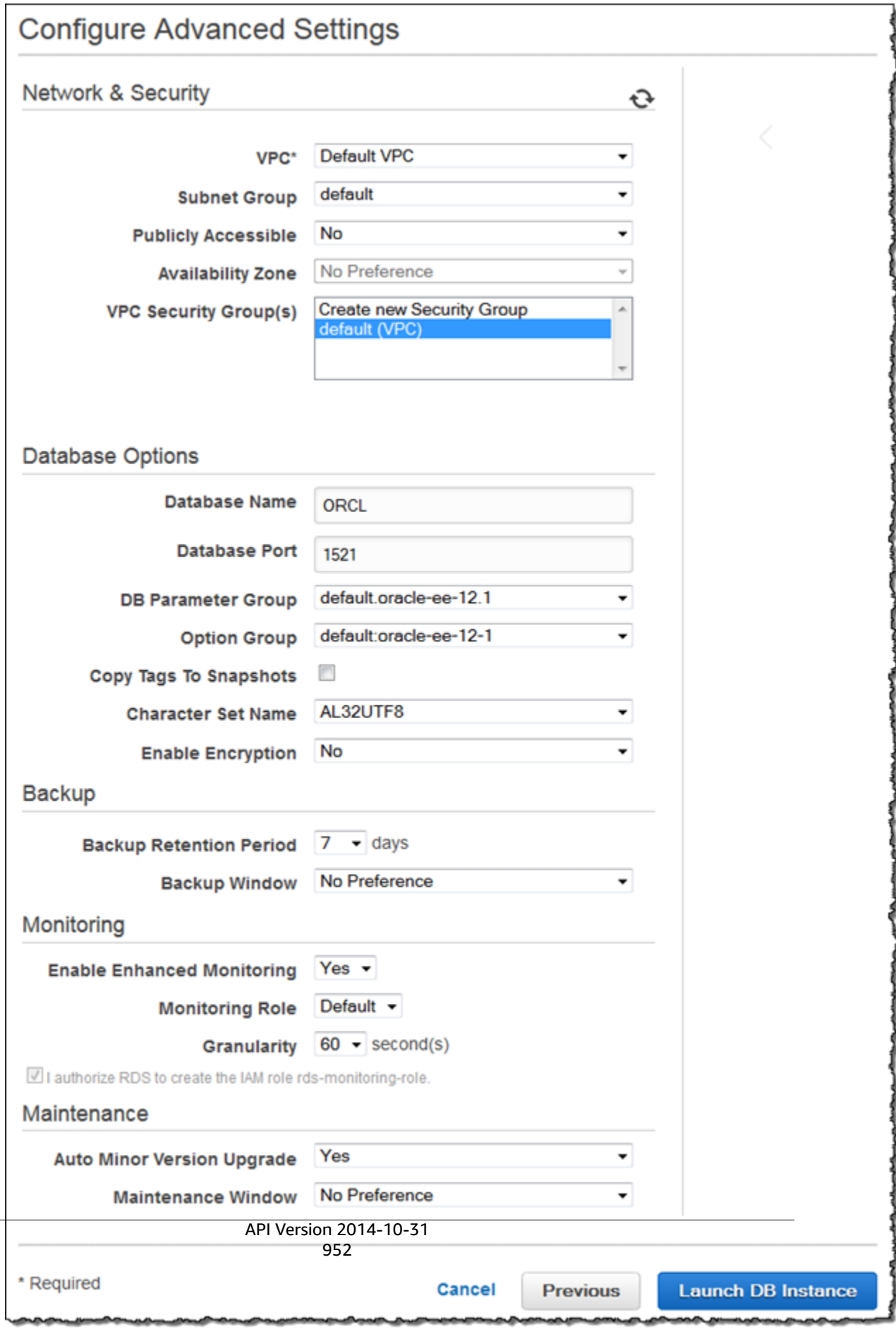
7. Choose **Next** to continue. The **Specify DB Details** page appears.

   On the **Specify DB Details** page, specify your DB instance information. For information about each setting, see Settings for Oracle DB Instances (p. 955).



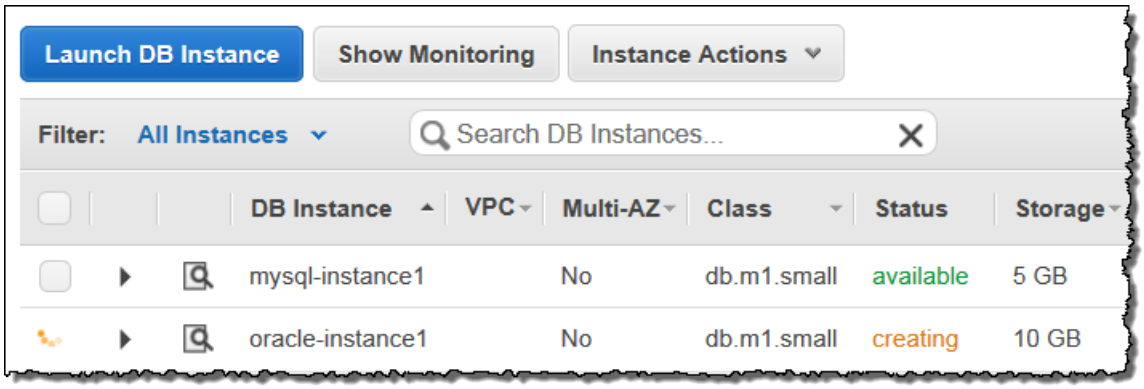
8. Choose **Next** to continue. The **Configure Advanced Settings** page appears.

   On the **Configure Advanced Settings** page, provide additional information that RDS needs to launch the DB instance. For information about each setting, see Settings for Oracle DB Instances (p. 955).


Configure Advanced Settings
Network & Security
VPC*
Default VPC
Subnet Group
default
Publicly Accessible
No
Availability Zone
No Preference
VPC Security Group(s)
Create new Security Group
default (VPC)
Database Options
Database Name
ORCL
Database Port
1521
DB Parameter Group
default.oracle-ee-12.1
Option Group
default:oracle-ee-12-1
Copy Tags To Snapshots
Character Set Name
AL32UTF8
Enable Encryption
No
Backup
Backup Retention Period
7
days
Backup Window
No Preference
Monitoring
Enable Enhanced Monitoring
Yes
Monitoring Role
Default
Granularity
60
second(s)
I authorize RDS to create the IAM role rds-monitoring-role.
Maintenance
Auto Minor Version Upgrade
Yes
Maintenance Window
No Preference
* Required
Cancel
Previous
Launch DB Instance

9. Choose **Launch DB Instance**.
10. On the final page of the wizard, choose **Close**.

On the RDS console, the new DB instance appears in the list of DB instances. The DB instance has a status of **creating** until the DB instance is created and ready for use. When the state changes to **available**, you can connect to the DB instance. Depending on the DB instance class and storage allocated, it could take several minutes for the new instance to be available.



CLI

To create an Oracle DB instance by using the AWS CLI, call the create-db-instance command with the parameters below. For information about each setting, see Settings for Oracle DB Instances (p. 955).

- `--db-instance-identifier`
- `--db-instance-class`
- `--db-security-groups`
- `--db-subnet-group`
- `--engine`
- `--master-user-name`
- `--master-user-password`
- `--allocated-storage`
- `--backup-retention-period`

Example

The following command will launch the example DB instance.

For Linux, OS X, or Unix:

```
aws rds create-db-instance \
    --engine oracle-se1 \
    --db-instance-identifier mydbinstance \
    --allocated-storage 20 \
    --db-instance-class db.m1.small \
    --db-security-groups mydbsecuritygroup \
    --db-subnet-group mydbsubnetgroup \
    --master-username masterawsuser \
    --master-user-password masteruserpassword \
    --backup-retention-period 3
```

For Windows:

```
aws rds create-db-instance ^
    --engine oracle-se1 ^
    --db-instance-identifier mydbinstance ^
    --allocated-storage 20 ^
    --db-instance-class db.m1.small ^
    --db-security-groups mydbsecuritygroup ^
    --db-subnet-group mydbsubnetgroup ^
    --master-username masterawsuser ^
    --master-user-password masteruserpassword ^
    --backup-retention-period 3
```

This command should produce output similar to the following:

```
DBINSTANCE  mydbinstance  db.m1.small  oracle-se1  20  sa  creating  3  ****  n
 11.2.0.4.v1
SECGROUP  default  active
PARAMGRP  default.oracle-se1-11.2  in-sync
```

API

To create an Oracle DB instance by using the Amazon RDS API, call the CreateDBInstance action with the parameters below. For information about each setting, see Settings for Oracle DB Instances (p. 955).

- `AllocatedStorage`
- `BackupRetentionPeriod`
- `DBInstanceClass`
- `DBInstanceIdentifier`
- `DBSecurityGroups`
- `DBSubnetGroup`
- `Engine`
- `MasterUsername`
- `MasterUserPassword`

Example

```
https://rds.amazonaws.com/
    ?Action=CreateDBInstance
    &AllocatedStorage=250
    &BackupRetentionPeriod=3
    &DBInstanceClass=db.m1.large
    &DBInstanceIdentifier=mydbinstance
    &DBSecurityGroups.member.1=mysecuritygroup
    &DBSubnetGroup=mydbsubnetgroup
    &Engine=oracle-se1
    &MasterUserPassword=masteruserpassword
    &MasterUsername=masterawsuser
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-10-31
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20140305/us-west-1/rds/aws4_request
    &X-Amz-Date=20140305T185838Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
```

```
&X-Amz-Signature=b441901545441d3c7a48f63b5b1522c5b2b37c137500c93c45e209d4b3a064a3
```

Settings for Oracle DB Instances

The following table contains details about settings that you choose when you create an Oracle DB instance.

Setting	Setting Description
Allocated Storage	The amount of storage to allocate your DB instance (in gigabytes). In some cases, allocating a higher amount of storage for your DB instance than the size of your database can improve I/O performance. For more information, see Storage for Amazon RDS (p. 410).
Auto Minor Version Upgrade	Amazon RDS does not support automatic minor version upgrades for DB instances running Oracle. You must modify your DB instance manually to perform a minor version upgrade. Some options, such as Oracle Locator, Oracle Multimedia, and Oracle Spatial, require that you enable automatic minor version upgrades. Upgrades for DB instances that use these options are installed during your scheduled maintenance window, and an outage occurs during the upgrade. You can't disable automatic minor version upgrades at the same time as you modify the option group to remove such an option.
Availability Zone	The availability zone for your DB instance. Use the default of **No Preference** unless you need to specify a particular Availability Zone. For more information, see Regions and Availability Zones (p. 97).
Backup Retention Period	The number of days that you want automatic backups of your DB instance to be retained. For any non-trivial instance, you should set this value to `1` or greater. For more information, see Working With Backups (p. 201).
Backup Window	The time period during which Amazon RDS automatically takes a backup of your DB instance. Unless you have a specific time that you want to have your database backup, use the default of **No Preference**. For more information, see Working With Backups (p. 201).
Character Set Name	The character set for your DB instance. The default value of **AL32UTF8** is for the Unicode 5.0 UTF-8 Universal character set. You cannot change the character set after the DB instance is created. For more information, see Oracle Character Sets Supported in Amazon RDS (p. 990).

Setting	Setting Description
Copy Tags To Snapshots	Select this option to copy any DB instance tags to a DB snapshot when you create a snapshot. For more information, see Tagging Amazon RDS Resources (p. 129).
Database Name	The name for the database on your DB instance. The name must begin with a letter and contain up to 8 alpha-numeric characters. You can't specify the string NULL, or any other reserved word, for the database name. If you do not provide a name, Amazon RDS does not create a database on the DB instance you are creating.
Database Port	The port that you want to access the DB instance through. Oracle installations default to port 1521.
DB Engine Version	The version of Oracle that you want to use.
DB Instance Class	The DB instance class that you want to use. For more information, see DB Instance Class (p. 92) and DB Instance Class Support for Oracle (p. 934).
DB Instance Identifier	The name for your DB instance. The name must be unique for your account and region. You can add some intelligence to the name, such as including the region and DB engine you chose, for example **`oracle-instance1`**.
DB Parameter Group	A parameter group for your DB instance. You can choose the default parameter group or you can create a custom parameter group. For more information, see Working with DB Parameter Groups (p. 170).
Enable Encryption	**Yes** to enable encryption at rest for this DB instance. For more information, see Encrypting Amazon RDS Resources (p. 355).
Enable Enhanced Monitoring	**Yes** to gather metrics in real time for the operating system that your DB instance runs on. For more information, see Enhanced Monitoring (p. 258).
License Model	The license model that you want to use. Choose **license-included** to use the general license agreement for Oracle. Choose **bring-your-own-license** to use your existing Oracle license. For more information, see Oracle Licensing (p. 933).
Maintenance Window	The 30 minute window in which pending modifications to your DB instance are applied. If the time period doesn't matter, choose **No Preference**. For more information, see The Amazon RDS Maintenance Window (p. 103).

Setting	Setting Description
Master User Name	The name that you use as the master user name to log on to your DB instance with all database privileges. This user account is used to log into the DB instance and is granted DBA privileges. For more information, see Oracle Security (p. 935).
Master User Password	The password for your master user account. The password must contain from 8 to 30 printable ASCII characters (excluding /,", and @).
Multi-AZ Deployment	**Yes** to create a standby replica of your DB instance in another availability zone for failover support. We recommend Multi-AZ for production workloads to maintain high availability. For development and testing, you can choose **No**. For more information, see Regions and Availability Zones (p. 97).
Option Group	An option group for your DB instance. You can choose the default option group or you can create a custom option group. For more information, see Working with Option Groups (p. 153).
Publicly Accessible	**Yes** to give your DB instance a public IP address. This means that it is accessible outside the VPC (the DB instance also needs to be in a public subnet in the VPC). Choose **No** if you want the DB instance to only be accessible from inside the VPC. For more information, see Hiding a DB Instance in a VPC from the Internet (p. 401).
Storage Type	The storage type for your DB instance. For more information, see Amazon RDS Storage Types (p. 410).
Subnet Group	This setting depends on the platform you are on. If you are a new customer to AWS, choose **default**, which is the default DB subnet group that was created for your account. If you are creating a DB instance on the previous E2-Classic platform and you want your DB instance in a specific VPC, choose the DB subnet group you created for that VPC.
VPC	This setting depends on the platform you are on. If you are a new customer to AWS, choose the default VPC. If you are creating a DB instance on the previous E2-Classic platform, choose **Not in VPC**. For more information, see Amazon Virtual Private Cloud (VPCs) and Amazon RDS (p. 390).

Setting	Setting Description
VPC Security Group	If you are a new customer to AWS, choose the default VPC. If you have created your own VPC security group, choose the VPC security group you previously created. For more information, see Working with DB Security Groups (EC2-Classic Platform) (p. 380).

Related Topics

- Tutorial: Create an Amazon VPC for Use with an Amazon RDS DB Instance (p. 406)
- Connecting to a DB Instance Running the Oracle Database Engine (p. 959)
- Modifying a DB Instance Running the Oracle Database Engine (p. 967)
- Deleting a DB Instance (p. 126)

Connecting to a DB Instance Running the Oracle Database Engine

After Amazon RDS provisions your Oracle DB instance, you can use any standard SQL client application to connect to the DB instance. In this topic, you connect to a DB instance that is running the Oracle database engine by using Oracle SQL Developer or SQL*Plus.

For an example that walks you through the process of creating and connecting to a sample DB instance, see Creating an Oracle DB Instance and Connecting to a Database on an Oracle DB Instance (p. 44).

Finding the Endpoint of Your DB Instance

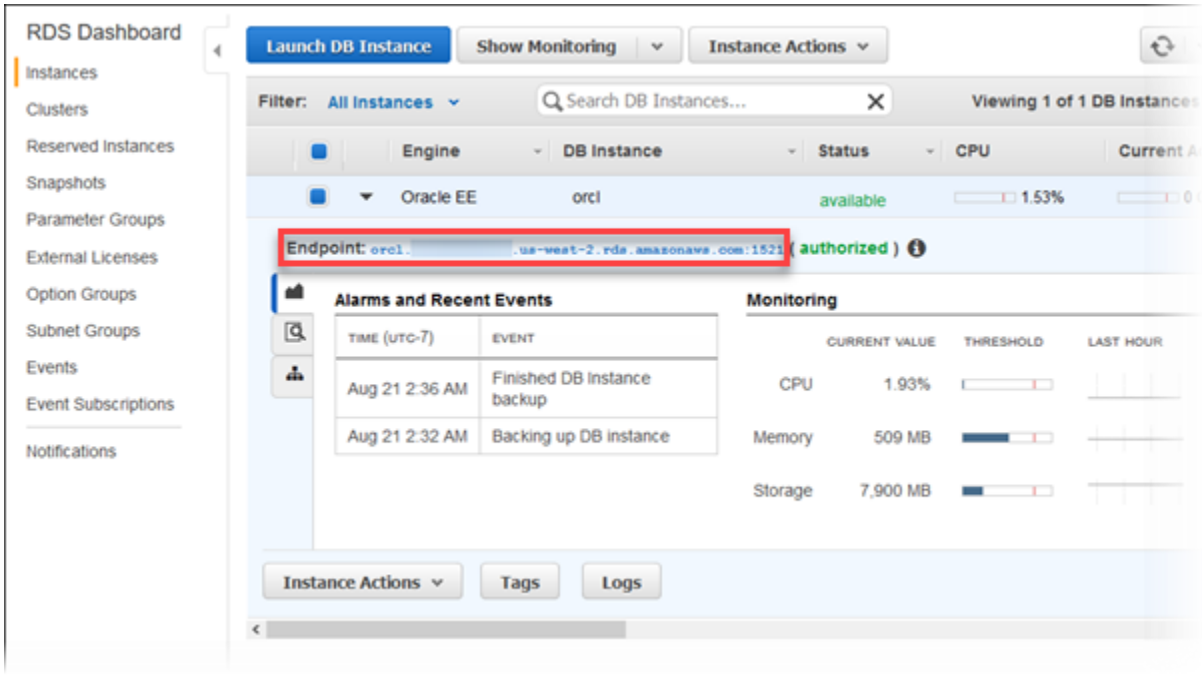
Each Amazon RDS DB instance has an endpoint, and each endpoint has the DNS name and port number for the DB instance. To connect to your DB instance using a SQL client application, you need the DNS name and port number for your DB instance.

You can find the endpoint for a DB instance using the Amazon RDS console or the AWS CLI.

AWS Management Console

To find the endpoint using the console

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the upper-right corner of the console, choose the AWS Region of your DB instance.
3. Find the DNS name and port number for your DB Instance.
   a. Choose **Instances** to display a list of your DB instances.
   b. Choose the row for your Oracle DB instance to display the summary information for the DB instance.



c. Copy the endpoint. The **Endpoint** field has two parts that are separated by a colon (:). The part before the colon is the DNS name for the DB instance, and the part following the colon is the port number. Make sure that you copy both parts.

CLI

To find the endpoint of an Oracle DB instance by using the AWS CLI, call the describe-db-instances command.

Example To find the endpoint using the AWS CLI

```
aws rds describe-db-instances
```

Search for `Endpoint` in the output to find the DNS name and port number for your DB instance. The `Address` line in the output contains the DNS name. The following is an example of the JSON endpoint output:

```
"Endpoint": {
    "HostedZoneId": "Z1PVIF0B656C1W",
    "Port": 3306,
    "Address": "myinstance.123456789012.us-west-2.rds.amazonaws.com"
},
```

Note
The output might contain information for multiple DB instances.

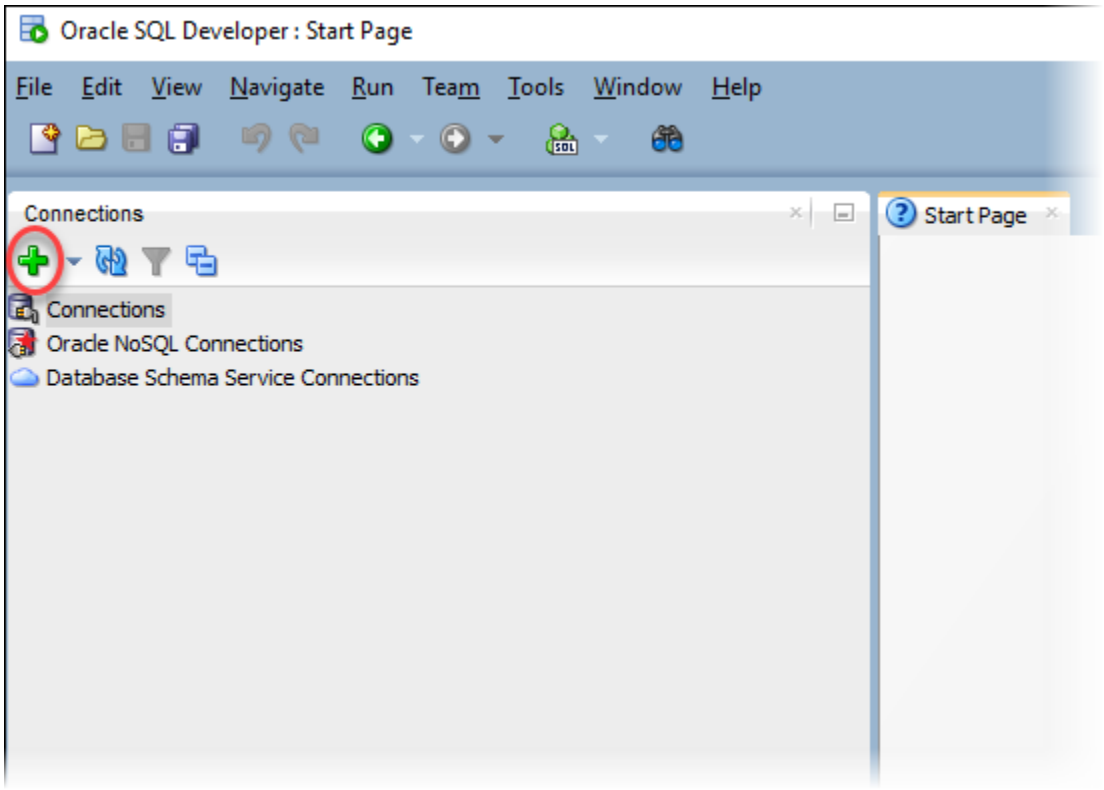
Connecting to Your DB Instance Using Oracle SQL Developer

In this procedure, you connect to your DB instance by using Oracle SQL Developer. To download a standalone version of this utility, see the Oracle SQL Developer Downloads page.

To connect to your DB instance, you need its DNS name and port number. For information about finding the DNS name and port number for a DB instance, see Finding the Endpoint of Your DB Instance (p. 959).

To connect to a DB instance using SQL Developer

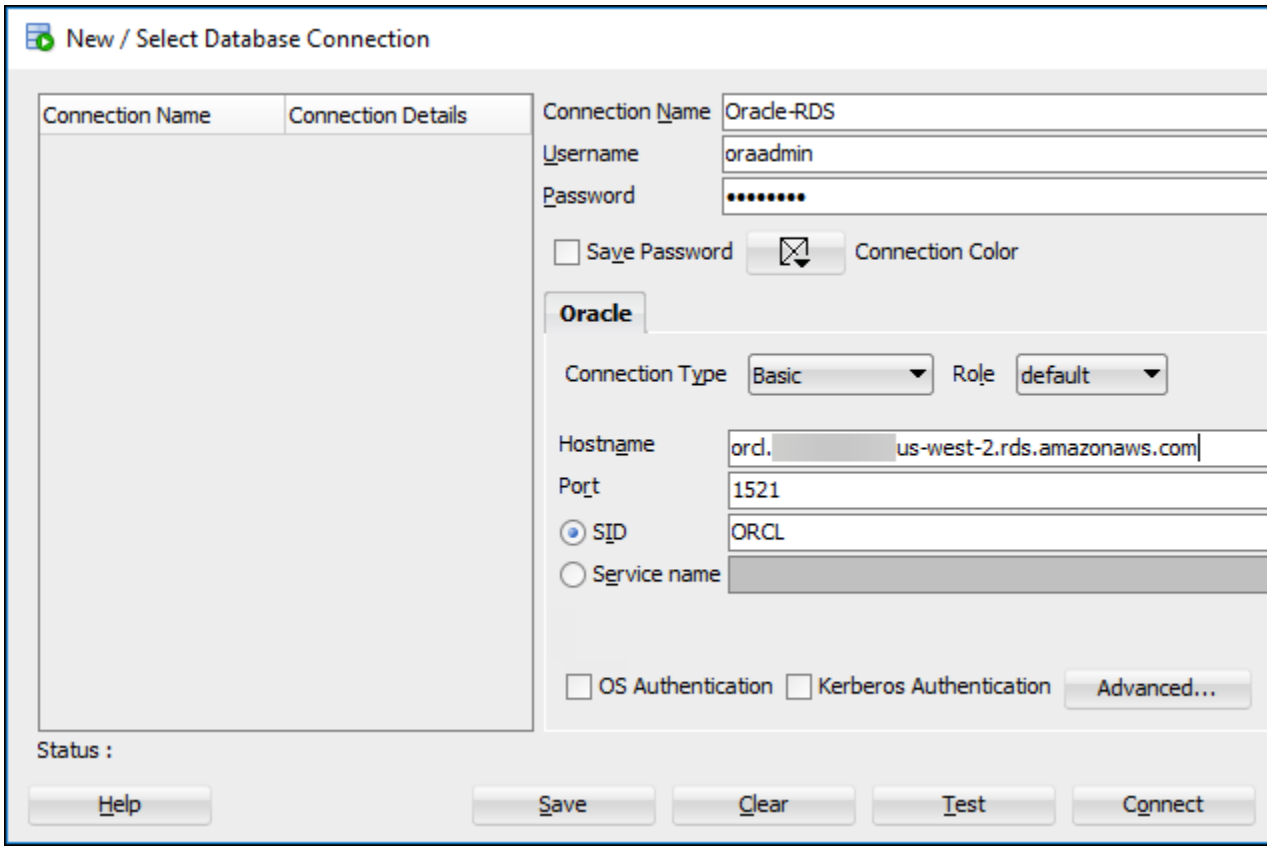
1. Start Oracle SQL Developer.
2. On the **Connections** tab, choose the **add (+)** icon.



3. In the **New/Select Database Connection** dialog box, provide the information for your DB instance:

   - For **Connection Name**, type a name that describes the connection, such as `Oracle-RDS`.
   - For **Username**, type the name of the database administrator for the DB instance.
   - For **Password**, type the password for the database administrator.
   - For **Hostname**, type or paste the DNS name of the DB instance.
   - For **Port**, type the port number.
   - For **SID**, type the Oracle database SID.

   The completed dialog box should look similar to the following.

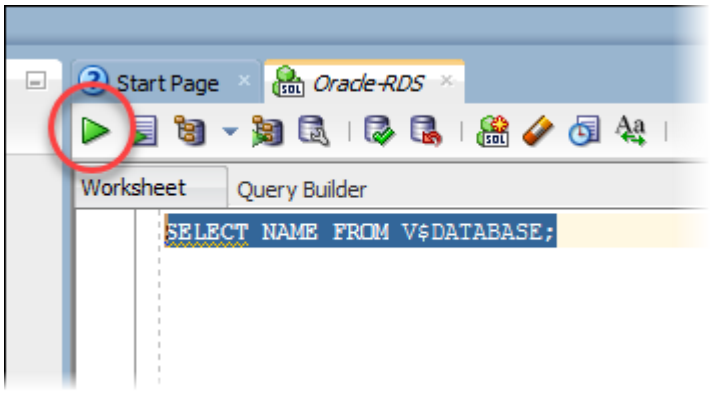


4. Click **Connect**.
5. You can now start creating your own databases and running queries against your DB instance and databases as usual. To run a test query against your DB instance, do the following:

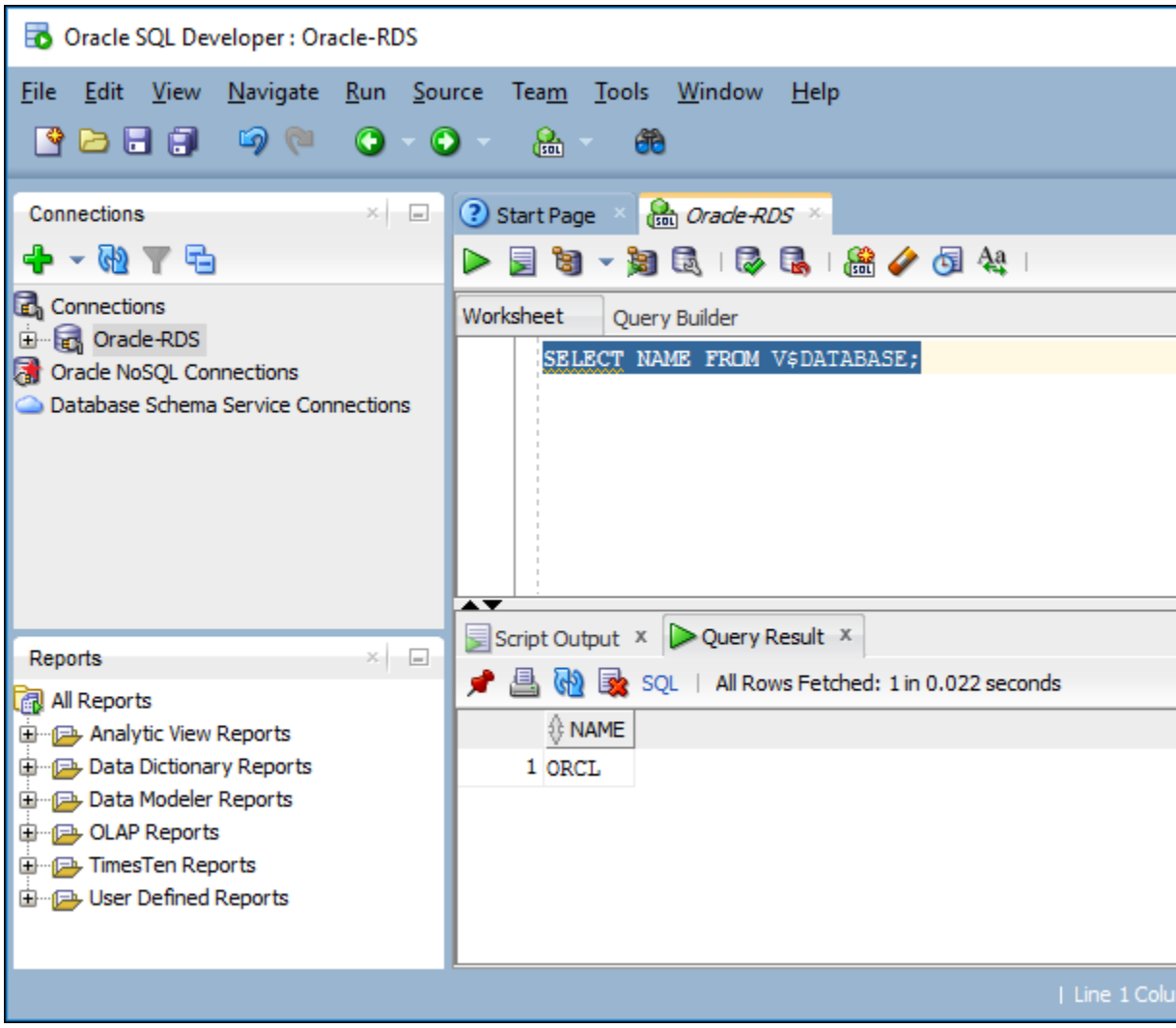
   a. In the **Worksheet** tab for your connection, type the following SQL query:

   ```
   SELECT NAME FROM V$DATABASE;
   ```

   b. Click the **execute** icon to run the query.

   SQL Developer returns the database name.



Connecting to Your DB Instance Using SQL*Plus

You can use a utility like SQL*Plus to connect to an Amazon RDS DB instance running Oracle. To download a standalone version of SQL*Plus, see SQL*Plus User's Guide and Reference.

To connect to your DB instance, you need its DNS name and port number. For information about finding the DNS name and port number for a DB instance, see Finding the Endpoint of Your DB Instance (p. 959).

Example To connect to an Oracle DB instance using SQL*Plus

In the following examples, substitute the DNS name for your DB instance, and then include the port number and the Oracle SID. The SID value is the name of the DB instance's database that you specified when you created the DB instance, and not the name of the DB instance.

For Linux, OS X, or Unix:

```
sqlplus 'mydbusr@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=dns_name)(PORT=port))
(CONNECT_DATA=(SID=database_name)))'
```

For Windows:

```
sqlplus mydbusr@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=dns_name)(PORT=port))
(CONNECT_DATA=(SID=database_name)))
```

You will see output similar to the following.

```
SQL*Plus: Release 12.1.0.2.0 Production on Mon Aug 21 09:42:20 2017
```

After you enter the password for the user, the SQL prompt appears.

```
SQL>
```

Note
The shorter format connection string (Easy connect or EZCONNECT), such as `sqlplus USER/PASSWORD@LONGER-THAN-63-CHARS-RDS-ENDPOINT-HERE:1521/DATABASE_IDENTIFIER`, might encounter a maximum character limit and should not be used to connect.

Security Group Considerations

For you to connect to your DB instance, it must be associated with a security group that contains the IP addresses and network configuration that you use to access the DB instance. You might have associated your DB instance with an appropriate security group when you created it. If you assigned a default, non-configured security group when you created the DB instance, the DB instance firewall prevents connections.

If you need to create a new security group to enable access, the type of security group that you create depends on which Amazon EC2 platform your DB instance is on. To determine your platform, see Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform (p. 391). In general, if your DB instance is on the *EC2-Classic* platform, you create a DB security group; if your DB instance is on the *VPC* platform, you create a VPC security group. For information about creating a new security group, see Amazon RDS Security Groups (p. 375).

After you create the new security group, you modify your DB instance to associate it with the security group. For more information, see Modifying a DB Instance Running the Oracle Database Engine (p. 967).

You can enhance security by using SSL to encrypt connections to your DB instance. For more information, see Oracle SSL (p. 1021).

Dedicated and Shared Server Processes

Server processes handle user connections to an Oracle DB instance. By default, the Oracle DB instance uses dedicated server processes. With dedicated server processes, each server process services only one user process. You can optionally configure shared server processes. With shared server processes, each server process can service multiple user processes.

You might consider using shared server processes when a high number of user sessions are using too much memory on the server. You might also consider shared server processes when sessions connect

and disconnect very often, resulting in performance issues. There are also disadvantages to using shared server processes. For example, they can strain CPU resources, and they are more complicated to configure and administer.

For more information about dedicated and shared server processes, see About Dedicated and Shared Server Processes in the Oracle documentation. For more information about configuring shared server processes on an Amazon RDS Oracle DB instance, see How do I configure Amazon RDS for Oracle Database to work with shared servers? in the Knowledge Center.

Troubleshooting the Connection to Your Oracle DB Instance

The following are issues you might encounter when you try to connect to your Oracle DB instance.

Issue	Troubleshooting Suggestions
Unable to connect to your DB instance.	For a newly created DB instance, the DB instance has a status of **creating** until it is ready to use. When the state changes to **available**, you can connect to the DB instance. Depending on the DB instance class and the amount of storage, it can take up to 20 minutes before the new DB instance is available.
Unable to connect to your DB instance.	If you can't send or receive communications over the port that you specified when you created the DB instance, you can't connect to the DB instance. Check with your network administrator to verify that the port you specified for your DB instance allows inbound and outbound communication.
Unable to connect to your DB instance.	The access rules enforced by your local firewall and the IP addresses you authorized to access your DB instance in the security group for the DB instance might not match. The problem is most likely the egress or ingress rules on your firewall. For more information about security groups, see Amazon RDS Security Groups (p. 375). To walk through the process of setting up rules for your security group, see Tutorial: Create an Amazon VPC for Use with an Amazon RDS DB Instance (p. 406).
Connect failed because target host or object does not exist – Oracle, Error: ORA-12545	Make sure that you specified the server name and port number correctly. For **Server name**, type or paste the DNS name from the console. For information about finding the DNS name and port number for a DB instance, see Finding the Endpoint of Your DB Instance (p. 959).
Invalid username/password; logon denied – Oracle, Error: ORA-01017	You were able to reach the DB instance, but the connection was refused. This is usually caused by providing an incorrect user name or password. Verify the user name and password, and then retry.

Related Topics

- Creating a DB Instance Running the Oracle Database Engine (p. 949)
- Modifying a DB Instance Running the Oracle Database Engine (p. 967)
- Deleting a DB Instance (p. 126)

Modifying a DB Instance Running the Oracle Database Engine

You can change the settings of a DB instance to accomplish tasks such as adding additional storage or changing the DB instance class. This topic guides you through modifying an Amazon RDS Oracle DB instance, and describes the settings for Oracle instances.

We recommend that you test any changes on a test instance before modifying a production instance, so that you fully understand the impact of each change. This is especially important when upgrading database versions.

After you modify your DB instance settings, you can apply the changes immediately, or apply them during the next maintenance window for the DB instance. Some modifications cause an interruption by restarting the DB instance.

AWS Management Console

To modify an Oracle DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **DB Instances**, and then select the DB instance that you want to modify.
3. Choose **Instance Actions**, and then choose **Modify**. The **Modify DB Instance** page appears.
4. Change any of the settings that you want. For information about each setting, see Settings for Oracle DB Instances (p. 968).
5. To apply the changes immediately, select **Apply Immediately**. Selecting this option can cause an outage in some cases. For more information, see The Impact of Apply Immediately (p. 114).
6. When all the changes are as you want them, choose **Continue**.
7. On the confirmation page, review your changes. If they are correct, choose **Modify DB Instance** to save your changes.

   Alternatively, choose **Back** to edit your changes, or choose **Cancel** to cancel your changes.

CLI

To modify an Oracle DB instance by using the AWS CLI, call the modify-db-instance command. Specify the DB instance identifier, and the parameters for the settings that you want to modify. For information about each parameter, see Settings for Oracle DB Instances (p. 968).

Example

The following code modifies `mydbinstance` by setting the backup retention period to 1 week (7 days). The code disables automatic minor version upgrades by using `--no-auto-minor-version-upgrade`. To allow automatic minor version upgrades, use `--auto-minor-version-upgrade`. The changes are applied during the next maintenance window by using `--no-apply-immediately`. Use `--apply-immediately` to apply the changes immediately. For more information, see The Impact of Apply Immediately (p. 114).

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --backup-retention-period 7 \
```

```
    --no-auto-minor-version-upgrade \
    --no-apply-immediately
```

For Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --backup-retention-period 7 ^
    --no-auto-minor-version-upgrade ^
    --no-apply-immediately
```

API

To modify an Oracle DB instance by using the Amazon RDS API, call the ModifyDBInstance action. Specify the DB instance identifier, and the parameters for the settings that you want to modify. For information about each parameter, see Settings for Oracle DB Instances (p. 968).

Example

The following code modifies `mydbinstance` by setting the backup retention period to 1 week (7 days) and disabling automatic minor version upgrades. These changes are applied during the next maintenance window.

```
https://rds.amazonaws.com/
    ?Action=ModifyDBInstance
    &ApplyImmediately=false
    &AutoMinorVersionUpgrade=false
    &BackupRetentionPeriod=7
    &DBInstanceIdentifier=mydbinstance
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-10-31
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20131016/us-west-1/rds/aws4_request
    &X-Amz-Date=20131016T233051Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=087a8eb41cb1ab0fc9ec1575f23e73757ffc6a1e42d7d2b30b9cc0be988cff97
```

Settings for Oracle DB Instances

The following table contains details about which settings you can modify, which settings you can't modify, when the changes can be applied, and whether the changes cause downtime for the DB instance.

Setting	Setting Description	When the Change Occurs	Downtime Notes
Allocated Storage	The storage, in gigabytes, that you want to allocate for your DB instance. You can only increase the allocated storage, you can't reduce the allocated storage. You can't modify allocated storage if the DB instance status is `storage-optimization` or if the allocated storage for the DB instance has been modified in the last six hours.	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	No downtime. Performance may be degraded during the change.

Setting	Setting Description	When the Change Occurs	Downtime Notes
	The maximum storage allowed depends on the storage type. For more information, see Storage for Amazon RDS (p. 410).		
Auto Minor Version Upgrade	Amazon RDS does not support automatic minor version upgrades for DB instances running Oracle. You must modify the DB instance manually to perform a minor version upgrade. Use the **DB Engine Version** field to manually upgrade your DB instance to a later minor version. Some options, such as Oracle Locator, Oracle Multimedia, and Oracle Spatial, require that you enable automatic minor version upgrades. Upgrades for DB instances that use these options are installed during your scheduled maintenance window, and an outage occurs during the upgrade. You can't disable automatic minor version upgrades at the same time as you modify the option group to remove such an option.	–	–
Backup Retention Period	The number of days that automatic backups are retained. To disable automatic backups, set the backup retention period to 0. For more information, see Working With Backups (p. 201).	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false and you change the setting from a non-zero value to another non-zero value, the change is applied asynchronously, as soon as possible. Otherwise, the change occurs during the next maintenance window.	An outage occurs if you change from 0 to a non-zero value, or from a non-zero value to 0.
Backup Window	The time range during which automated backups of your databases occur. The backup window is a start time in Universal Coordinated Time (UTC), and a duration in hours. For more information, see Working With Backups (p. 201).	The change is applied asynchronously, as soon as possible.	–

Setting	Setting Description	When the Change Occurs	Downtime Notes
Certificate Authority	The certificate that you want to use.	–	–
Copy Tags to Snapshots	If you have any DB instance tags, this option copies them when you create a DB snapshot. For more information, see Tagging Amazon RDS Resources (p. 129).	–	–
Database Port	The port that you want to use to access the database. The port value must not match any of the port values specified for options in the option group for the DB instance.	The change occurs immediately. This setting ignores the **Apply Immediately** setting.	The DB instance is rebooted immediately.
DB Engine Version	The version of the Oracle database engine that you want to use. Before you upgrade your production DB instances, we recommend that you test the upgrade process on a test instance to verify its duration and to validate your applications. We do not recommend upgrading micro DB instances because they have limited CPU resources and the upgrade process may take hours to complete. An alternative to upgrading micro DB instances with small storage (10-20 GB) is to copy your data using Data Pump, where we also recommend testing before migrating your production instances. For more information, see Upgrading the Oracle DB Engine (p. 975).	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	An outage occurs during this change.
DB Instance Class	The DB instance class that you want to use. For more information, see DB Instance Class (p. 92) and DB Instance Class Support for Oracle (p. 934).	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	An outage occurs during this change.

Setting	Setting Description	When the Change Occurs	Downtime Notes
DB Instance Identifier	The DB instance identifier. This value is stored as a lowercase string. For more information about the effects of renaming a DB instance, see Renaming a DB Instance (p. 116).	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	An outage occurs during this change. The DB instance is rebooted.
DB Parameter Group	The parameter group that you want associated with the DB instance. For more information, see Working with DB Parameter Groups (p. 170).	The parameter group change occurs immediately. However, parameter changes only occur when you reboot the DB instance manually without failover. For more information, see Rebooting a DB Instance (p. 119).	An outage doesn't occur during this change. However, parameter changes only occur when you reboot the DB instance manually without failover.
Enable Enhanced Monitoring	**Yes** to enable gathering metrics in real time for the operating system that your DB instance runs on. For more information, see Enhanced Monitoring (p. 258).	–	–
License Model	**license-included** to use the general license agreement for Oracle. **bring-your-own-license** to use your existing Oracle license. For more information, see Oracle Licensing (p. 933).	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	An outage occurs during this change.
Maintenance Window	The time range during which system maintenance occurs. System maintenance includes upgrades, if applicable. The maintenance window is a start time in Universal Coordinated Time (UTC), and a duration in hours. If you set the window to the current time, there must be at least 30 minutes between the current time and end of the window to ensure any pending changes are applied. For more information, see The Amazon RDS Maintenance Window (p. 103).	The change occurs immediately. This setting ignores the **Apply Immediately** setting.	If there are one or more pending actions that cause an outage, and the maintenance window is changed to include the current time, then those pending actions are applied immediately, and an outage occurs.

Setting	Setting Description	When the Change Occurs	Downtime Notes
Multi-AZ Deployment	**Yes** to deploy your DB instance in multiple Availability Zones; otherwise, **No**. For more information, see Regions and Availability Zones (p. 97).	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	–
New Master Password	The password for your master user. The password must contain from 8 to 30 alphanumeric characters.	The change is applied asynchronously, as soon as possible. This setting ignores the **Apply Immediately** setting.	–
Option Group	The option group that you want associated with the DB instance. For more information, see Working with Option Groups (p. 153).	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	When you add the APEX options to an existing DB instance, a brief outage occurs while your DB instance is automatically restarted. When you add the OEM option to an existing DB instance, the change can cause a brief (sub-second) period during which new connections are rejected. Existing connections are not interrupted.
Publicly Accessible	**Yes** to give the DB instance a public IP address, meaning that it is accessible outside the VPC. To be publicly accessible, the DB instance also has to be in a public subnet in the VPC. **No** to make the DB instance accessible only from inside the VPC. For more information, see Hiding a DB Instance in a VPC from the Internet (p. 401).	The change occurs immediately. This setting ignores the **Apply Immediately** setting.	–
Security Group	The security group you want associated with the DB instance. For more information, see Working with DB Security Groups (EC2-Classic Platform) (p. 380).	The change is applied asynchronously, as soon as possible. This setting ignores the **Apply Immediately** setting.	–

Setting	Setting Description	When the Change Occurs	Downtime Notes
Storage Type	The storage type that you want to use. For more information, see Amazon RDS Storage Types (p. 410).	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	The following changes all result in a brief outage while the process starts. After that, you can use your database normally while the change takes place. • From **General Purpose (SSD)** to **Magnetic**. • From **General Purpose (SSD)** to **Provisioned IOPS (SSD)**, if the DB instance is single-AZ. There is no outage for a multi-AZ DB instance. • From **Magnetic** to **General Purpose (SSD)**. • From **Magnetic** to **Provisioned IOPS (SSD)**. • From **Provisioned IOPS (SSD)** to **Magnetic**. • From **Provisioned IOPS (SSD)** to **General Purpose (SSD)**, if the DB instance is single-AZ. There is no outage for a multi-AZ DB instance.

Setting	Setting Description	When the Change Occurs	Downtime Notes
Subnet Group	The subnet group for the DB instance. You can use this setting to move your DB instance to a different VPC. If your DB instance is not in a VPC, you can use this setting to move your DB instance into a VPC. For more information, see Moving a DB Instance Not in a VPC into a VPC (p. 405).	–	–

Related Topics

- Rebooting a DB Instance (p. 119)
- Connecting to a DB Instance Running the Oracle Database Engine (p. 959)
- Upgrading the Oracle DB Engine (p. 975)
- Deleting a DB Instance (p. 126)

Upgrading the Oracle DB Engine

When Amazon RDS supports a new version of Oracle, you can upgrade your DB instances to the new version. Amazon RDS supports the following upgrades to an Oracle DB instance:

- **Major Version Upgrades** – from 11g to 12c.
- **Minor Version Upgrades**

You must perform all upgrades manually, and an outage occurs while the upgrade takes place. The time for the outage varies based on your engine version and the size of your DB instance.

For information about what Oracle versions are available on Amazon RDS, see Appendix: Oracle Database Engine Release Notes (p. 1120).

Overview of Upgrading

Amazon RDS takes two DB snapshots during the upgrade process. The first DB snapshot is of the DB instance before any upgrade changes have been made. If the upgrade doesn't work for your databases, you can restore this snapshot to create a DB instance running the old version. The second DB snapshot is taken after the upgrade completes.

> **Note**
> Amazon RDS only takes DB snapshots if you have set the backup retention period for your DB instance to a number greater than 0. To change your backup retention period, see Modifying a DB Instance Running the Oracle Database Engine (p. 967).

After an upgrade is complete, you can't revert to the previous version of the database engine. If you want to return to the previous version, restore the DB snapshot that was taken before the upgrade to create a new DB instance.

If your DB instance is in a Multi-AZ deployment, both the primary and standby replicas are upgraded. The primary and standby DB instances are upgraded at the same time, and you experience an outage until the upgrade is complete.

Major Version Upgrades

Amazon RDS supports the following upgrades to an Oracle DB instance.

Current Version	Upgrade Path
11.2.0.4.v1, 11.2.0.4.v3 – 11.2.0.4.v13	Upgrade directly to 12.1.0.2.v9.

Oracle Minor Version Upgrades

You must modify the DB instance manually to perform a minor version upgrade. Minor version upgrades do not occur automatically. A minor version upgrade applies an Oracle PSU.

The following minor version upgrades are not supported.

Current Version	Upgrade Not Supported
12.1.0.2.v6	12.1.0.2.v7
12.1.0.2.v5	12.1.0.2.v7

Current Version	Upgrade Not Supported
12.1.0.2.v5	12.1.0.2.v6

Oracle SE2 Upgrade Paths

The following table shows supported upgrade paths to Standard Edition Two (SE2). For more information about the License Included and Bring Your Own License (BYOL) models, see Oracle Licensing (p. 933).

Your Existing Configuration	Supported SE2 Configuration
12.1.0.2 SE2, BYOL	12.1.0.2 SE2, BYOL or License Included
11.2.0.4 SE1, BYOL or License Included 11.2.0.4 SE, BYOL	12.1.0.2 SE2, BYOL or License Included

To upgrade from your existing configuration to a supported SE2 configuration, use a supported upgrade path. For more information, see Major Version Upgrades (p. 975).

Option and Parameter Group Considerations

Option Group Considerations

If your DB instance uses a custom option group, in some cases Amazon RDS can't automatically assign your DB instance a new option group. For example, this occurs when you upgrade to a new major version. In those cases, you must specify a new option group when you upgrade. We recommend that you create a new option group, and add the same options to it as in your existing custom option group.

For more information, see Creating an Option Group (p. 154) or Making a Copy of an Option Group (p. 156).

If your DB instance uses a custom option group that contains the APEX option, in some cases you can reduce the time it takes to upgrade your DB instance by upgrading your version of APEX at the same time as your DB instance. For more information, see Upgrading the APEX Version (p. 998).

Parameter Group Considerations

If your DB instance uses a custom parameter group, in some cases Amazon RDS can't automatically assign your DB instance a new parameter group. For example, this occurs when you upgrade to a new major version. In those cases, you must specify a new parameter group when you upgrade. We recommend that you create a new parameter group, and configure the parameters as in your existing custom parameter group.

For more information, see Creating a DB Parameter Group (p. 171) or Copying a DB Parameter Group (p. 175).

Testing an Upgrade

Before you perform a major version upgrade on your DB instance, you should thoroughly test your database and all applications that access the database for compatibility with the new version. We recommend that you use the following procedure.

To test a major version upgrade

1. Review the Oracle upgrade documentation for the new version of the database engine to see if there are compatibility issues that might affect your database or applications. For more information, see Database Upgrade Guide in the Oracle documentation.
2. If your DB instance uses a custom option group, create a new option group compatible with the new version you are upgrading to. For more information, see Option Group Considerations (p. 976).
3. If your DB instance uses a custom parameter group, create a new parameter group compatible with the new version you are upgrading to. For more information, see Parameter Group Considerations (p. 976).
4. Create a DB snapshot of the DB instance to be upgraded. For more information, see Creating a DB Snapshot (p. 207).
5. Restore the DB snapshot to create a new test DB instance. For more information, see Restoring from a DB Snapshot (p. 209).
6. Modify this new test DB instance to upgrade it to the new version, by using one of the following methods:
   - AWS Management Console (p. 977)
   - CLI (p. 977)
   - API (p. 978)
7. Perform testing:
   - Run as many of your quality assurance tests against the upgraded DB instance as needed to ensure that your database and application work correctly with the new version.
   - Implement any new tests needed to evaluate the impact of any compatibility issues that you identified in step 1.
   - Test all stored procedures, functions, and triggers.
   - Direct test versions of your applications to the upgraded DB instance. Verify that the applications work correctly with the new version.
   - Evaluate the storage used by the upgraded instance to determine if the upgrade requires additional storage. You might need to choose a larger instance class to support the new version in production. For more information, see DB Instance Class (p. 92).
8. If all tests pass, then perform the upgrade on your production DB instance. We recommend that you don't allow write operations to the DB instance until you confirm that everything is working correctly.

AWS Management Console

To upgrade an Oracle DB instance by using the AWS Management Console, you follow the same procedure as when you modify the DB instance. For more detailed instructions, see Modifying a DB Instance Running the Oracle Database Engine (p. 967).

CLI

To upgrade an Oracle DB instance by using the AWS CLI, call the modify-db-instance command with the following parameters:

- `--db-instance-identifier` – the name of the DB instance.
- `--engine-version` – the version number of the database engine to upgrade to.
- `--allow-major-version-upgrade` – to upgrade major version.

- `--no-apply-immediately` – apply changes during the next maintenance window. To apply changes immediately, use `--apply-immediately`. For more information, see The Impact of Apply Immediately (p. 114).

You might also need to include the following parameters. For more information, see Option Group Considerations (p. 976) and Parameter Group Considerations (p. 976).

- `--option-group-name` – the option group for the upgraded DB instance.
- `--db-parameter-group-name` – the parameter group for the upgraded DB instance.

Example

The following code upgrades a DB instance. These changes are applied during the next maintenance window.

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier <mydbinstance> \
    --engine-version <12.1.0.2.v9> \
    --option-group-name <default:oracle-ee-12-1> \
    --db-parameter-group-name <default.oracle-ee-12.1> \
    --allow-major-version-upgrade \
    --no-apply-immediately
```

For Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier <mydbinstance> ^
    --engine-version <12.1.0.2.v9> ^
    --option-group-name <default:oracle-ee-12-1> ^
    --db-parameter-group-name <default.oracle-ee-12.1> ^
    --allow-major-version-upgrade ^
    --no-apply-immediately
```

API

To upgrade an Oracle DB instance by using the Amazon RDS API, call the ModifyDBInstance action with the following parameters:

- `DBInstanceIdentifier` – the name of the DB instance.
- `EngineVersion` – the version number of the database engine to upgrade to.
- `AllowMajorVersionUpgrade` – set to `true` to upgrade major version.
- `ApplyImmediately` – whether to apply changes immediately or during the next maintenance window. To apply changes immediately, set the value to `true`. To apply changes during the next maintenance window, set the value to `false`. For more information, see The Impact of Apply Immediately (p. 114).

You might also need to include the following parameters. For more information, see Option Group Considerations (p. 976) and Parameter Group Considerations (p. 976).

- `OptionGroupName` – the option group for the upgraded DB instance.
- `DBParameterGroupName` – the parameter group for the upgraded DB instance.

Example

The following code upgrades a DB instance. These changes are applied during the next maintenance window.

```
https://rds.amazonaws.com/
    ?Action=ModifyDBInstance
    &AllowMajorVersionUpgrade=true
    &ApplyImmediately=false
    &DBInstanceIdentifier=mydbinstance
    &DBParameterGroupName=default.oracle-ee-12.1
    &EngineVersion=12.1.0.2.v9
    &OptionGroupName=default:oracle-ee-12-1
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-10-31
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20131016/us-west-1/rds/aws4_request
    &X-Amz-Date=20131016T233051Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=087a8eb41cb1ab5f99e81575f23e73757ffc6a1e42d7d2b30b9cc0be988cff97
```

Related Topics

- Upgrading an Oracle DB Snapshot (p. 980)
- Updating the Operating System for a DB Instance or DB Cluster (p. 108)
- Modifying a DB Instance Running the Oracle Database Engine (p. 967)

Upgrading an Oracle DB Snapshot

If you have existing manual DB snapshots, you might want to upgrade a snapshot to a later version of the Oracle database engine.

When Oracle stops providing patches for a version, and therefore Amazon RDS deprecates the version, you can upgrade your snapshots that correspond to the deprecated version. For more information, see Oracle Engine Version Management (p. 944).

The following snapshot upgrades are currently supported.

Current Snapshot Version	Supported Snapshot Upgrade
12.1.0.1	12.1.0.2.v8
11.2.0.3	11.2.0.4.v11
11.2.0.2	11.2.0.4.v12

Amazon RDS supports upgrading snapshots in all AWS Regions except the following:

- EU (Frankfurt)
- China (Beijing)
- AWS GovCloud (US)

AWS Management Console

To upgrade an Oracle DB snapshot

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Snapshots**, and then select the DB snapshot that you want to upgrade.
3. Choose **Snapshot Actions**, and then choose **Modify Snapshot**. The **Modify DB Snapshot** page appears.
4. For **DB Engine Version**, choose the version to upgrade the snapshot to.
5. (Optional) For **Option Group**, choose the option group for the upgraded DB snapshot. The same option group considerations apply when upgrading a DB snapshot as when upgrading a DB instance. For more information, see Option Group Considerations (p. 976).
6. Choose **Modify Snapshot** to save your changes.

   Alternatively, choose **Cancel** to cancel your changes.

CLI

To upgrade an Oracle DB snapshot by using the AWS CLI, call the modify-db-snapshot command with the following parameters:

- `--db-snapshot-identifier` – The name of the DB snapshot.
- `--engine-version` – The version to upgrade the snapshot to.

You might also need to include the following parameter. The same option group considerations apply when upgrading a DB snapshot as when upgrading a DB instance. For more information, see Option Group Considerations (p. 976).

- `--option-group-name` – The option group for the upgraded DB snapshot.

Example

The following example upgrades a DB snapshot.

For Linux, OS X, or Unix:

```
aws rds modify-db-snapshot \
    --db-snapshot-identifier <mydbsnapshot> \
    --engine-version <11.2.0.4.v12> \
    --option-group-name <default:oracle-se1-11-2>
```

For Windows:

```
aws rds modify-db-snapshot ^
    --db-snapshot-identifier <mydbsnapshot> ^
    --engine-version <11.2.0.4.v12> ^
    --option-group-name <default:oracle-se1-11-2>
```

API

To upgrade an Oracle DB snapshot by using the Amazon RDS API, call the ModifyDBSnapshot action with the following parameters:

- `DBSnapshotIdentifier` – The name of the DB snapshot.
- `EngineVersion` – The version to upgrade the snapshot to.

You might also need to include the following parameter. The same option group considerations apply when upgrading a DB snapshot as when upgrading a DB instance. For more information, see Option Group Considerations (p. 976).

- `OptionGroupName` – The option group for the upgraded DB snapshot.

Example

The following example upgrades a DB snapshot.

```
https://rds.amazonaws.com/
    ?Action=ModifyDBSnapshot
    &DBSnapshotIdentifier=mydbsnapshot
    &EngineVersion=11.2.0.4.v12
    &OptionGroupName=default:oracle-se1-11-2
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-10-31
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20131016/us-west-1/rds/aws4_request
    &X-Amz-Date=20131016T233051Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=087a8eb41cb1ab5f99e81575f23e73757ffc6a1e42d7d2b30b9cc0be988cff97
```

Related Topics

- Appendix: Oracle Database Engine Release Notes (p. 1120)
- Upgrading the Oracle DB Engine (p. 975)
- Updating the Operating System for a DB Instance or DB Cluster (p. 108)

Importing Data into Oracle on Amazon RDS

How you import data into an Amazon RDS DB instance depends on the amount of data you have and the number and variety of database objects in your database. For example, you can use Oracle SQL Developer to import a simple, 20 MB database. You can use Oracle Data Pump to import complex databases, or databases that are several hundred megabytes or several terabytes in size.

You can also use AWS Database Migration Service (AWS DMS) to import data into an Amazon RDS DB instance. AWS DMS can migrate databases without downtime and, for many database engines, continue ongoing replication until you are ready to switch over to the target database. You can migrate to Oracle from either the same database engine or a different database engine using AWS DMS. If you are migrating from a different database engine, you can use the AWS Schema Conversion Tool to migrate schema objects that are not migrated by AWS DMS. For more information about AWS DMS, see What is AWS Database Migration Service.

Before you use any of these migration techniques, we recommend the best practice of taking a backup of your database. You can back up your Amazon RDS instances by creating snapshots. Later, you can restore the database from the snapshots. For more information, see Backing Up and Restoring Amazon RDS DB Instances (p. 200).

Oracle SQL Developer

For small databases, you can use Oracle SQL Developer, a graphical Java tool distributed without cost by Oracle. You can install this tool on your desktop computer (Windows, Linux, or Mac) or on one of your servers. Oracle SQL Developer provides options for migrating data between two Oracle databases, or for migrating data from other databases, such as MySQL, to Oracle. Oracle SQL Developer is best suited for migrating small databases. We recommend that you read the Oracle SQL Developer product documentation before you begin migrating your data.

After you install SQL Developer, you can use it to connect to your source and target databases. Use the **Database Copy** command on the Tools menu to copy your data to your Amazon RDS instance.

To download Oracle SQL Developer, go to http://www.oracle.com/technetwork/developer-tools/sql-developer.

Oracle also has documentation on how to migrate from other databases, including MySQL and SQL Server. For more information, see http://www.oracle.com/technetwork/database/migration in the Oracle documentation.

Oracle Data Pump

Oracle Data Pump is a long-term replacement for the Oracle Export/Import utilities and is the preferred way to move large amounts of data from an Oracle installation to an Amazon RDS DB instance. You can use Oracle Data Pump for several scenarios:

- Import data from an Oracle database (either on-premises or Amazon EC2 instance) to an Amazon RDS Oracle DB instance
- Import data from an Amazon RDS Oracle DB instance to an Oracle database (either on-premises or Amazon EC2 instance)
- Import data between Amazon RDS Oracle DB instances (for example, to migrate data from EC2-Classic to VPC)

To download Oracle Data Pump utilities, go to http://www.oracle.com/technetwork/database/features/instant-client.

The following process uses Oracle Data Pump and the DBMS_FILE_TRANSFER package. The process connects to a source Oracle instance (which can be an on-premises or Amazon EC2 instance, or an Amazon RDS Oracle DB instance) and exports data using the DBMS_DATAPUMP package. It then uses the DBMS_FILE_TRANSFER.PUT_FILE method to copy the dump file from the Oracle instance to the DATA_PUMP_DIR directory on the target Amazon RDS Oracle DB instance that is connected using a database link. The final step imports the data from the copied dump file into the Amazon RDS Oracle DB instance using the DBMS_DATAPUMP package.

The process has the following requirements:

- You must have execute privileges on the DBMS_FILE_TRANSFER and DBMS_DATAPUMP packages.
- You must have write privileges to the DATA_PUMP_DIR directory on the source DB instance.
- You must ensure that you have enough storage space to store the dump file on the source instance and the target DB instance.

> **Note**
> This process imports a dump file into the DATA_PUMP_DIR directory, a preconfigured directory on all Oracle DB instances. This directory is located on the same storage volume as your data files. When you import the dump file, the existing Oracle data files will use more space, so you should make sure that your DB instance can accommodate that additional use of space as well. The imported dump file is not automatically deleted or purged from the DATA_PUMP_DIR directory. Use UTL_FILE.FREMOVE to remove the imported dump file.

The import process using Oracle Data Pump and the DBMS_FILE_TRANSFER package has the following steps:

- Step 1: Grant privileges to user on the Amazon RDS target instance
- Step 2: Grant privileges to user on source database
- Step 3: Use DBMS_DATAPUMP to create a dump file
- Step 4: Create a database link to the target DB instance
- Step 5: Use DBMS_FILE_TRANSFER to copy the exported dump file to the target DB instance
- Step 6: Use DBMS_DATAPUMP to import the data file on the target DB instance
- Step 7: Clean up

Step 1: Grant privileges to user on the Amazon RDS target instance

1. Use SQL Plus or Oracle SQL Developer to connect to the Amazon RDS target Oracle DB instance into which the data will be imported. Connect as the Amazon RDS master user. For information about connecting to the DB instance, see Connecting to a DB Instance Running the Oracle Database Engine (p. 959).
2. Create the required tablespaces before you import the data. For more information, see Creating and Sizing Tablespaces (p. 1054).
3. If the user account into which the data will be imported does not exist, create the user account and grant the necessary permissions and roles. If you will import data into multiple user schemas, create each user account and grant the necessary privileges and roles to it.

   For example, the following commands create a new user and grant the necessary permissions and roles to import the data into the user's schema:

```
create user schema_1 identified by <password>;
```

```
grant create session, resource to schema_1;
alter user schema_1 quota 100M on users;
```

This example grants the new user the CREATE SESSION privilege and the RESOURCE role. Additional privileges and roles might be required depending on the database objects you will import.

Step 2: Grant privileges to user on source database

Use SQL Plus or Oracle SQL Developer to connect to the Oracle instance that contains the data to be imported. If necessary, create a user account and grant the necessary permissions.

Note
If the source database is an Amazon RDS instance, you can skip this step. You will use your Amazon RDS master user account to perform the export.

The following commands create a new user and grant the necessary permissions:

```
create user export_user identified by <password>;
grant create session, create table, create database link to export_user;
alter user export_user quota 100M on users;
grant read, write on directory data_pump_dir to export_user;
grant select_catalog_role to export_user;
grant execute on dbms_datapump to export_user;
grant execute on dbms_file_transfer to export_user;
```

Step 3: Use DBMS_DATAPUMP to create a dump file

Use SQL Plus or Oracle SQL Developer to connect to the source Oracle instance with an administrative user or with the user you created in Step 2. If the source database is an Amazon RDS Oracle DB instance, connect with the Amazon RDS master user. Next, use the Oracle Data Pump utility to create a dump file.

The following script creates a dump file named *sample.dmp* in the DATA_PUMP_DIR directory.

```
DECLARE
hdnl NUMBER;
BEGIN
hdnl := DBMS_DATAPUMP.OPEN( operation => 'EXPORT', job_mode => 'SCHEMA', job_name=>null);
DBMS_DATAPUMP.ADD_FILE( handle => hdnl, filename => 'sample.dmp', directory =>
 'DATA_PUMP_DIR', filetype => dbms_datapump.ku$_file_type_dump_file);
DBMS_DATAPUMP.ADD_FILE( handle => hdnl, filename => 'exp.log', directory =>
 'DATA_PUMP_DIR', filetype => dbms_datapump.ku$_file_type_log_file);
DBMS_DATAPUMP.METADATA_FILTER(hdnl,'SCHEMA_EXPR','IN (''SCHEMA_1'')');
DBMS_DATAPUMP.START_JOB(hdnl);
END;
/
```

Step 4: Create a database link to the target DB instance

Create a database link between your source instance and your target DB instance. Note that your local Oracle instance must have network connectivity to the DB instance in order to create a database link and to transfer your export dump file.

Perform this step connected with the same user account as the previous step.

If you are creating a database link between two DB instances inside the same VPC or peered VPCs, the two DB instances should have a valid route between them. The security group of each DB instance must allow ingress to and egress from the other DB instance. The security group inbound and outbound rules can refer to security groups from the same VPC or a peered VPC. For more information, see Adjusting Database Links for Use with DB Instances in a VPC (p. 1058).

The following command creates a database link named *to_rds* that connects to the Amazon RDS master user at the target DB instance:

```
create database link to_rds connect to <master_user_account> identified by <password>
using '(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=<dns or ip address of remote db>)
(PORT=<listener port>))(CONNECT_DATA=(SID=<remote SID>)))';
```

Step 5: Use DBMS_FILE_TRANSFER to copy the exported dump file to the target DB instance

Use DBMS_FILE_TRANSFER to copy the dump file from the source database instance to the target DB instance. The following script copies a dump file named sample.dmp from the source instance to a target database link named *to_rds* (created in the previous step):

```
BEGIN
DBMS_FILE_TRANSFER.PUT_FILE(
source_directory_object       => 'DATA_PUMP_DIR',
source_file_name              => 'sample.dmp',
destination_directory_object  => 'DATA_PUMP_DIR',
destination_file_name         => 'sample_copied.dmp',
destination_database          => 'to_rds'
);
END;
/
```

Step 6: Use DBMS_DATAPUMP to import the data file on the target DB instance

Use Oracle Data Pump to import the schema in the DB instance. Note that additional options such as METADATA_REMAP might be required.

Connect to the DB instance with the Amazon RDS master user account to perform the import.

```
DECLARE
hdnl NUMBER;
BEGIN
hdnl := DBMS_DATAPUMP.OPEN( operation => 'IMPORT', job_mode => 'SCHEMA', job_name=>null);
DBMS_DATAPUMP.ADD_FILE( handle => hdnl, filename => 'sample_copied.dmp', directory =>
 'DATA_PUMP_DIR', filetype => dbms_datapump.ku$_file_type_dump_file);
DBMS_DATAPUMP.METADATA_FILTER(hdnl,'SCHEMA_EXPR','IN (''SCHEMA_1'')');
DBMS_DATAPUMP.START_JOB(hdnl);
END;
/
```

You can verify the data import by viewing the user's tables on the DB instance. For example, the following query returns the number of tables for schema_1:

```
select count(*) from dba_tables where owner='SCHEMA_1';
```

Step 7: Clean up

After the data has been imported, you can delete the files you no longer want to keep. You can list the files in the DATA_PUMP_DIR using the following command:

```
select * from table(RDSADMIN.RDS_FILE_UTIL.LISTDIR('DATA_PUMP_DIR')) order by mtime;
```

The following command can be used to delete files in the DATA_PUMP_DIR that you no longer require:

```
exec utl_file.fremove('DATA_PUMP_DIR','<file name>');
```

For example, the following command deletes the file named "sample_copied.dmp":

```
exec utl_file.fremove('DATA_PUMP_DIR','sample_copied.dmp');
```

Oracle Export/Import Utilities

The Oracle Export/Import utilities are best suited for migrations where the data size is small and data types such as binary float and double are not required. The import process creates the schema objects so you do not need to run a script to create them beforehand, making this process well suited for databases with small tables. The following example demonstrates how these utilities can be used to export and import specific tables.

To download Oracle export and import utilities, go to http://www.oracle.com/technetwork/database/features/instant-client.

Export the tables from the source database using the command below. Substitute username/password as appropriate.

```
exp cust_dba@ORCL FILE=exp_file.dmp TABLES=(tab1,tab2,tab3) LOG=exp_file.log
```

The export process creates a binary dump file that contains both the schema and data for the specified tables. Now this schema and data can be imported into a target database using the command:

```
imp cust_dba@targetdb FROMUSER=cust_schema TOUSER=cust_schema \
TABLES=(tab1,tab2,tab3) FILE=exp_file.dmp LOG=imp_file.log
```

There are other variations of the Export and Import commands that might be better suited to your needs. See Oracle's documentation for full details.

Oracle SQL*Loader

Oracle SQL*Loader is well suited for large databases that have a limited number of objects in them. Since the process involved in exporting from a source database and loading to a target database is very specific to the schema, the following example creates the sample schema objects, exports from a source, and then loads it into a target database.

To download Oracle SQL*Loader, go to http://www.oracle.com/technetwork/database/features/instant-client.

1. Create a sample source table using the command below.

```
create table customer_0 tablespace users as select rownum id, o.* from
```

```
all_objects o, all_objects x where rownum <= 1000000;
```

2. On the target Amazon RDS instance, create a destination table that is used to load the data.

```
create table customer_1 tablespace users as select 0 as id, owner,
object_name, created from all_objects where 1=2;
```

3. The data is exported from the source database to a flat file with delimiters. This example uses SQL*Plus for this purpose. For your data, you will likely need to generate a script that does the export for all the objects in the database.

```
alter session set nls_date_format = 'YYYY/MM/DD HH24:MI:SS'; set linesize 800
HEADING OFF FEEDBACK OFF array 5000 pagesize 0 spool customer_0.out SET
MARKUP HTML PREFORMAT ON SET COLSEP ',' SELECT id, owner, object_name,
created FROM customer_0; spool off
```

4. You need to create a control file to describe the data. Again, depending on your data, you will need to build a script that does this step.

```
cat << EOF > sqlldr_1.ctl
load data
infile customer_0.out
into table customer_1
APPEND
fields terminated by "," optionally enclosed by '"'
(
id              POSITION(01:10)         INTEGER EXTERNAL,
owner             POSITION(12:41)         CHAR,
object_name       POSITION(43:72)         CHAR,
created            POSITION(74:92)         date "YYYY/MM/DD HH24:MI:SS"
)
```

   If needed, copy the files generated by the preceding code to a staging area, such as an Amazon EC2 instance.

5. Finally, import the data using SQL*Loader with the appropriate username and password for the target database.

```
sqlldr cust_dba@targetdb control=sqlldr_1.ctl BINDSIZE=10485760 READSIZE=10485760
 ROWS=1000
```

Oracle Materialized Views

You can also make use of Oracle materialized view replication to migrate large datasets efficiently. Replication allows you to keep the target tables in sync with the source on an ongoing basis, so the actual cutover to Amazon RDS can be done later, if needed. The replication is set up using a database link from the Amazon RDS instance to the source database.

One requirement for materialized views is to allow access from the target database to the source database. In the following example, access rules were enabled on the source database to allow the Amazon RDS target database to connect to the source over SQLNet.

1. Create a user account on both source and Amazon RDS target instances that can authenticate with the same password.

```
create user dblink_user identified by <password>
default tablespace users
```

```
temporary tablespace temp; grant create session to dblink_user; grant select
any table to dblink_user; grant select any dictionary to dblink_user;
```

2. Create a database link from the Amazon RDS target instance to the source instance using the newly created dblink_user.

```
create database link remote_site
connect to dblink_user identified by <password>
using '(description=(address=(protocol=tcp) (host=<myhost>) (port=<listener port>))

(connect_data=(sid=<sourcedb sid>)))';
```

3. Test the link:

```
select * from v$instance@remote_site;
```

4. Create a sample table with primary key and materialized view log on the source instance.

```
create table customer_0 tablespace users as select rownum id, o.* from
all_objects o, all_objects x where rownum <= 1000000; alter table customer_0
add constraint pk_customer_0 primary key (id) using index; create
materialized view log on customer_0;
```

5. On the target Amazon RDS instance, create a materialized view.

```
CREATE MATERIALIZED VIEW customer_0    BUILD IMMEDIATE   REFRESH FAST    AS
SELECT * FROM cust_dba.customer_0@remote_site;
```

Oracle Character Sets Supported in Amazon RDS

The following table lists the Oracle database character sets that are supported in Amazon RDS. You can use a value from this table with the `--character-set-name` parameter of the AWS CLI create-db-instance command or with the `CharacterSetName` parameter of the Amazon RDS API CreateDBInstance action.

Setting the NLS_LANG environment parameter in your client's environment is the simplest way to specify locale behavior for Oracle. This parameter sets the language and territory used by the client application and the database server. It also indicates the client's character set, which corresponds to the character set for data entered or displayed by a client application. Amazon RDS lets you set the character set when you create a DB instance. For more information on the NLS_LANG and character sets, see What is a Character set or Code Page? in the Oracle documentation.

Value	Description
AL32UTF8	Unicode 5.0 UTF-8 Universal character set (default)
AR8ISO8859P6	ISO 8859-6 Latin/Arabic
AR8MSWIN1256	Microsoft Windows Code Page 1256 8-bit Latin/Arabic
BLT8ISO8859P13	ISO 8859-13 Baltic
BLT8MSWIN1257	Microsoft Windows Code Page 1257 8-bit Baltic
CL8ISO8859P5	ISO 88559-5 Latin/Cyrillic
CL8MSWIN1251	Microsoft Windows Code Page 1251 8-bit Latin/Cyrillic
EE8ISO8859P2	ISO 8859-2 East European
EL8ISO8859P7	ISO 8859-7 Latin/Greek
EE8MSWIN1250	Microsoft Windows Code Page 1250 8-bit East European
EL8MSWIN1253	Microsoft Windows Code Page 1253 8-bit Latin/Greek
IW8ISO8859P8	ISO 8859-8 Latin/Hebrew
IW8MSWIN1255	Microsoft Windows Code Page 1255 8-bit Latin/Hebrew
JA16EUC	EUC 24-bit Japanese
JA16EUCTILDE	Same as JA16EUC except for mapping of wave dash and tilde to and from Unicode
JA16SJIS	Shift-JIS 16-bit Japanese
JA16SJISTILDE	Same as JA16SJIS except for mapping of wave dash and tilde to and from Unicode
KO16MSWIN949	Microsoft Windows Code Page 949 Korean

Value	Description
NE8ISO8859P10	ISO 8859-10 North European
NEE8ISO8859P4	ISO 8859-4 North and Northeast European
TH8TISASCII	Thai Industrial Standard 620-2533-ASCII 8-bit
TR8MSWIN1254	Microsoft Windows Code Page 1254 8-bit Turkish
US7ASCII	ASCII 7-bit American
UTF8	Unicode 3.0 UTF-8 Universal character set, CESU-8 compliant
VN8MSWIN1258	Microsoft Windows Code Page 1258 8-bit Vietnamese
WE8ISO8859P1	Western European 8-bit ISO 8859 Part 1
WE8ISO8859P15	ISO 8859-15 West European
WE8ISO8859P9	ISO 8859-9 West European and Turkish
WE8MSWIN1252	Microsoft Windows Code Page 1252 8-bit West European
ZHS16GBK	GBK 16-bit Simplified Chinese
ZHT16HKSCS	Microsoft Windows Code Page 950 with Hong Kong Supplementary Character Set HKSCS-2001. Character set conversion is based on Unicode 3.0.
ZHT16MSWIN950	Microsoft Windows Code Page 950 Traditional Chinese
ZHT32EUC	EUC 32-bit Traditional Chinese

You can also set the following National Language Support (NLS) initialization parameters at the instance level for an Oracle DB instance in Amazon RDS:

- NLS_DATE_FORMAT
- NLS_LENGTH_SEMANTICS
- NLS_NCHAR_CONV_EXCP
- NLS_TIME_FORMAT
- NLS_TIME_TZ_FORMAT
- NLS_TIMESTAMP_FORMAT
- NLS_TIMESTAMP_TZ_FORMAT

For information about modifying instance parameters, see Working with DB Parameter Groups (p. 170).

You can set other NLS initialization parameters in your SQL client. For example, the following statement sets the NLS_LANGUAGE initialization parameter to GERMAN in a SQL client that is connected to an Oracle DB instance:

```
ALTER SESSION SET NLS_LANGUAGE=GERMAN;
```

For information about connecting to an Oracle DB instance with a SQL client, see Connecting to a DB Instance Running the Oracle Database Engine (p. 959).

Options for Oracle DB Instances

This section describes options, or additional features, that are available for Amazon RDS instances running the Oracle DB engine. To enable these options, you add them to an option group, and then associate the option group with your DB instance. For more information, see Working with Option Groups (p. 153).

Some options require additional memory to run on your DB instance. For example, Oracle Enterprise Manager Database Control uses about 300 MB of RAM. If you enable this option for a small DB instance, you might encounter performance problems due to memory constraints. You can adjust the Oracle parameters so that the database requires less RAM; alternatively, you can scale up to a larger DB instance.

Amazon RDS supports the following options for Oracle DB instances.

Option	Option ID
Oracle Application Express (p. 994)	`APEX` `APEX-DEV`
Oracle Enterprise Manager (p. 1005)	`OEM` `OEM_AGENT`
Oracle Label Security (p. 1000)	`OLS`
Oracle Locator (p. 1014)	`LOCATOR`
Oracle Multimedia (p. 1017)	`MULTIMEDIA`
Oracle Native Network Encryption (p. 1003)	`NATIVE_NETWORK_ENCRYPTION`
Oracle SQLT (p. 1025)	`SQLT`
Oracle SSL (p. 1021)	`SSL`
Oracle Spatial (p. 1019)	`SPATIAL`
Oracle Statspack (p. 1029)	`STATSPACK`
Oracle Time Zone (p. 1033)	`Timezone`
Oracle Transparent Data Encryption (p. 1036)	`TDE`
Oracle UTL_MAIL (p. 1038)	`UTL_MAIL`
Oracle XML DB (p. 1040)	`XMLDB`

Oracle Application Express

Amazon RDS supports Oracle Application Express (APEX) through the use of the `APEX` and `APEX-DEV` options. Oracle APEX can be deployed as a run-time environment or as a full development environment for web-based applications. Using Oracle APEX, developers can build applications entirely within the web browser. For more information, see Oracle Application Express in the Oracle documentation.

Oracle APEX consists of two main components:

- A *repository* that stores the metadata for APEX applications and components. The repository consists of tables, indexes, and other objects that are installed in your Amazon RDS DB instance.
- A *listener* that manages HTTP communications with Oracle APEX clients. The listener accepts incoming connections from web browsers, forwards them to the Amazon RDS DB instance for processing, and then sends results from the repository back to the browsers. The APEX Listener was renamed Oracle Rest Data Services (ORDS) in Oracle 12c.

When you add the Amazon RDS APEX options to your DB instance, Amazon RDS installs the Oracle APEX repository only. You must install the Oracle APEX Listener on a separate host, such as an Amazon EC2 instance, an on-premises server at your company, or your desktop computer.

Amazon RDS supports the following versions of Oracle APEX for Oracle 12c:

- Oracle APEX version 5.1.2.v1
- Oracle APEX version 5.0.4.v1
- Oracle APEX version 4.2.6.v1

Amazon RDS supports the following versions of Oracle APEX for Oracle 11g:

- Oracle APEX version 5.1.2.v1
- Oracle APEX version 5.0.4.v1
- Oracle APEX version 4.2.6.v1
- Oracle APEX version 4.1.1.v1

Prerequisites for Oracle APEX and APEX Listener

The following are prerequisites for using Oracle APEX and APEX Listener:

- You must have SQL*Plus to perform administrative tasks on your DB instance.
- You must have the following software installed on the host computer that acts as the Oracle APEX Listener:
  - The Java Runtime Environment (JRE).
  - Oracle Net Services, to enable the Oracle APEX Listener to connect to your Amazon RDS instance.

Adding the Amazon RDS APEX Options

The general process for adding the Amazon RDS APEX options to a DB instance is the following:

1. Create a new option group, or copy or modify an existing option group.
2. Add the options to the option group.
3. Associate the option group with the DB instance.

When you add the Amazon RDS APEX options, a brief outage occurs while your DB instance is automatically restarted.

To add the APEX options to a DB instance

1. Determine the option group you want to use. You can create a new option group or use an existing option group. If you want to use an existing option group, skip to the next step. Otherwise, create a custom DB option group with the following settings:

   a. For **Engine**, choose the Oracle edition that you want to use. The APEX options are supported on all editions.
   b. For **Major Engine Version**, choose **11.2** or **12.1**.
   c. For **APEX Version**, choose the version of `APEX` that you want to use. If you don't choose a version, version 4.1.1.v1 is the default for 11g, and version 4.2.6.v1 is the default for 12c.

   For more information, see Creating an Option Group (p. 154).
2. Add the options to the option group. If you want to deploy only the Oracle APEX run-time environment, add only the `APEX` option. If you want to deploy the full development environment, add both the `APEX` and `APEX-DEV` options.

   - For Oracle 12c, add the **APEX** and **APEX-DEV** options.
   - For Oracle 11g, first add the **XMLDB** option as a prerequisite, then add the **APEX** and **APEX-DEV** options.

   Important
   If you add the APEX options to an existing option group that is already attached to one or more DB instances, a brief outage occurs while all the DB instances are automatically restarted.

   For more information about adding options, see Adding an Option to an Option Group (p. 157).
3. Apply the option group to a new or existing DB instance:

   - For a new DB instance, you apply the option group when you launch the instance. For more information, see Creating a DB Instance Running the Oracle Database Engine (p. 949).
   - For an existing DB instance, you apply the option group by modifying the instance and attaching the new option group. When you add the APEX options to an existing DB instance, a brief outage occurs while your DB instance is automatically restarted. For more information, see Modifying a DB Instance Running the Oracle Database Engine (p. 967).

Unlocking the Public User Account

After the Amazon RDS APEX options are installed, you must change the password for the APEX public user account, and then unlock the account. You can do this by using the Oracle SQL*Plus command line utility. Connect to your DB instance as the master user, and issue the following commands. Replace `new_password` with a password of your choice.

```
alter user APEX_PUBLIC_USER identified by new_password;
alter user APEX_PUBLIC_USER account unlock;
```

Installing and Configuring the APEX Listener

You are now ready to install and configure a listener for use with Oracle APEX. You can use one of these products for this purpose:

- For APEX version 5.0 and later, use Oracle Rest Data Services (ORDS)
- For APEX version 4.1.1, use Oracle APEX Listener version 1.1.4
- Oracle HTTP Server and `mod_plsql`

> **Note**
> Amazon RDS doesn't support the Oracle XML DB HTTP server with the embedded PL/SQL gateway; you can't use this as an APEX Listener. In general, Oracle recommends against using the embedded PL/SQL gateway for applications that run on the internet.

You must install the APEX Listener on a separate host such as an Amazon EC2 instance, an on-premises server at your company, or your desktop computer.

The following procedure shows you how to install and configure the APEX Listener. We assume that the name of your host is `myapexhost.example.com`, and that your host is running Linux.

To install and configure the APEX Listener

1. Log in to `myapexhost.example.com` as `root`.
2. Create a nonprivileged OS user to own the APEX Listener installation. The following command creates a new user named *apexuser*.

   ```
   useradd -d /home/apexuser apexuser
   ```

   The following command assigns a password to the new user.

   ```
   passwd apexuser;
   ```
3. Log in to `myapexhost.example.com` as `apexuser`, and download the APEX and APEX Listener installation files from Oracle:
   - http://www.oracle.com/technetwork/developer-tools/apex/downloads/index.html
   - http://www.oracle.com/technetwork/developer-tools/apex-listener/downloads/index.html
   - Oracle Application Express Prior Release Archives
4. Unzip the APEX file:

Listener Type	Instructions
ORDS	Run the following code: `unzip ords.<version>.zip`
APEX Listener	Run the following code: `unzip apex_<version>.zip`

5. Create a new directory and open the APEX Listener file:

Listener Type	Instructions
ORDS	Run the following code: `mkdir /home/apexuser/ORDS`

Listener Type	Instructions
	`cd /home/apexuser/ORDS` `unzip ../ords.<version>.zip`

APEX Listener	Run the following code:
	`mkdir /home/apexuser/apexlistener`
	`cd /home/apexuser/apexlistener`
	`unzip ../apex_listener.<version>.zip`

6. While you are still in the directory from the previous step, run the listener program.

Listener Type	Instructions
ORDS	Run the following code: `java -jar ords.war setup` The program prompts you for the following information. The default values are in brackets. • The name of the database server [localhost] • The database listen port [1521] • Database service name or database SID [1] 1 to specify the database service name, 2 to specify the database SID • Database SID [xe] • Database user name [APEX_PUBLIC_USER] • Database password

APEX Listener	Run the following code:
	`java -Dapex.home=./apex -Dapex.images=/home/apexuser/apex/images -Dapex.erase -jar ./apex.war`
	The program prompts you for the following:
	• The APEX Listener Administrator user name. The default is *adminlistener*.
	• A password for the APEX Listener Administrator.
	• The APEX Listener Manager user name. The default is *managerlistener*.
	• A password for the APEX Listener Administrator.
	The program prints a URL that you need in order to complete the configuration, as follows:
	`INFO: Please complete configuration at: http://localhost:8080/apex/listenerConfigure`
	`Database is not yet configured`
	Leave the APEX Listener running. It needs to continue running for you to use Oracle Application Express. When you have finished this configuration procedure, you can run the listener in the background.

Listener Type	Instructions
	From your web browser, go to the URL provided by the APEX Listener program. The Oracle Application Express Listener administration window appears. Type the following information: • **Username** – `APEX_PUBLIC_USER` • **Password** – the password for *APEX_PUBLIC_USER*. This password is the one that you specified earlier when you configured the APEX repository. For more information, see Unlocking the Public User Account (p. 995). • **Connection Type** – Basic • **Hostname** – the endpoint of your Amazon RDS DB instance, such as `mydb.f9rbfa893tft.us-east-1.rds.amazonaws.com`. • **Port** – 1521 • **SID** – the name of the database on your Amazon RDS DB instance, such as `mydb`. Choose **Apply**. The APEX administration window appears.

7. You must set a password for the APEX `admin` user. To do this, use SQL*Plus to connect to your DB instance as the master user, and then issue the following commands:

```
EXEC rdsadmin.rdsadmin_util.grant_apex_admin_role;
grant APEX_ADMINISTRATOR_ROLE to master;
@/home/apexuser/apex/apxchpwd.sql
```

   Replace `master` with your master user name. When the `apxchpwd.sql` script prompts you, type a new `admin` password.

8. For ORDS, start the APEX Listener. Run the following code:

```
java -jar ords.war
```

   The first time you start the APEX Listener, you are prompted to provide the location of the APEX Static resources. This images folder is located in the /apex/images directory in the installation directory for APEX.

9. Return to the APEX administration window in your browser and choose **Administration**. Next, choose **Application Express Internal Administration**. When you are prompted for credentials, type the following information:

   • **User name** – `admin`
   • **Password** – the password you set using the `apxchpwd.sql` script

   Choose **Login**, and then set a new password for the `admin` user.

The APEX Listener is now ready for use.

Upgrading the APEX Version

If you are planning to do a major version upgrade of your DB instance, and you are using an APEX version that is not compatible with your target database version, you can upgrade your version of APEX at the same time as your DB instance. This can reduce the time it takes to upgrade your DB instance.

Important
Back up your DB instance before you upgrade APEX. For more information, see Creating a DB Snapshot (p. 207) and Testing an Upgrade (p. 976).

To upgrade APEX with your DB instance, do the following:

- Create a new option group for the upgraded version of your DB instance.
- Add the upgraded versions of APEX and APEX-DEV to the new option group. Be sure to include any other options that your DB instance uses. For more information, see Option Group Considerations (p. 976).
- When you upgrade your DB instance, specify the new option group for your upgraded DB instance.

After you upgrade your version of APEX, the APEX schema for the previous version might still exist in your database. If you don't need it anymore, you can drop the old APEX schema from your database after you upgrade.

Removing the APEX Option

You can remove the Amazon RDS APEX options from a DB instance. To remove the APEX options from a DB instance, do one of the following:

- To remove the APEX options from multiple DB instances, remove the APEX options from the option group they belong to. This change affects all DB instances that use the option group. When you remove the APEX options from an option group that is attached to multiple DB instances, a brief outage occurs while all the DB instances are restarted.

  For more information, see Removing an Option from an Option Group (p. 167).
- To remove the APEX options from a single DB instance, modify the DB instance and specify a different option group that doesn't include the APEX options. You can specify the default (empty) option group, or a different custom option group. When you remove the APEX options, a brief outage occurs while your DB instance is automatically restarted.

  For more information, see Modifying a DB Instance Running the Oracle Database Engine (p. 967).

When you remove the APEX options from a DB instance, the APEX schema is removed from your database.

Related Topics

- Working with Option Groups (p. 153)
- Options for Oracle DB Instances (p. 993)

Oracle Label Security

Amazon RDS supports Oracle Label Security for Oracle Enterprise Edition, version 12c, through the use of the OLS option.

Most database security controls access at the object level. Oracle Label Security provides fine-grained control of access to individual table rows. For example, you can use Label Security to enforce regulatory compliance with a policy-based administration model. You can use Label Security policies to control access to sensitive data, and restrict access to only users with the appropriate clearance level. For more information, see Introduction to Oracle Label Security in the Oracle documentation.

Prerequisites for Oracle Label Security

The following are prerequisites for using Oracle Label Security:

- Your DB instance must use the Bring Your Own License model. For more information, see Oracle Licensing (p. 933).
- You must have a valid license for Oracle Enterprise Edition with Software Update License and Support.
- Your Oracle license must include the Label Security option.

Adding the Oracle Label Security Option

The general process for adding the Oracle Label Security option to a DB instance is the following:

1. Create a new option group, or copy or modify an existing option group.
2. Add the option to the option group.
3. Associate the option group with the DB instance.

After you add the Label Security option, as soon as the option group is active, Label Security is active.

To add the Label Security option to a DB instance

1. Determine the option group you want to use. You can create a new option group or use an existing option group. If you want to use an existing option group, skip to the next step. Otherwise, create a custom DB option group with the following settings:

   a. For **Engine**, choose **oracle-ee**.
   b. For **Major Engine Version**, choose **12.1**.

   For more information, see Creating an Option Group (p. 154).
2. Add the **OLS** option to the option group. For more information about adding options, see Adding an Option to an Option Group (p. 157).

   Important
   If you add Label Security to an existing option group that is already attached to one or more DB instances, all the DB instances are restarted.
3. Apply the option group to a new or existing DB instance:

   - For a new DB instance, you apply the option group when you launch the instance. For more information, see Creating a DB Instance Running the Oracle Database Engine (p. 949).
   - For an existing DB instance, you apply the option group by modifying the instance and attaching the new option group. When you add the Label Security option to an existing DB instance, a brief outage occurs while your DB instance is automatically restarted. For more information, see Modifying a DB Instance Running the Oracle Database Engine (p. 967).

Using Oracle Label Security

To use Oracle Label Security, you create policies that control access to specific rows in your tables. For more information, see Creating an Oracle Label Security Policy in the Oracle documentation.

When you work with Label Security, you perform all actions as the LBAC_DBA role. The master user for your DB instance is granted the LBAC_DBA role. You can grant the LBAC_DBA role to other users so that they can administer Label Security policies.

You can configure Label Security through the Oracle Enterprise Manager (OEM) Cloud Control. Amazon RDS supports the OEM Cloud Control through the Management Agent option. For more information, see Oracle Management Agent for Enterprise Manager Cloud Control (p. 1010).

Removing the Oracle Label Security Option

You can remove Oracle Label Security from a DB instance.

To remove Label Security from a DB instance, do one of the following:

- To remove Label Security from multiple DB instances, remove the Label Security option from the option group they belong to. This change affects all DB instances that use the option group. When you remove Label Security from an option group that is attached to multiple DB instances, all the DB instances are restarted. For more information, see Removing an Option from an Option Group (p. 167).
- To remove Label Security from a single DB instance, modify the DB instance and specify a different option group that doesn't include the Label Security option. You can specify the default (empty) option group, or a different custom option group. When you remove the Label Security option, a brief outage occurs while your DB instance is automatically restarted. For more information, see Modifying a DB Instance Running the Oracle Database Engine (p. 967).

Troubleshooting

The following are issues you might encounter when you use Oracle Label Security.

Issue	Troubleshooting Suggestions
When you try to create a policy, you see an error message similar to the following: `insufficient authorization for the SYSDBA package`.	A known issue with Oracle's Label Security feature prevents users with usernames of 16 or 24 characters from running Label Security commands. You can create a new user with a different number of characters, grant LBAC_DBA to the new user, log in as the new user, and run the OLS commands as the new user. For additional information, please contact Oracle support.

Related Topics

- Working with Option Groups (p. 153)
- Options for Oracle DB Instances (p. 993)

Oracle Native Network Encryption

Amazon RDS supports Oracle native network encryption (NNE). With native network encryption, you can encrypt data as it moves to and from a DB instance. Amazon RDS supports NNE for all editions of Oracle.

A detailed discussion of Oracle native network encryption is beyond the scope of this guide, but you should understand the strengths and weaknesses of each algorithm and key before you decide on a solution for your deployment. For information about the algorithms and keys that are available through Oracle native network encryption, see Configuring Network Data Encryption in the Oracle documentation. For more information about AWS security, see the AWS Security Center.

Note
You can use Native Network Encryption or Secure Sockets Layer, but not both. For more information, see Oracle SSL (p. 1021).

NNE Option Settings

Amazon RDS supports the following settings for the NNE option.

Option Setting	Valid Values	Default Value	Description
SQLNET.ENCRYPTION_SERVER	`Accepted`, `Rejected`, `Requested`, `Required`	`Requested`	The encryption behavior when a client, or a server acting as a client, connects to the DB instance. `Requested` indicates that the DB instance does not require traffic from the client to be encrypted.
SQLNET.CRYPTO_CHECKSUM_SERVER	`Accepted`, `Rejected`, `Requested`, `Required`	`Requested`	The data integrity behavior when a client, or a server acting as a client, connects to the DB instance. `Requested` indicates that the DB instance does not require the client to perform a checksum.
SQLNET.ENCRYPTION_TYPES_SERVER	`AES256`, `AES192`, `3DES168`, `AES128`, `3DES112`, `RC4_256`, `RC4_128`, `RC4_56`, `RC4_40`, `DES`, `DES40`	`AES256`, `AES192`, `3DES168`, `AES128`, `3DES112`, `RC4_256`, `RC4_128`, `RC4_56`, `RC4_40`, `DES`, `DES40`	A list of encryption algorithms used by the DB instance. The DB instance will use each algorithm, in order, to attempt to decrypt the client input until an algorithm succeeds or until the end of the list is reached. Amazon RDS uses the following default list from Oracle. You can change the order or limit the algorithms that the DB instance will accept. 1. `RC4_256`: RSA RC4 (256-bit key size) 2. `AES256`: AES (256-bit key size) 3. `AES192`: AES (192-bit key size) 4. `3DES168`: 3-key Triple-DES (112-bit effective key size)

Option Setting	Valid Values	Default Value	Description
			5. `RC4_128`: RSA RC4 (128-bit key size) 6. `AES128`: AES (128-bit key size) 7. `3DES112`: 2-key Triple-DES (80-bit effective key size) 8. `RC4_56`: RSA RC4 (56-bit key size) 9. `DES`: Standard DES (56-bit key size) 10`RC4_40`: RSA RC4 (40-bit key size) 11`DES40`: DES40 (40-bit key size)
SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER	SHA1, MD5	SHA1	The checksum algorithm.

Adding the NNE Option

The general process for adding the NNE option to a DB instance is the following:

1. Create a new option group, or copy or modify an existing option group.
2. Add the option to the option group.
3. Associate the option group with the DB instance.

After you add the NNE option, as soon as the option group is active, NNE is active.

To add the NNE option to a DB instance

1. For **Engine**, choose the Oracle edition that you want to use. NNE is supported on all editions.
2. For **Major Engine Version**, choose **11.2** or **12.1**.

   For more information, see Creating an Option Group (p. 154).
3. Add the **NNE** option to the option group. For more information about adding options, see Adding an Option to an Option Group (p. 157).

   Note
   After you add the NNE option, you don't need to restart your DB instances. As soon as the option group is active, NNE is active.
4. Apply the option group to a new or existing DB instance:

   - For a new DB instance, you apply the option group when you launch the instance. For more information, see Creating a DB Instance Running the Oracle Database Engine (p. 949).
   - For an existing DB instance, you apply the option group by modifying the instance and attaching the new option group. After you add the NNE option, you don't need to restart your DB instance. As soon as the option group is active, NNE is active. For more information, see Modifying a DB Instance Running the Oracle Database Engine (p. 967).

Using NNE

With Oracle native network encryption, you can also specify network encryption on the client side. On the client (the computer used to connect to the DB instance), you can use the sqlnet.ora file to specify the following client settings: SQLNET.CRYPTO_CHECKSUM_CLIENT , SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT, SQLNET.ENCRYPTION_CLIENT, and

SQLNET.ENCRYPTION_TYPES_CLIENT. For information, see Configuring Network Data Encryption and Integrity for Oracle Servers and Clients in the Oracle documentation.

Sometimes, the DB instance will reject a connection request from an application, for example, if there is a mismatch between the encryption algorithms on the client and on the server.

To test Oracle native network encryption , add the following lines to the sqlnet.ora file on the client:

```
DIAG_ADR_ENABLED=off
TRACE_DIRECTORY_CLIENT=/tmp
TRACE_FILE_CLIENT=nettrace
TRACE_LEVEL_CLIENT=16
```

These lines generate a trace file on the client called `/tmp/nettrace*` when the connection is attempted. The trace file contains information on the connection. For more information about connection-related issues when you are using Oracle Native Network Encryption, see About Negotiating Encryption and Integrity in the Oracle documentation.

Modifying NNE Settings

After you enable NNE, you can modify settings for the option. For more information about how to modify option settings, see Modifying an Option Setting (p. 163). For more information about each setting, see NNE Option Settings (p. 1003).

Removing the NNE Option

You can remove NNE from a DB instance.

To remove NNE from a DB instance, do one of the following:

- To remove NNE from multiple DB instances, remove the NNE option from the option group they belong to. This change affects all DB instances that use the option group. After you remove the NNE option, you don't need to restart your DB instances. For more information, see Removing an Option from an Option Group (p. 167).
- To remove NNE from a single DB instance, modify the DB instance and specify a different option group that doesn't include the NNE option. You can specify the default (empty) option group, or a different custom option group. After you remove the NNE option, you don't need to restart your DB instance. For more information, see Modifying a DB Instance Running the Oracle Database Engine (p. 967).

Related Topics

- Working with Option Groups (p. 153)
- Options for Oracle DB Instances (p. 993)

Oracle Enterprise Manager

Amazon RDS supports Oracle Enterprise Manager (OEM). OEM is Oracle's integrated enterprise information technology management product line.

Amazon RDS supports OEM through the following options.

Option	Option ID	Support For
OEM Database (p. 1007)	`OEM`	OEM Database Express 12c

Option	Option ID	Support For
		OEM 11g Database Control
OEM Management Agent (p. 1010)	`OEM_AGENT`	OEM Cloud Control for 13c OEM Cloud Control for 12c

Note
You can use OEM Database or OEM Management Agent, but not both.

Oracle Enterprise Manager Database Express

Amazon RDS supports Oracle Enterprise Manager (OEM) Database Express through the use of the OEM option. Amazon RDS supports the following versions of OEM database:

- Oracle Enterprise Manager Database Express 12c
- Oracle Enterprise Manager 11g Database Control

OEM Database Express and Database Control are similar tools that have a web-based interface for Oracle database administration. For more information about these tools, see Accessing Enterprise Manager Database Express 12c and Accessing Enterprise Manager 11g Database Control in the Oracle documentation.

The following are some limitations to using OEM Database:

- OEM Database is not supported on the following DB instance classes: db.t2.micro, db.t2.small, db.m1.small.

  For more information about DB instance classes, see DB Instance Class Support for Oracle (p. 934).
- OEM 11g Database Control is not compatible with the following time zones: America/Argentina/Buenos_Aires, America/Matamoros, America/Monterrey, America/Toronto, Asia/Ashgabat, Asia/Dhaka, Asia/Kathmandu, Asia/Kolkata, Asia/Ulaanbaatar, Atlantic/Cape_Verde, Australia/Eucla, Pacific/Kiritimati.

  For more information about time zone support, see Oracle Time Zone (p. 1033).

OEM Database Option Settings

Amazon RDS supports the following settings for the OEM option.

Option Setting	Valid Values	Description
Port	An integer value	The port on the DB instance that listens for OEM Database. The default for OEM Database Express 12c is 5500. The default for OEM 11g Database Control is 1158.
Security Groups	—	A security group that has access to **Port**.

Adding the OEM Database Option

The general process for adding the OEM option to a DB instance is the following:

1. Create a new option group, or copy or modify an existing option group.
2. Add the option to the option group.
3. Associate the option group with the DB instance.

After you add the OEM option, you don't need to restart your DB instance. As soon as the option group is active, the OEM Database is active.

To add the OEM option to a DB instance

1. Determine the option group you want to use. You can create a new option group or use an existing option group. If you want to use an existing option group, skip to the next step. Otherwise, create a custom DB option group with the following settings:

   a. For **Engine** choose the oracle edition for your DB instance.
   b. For **Major Engine Version** choose **11.2** or **12.1** for your DB instance.

   For more information, see Creating an Option Group (p. 154).
2. Add the **OEM** option to the option group, and configure the option settings. For more information about adding options, see Adding an Option to an Option Group (p. 157). For more information about each setting, see OEM Database Option Settings (p. 1007).
3. Apply the option group to a new or existing DB instance:

   - For a new DB instance, you apply the option group when you launch the instance. For more information, see Creating a DB Instance Running the Oracle Database Engine (p. 949).
   - For an existing DB instance, you apply the option group by modifying the instance and attaching the new option group. For more information, see Modifying a DB Instance Running the Oracle Database Engine (p. 967).

Using OEM Database

After you enable the OEM option, you can begin using the OEM Database tool from your web browser.

You can access either OEM Database Control or OEM Database Express from your web browser. For example, if the endpoint for your Amazon RDS DB instance is `mydb.f9rbfa893tft.us-east-1.rds.amazonaws.com`, and your OEM port is 1158, then the URL to access the OEM Database Control the following.

```
https://mydb.f9rbfa893tft.us-east-1.rds.amazonaws.com:1158/em
```

When you access either tool from you web browser, a login window appears that prompts you for a user name and password. Type the master user name and master password for your DB instance. You are now ready to manage your Oracle databases.

Modifying OEM Database Settings

After you enable OEM Database, you can modify the Security Groups setting for the option.

You can't modify the OEM port number after you have associated the option group with a a DB instance. To change the OEM port number for a DB instance, do the following:

1. Create a new option group.
2. Add the OEM option with the new port number to the new option group.
3. Remove the existing option group from the DB instance.
4. Add the new option group to the DB instance.

For more information about how to modify option settings, see Modifying an Option Setting (p. 163). For more information about each setting, see OEM Database Option Settings (p. 1007).

Removing the OEM Database Option

You can remove the OEM option from a DB instance. After you remove the OEM option, you don't need to restart your DB instance.

To remove the OEM option from a DB instance, do one of the following:

- Remove the OEM option from the option group it belongs to. This change affects all DB instances that use the option group. For more information, see Removing an Option from an Option Group (p. 167)
- Modify the DB instance and specify a different option group that doesn't include the OEM option. This change affects a single DB instance. You can specify the default (empty) option group, or a different custom option group. For more information, see Modifying a DB Instance Running the Oracle Database Engine (p. 967).

Related Topics

- Working with Option Groups (p. 153)
- Options for Oracle DB Instances (p. 993)

Oracle Management Agent for Enterprise Manager Cloud Control

Amazon RDS supports Oracle Enterprise Manager (OEM) Management Agent through the use of the OEM_AGENT option. Amazon RDS supports Management Agent for the following versions of OEM:

- Oracle Enterprise Manager Cloud Control for 13c
- Oracle Enterprise Manager Cloud Control for 12c

Management Agent is a software component that monitors targets running on hosts and communicates that information to the middle-tier Oracle Management Service (OMS). For more information, see Overview of Oracle Enterprise Manager Cloud Control 12c and Overview of Oracle Enterprise Manager Cloud Control 13c in the Oracle documentation.

The following are some limitations to using Management Agent:

- Administrative tasks such as job execution and database patching, that require host credentials, are not supported.
- Host metrics and the process list are not guaranteed to reflect the actual system state.
- Autodiscovery is not supported. You must manually add database targets.
- OMS module availability depends your database edition. For example, the database performance diagnosis and tuning module is only available for Oracle Database Enterprise Edition.
- Management Agent consumes additional memory and computing resources. If you experience performance problems after enabling the OEM_AGENT option, we recommend that you scale up to a larger DB instance class. For more information, see DB Instance Class (p. 92) and Modifying a DB Instance Running the Oracle Database Engine (p. 967).

Prerequisites for Management Agent

The following are prerequisites for using Management Agent:

- An Amazon RDS DB instance running Oracle version 12.1.0.2 or 11.2.0.4.
- At least 3.3 GB of storage space for OEM 13c2.
- At least 3 GB of storage space for OEM 13c1.
- At least 2 GB of storage space for OEM 12c.
- An Oracle Management Service (OMS), configured to connect to your Amazon RDS DB instance.
  - For OMS 13c2 with Oracle patch 25163555 applied, use OEM Agent 13.2.0.0.v2 or later.
  - For unpatched OMS 13c2, use OEM Agent 13.2.0.0.v1.
- In most cases, you need to configure your VPC to allow connections from OMS to your DB instance. If you are not familiar with Amazon Virtual Private Cloud (Amazon VPC), we recommend that you complete the steps in Tutorial: Create an Amazon VPC for Use with an Amazon RDS DB Instance (p. 406) before continuing.

Additional configuration is required to allow your OMS host and your Amazon RDS DB instance to communicate. You must also do the following:

- To connect from the Management Agent to your OMS, if your OMS is behind a firewall, you must add the IP addresses of your DB instances to your OMS.
- To connect from your OMS to the Management Agent, if your OMS has a publicly resolvable host name, you must add the OMS address to a security group. Your security group must have inbound rules that allow access to the DB instance port and the Management Agent port. For an example of

creating a security and adding inbound rules, see Tutorial: Create an Amazon VPC for Use with an Amazon RDS DB Instance (p. 406).

- To connect from your OMS to the Management Agent, if your OMS doesn't have a publicly resolvable host name, use one of the following:
  - If your OMS is hosted on an Amazon Elastic Compute Cloud (Amazon EC2) instance in a private VPC, you can set up VPC peering to connect from OMS to Management Agent. For more information, see A DB Instance in a VPC Accessed by an EC2 Instance in a Different VPC (p. 394).
  - If your OMS is hosted on-premises, you can set up a VPN connection to allow access from OMS to Management Agent. For more information, see A DB Instance in a VPC Accessed by a Client Application Through the Internet (p. 396) or VPN Connections.

Management Agent Option Settings

Amazon RDS supports the following settings for the Management Agent option.

Option Setting	Valid Values	Description
Version (`AGENT_VERSION`)	13.2.0.0 13.1.0.0 12.1.0.4 12.1.0.5	The version of the Management Agent software.
Port (`AGENT_PORT`)	An integer value	The port on the DB instance that listens for the OMS host. The default is 3872. Your OMS host must belong to a security group that has access to this port.
Security Groups	—	A security group that has access to **Port**. Your OMS host must belong to this security group.
OMS_HOST	A string value, for example *`my.example.oms`*	The publicly accessible host name or IP address of the OMS.
OMS_PORT	An integer value	The port on the OMS host that listens for the Management Agent.
AGENT_REGISTRATION_PASSWORD	A string value	The password that the Management Agent uses to authenticate itself with the OMS. We recommend that you create a persistent password in your OMS before enabling the OEM_AGENT option. With a persistent password you can share a single Management Agent option group among multiple Amazon RDS databases.

Adding the Management Agent Option

The general process for adding the Management Agent option to a DB instance is the following:

1. Create a new option group, or copy or modify an existing option group.
2. Add the option to the option group.
3. Associate the option group with the DB instance.

After you add the Management Agent option, you don't need to restart your DB instance. As soon as the option group is active, the OEM Agent is active.

To add the Management Agent option to a DB instance

1. Determine the option group you want to use. You can create a new option group or use an existing option group. If you want to use an existing option group, skip to the next step. Otherwise, create a custom DB option group with the following settings:

   a. For **Engine** choose the oracle edition for your DB instance.
   b. For **Major Engine Version** choose **11.2** or **12.1** for your DB instance.

   For more information, see Creating an Option Group (p. 154).
2. Add the **OEM_AGENT** option to the option group, and configure the option settings. For more information about adding options, see Adding an Option to an Option Group (p. 157). For more information about each setting, see Management Agent Option Settings (p. 1011).
3. Apply the option group to a new or existing DB instance:

   - For a new DB instance, you apply the option group when you launch the instance. For more information, see Creating a DB Instance Running the Oracle Database Engine (p. 949).
   - For an existing DB instance, you apply the option group by modifying the instance and attaching the new option group. For more information, see Modifying a DB Instance Running the Oracle Database Engine (p. 967).

Using the Management Agent

After you enable the Management Agent option, use the following procedure to begin using it.

To use the Management Agent

1. Unlock and reset the DBSNMP account credential, by running the following code on your target database on your DB instance, and using your master user account.

```
ALTER USER dbsnmp IDENTIFIED BY new_password ACCOUNT UNLOCK;
```

2. Add your targets to the OMS console manually:

   a. In your OMS console, choose **Setup**, **Add Target**, **Add Targets Manually**.
   b. Choose **Add Targets Declaratively by Specifying Target Monitoring Properties**.
   c. For **Target Type**, choose **Database Instance**.
   d. For **Monitoring Agent**, choose the agent with the same identifier as your Amazon RDS DB instance identifier.
   e. Choose **Add Manually**.
   f. Specify the following database properties:

- For **Target name**, type a name.
- For **Database system name**, type a name.
- For **Monitor username**, type `dbsnmp`.
- For **Monitor password**, type the password from Step 1.
- For **Role**, type **normal**.
- For **Oracle home path**, type **/oracle**.
- For **Listener Machine name**, the agent identifier already appears.
- For **Port**, type the database port. The RDS default port is 1521.
- For **Database name**, type the name of your database.

g. Choose **Test Connection**.

h. Choose **Next**. The target database appears in your list of monitored resources.

Modifying Management Agent Settings

After you enable the Management Agent, you can modify settings for the option. For more information about how to modify option settings, see Modifying an Option Setting (p. 163). For more information about each setting, see Management Agent Option Settings (p. 1011).

Removing the Management Agent Option

You can remove the OEM Agent from a DB instance. After you remove the OEM Agent, you don't need to restart your DB instance.

To remove the OEM Agent from a DB instance, do one of the following:

- Remove the OEM Agent option from the option group it belongs to. This change affects all DB instances that use the option group. For more information, see Removing an Option from an Option Group (p. 167)
- Modify the DB instance and specify a different option group that doesn't include the OEM Agent option. This change affects a single DB instance. You can specify the default (empty) option group, or a different custom option group. For more information, see Modifying a DB Instance Running the Oracle Database Engine (p. 967).

Related Topics

- Working with Option Groups (p. 153)
- Options for Oracle DB Instances (p. 993)

Oracle Locator

Amazon RDS supports Oracle Locator through the use of the `LOCATOR` option. Oracle Locator provides capabilities that are typically required to support internet and wireless service-based applications and partner-based GIS solutions. Oracle Locator is a limited subset of Oracle Spatial. For more information, see Oracle Locator in the Oracle documentation.

> **Important**
> If you use Oracle Locator, Amazon RDS automatically updates your DB instance to the latest Oracle PSU if there are security vulnerabilities with a Common Vulnerability Scoring System (CVSS) score of 9+ or other announced security vulnerabilities.

Amazon RDS supports Oracle Locator for the following editions and versions of Oracle:

- Oracle Standard Edition (SE2) or Enterprise Edition, version 12.1.0.2.v6 or later
- Oracle Standard Edition (SE, SE1) or Enterprise Edition, version 11.2.0.4.v10 or later

Prerequisites for Oracle Locator

The following are prerequisites for using Oracle Locator:

- Your DB instance must be inside a virtual private cloud (VPC). For more information, see Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform (p. 391).
- Your DB instance must be of sufficient class. Oracle Locator is not supported for the db.m1.small, db.t2.micro, or db.t2.small DB instance classes. For more information, see DB Instance Class Support for Oracle (p. 934).
- Your DB instance must have Auto Minor Version Upgrade enabled. Amazon RDS updates your DB instance to the latest Oracle PSU if there are security vulnerabilities with a CVSS score of 9+ or other announced security vulnerabilities. For more information, see Settings for Oracle DB Instances (p. 968).
- If your DB instance is version 11.2.0.4.v10 or later, you must install the `XMLDB` option. For more information, see Oracle XML DB (p. 1040).

Best Practices for Oracle Locator

The following are best practices for using Oracle Locator:

- For maximum security, use the `LOCATOR` option with Secure Sockets Layer (SSL). For more information, see Oracle SSL (p. 1021).
- Configure your DB instance to restrict access to your DB instance. For more information, see Scenarios for Accessing a DB Instance in a VPC (p. 392) and Working with an Amazon RDS DB Instance in a VPC (p. 399).

Adding the Oracle Locator Option

The following is the general process for adding the `LOCATOR` option to a DB instance:

1. Create a new option group, or copy or modify an existing option group.
2. Add the option to the option group.
3. Associate the option group with the DB instance.

There is a brief outage while the `LOCATOR` option is added. After you add the option, you don't need to restart your DB instance. As soon as the option group is active, Oracle Locator is available.

To add the `LOCATOR` option to a DB instance

1. Determine the option group that you want to use. You can create a new option group or use an existing option group. If you want to use an existing option group, skip to the next step. Otherwise, create a custom DB option group with the following settings:

   a. For **Engine**, choose the oracle edition for your DB instance.
   b. For **Major Engine Version**, choose **11.2** or **12.1** for your DB instance.

   For more information, see Creating an Option Group (p. 154).
2. Add the **LOCATOR** option to the option group. For more information about adding options, see Adding an Option to an Option Group (p. 157).
3. Apply the option group to a new or existing DB instance:

   - For a new DB instance, you apply the option group when you launch the instance. For more information, see Creating a DB Instance Running the Oracle Database Engine (p. 949).
   - For an existing DB instance, you apply the option group by modifying the instance and attaching the new option group. For more information, see Modifying a DB Instance Running the Oracle Database Engine (p. 967).

Using Oracle Locator

After you enable the Oracle Locator option, you can begin using it. You should only use Oracle Locator features. Don't use any Oracle Spatial features unless you have a license for Oracle Spatial.

For a list of features that are supported for Oracle Locator, see Features Included with Locator in the Oracle documentation.

For a list of features that are not supported for Oracle Locator, see Features Not Included with Locator in the Oracle documentation.

Removing the Oracle Locator Option

You can remove the `LOCATOR` option from a DB instance. There is a brief outage while the option is removed. After you remove the `LOCATOR` option, you don't need to restart your DB instance.

> **Warning**
> Removing the `LOCATOR` option can result in data loss if the DB instance is using data types that were enabled as part of the option. Back up your data before proceeding. For more information, see Backing Up and Restoring Amazon RDS DB Instances (p. 200).

To remove the `LOCATOR` option from a DB instance, do one of the following:

- Remove the `LOCATOR` option from the option group it belongs to. This change affects all DB instances that use the option group. For more information, see Removing an Option from an Option Group (p. 167).
- Modify the DB instance and specify a different option group that doesn't include the `LOCATOR` option. This change affects a single DB instance. You can specify the default (empty) option group or a different custom option group. For more information, see Modifying a DB Instance Running the Oracle Database Engine (p. 967).

Related Topics

- Oracle Spatial (p. 1019)

- Options for Oracle DB Instances (p. 993)
- Working with Option Groups (p. 153)

Oracle Multimedia

Amazon RDS supports Oracle Multimedia through the use of the `MULTIMEDIA` option. You can use Oracle Multimedia to store, manage, and retrieve images, audio, video, and other heterogeneous media data. For more information, see Oracle Multimedia in the Oracle documentation.

> **Important**
> If you use Oracle Multimedia, Amazon RDS automatically updates your DB instance to the latest Oracle PSU if there are security vulnerabilities with a Common Vulnerability Scoring System (CVSS) score of 9+ or other announced security vulnerabilities.

Amazon RDS supports Oracle Multimedia for the following editions and versions of Oracle:

- Oracle Enterprise Edition, version 12.1.0.2.v6 or later
- Oracle Enterprise Edition, version 11.2.0.4.v10 or later

Prerequisites for Oracle Multimedia

The following are prerequisites for using Oracle Multimedia:

- Your DB instance must be inside a virtual private cloud (VPC). For more information, see Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform (p. 391).
- Your DB instance must be of sufficient class. Oracle Multimedia is not supported for the db.m1.small, db.t2.micro, or db.t2.small DB instance classes. For more information, see DB Instance Class Support for Oracle (p. 934).
- Your DB instance must have Auto Minor Version Upgrade enabled. Amazon RDS updates your DB instance to the latest Oracle PSU if there are security vulnerabilities with a CVSS score of 9+ or other announced security vulnerabilities. For more information, see Settings for Oracle DB Instances (p. 968).
- If your DB instance is version 11.2.0.4.v10 or later, you must install the `XMLDB` option. For more information, see Oracle XML DB (p. 1040).

Best Practices for Oracle Multimedia

The following are best practices for using Oracle Multimedia:

- For maximum security, use the `MULTIMEDIA` option with Secure Sockets Layer (SSL). For more information, see Oracle SSL (p. 1021).
- Configure your DB instance to restrict access to your DB instance. For more information, see Scenarios for Accessing a DB Instance in a VPC (p. 392) and Working with an Amazon RDS DB Instance in a VPC (p. 399).

Adding the Oracle Multimedia Option

The following is the general process for adding the `MULTIMEDIA` option to a DB instance:

1. Create a new option group, or copy or modify an existing option group.
2. Add the option to the option group.
3. Associate the option group with the DB instance.

There is a brief outage while the `MULTIMEDIA` option is added. After you add the option, you don't need to restart your DB instance. As soon as the option group is active, Oracle Multimedia is available.

To add the `MULTIMEDIA` option to a DB instance

1. Determine the option group that you want to use. You can create a new option group or use an existing option group. If you want to use an existing option group, skip to the next step. Otherwise, create a custom DB option group with the following settings:

   a. For **Engine**, choose **oracle-ee**.
   b. For **Major Engine Version**, choose **11.2** or **12.1** for your DB instance.

   For more information, see Creating an Option Group (p. 154).
2. Add the **MULTIMEDIA** option to the option group. For more information about adding options, see Adding an Option to an Option Group (p. 157).
3. Apply the option group to a new or existing DB instance:

   - For a new DB instance, you apply the option group when you launch the instance. For more information, see Creating a DB Instance Running the Oracle Database Engine (p. 949).
   - For an existing DB instance, you apply the option group by modifying the instance and attaching the new option group. For more information, see Modifying a DB Instance Running the Oracle Database Engine (p. 967).

Removing the Oracle Multimedia Option

You can remove the `MULTIMEDIA` option from a DB instance. There is a brief outage while the option is removed. After you remove the `MULTIMEDIA` option, you don't need to restart your DB instance.

> **Warning**
> Removing the `MULTIMEDIA` option can result in data loss if the DB instance is using data types that were enabled as part of the option. Back up your data before proceeding. For more information, see Backing Up and Restoring Amazon RDS DB Instances (p. 200).

To remove the `MULTIMEDIA` option from a DB instance, do one of the following:

- Remove the `MULTIMEDIA` option from the option group it belongs to. This change affects all DB instances that use the option group. For more information, see Removing an Option from an Option Group (p. 167).
- Modify the DB instance and specify a different option group that doesn't include the `MULTIMEDIA` option. This change affects a single DB instance. You can specify the default (empty) option group or a different custom option group. For more information, see Modifying a DB Instance Running the Oracle Database Engine (p. 967).

Related Topics

- Working with Option Groups (p. 153)
- Options for Oracle DB Instances (p. 993)

Oracle Spatial

Amazon RDS supports Oracle Spatial through the use of the `SPATIAL` option. Oracle Spatial provides a SQL schema and functions that facilitate the storage, retrieval, update, and query of collections of spatial data in an Oracle database. For more information, see Spatial Concepts in the Oracle documentation.

> **Important**
> If you use Oracle Spatial, Amazon RDS automatically updates your DB instance to the latest Oracle PSU if there are security vulnerabilities with a Common Vulnerability Scoring System (CVSS) score of 9+ or other announced security vulnerabilities.

Amazon RDS supports Oracle Spatial for the following editions and versions of Oracle:

- Oracle Enterprise Edition, version 12.1.0.2.v6 or later
- Oracle Enterprise Edition, version 11.2.0.4.v10 or later

Prerequisites for Oracle Spatial

The following are prerequisites for using Oracle Spatial:

- An Amazon RDS DB instance that's running Oracle Enterprise Edition version 12.1.0.2.v6 or later, or 11.2.0.4.v10 or later.
- Your DB instance must be inside a virtual private cloud (VPC). For more information, see Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform (p. 391).
- Your DB instance must be of sufficient class. Oracle Spatial is not supported for the db.m1.small, db.t2.micro, or db.t2.small DB instance classes. For more information, see DB Instance Class Support for Oracle (p. 934).
- Your DB instance must have Auto Minor Version Upgrade enabled. Amazon RDS updates your DB instance to the latest Oracle PSU if there are security vulnerabilities with a CVSS score of 9+ or other announced security vulnerabilities. For more information, see Settings for Oracle DB Instances (p. 968).
- If your DB instance is version 11.2.0.4.v10 or later, you must install the `XMLDB` option. For more information, see Oracle XML DB (p. 1040).
- An Oracle Spatial license from Oracle. For more information, see Oracle Spatial and Graph in the Oracle documentation.

Best Practices for Oracle Spatial

The following are best practices for using Oracle Spatial:

- For maximum security, use the `SPATIAL` option with Secure Sockets Layer (SSL). For more information, see Oracle SSL (p. 1021).
- Configure your DB instance to restrict access to your DB instance. For more information, see Scenarios for Accessing a DB Instance in a VPC (p. 392) and Working with an Amazon RDS DB Instance in a VPC (p. 399).

Adding the Oracle Spatial Option

The following is the general process for adding the `SPATIAL` option to a DB instance:

1. Create a new option group, or copy or modify an existing option group.

2. Add the option to the option group.
3. Associate the option group with the DB instance.

There is a brief outage while the `SPATIAL` option is added. After you add the option, you don't need to restart your DB instance. As soon as the option group is active, Oracle Spatial is available.

To add the `SPATIAL` option to a DB instance

1. Determine the option group that you want to use. You can create a new option group or use an existing option group. If you want to use an existing option group, skip to the next step. Otherwise, create a custom DB option group with the following settings:

   a. For **Engine**, choose **oracle-ee**.

   b. For **Major Engine Version**, choose **11.2** or **12.1** for your DB instance.

   For more information, see Creating an Option Group (p. 154).
2. Add the **SPATIAL** option to the option group. For more information about adding options, see Adding an Option to an Option Group (p. 157).
3. Apply the option group to a new or existing DB instance:

   - For a new DB instance, you apply the option group when you launch the instance. For more information, see Creating a DB Instance Running the Oracle Database Engine (p. 949).
   - For an existing DB instance, you apply the option group by modifying the instance and attaching the new option group. For more information, see Modifying a DB Instance Running the Oracle Database Engine (p. 967).

Removing the Oracle Spatial Option

You can remove the `SPATIAL` option from a DB instance. There is a brief outage while the option is removed. After you remove the `SPATIAL` option, you don't need to restart your DB instance.

> **Warning**
> Removing the `SPATIAL` option can result in data loss if the DB instance is using data types that were enabled as part of the option. Back up your data before proceeding. For more information, see Backing Up and Restoring Amazon RDS DB Instances (p. 200).

To remove the `SPATIAL` option from a DB instance, do one of the following:

- Remove the `SPATIAL` option from the option group it belongs to. This change affects all DB instances that use the option group. For more information, see Removing an Option from an Option Group (p. 167).
- Modify the DB instance and specify a different option group that doesn't include the `SPATIAL` option. This change affects a single DB instance. You can specify the default (empty) option group, or a different custom option group. For more information, see Modifying a DB Instance Running the Oracle Database Engine (p. 967).

Related Topics

- Oracle Locator (p. 1014)
- Options for Oracle DB Instances (p. 993)
- Working with Option Groups (p. 153)

Oracle SSL

You enable Secure Sockets Layer (SSL) encryption for an Oracle DB instance by adding the Oracle SSL option to the option group associated with an Oracle DB instance. You specify the port you want to communicate over using SSL. You must configure the Oracle client as shown in this following section.

You enable SSL encryption for an Oracle DB instance by adding the Oracle SSL option to the option group associated with the DB instance. Amazon RDS uses a second port, as required by Oracle, for SSL connections which allows both clear text and SSL-encrypted communication to occur at the same time between a DB instance and an Oracle client. For example, you can use the port with clear text communication to communicate with other resources inside a VPC while using the port with SSL-encrypted communication to communicate with resources outside the VPC.

Note
You can use Secure Sockets Layer or Native Network Encryption, but not both. For more information, see Oracle Native Network Encryption (p. 1003).

You can use SSL encryption with the following Oracle database versions and editions:

- 12.1.0.2: all versions, all editions including Standard Edition Two
- 11.2.0.4: all versions, Enterprise Edition
- 11.2.0.4: v6 and later, Standard Edition, Standard Edition One, Enterprise Edition

Note
You cannot use both SSL and Oracle native network encryption (NNE) on the same instance. If you use SSL encryption, you must disable any other connection encryption.

Configuring an Oracle Client to Use SSL with an Oracle DB Instance

You must configure the Oracle client before connecting to an Oracle DB instance that uses the Oracle SSL option.

To configure an Oracle client to use SSL to connect to an Oracle DB instance

1. Set the ORACLE_HOME environment variable to the location of your Oracle home directory.

   The path to your Oracle home directory depends on your installation. The following is an example that sets the ORACLE_HOME environment variable:

   ```
   prompt>export ORACLE_HOME=/home/user/app/user/product/12.1.0/dbhome_1
   ```

   For information about setting Oracle environment variables, see SQL*Plus Environment Variables in the Oracle documentation and the Oracle installation guide for your operating system.
2. Append **`$ORACLE_HOME/lib`** to the LD_LIBRARY_PATH environment variable.

   The following is an example that sets the LD_LIBRARY_PATH environment variable:

   ```
   prompt>export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ORACLE_HOME/lib
   ```
3. Create a directory for the Oracle wallet at $ORACLE_HOME/ssl_wallet.

The following is an example that creates the Oracle wallet directory:

```
prompt>mkdir $ORACLE_HOME/ssl_wallet
```

4. Download the RDS CA certificates file from https://s3.amazonaws.com/rds-downloads/rds-ca-2015-root.pem and then put the file in the ssl_wallet directory.

   The RDS CA certificates file for AWS GovCloud (US) is available at https://s3-us-gov-west-1.amazonaws.com/rds-downloads/rds-ca-2012-us-gov-west-1.pem.
5. In the $ORACLE_HOME/network/admin directory, modify or create the *tnsnames.ora* file and include the following entry:

```
<database name>= (DESCRIPTION = (ADDRESS_LIST = (ADDRESS = (PROTOCOL = TCPS)
   (HOST = <endpoint>) (PORT = <ssl port number>)))(CONNECT_DATA = (SID = <database
 name>))
   (SECURITY = (SSL_SERVER_CERT_DN =
 "C=US,ST=Washington,L=Seattle,O=Amazon.com,OU=RDS,CN=<endpoint>")))
```

6. In the same directory, modify or create the *sqlnet.ora* file and include the following parameters:

```
WALLET_LOCATION = (SOURCE = (METHOD = FILE) (METHOD_DATA = (DIRECTORY = $ORACLE_HOME/
ssl_wallet)))
SSL_CLIENT_AUTHENTICATION = FALSE
SSL_VERSION = 1.0
SSL_CIPHER_SUITES = (SSL_RSA_WITH_AES_256_CBC_SHA)
SSL_SERVER_DN_MATCH = ON
```

7. Run the following commands to create the Oracle wallet:

```
prompt>orapki wallet create -wallet $ORACLE_HOME/ssl_wallet -auto_login_only

prompt>orapki wallet add -wallet $ORACLE_HOME/ssl_wallet -trusted_cert -cert
      $ORACLE_HOME/ssl_wallet/rds-ca-2015-root.pem -auto_login_only
```

Connecting to an Oracle DB Instance Using SSL

After you configure the Oracle client to use SSL as described preceding, you can connect to the Oracle DB instance with the SSL option. For example, to connect to the DB instance using SQL*Plus, use the following command:

```
sqlplus '<mydbuser>@(DESCRIPTION = (ADDRESS = (PROTOCOL = TCPS)(HOST = <endpoint>) (PORT
 = <ssl port number>))(CONNECT_DATA = (SID = <database name>)))'
```

You can also connect to the Oracle DB instance without using SSL. For example, the following command connects to the DB instance through the clear text port without SSL encryption:

```
sqlplus '<mydbuser>@(DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = <endpoint>) (PORT
 = <port number>))(CONNECT_DATA = (SID = <database name>)))'
```

If you want to close Transmission Control Protocol (TCP) port access, create a security group with no IP address ingresses and add it to the instance. This addition closes connections over the TCP port, while still allowing connections over the SSL port that are specified from IP addresses within the range permitted by the SSL option security group.

Setting Up an SSL Connection Over JDBC

To use an SSL connection over JDBC, you must create a keystore, trust the Amazon RDS root CA certificate, and use the code snippet specified below.

To create the keystore in JKS format, use the following command. For more information about creating the keystore, see the Oracle documentation.

```
keytool -keystore clientkeystore -genkey -alias client
```

Next, follow these steps to trust the Amazon RDS root CA certificate:

1. 1. Download the Amazon RDS root CA certificate from https://s3.amazonaws.com/rds-downloads/rds-ca-2015-root.pem.
2. Convert the certificate to DER format using the following command:

   ```
   openssl x509 -outform der -in rds-ca-2015-root.pem -out rds-ca-2015-root.der
   ```

3. Import the certificate into the keystore using the following command:

   ```
   keytool -import -alias rds-root -keystore clientkeystore -file rds-ca-2015-root.der
   ```

The following code snippet shows how to setup the SSL connection using JDBC:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Properties;

public class OracleSslConnectionTest {
    private static final String DB_SERVER_NAME = "<dns-name-provided-by-amazon-rds>";
    private static final Integer SSL_PORT = "<ssl-option-port-configured-in-option-group>";
    private static final String DB_SID = "<oracle-sid>";
    private static final String DB_USER = "<user name>";
    private static final String DB_PASSWORD = "<password>";
    // This key store has only the prod root ca: https://s3.amazonaws.com/rds-downloads/
rds-ca-2015-root.pem
    private static final String KEY_STORE_FILE_PATH = "<file-path-to-keystore>";
    private static final String KEY_STORE_PASS = "<keystore-password>";
```

```
    public static void main(String[] args) throws SQLException {
        final Properties properties = new Properties();
        final String connectionString = String.format(
                "jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCPS)(HOST=%s)(PORT=%d))
(CONNECT_DATA=(SID=%s)))",
                DB_SERVER_NAME, SSL_PORT, DB_SID);
        properties.put("user", DB_USER);
        properties.put("password", DB_PASSWORD);
        properties.put("oracle.jdbc.J2EE13Compliant", "true");
        properties.put("javax.net.ssl.trustStore", KEY_STORE_FILE_PATH);
        properties.put("javax.net.ssl.trustStoreType", "JKS");
        properties.put("javax.net.ssl.trustStorePassword", KEY_STORE_PASS);
        final Connection connection = DriverManager.getConnection(connectionString,
 properties);
        // If no exception, that means handshake has passed, and an SSL connection can be
 opened
    }
}
```

Enforcing a DN Match with an SSL Connection

The Oracle parameter SSL_SERVER_DN_MATCH can be used to enforce that the distinguished name (DN) for the database server matches its service name. If you enforce the match verifications, then SSL ensures that the certificate is from the server. If you do not enforce the match verification, then SSL performs the check but allows the connection, regardless if there is a match. If you do not enforce the match, you allow the server to potentially fake its identify.

To enforce DN matching, add the DN match property and use the connection string specified below.

Add the property to the client connection to enforce DN matching:

```
properties.put("oracle.net.ssl_server_dn_match", "TRUE”);
```

Use the following connection string to enforce DN matching when using SSL:

```
final String connectionString = String.format(
    "jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCPS)(HOST=%s)(PORT=%d))" +
    "(CONNECT_DATA=(SID=%s))" +
    "(SECURITY = (SSL_SERVER_CERT_DN =
\"C=US,ST=Washington,L=Seattle,O=Amazon.com,OU=RDS,CN=%s\")))",
    DB_SERVER_NAME, SSL_PORT, DB_SID, DB_SERVER_NAME);
```

Related Topics

- Working with Option Groups (p. 153)
- Options for Oracle DB Instances (p. 993)

Oracle SQLT

Amazon RDS supports Oracle SQLTXPLAIN (SQLT) through the use of the SQLT option.

The Oracle `EXPLAIN PLAN` statement can determine the execution plan of a SQL statement. It can verify whether the Oracle optimizer chooses a certain execution plan, such as a nested loops join. It also helps you understand the optimizer's decisions, such as why it chose a nested loops join over a hash join. So `EXPLAIN PLAN` helps you understand the statement's performance.

SQLT is an Oracle utility that produces a report. The report includes object statistics, object metadata, optimizer-related initialization parameters, and other information that a database administrator can use to tune a SQL statement for optimal performance. SQLT produces an HTML report with hyperlinks to all of the sections in the report.

Unlike Automatic Workload Repository or Statspack reports, SQLT works on individual SQL statements. SQLT is a collection of SQL, PL/SQL, and SQL*Plus files that collect, store, and display performance data.

To download SQLT and access instructions for using it, log in to your My Oracle Support account, and open the following documents:

- To download SQLT: Document 215187.1
- For SQLT usage instructions: Document 1614107.1
- For frequently asked questions about SQLT: Document 1454160.1
- For information about reading SQLT output: Document 1456176.1

You can use SQLT with any edition of the following Oracle Database versions:

- Oracle 12c, 12.1.0.2
- Oracle 11g, 11.2.0.4

Amazon RDS does not support the following SQLT methods:

- `XPLORE`
- `XHUME`

Prerequisites for SQLT

The following are prerequisites for using SQLT:

- You must remove users and roles that are required by SQLT, if they exist.

  The SQLT option creates the following users and roles on a DB instance:

  - `SQLTXPLAIN` user
  - `SQLTXADMIN` user
  - `SQLT_USER_ROLE` role

  If your DB instance has any of these users or roles, log in to the DB instance using a SQL client, and drop them using the following statements:

```
DROP USER SQLTXPLAIN CASCADE;
DROP USER SQLTXADMIN CASCADE;
DROP ROLE SQLT_USER_ROLE CASCADE;
```

- You must remove tablespaces that are required by SQLT, if they exist.

  The SQLT option creates the following tablespaces on a DB instance:

  - `RDS_SQLT_TS`
  - `RDS_TEMP_SQLT_TS`

  If your DB instance has these tablespaces, log in to the DB instance using a SQL client, and drop them.

SQLT Option Settings

SQLT can work with licensed features that are provided by the Oracle Tuning Pack and the Oracle Diagnostics Pack. The Oracle Tuning Pack includes the SQL Tuning Advisor, and the Oracle Diagnostics Pack includes the Automatic Workload Repository. The SQLT settings enable or disable access to these features from SQLT.

Amazon RDS supports the following settings for the SQLT option.

Option Setting	Valid Values	Default Value	Description
`LICENSE_PACK`	`T,D,N`	`T`	The Oracle Management Packs that you want to access with SQLT. Enter one of the following values: • `T` indicates that you have a license for the Oracle Tuning Pack and the Oracle Diagnostics Pack, and you want to access the SQL Tuning Advisor and Automatic Workload Repository from SQLT. • `D` indicates that you have a license for the Oracle Diagnostics Pack, and you want to access the Automatic Workload Repository from SQLT. • `N` indicates that you don't have a license for the Oracle Tuning Pack and the Oracle Diagnostics Pack, or that you have a license for one or both of them, but you don't want SQLT to access them. **Note** Amazon RDS does not provide licenses for these Oracle Management Packs. If you indicate that you want to use a pack that is not included in your DB instance, you can use SQLT with the DB instance. However, SQLT can't access the pack, and the SQLT report doesn't include the data for the pack. For example, if you specify `T`, but the DB instance doesn't include the Oracle Tuning Pack, SQLT works on the DB instance, but the report it generates doesn't contain data related to the Oracle Tuning Pack.

Adding the SQLT Option

The following is the general process for adding the SQLT option to a DB instance:

1. Create a new option group, or copy or modify an existing option group.

2. Add the SQLT option to the option group.
3. Associate the option group with the DB instance.

After you add the SQLT option, as soon as the option group is active, SQLT is active.

To add the SQLT option to a DB instance

1. Determine the option group that you want to use. You can create a new option group or use an existing option group. If you want to use an existing option group, skip to the next step. Otherwise, create a custom DB option group with the following settings:

   a. For **Engine**, choose the Oracle edition that you want to use. The SQLT option is supported on all editions.
   b. For **Major Engine Version**, choose **11.2** or **12.1**.

   For more information, see Creating an Option Group (p. 154).
2. Add the **SQLT** option to the option group. For more information about adding options, see Adding an Option to an Option Group (p. 157).
3. Apply the option group to a new or existing DB instance:

   - For a new DB instance, you apply the option group when you launch the instance. For more information, see Creating a DB Instance Running the Oracle Database Engine (p. 949).
   - For an existing DB instance, you apply the option group by modifying the instance and attaching the new option group. For more information, see Modifying a DB Instance Running the Oracle Database Engine (p. 967).
4. (Optional) Verify the SQLT installation on each DB instance with the SQLT option.

   a. Use a SQL client to connect to the DB instance as the master user.

      For information about connecting to an Oracle DB instance using a SQL client, see Connecting to a DB Instance Running the Oracle Database Engine (p. 959).
   b. Run the following query:

```
SELECT sqltxplain.sqlt$a.get_param('tool_version') sqlt_version FROM DUAL;
```

      The query returns the current version of the SQLT option on Amazon RDS. `12.1.160429` is an example of a version of SQLT that is available on Amazon RDS.
5. Change the passwords of the users that are created by the SQLT option.

   a. Use a SQL client to connect to the DB instance as the master user.
   b. Run the following SQL statement to change the password for the `SQLTXADMIN` user:

```
ALTER USER SQLTXADMIN IDENTIFIED BY new_password ACCOUNT UNLOCK;
```

   c. Run the following SQL statement to change the password for the `SQLTXPLAIN` user:

```
ALTER USER SQLTXPLAIN IDENTIFIED BY new_password ACCOUNT UNLOCK;
```

Note
Upgrading SQLT requires uninstalling an older version of SQLT and then installing the new version. So, all SQLT metadata can be lost when you upgrade SQLT. A major version upgrade of a database also uninstalls and re-installs SQLT. An example of a major version upgrade is an upgrade from Oracle 11g to Oracle 12c.

Using SQLT

SQLT works with the Oracle SQL*Plus utility.

To use SQLT

1. Download the SQLT .zip file from Document 215187.1 on the My Oracle Support site.
2. Unzip the SQLT .zip file.
3. From a command prompt, change to the `sqlt/run` directory on your file system.
4. From the command prompt, open SQL*Plus, and connect to the DB instance as the master user.

   For information about connecting to a DB instance using SQL*Plus, see Connecting to a DB Instance Running the Oracle Database Engine (p. 959).
5. Get the SQL ID of a SQL statement:

   ```
   SELECT SQL_ID FROM V$SQL WHERE SQL_TEXT='sql_statement';
   ```

   Your output is similar to the following:

   ```
   SQL_ID
   -------------
   chvsmttqjzjkn
   ```
6. Analyze a SQL statement with SQLT:

   ```
   START sqltxtract.sql sql_id sqltxplain_user_password
   ```

   For example, for the SQL ID `chvsmttqjzjkn`, enter the following:

   ```
   START sqltxtract.sql chvsmttqjzjkn sqltxplain_user_password
   ```

   SQLT generates the HTML report and related resources as a .zip file in the directory from which the SQLT command was run.
7. (Optional) To enable application users to diagnose SQL statements with SQLT, grant `SQLT_USER_ROLE` to each application user with the following statement:

   ```
   GRANT ROLE SQLT_USER_ROLE TO application_user_name;
   ```

Note
Oracle does not recommend running SQLT with the `SYS` user or with users that have the `DBA` role. It is a best practice to run SQLT diagnostics using the application user's account, by granting `SQLT_USER_ROLE` to the application user.

Modifying SQLT Settings

After you enable SQLT, you can modify the `LICENSE_PACK` setting for the option.

For more information about how to modify option settings, see Modifying an Option Setting (p. 163). For more information about each setting, see SQLT Option Settings (p. 1026).

Removing the SQLT Option

You can remove SQLT from a DB instance.

To remove SQLT from a DB instance, do one of the following:

- To remove SQLT from multiple DB instances, remove the SQLT option from the option group to which the DB instances belong. This change affects all DB instances that use the option group. For more information, see Removing an Option from an Option Group (p. 167).
- To remove SQLT from a single DB instance, modify the DB instance and specify a different option group that doesn't include the SQLT option. You can specify the default (empty) option group or a different custom option group. For more information, see Modifying a DB Instance Running the Oracle Database Engine (p. 967).

Related Topics

- Working with Option Groups (p. 153)
- Options for Oracle DB Instances (p. 993)

Oracle Statspack

The Oracle Statspack option installs and enables the Oracle Statspack performance statistics feature. Oracle Statspack is a collection of SQL, PL/SQL, and SQL*Plus scripts that collect, store, and display performance data. For information about using Oracle Statspack, see Oracle Statspack in the Oracle documentation.

Note
Oracle Statspack is no longer supported by Oracle and has been replaced by the more advanced Automatic Workload Repository (AWR). AWR is available only for Oracle Enterprise Edition customers who have purchased the Diagnostics Pack. Oracle Statspack can be used with any Oracle DB engine on Amazon RDS.

The following steps show you how to work with Oracle Statspack on Amazon RDS:

1. If you have an existing DB instance that has the PERFSTAT account already created and you want to use Oracle Statspack with it, you must drop the PERFSTAT account before adding the Statspack option to the option group associated with your DB instance. If you attempt to add the Statspack option to an option group associated with a DB instance that already has the PERFSTAT account created, you get an error and the RDS event RDS-Event-0058 is generated.

If you have already installed Statspack, and the PERFSTAT account is associated with Statspack, then skip this step, and do not drop the PERFSTAT user.

You can drop the PERFSTAT account by running the following command:

```
DROP USER perfstat CASCADE;
```

2. Add the Statspack option to an option group and then associate that option group with your DB instance. Amazon RDS installs the Statspack scripts on the DB instance and then sets up the PERFSTAT user account, the account you use to run the Statspack scripts. If you have installed Statspack, skip this step.
3. After Amazon RDS has installed Statspack on your DB instance, you must log in to the DB instance using your master user name and master password. You must then reset the PERFSTAT password from the randomly generated value Amazon RDS created when Statspack was installed. After you have reset the PERFSTAT password, you can log in using the PERFSTAT user account and run the Statspack scripts.

   Use the following command to reset the password:

   ```
   ALTER USER perfstat IDENTIFIED BY <new_password> ACCOUNT UNLOCK;
   ```

4. After you have logged on using the PERFSTAT account, you can either manually create a Statspack snapshot or create a job that will take a Statspack snapshot after a given time interval. For example, the following job creates a Statspack snapshot every hour:

   ```
   variable jn number;
   execute dbms_job.submit(:jn, 'statspack.snap;',sysdate,'trunc(SYSDATE+1/24,''HH24'')');
   commit;
   ```

5. Once you have created at least two Statspack snapshots, you can view them using the following query:

   ```
   select snap_id, snap_time from stats$snapshot order by 1;
   ```

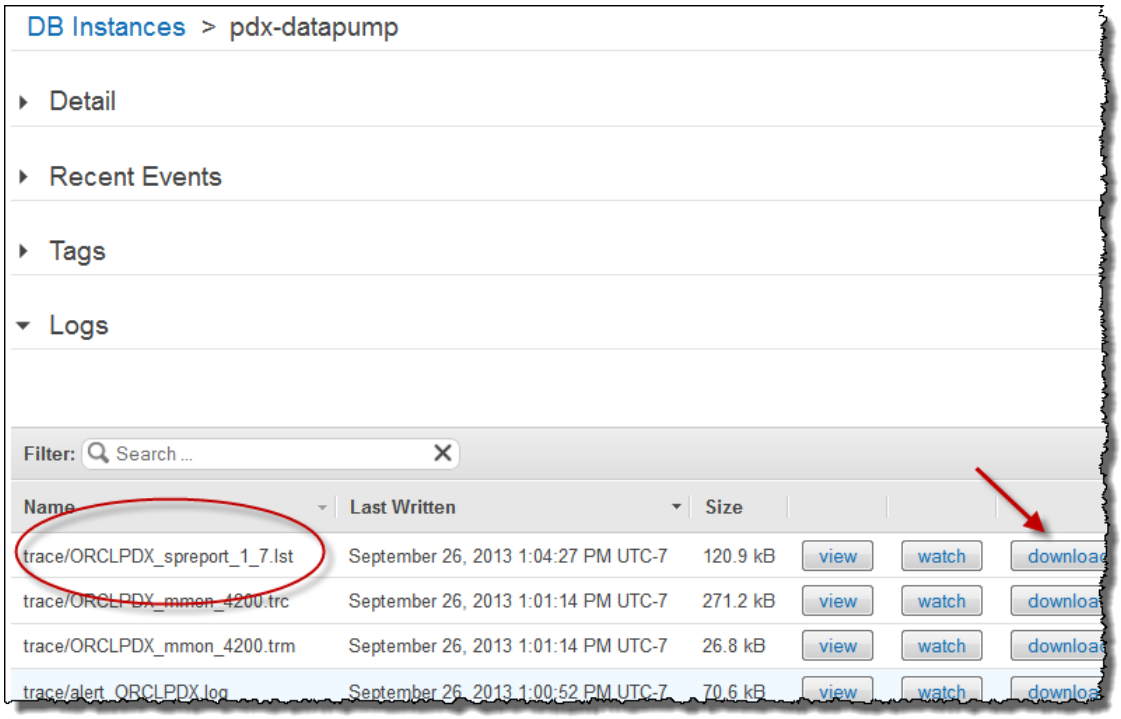
6. To create a Statspack report, you choose two snapshots to analyze and run the following Amazon RDS command:

   ```
   exec RDSADMIN.RDS_RUN_SPREPORT(<begin snap>,<end snap>);
   ```

   For example, the following Amazon RDS command would create a report based on the interval between Statspack snapshots 1 and 7:

   ```
   exec RDSADMIN.RDS_RUN_SPREPORT(1,7);
   ```

The file name of the Statspack report that is generated includes the number of the two Statspack snapshots used. For example, a report file created using Statspack snapshots 1 and 7 would be named ORCL_spreport_1_7.lst. You can download the Statspack report by selecting the report in the Log section of the RDS console and clicking **Download** or you can use the trace file procedures explained in Working with Oracle Trace Files (p. 318).



If an error occurs when producing the report, an error file is created using the same naming conventions but with an extension of .err. For example, if an error occurred while creating a report using Statspack snapshots 1 and 7, the report file would be named ORCL_spreport_1_7.err. You can download the error report by selecting the report in the Log section of the RDS console and clicking **Download** or use the trace file procedures explained in Working with Oracle Trace Files (p. 318).

Oracle Statspack does some basic checking before running the report, so you could also see error messages displayed at the command prompt. For example, if you attempt to generate a report based on an invalid range, such as the beginning Statspack snapshot value is larger than the ending Statspack snapshot value, the error message is displayed at the command prompt and no error file is created.

```
exec RDSADMIN.RDS_RUN_SPREPORT(2,1);
*
ERROR at line 1:
ORA-20000: Invalid snapshot IDs. Find valid ones in perfstat.stats$snapshot.
```

If you use an invalid number for one of the Statspack snapshots, the error message will also be displayed at the command prompt. For example, if you have 20 Statspack snapshots but request that a report be run using Statspack snapshots 1 and 50, the command prompt will display an error.

```
exec RDSADMIN.RDS_RUN_SPREPORT(1,50);
*
ERROR at line 1:
ORA-20000: Could not find both snapshot IDs
```

For more information about how to use Oracle Statspack, including information on adjusting the amount of data captured by adjusting the snapshot level, go to the Oracle Statspack documentation page.

To remove Oracle Statspack files, use the following command:

```
execute statspack.purge(<begin snap>, <end snap>);
```

Related Topics

- Working with Option Groups (p. 153)
- Options for Oracle DB Instances (p. 993)

Oracle Time Zone

You can use the time zone option to change the system time zone used by your Oracle DB instance. For example, you might change the time zone of a DB instance to be compatible with an on-premises environment, or a legacy application. The time zone option changes the time zone at the host level. Changing the time zone impacts all date columns and values, including `SYSDATE` and `SYSTIMESTAMP`.

The time zone option differs from the `rdsadmin_util.alter_db_time_zone` command. The `alter_db_time_zone` command changes the time zone only for certain data types. The time zone option changes the time zone for all date columns and values. For more information about `alter_db_time_zone`, see Setting the Database Time Zone (p. 1056).

Prerequisites for Time Zone

The time zone option is a permanent and persistent option. You can't remove the option from an option group after you add it. You can't remove the option group from a DB instance after you add it. You can't modify the time zone setting of the option to a different time zone.

We strongly urge you to take a DB snapshot of your DB instance before adding the time zone option to a DB instance. By using a snapshot you can recover the DB instance if you set the time zone option incorrectly. For more information, see Creating a DB Snapshot (p. 207).

We strongly urge you to test the time zone option on a test DB instance before you add it to a production DB instance. Adding the time zone option can cause problems with tables that use system date to add dates or times. You should analyze your data and applications to determine the impact of changing the time zone.

Time Zone Option Settings

Amazon RDS supports the following settings for the time zone option.

Option Setting	Valid Values	Description
Time Zone	One of the available time zones. For the full list, see Available Time Zones (p. 1035).	The new time zone for your DB instance.

Adding the Time Zone Option

The general process for adding the time zone option to a DB instance is the following:

1. Create a new option group, or copy or modify an existing option group.
2. Add the option to the option group.
3. Associate the option group with the DB instance.

When you add the time zone option, a brief outage occurs while your DB instance is automatically restarted.

AWS Management Console

To add the time zone option to a DB instance

1. Determine the option group you want to use. You can create a new option group or use an existing option group. If you want to use an existing option group, skip to the next step. Otherwise, create a custom DB option group with the following settings:

a. For **Engine** choose the oracle edition for your DB instance.
b. For **Major Engine Version** choose **11.2** or **12.1** for your DB instance.

For more information, see Creating an Option Group (p. 154).

2. Add the **Timezone** option to the option group, and configure the option settings.

   Important
   If you add the time zone option to an existing option group that is already attached to one or more DB instances, a brief outage occurs while all the DB instances are automatically restarted.

   For more information about adding options, see Adding an Option to an Option Group (p. 157). For more information about each setting, see Time Zone Option Settings (p. 1033).

3. Apply the option group to a new or existing DB instance:

   - For a new DB instance, you apply the option group when you launch the instance. For more information, see Creating a DB Instance Running the Oracle Database Engine (p. 949).
   - For an existing DB instance, you apply the option group by modifying the instance and attaching the new option group. When you add the time zone option to an existing DB instance, a brief outage occurs while your DB instance is automatically restarted. For more information, see Modifying a DB Instance Running the Oracle Database Engine (p. 967).

CLI

The following example uses the AWS CLI add-option-to-option-group command to add the `Timezone` option and the `TIME_ZONE` option setting to an option group called `myoptiongroup`. The time zone is set to `Africa/Cairo`.

For Linux, OS X, or Unix:

```
aws rds add-option-to-option-group \
    --option-group-name "myoptiongroup" \
    --options "OptionName=Timezone,OptionSettings=[{Name=TIME_ZONE,Value=Africa/Cairo}]" \
    --apply-immediately
```

For Windows:

```
aws rds add-option-to-option-group ^
    --option-group-name "myoptiongroup" ^
    --options "OptionName=Timezone,OptionSettings=[{Name=TIME_ZONE,Value=Africa/Cairo}]" ^
    --apply-immediately
```

Modifying Time Zone Settings

The time zone option is a permanent and persistent option. You can't remove the option from an option group after you add it. You can't remove the option group from a DB instance after you add it. You can't modify the time zone setting of the option to a different time zone. If you set the time zone incorrectly, restore a snapshot of your DB instance from before you added the time zone option.

Removing the Time Zone Option

The time zone option is a permanent and persistent option. You can't remove the option from an option group after you add it. You can't remove the option group from a DB instance after you add it. To remove

the time zone option, restore a snapshot of your DB instance from before you added the time zone option.

Available Time Zones

The following values can be used for the time zone option.

Zone	Time Zone
Africa	Africa/Cairo, Africa/Casablanca, Africa/Harare, Africa/Lagos, Africa/Luanda, Africa/Monrovia, Africa/Nairobi, Africa/Tripoli, Africa/Windhoek
America	America/Araguaina, America/Argentina/Buenos_Aires, America/Asuncion, America/Bogota, America/Caracas, America/Chicago, America/Chihuahua, America/Cuiaba, America/Denver, America/Detroit, America/Fortaleza, America/Godthab, America/Guatemala, America/Halifax, America/Lima, America/Los_Angeles, America/Manaus, America/Matamoros, America/Mexico_City, America/Monterrey, America/Montevideo, America/New_York, America/Phoenix, America/Santiago, America/Sao_Paulo, America/Tijuana, America/Toronto
Asia	Asia/Amman, Asia/Ashgabat, Asia/Baghdad, Asia/Baku, Asia/Bangkok, Asia/Beirut, Asia/Calcutta, Asia/Damascus, Asia/Dhaka, Asia/Hong_Kong, Asia/Irkutsk, Asia/Jakarta, Asia/Jerusalem, Asia/Kabul, Asia/Karachi, Asia/Kathmandu, Asia/Kolkata, Asia/Krasnoyarsk, Asia/Magadan, Asia/Manila, Asia/Muscat, Asia/Novosibirsk, Asia/Rangoon, Asia/Riyadh, Asia/Seoul, Asia/Shanghai, Asia/Singapore, Asia/Taipei, Asia/Tehran, Asia/Tokyo, Asia/Ulaanbaatar, Asia/Vladivostok, Asia/Yakutsk, Asia/Yerevan
Atlantic	Atlantic/Azores, Atlantic/Cape_Verde
Australia	Australia/Adelaide, Australia/Brisbane, Australia/Darwin, Australia/Eucla, Australia/Hobart, Australia/Lord_Howe, Australia/Perth, Australia/Sydney
Brazil	Brazil/DeNoronha, Brazil/East
Canada	Canada/Newfoundland, Canada/Saskatchewan
Etc	Etc/GMT-3
Europe	Europe/Amsterdam, Europe/Athens, Europe/Berlin, Europe/Dublin, Europe/Helsinki, Europe/Kaliningrad, Europe/London, Europe/Madrid, Europe/Moscow, Europe/Paris, Europe/Prague, Europe/Rome, Europe/Sarajevo
Pacific	Pacific/Apia, Pacific/Auckland, Pacific/Chatham, Pacific/Fiji, Pacific/Guam, Pacific/Honolulu, Pacific/Kiritimati, Pacific/Marquesas, Pacific/Samoa, Pacific/Tongatapu, Pacific/Wake
US	US/Alaska, US/Central, US/East-Indiana, US/Eastern, US/Pacific
UTC	UTC

Related Topics

- Working with Option Groups (p. 153)
- Options for Oracle DB Instances (p. 993)

Oracle Transparent Data Encryption

Amazon RDS supports Oracle Transparent Data Encryption (TDE), a feature of the Oracle Advanced Security option available in Oracle Enterprise Edition. This feature automatically encrypts data before it is written to storage and automatically decrypts data when the data is read from storage.

Oracle Transparent Data Encryption is used in scenarios where you need to encrypt sensitive data in case data files and backups are obtained by a third party or when you need to address security-related regulatory compliance issues.

> **Note**
> You can use the TDE option or AWS CloudHSM Classic, but not both. For more information, see Using AWS CloudHSM Classic to Store Amazon RDS Oracle TDE Keys (p. 1086).

The TDE option is a permanent option that cannot be removed from an option group, and that option group cannot be removed from a DB instance once it is associated with a DB instance. You cannot disable TDE from a DB instance once that instance is associated with an option group with the Oracle TDE option.

A detailed explanation about Oracle Transparent Data Encryption is beyond the scope of this guide. For information about using Oracle Transparent Data Encryption, see Securing Stored Data Using Transparent Data Encryption. For more information about Oracle Advanced Security, see Oracle Advanced Security in the Oracle documentation. For more information on AWS security, see the AWS Security Center.

TDE Encryption Modes

Oracle Transparent Data Encryption supports two encryption modes: TDE tablespace encryption and TDE column encryption. TDE tablespace encryption is used to encrypt entire application tables. TDE column encryption is used to encrypt individual data elements that contain sensitive data. You can also apply a hybrid encryption solution that uses both TDE tablespace and column encryption.

> **Note**
> Amazon RDS manages the Oracle Wallet and TDE master key for the DB instance. You do not need to set the encryption key using the command `ALTER SYSTEM set encryption key`.

For information about TDE best practices, see Oracle Advanced Security Transparent Data Encryption Best Practices.

Once the option is enabled, you can check the status of the Oracle Wallet by using the following command:

```
SELECT * FROM v$encryption_wallet;
```

To create an encrypted tablespace, use the following command:

```
CREATE TABLESPACE encrypt_ts ENCRYPTION DEFAULT STORAGE (ENCRYPT);
```

To specify the encryption algorithm, use the following command:

```
CREATE TABLESPACE encrypt_ts ENCRYPTION USING 'AES256' DEFAULT STORAGE (ENCRYPT);
```

Note that the previous commands for encrypting a tablespace are the same as the commands you would use with an Oracle installation not on Amazon RDS, and the ALTER TABLE syntax to encrypt a column is also the same as the commands you would use for an Oracle installation not on Amazon RDS.

You should determine if your DB instance is associated with an option group that has the **TDE** option. To view the option group that a DB instance is associated with, you can use the RDS console, the describe-db-instance AWS CLI command, or the API action DescribeDBInstances.

To comply with several security standards, Amazon RDS is working to implement automatic periodic master key rotation.

Adding the TDE Option

The process for using Oracle Transparent Data Encryption (TDE) with Amazon RDS is as follows:

1. If the DB instance is not associated with an option group that has the **TDE** option enabled, you must either create an option group and add the **TDE** option or modify the associated option group to add the **TDE** option. For information about creating or modifying an option group, see Working with Option Groups (p. 153). For information about adding an option to an option group, see Adding an Option to an Option Group (p. 157).
2. Associate the DB instance with the option group with the **TDE** option. For information about associating a DB instance with an option group, see Modifying a DB Instance Running the Oracle Database Engine (p. 967).

Removing the TDE Option

If you no longer want to use the TDE option with a DB instance, you must decrypt all your data on the DB instance, copy the data to a new DB instance that is not associated with an option group with TDE enabled, and then delete the original instance. You can rename the new instance to be the same name as the previous DB instance if you prefer.

Using TDE with Data Pump

You can use Oracle Data Pump to import or export encrypted dump files. Amazon RDS supports the password encryption mode (ENCRYPTION_MODE=PASSWORD) for Oracle Data Pump. Amazon RDS does not support transparent encryption mode (ENCRYPTION_MODE=TRANSPARENT) for Oracle Data Pump. For more information about using Oracle Data Pump with Amazon RDS, see Oracle Data Pump (p. 983).

Related Topics

- Working with Option Groups (p. 153)
- Options for Oracle DB Instances (p. 993)

Oracle UTL_MAIL

Amazon RDS supports Oracle UTL_MAIL through the use of the UTL_MAIL option and SMTP servers. You can send email directly from your database by using the UTL_MAIL package. Amazon RDS supports UTL_MAIL for the following versions of Oracle:

- Oracle version 12.1.0.2.v5 and later
- Oracle version 11.2.0.4.v9 and later

The following are some limitations to using UTL_MAIL:

- UTL_MAIL does not support Transport Layer Security (TLS) and therefore emails are not encrypted.
- UTL_MAIL does not support authentication with SMTP servers.
- You can only send a single attachment in an email.
- You can't send attachments larger than 32 K.
- You can only use ASCII and Extended Binary Coded Decimal Interchange Code (EBCDIC) character encodings.
- SMTP port (25) is throttled based on the elastic network interface owner's policies.

When you enable UTL_MAIL, only the master user for your DB instance is granted the execute privilege. If necessary, the master user can grant the execute privilege to other users so that they can use UTL_MAIL.

> **Important**
> We recommend that you enable Oracle's built-in auditing feature to track the use of UTL_MAIL procedures.

Prerequisites for Oracle UTL_MAIL

The following are prerequisites for using Oracle UTL_MAIL:

- One or more SMTP servers, and the corresponding IP addresses or public or private Domain Name Server (DNS) names. For more information about private DNS names resolved through a custom DNS server, see Setting Up a Custom DNS Server (p. 1053).
- For Oracle versions prior to 12c, your DB instance must also use the XML DB option. For more information, see Oracle XML DB (p. 1040).

Adding the Oracle UTL_MAIL Option

The general process for adding the Oracle UTL_MAIL option to a DB instance is the following:

1. Create a new option group, or copy or modify an existing option group.
2. Add the option to the option group.
3. Associate the option group with the DB instance.

After you add the UTL_MAIL option, as soon as the option group is active, UTL_MAIL is active.

To add the UTL_MAIL option to a DB instance

1. Determine the option group you want to use. You can create a new option group or use an existing option group. If you want to use an existing option group, skip to the next step. Otherwise, create a custom DB option group with the following settings:

   a. For **Engine**, choose the edition of Oracle you want to use.

b. For **Major Engine Version**, choose **11.2** or **12.1**.

For more information, see Creating an Option Group (p. 154).

2. Add the **UTL_MAIL** option to the option group. For more information about adding options, see Adding an Option to an Option Group (p. 157).
3. Apply the option group to a new or existing DB instance:

   - For a new DB instance, you apply the option group when you launch the instance. For more information, see Creating a DB Instance Running the Oracle Database Engine (p. 949).
   - For an existing DB instance, you apply the option group by modifying the instance and attaching the new option group. For more information, see Modifying a DB Instance Running the Oracle Database Engine (p. 967).

Using Oracle UTL_MAIL

After you enable the UTL_MAIL option, you must configure the SMTP server before you can begin using it.

You configure the SMTP server by setting the SMTP_OUT_SERVER parameter to a valid IP address or public DNS name. For the SMTP_OUT_SERVER parameter, you can specify a comma-separated list of the addresses of multiple servers. If the first server is unavailable, UTL_MAIL tries the next server, and so on.

You can set the default SMTP_OUT_SERVER for a DB instance by using a DB parameter group. You can set the SMTP_OUT_SERVER parameter for a session by running the following code on your database on your DB instance.

```
ALTER SESSION SET smtp_out_server = mailserver.domain.com:25;
```

After the UTL_MAIL option is enabled, and your SMTP_OUT_SERVER is configured, you can send mail by using the `SEND` procedure. For more information, see UTL_MAIL in the Oracle documentation.

Removing the Oracle UTL_MAIL Option

You can remove Oracle UTL_MAIL from a DB instance.

To remove UTL_MAIL from a DB instance, do one of the following:

- To remove UTL_MAIL from multiple DB instances, remove the UTL_MAIL option from the option group they belong to. This change affects all DB instances that use the option group. For more information, see Removing an Option from an Option Group (p. 167).
- To remove UTL_MAIL from a single DB instance, modify the DB instance and specify a different option group that doesn't include the UTL_MAIL option. You can specify the default (empty) option group, or a different custom option group. For more information, see Modifying a DB Instance Running the Oracle Database Engine (p. 967).

Troubleshooting

The following are issues you might encounter when you use UTL_MAIL with Amazon RDS.

- Throttling. SMTP port (25) is throttled based on the elastic network interface owner's policies. If you can successfully send email by using UTL_MAIL, and you see the error `ORA-29278: SMTP transient error: 421 Service not available`, you are possibly being throttled. If you experience throttling with email delivery, we recommend that you implement a backoff algorithm. For more

information about backoff algorithms, see Error Retries and Exponential Backoff in AWS and How to handle a "Throttling – Maximum sending rate exceeded" error.

You can request that this throttle be removed. For more information, see How do I remove the throttle on port 25 from my EC2 instance?.

Related Topics

- Working with Option Groups (p. 153)
- Options for Oracle DB Instances (p. 993)

Oracle XML DB

Oracle XML DB adds native XML support to your DB instance. With XML DB, you can store and retrieve structured or unstructured XML, in addition to relational data.

XML DB is pre-installed on Oracle version 12c and later. Amazon RDS supports Oracle XML DB for version 11g through the use of the XMLDB option. After you apply the XMLDB option to your DB instance, you have full access to the Oracle XML DB repository; no post-installation tasks are required.

Note
The Amazon RDS XMLDB option does not provide support for the Oracle XML DB Protocol Server.

Adding the Oracle XML DB Option

The general process for adding the Oracle XML DB option to a DB instance is the following:

1. Create a new option group, or copy or modify an existing option group.
2. Add the option to the option group.
3. Associate the option group with the DB instance.

After you add the XML DB option, as soon as the option group is active, XML DB is active.

To add the XML DB option to a DB instance

1. Determine the option group you want to use. You can create a new option group or use an existing option group. If you want to use an existing option group, skip to the next step. Otherwise, create a custom DB option group with the following settings:

   a. For **Engine**, choose the edition of Oracle you want to use.
   b. For **Major Engine Version**, choose **11.2**.

   For more information, see Creating an Option Group (p. 154).
2. Add the **XMLDB** option to the option group. For more information about adding options, see Adding an Option to an Option Group (p. 157).
3. Apply the option group to a new or existing DB instance:

   - For a new DB instance, you apply the option group when you launch the instance. For more information, see Creating a DB Instance Running the Oracle Database Engine (p. 949).
   - For an existing DB instance, you apply the option group by modifying the instance and attaching the new option group. For more information, see Modifying a DB Instance Running the Oracle Database Engine (p. 967).

Removing the Oracle XML DB Option

You can remove the XML DB option from a DB instance running version 11g.

To remove the XML DB option from a DB instance running version 11g, do one of the following:

- To remove the XMLDB option from multiple DB instances, remove the XMLDB option from the option group they belong to. This change affects all DB instances that use the option group. For more information, see Removing an Option from an Option Group (p. 167).
- To remove the XMLDB option from a single DB instance, modify the DB instance and specify a different option group that doesn't include the XMLDB option. You can specify the default (empty) option group, or a different custom option group. For more information, see Modifying a DB Instance Running the Oracle Database Engine (p. 967).

Related Topics

- Working with Option Groups (p. 153)
- Options for Oracle DB Instances (p. 993)

Common DBA Tasks for Oracle DB Instances

This section describes the Amazon RDS-specific implementations of some common DBA tasks for DB instances running the Oracle database engine. To deliver a managed service experience, Amazon RDS doesn't provide shell access to DB instances, and restricts access to certain system procedures and tables that require advanced privileges.

The following are common DBA tasks for DB instances running Oracle:

	Amazon RDS method: `create_passthrough_verify_fcn`
Setting Up a Custom DNS Server (p. 1053)	—

- Database Tasks (p. 1054)

Changing the Global Name of a Database (p. 1054)	Amazon RDS method: `rename_global_name` Oracle method: `alter database rename`
Creating and Sizing Tablespaces (p. 1054)	Amazon RDS method: `create tablespace` Oracle method: `alter database`
Setting the Default Tablespace (p. 1055)	Amazon RDS method: `alter_default_tablespace` Oracle method: `alter database default tablespace`
Setting the Default Temporary Tablespace (p. 1055)	Amazon RDS method: `alter_default_temp_tablespace` Oracle method: `alter database default temporary tablespace`
Checkpointing the Database (p. 1055)	Amazon RDS method: `checkpoint` Oracle method: `alter system checkpoint`
Setting Distributed Recovery (p. 1056)	Amazon RDS method: `enable_distr_recovery` Oracle method: `alter system enable distributed recovery`
Setting the Database Time Zone (p. 1056)	Amazon RDS method: `alter_db_time_zone` Oracle method: `alter database set time_zone`
Working with Automatic Workload Repository (AWR) (p. 1058)	—
Adjusting Database Links for Use with DB Instances in a VPC (p. 1058)	—

- Log Tasks (p. 1065)

Setting Force Logging (p. 1065)	Amazon RDS method: `force_logging` Oracle method: `alter database force logging`
Setting Supplemental Logging (p. 1066)	Amazon RDS method: `alter_supplemental_logging` Oracle method: `alter database add supplemental log`
Switching Online Log Files (p. 1067)	Amazon RDS method: `switch_logfile` Oracle method: `alter system switch logfile`
Adding Online Redo Logs (p. 1067)	Amazon RDS method: `add_logfile`
Dropping Online Redo Logs (p. 1067)	Amazon RDS method: `drop_logfile`
Resizing Online Redo Logs (p. 1068)	—
Retaining Archived Redo Logs (p. 1070)	Amazon RDS method: `set_configuration`
Accessing Transaction Logs (p. 1071)	Amazon RDS method: `create_archivelog_dir` Amazon RDS method: `create_onlinelog_dir`

- Miscellaneous Tasks (p. 1072)

Creating New Directories in the Main Data Storage Space (p. 1072)	Amazon RDS method: `create_directory` Oracle method: `create directory`
Listing Files in a DB Instance Directory (p. 1072)	Amazon RDS method: `listdir` Oracle method: —
Reading Files in a DB Instance Directory (p. 1073)	Amazon RDS method: `read_text_file` Oracle method: —

Common DBA System Tasks for Oracle DB Instances

This section describes how you can perform common DBA tasks related to the system on your Amazon RDS DB instances running Oracle. To deliver a managed service experience, Amazon RDS doesn't provide shell access to DB instances, and restricts access to certain system procedures and tables that require advanced privileges.

Disconnecting a Session

You can use the Amazon RDS procedure `rdsadmin.rdsadmin_util.disconnect` to disconnect the current session by ending the dedicated server process. The `disconnect` procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
`sid`	number	—	required	The session identifier.
`serial`	number	—	required	The serial number of the session.
`method`	varchar	'IMMEDIATE'	optional	Valid values are `'IMMEDIATE'` or `'POST_TRANSACTION'`.

The following example disconnects a session:

```
begin
    rdsadmin.rdsadmin_util.disconnect(
        sid    => sid,
        serial => serial_number);
end;
/
```

To get the session identifier and the session serial number, query the `V$SESSION` view. The following example gets all sessions for the user `AWSUSER`:

```
select SID, SERIAL#, STATUS from V$SESSION where USERNAME = 'AWSUSER';
```

The database must be open to use this method. For more information about disconnecting a session, see ALTER SYSTEM in the Oracle documentation.

Killing a Session

You can use the Amazon RDS procedure `rdsadmin.rdsadmin_util.kill` to kill a session. The `kill` procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
`sid`	number	—	required	The session identifier.
`serial`	number	—	required	The serial number of the session.

Parameter Name	Data Type	Default	Required	Description
method	varchar	null	optional	Valid values are 'IMMEDIATE' or 'PROCESS'.

The following example kills a session:

```
begin
    rdsadmin.rdsadmin_util.kill(
        sid    => sid,
        serial => serial_number);
end;
/
```

To get the session identifier and the session serial number, query the V$SESSION view. The following example gets all sessions for the user AWSUSER:

```
select SID, SERIAL#, STATUS from V$SESSION where USERNAME = 'AWSUSER';
```

You can specify either IMMEDIATE or PROCESS as a value for the method parameter. Specifying PROCESS as the enables you to kill the processes associated with a session. You should only do this if killing the session using IMMEDIATE as the method value was unsuccessful.

Enabling and Disabling Restricted Sessions

You can use the Amazon RDS procedure rdsadmin.rdsadmin_util.restricted_session to enable and disable restricted sessions. The restricted_session procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
p_enable	boolean	true	optional	Set to true to enable restricted sessions, false to disable restricted sessions.

The following example shows how to enable and disable restricted sessions.

```
/* Verify that the database is currently unrestricted. */

select LOGINS from V$INSTANCE;

LOGINS
-------
ALLOWED


/* Enable restricted sessions */

exec rdsadmin.rdsadmin_util.restricted_session(p_enable => true);


/* Verify that the database is now restricted. */

select LOGINS from V$INSTANCE;
```

```
LOGINS
----------
RESTRICTED


/* Disable restricted sessions */

exec rdsadmin.rdsadmin_util.restricted_session(p_enable => false);


/* Verify that the database is now unrestricted again. */

select LOGINS from V$INSTANCE;

LOGINS
-------
ALLOWED
```

Flushing the Shared Pool

You can use the Amazon RDS procedure `rdsadmin.rdsadmin_util.flush_shared_pool` to flush the shared pool. The `flush_shared_pool` procedure has no parameters.

The following example flushes the shared pool.

```
exec rdsadmin.rdsadmin_util.flush_shared_pool;
```

Flushing the Buffer Cache

You can use the Amazon RDS procedure `rdsadmin.rdsadmin_util.flush_buffer_cache` flush the buffer cache. The `flush_buffer_cache` procedure has no parameters.

The following example flushes the buffer cache.

```
exec rdsadmin.rdsadmin_util.flush_buffer_cache;
```

Granting SELECT or EXECUTE Privileges to SYS Objects

Usually you transfer privileges by using roles, which can contain many objects.
You can grant privileges to a single object by using the Amazon RDS procedure `rdsadmin.rdsadmin_util.grant_sys_object`. The procedure only grants privileges that the master account already has via a role or direct grant.

The `grant_sys_object` procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
`p_obj_name`	varchar2	—	required	The name of the object to grant privileges for. The object can be a directory, function, package, procedure, sequence, table, or view. Object names must be spelled exactly as they appear in `DBA_OBJECTS`. Most system objects are defined in upper case, so

Parameter Name	Data Type	Default	Required	Description
				we recommend you try that first.
p_grantee	varchar2	—	required	The name of the object to grant privileges to. The object can be a schema or a role.
p_privilege	varchar2	null	required	—
p_grant_option	boolean	false	optional	Set to true to use the with grant option. The p_grant_option parameter is supported for Oracle versions 11.2.0.4.v8 and later, and 12.1.0.2.v4 and later.

The following example grants select privileges on an object named V_$SESSION to a user named USER1:

```
begin
    rdsadmin.rdsadmin_util.grant_sys_object(
        p_obj_name  => 'V_$SESSION',
        p_grantee   => 'USER1',
        p_privilege => 'SELECT');
end;
/
```

The following example grants select privileges on an object named V_$SESSION to a user named USER1 with the grant option:

```
begin
    rdsadmin.rdsadmin_util.grant_sys_object(
        p_obj_name     => 'V_$SESSION',
        p_grantee      => 'USER1',
        p_privilege    => 'SELECT',
        p_grant_option => true);
end;
/
```

To be able to grant privileges on an object, your account must have those privileges granted to it directly with the grant option, or via a role granted using with admin option. In the most common case, you may want to grant SELECT on a DBA view that has been granted to the SELECT_CATALOG_ROLE role. If that role isn't already directly granted to your user using with admin option, then you won't be able to transfer the privilege. If you have the DBA privilege, then you can grant the role directly to another user.

The following example grants the SELECT_CATALOG_ROLE and EXECUTE_CATALOG_ROLE to USER1. Since the with admin option is used, USER1 can now grant access to SYS objects that have been granted to SELECT_CATALOG_ROLE.

```
grant SELECT_CATALOG_ROLE to USER1 with admin option;
grant EXECUTE_CATALOG_ROLE to USER1 with admin option;
```

Objects already granted to PUBLIC do not need to be re-granted. If you use the grant_sys_object procedure to re-grant access, the procedure call succeeds.

Granting Privileges to Non-Master Users

You can grant select privileges for many objects in the `SYS` schema by using the `SELECT_CATALOG_ROLE` role. The `SELECT_CATALOG_ROLE` role gives users `SELECT` privileges on data dictionary views. The following example grants the role `SELECT_CATALOG_ROLE` to a user named `user1`.

```
grant SELECT_CATALOG_ROLE to user1;
```

You can grant execute privileges for many objects in the `SYS` schema by using the `EXECUTE_CATALOG_ROLE` role. The `EXECUTE_CATALOG_ROLE` role gives users `EXECUTE` privileges for packages and procedures in the data dictionary. The following example grants the role `EXECUTE_CATALOG_ROLE` to a user named *user1*:

```
grant EXECUTE_CATALOG_ROLE to user1;
```

The following example gets the permissions that the roles `SELECT_CATALOG_ROLE` and `EXECUTE_CATALOG_ROLE` allow:

```
  select *
    from ROLE_TAB_PRIVS
   where ROLE in ('SELECT_CATALOG_ROLE','EXECUTE_CATALOG_ROLE')
order by ROLE, TABLE_NAME asc;
```

The following example creates a non-master user named `user1`, grants the `CREATE SESSION` privilege, and grants the `SELECT` privilege on a database named *sh.sales*:

```
create user user1 identified by password;
grant CREATE SESSION to user1;
grant SELECT on sh.sales TO user1;
```

Modifying DBMS_SCHEDULER Jobs

You can use the Oracle procedure `dbms_scheduler.set_attribute` to modify DBMS_SCHEDULER jobs. For more information, see DBMS_SCHEDULER and SET_ATTRIBUTE Procedure in the Oracle documentation.

When working with Amazon RDS DB instances, prepend the schema name `SYS` to the object name. The following example sets the resource plan attribute for the monday window object.

```
begin
    dbms_scheduler.set_attribute(
        name      => 'SYS.MONDAY_WINDOW',
        attribute => 'RESOURCE_PLAN',
        value     => 'resource_plan_1');
end;
/
```

Creating Custom Functions to Verify Passwords

You can create a custom password verification function in two ways. If you want to use standard verification logic, and to store your function in the `SYS` schema, use the `create_verify_function` procedure. If you want to use custom verification logic, or you don't want to store your function in the `SYS` schema, use the `create_passthrough_verify_fcn` procedure.

The `create_verify_function` Procedure

The `create_verify_function` procedure is supported for Oracle version 11.2.0.4.v9 and later, and 12.1.0.2.v5 and later.

You can create a custom function to verify passwords by using the Amazon RDS procedure `rdsadmin.rdsadmin_password_verify.create_verify_function`. The `create_verify_function` procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
`p_verify_function_name`	varchar2	—	required	The name for your custom function. This function is created for you in the SYS schema. You assign this function to user profiles.
`p_min_length`	number	8	optional	The minimum number of characters required.
`p_max_length`	number	256	optional	The maximum number of characters allowed.
`p_min_letters`	number	1	optional	The minimum number of letters required.
`p_min_uppercase`	number	0	optional	The minimum number of uppercase letters required.
`p_min_lowercase`	number	0	optional	The minimum number of lowercase letters required.
`p_min_digits`	number	1	optional	The minimum number of digits required.
`p_min_special`	number	0	optional	The minimum number of special characters required.
`p_min_different_chars`	number	3	optional	The minimum number of distinct characters required.
`p_disallow_username`	boolean	true	optional	Set to `true` to disallow the username in the password.
`p_disallow_reverse`	boolean	true	optional	Set to `true` to disallow the reverse of the username in the password.
`p_disallow_db_name`	boolean	true	optional	Set to `true` to disallow the database or server name in the password.
`p_disallow_simple_strings`	boolean	true	optional	Set to `true` to disallow simple strings as the password.
`p_disallow_whitespace`	boolean	false	optional	Set to `true` to disallow white space characters in the password.

Parameter Name	Data Type	Default	Required	Description
p_disallow_at_sign	boolean	false	optional	Set to true to disallow the @ character in the password.

You can create multiple password verification functions.

There are restrictions on the name of your custom function. Your custom function can't have the same name as an existing system object, the name can be no more than 30 characters long, and the name must include one of the following strings: PASSWORD, VERIFY, COMPLEXITY, ENFORCE, or STRENGTH.

The following example creates a function named CUSTOM_PASSWORD_FUNCTION. The function requires that a password has at least 12 characters, 2 uppercase characters, 1 digit, and 1 special character, and that the password disallows the @ character.

```
begin
    rdsadmin.rdsadmin_password_verify.create_verify_function(
        p_verify_function_name => 'CUSTOM_PASSWORD_FUNCTION',
        p_min_length           => 12,
        p_min_uppercase        => 2,
        p_min_digits           => 1,
        p_min_special          => 1,
        p_disallow_at_sign     => true);
end;
/
```

To see the text of your verification function, query DBA_SOURCE. The following example gets the text of a custom password function named CUSTOM_PASSWORD_FUNCTION.

```
col text format a150

  select TEXT
    from DBA_SOURCE
   where OWNER = 'SYS' and NAME = 'CUSTOM_PASSWORD_FUNCTION'
order by LINE;
```

To associate your verification function with a user profile, use alter profile. The following example associates a verification function with the DEFAULT user profile.

```
alter profile DEFAULT limit PASSWORD_VERIFY_FUNCTION CUSTOM_PASSWORD_FUNCTION;
```

To see what user profiles are associated with what verification functions, query DBA_PROFILES. The following example gets the profiles that are associated with the custom verification function named CUSTOM_PASSWORD_FUNCTION.

```
select *
  from DBA_PROFILES
 where RESOURCE = 'PASSWORD' and LIMIT = 'CUSTOM_PASSWORD_FUNCTION';


PROFILE                    RESOURCE_NAME                    RESOURCE  LIMIT
-------------------------  -------------------------------  --------
 ------------------------
DEFAULT                    PASSWORD_VERIFY_FUNCTION         PASSWORD
 CUSTOM_PASSWORD_FUNCTION
```

The following example gets all profiles and the password verification functions that they are associated with.

```
select *
  from DBA_PROFILES
 where RESOURCE_NAME = 'PASSWORD_VERIFY_FUNCTION';


PROFILE                        RESOURCE_NAME                    RESOURCE  LIMIT
------------------------------ -------------------------------- --------
 ------------------------
DEFAULT                        PASSWORD_VERIFY_FUNCTION         PASSWORD
 CUSTOM_PASSWORD_FUNCTION
RDSADMIN                       PASSWORD_VERIFY_FUNCTION         PASSWORD  NULL
```

The `create_passthrough_verify_fcn` Procedure

The `create_passthrough_verify_fcn` procedure is supported for Oracle version 11.2.0.4.v11 and later, and 12.1.0.2.v7 and later.

You can create a custom function to verify passwords by using the Amazon RDS procedure `rdsadmin.rdsadmin_password_verify.create_passthrough_verify_fcn`. The `create_passthrough_verify_fcn` procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
`p_verify_function_name`	varchar2	—	required	The name for your custom verification function. This is a wrapper function that is created for you in the SYS schema, and it doesn't contain any verification logic. You assign this function to user profiles.
`p_target_owner`	varchar2	—	required	The schema owner for your custom verification function.
`p_target_function_name`	varchar2	—	required	The name of your existing custom function that contains the verification logic. Your custom function must return a boolean. Your function should return `true` if the password is valid and `false` if the password is invalid.

The following example creates a password verification function that uses the logic from the function named `PASSWORD_LOGIC_EXTRA_STRONG`.

```
begin
    rdsadmin.rdsadmin_password_verify.create_passthrough_verify_fcn(
        p_verify_function_name => 'CUSTOM_PASSWORD_FUNCTION',
        p_target_owner         => 'TEST_USER',
        p_target_function_name => 'PASSWORD_LOGIC_EXTRA_STRONG');
```

```
end;
/
```

To associate the verification function with a user profile, use `alter profile`. The following example associates the verification function with the `DEFAULT` user profile.

```
alter profile DEFAULT limit PASSWORD_VERIFY_FUNCTION CUSTOM_PASSWORD_FUNCTION;
```

Setting Up a Custom DNS Server

Amazon RDS supports outbound network access on your DB instances running Oracle. For more information about outbound network access, including prerequisites, see Using utl_http, utl_tcp, and utl_smtp with an Oracle DB Instance (p. 947).

Amazon RDS Oracle allows Domain Name Service (DNS) resolution from a custom DNS server owned by the customer. You can resolve only fully qualified domain names from your Amazon RDS DB instance through your custom DNS server.

After you set up your custom DNS name server, it takes up to 30 minutes to propagate the changes to your DB instance. After the changes are propagated to your DB instance, all outbound network traffic requiring a DNS lookup queries your DNS server over port 53.

To set up a custom DNS server for your Oracle Amazon RDS DB instance, do the following:

- From the DHCP options set attached to your VPC, set the `domain-name-servers` option to the IP address of your DNS name server. For more information, see DHCP Options Sets.

  Note
  The `domain-name-servers` option accepts up to four values, but your Amazon RDS DB instance uses only the first value.
- Ensure that your DNS server can resolve all lookup queries, including public DNS names, Amazon EC2 private DNS names, and customer-specific DNS names. If the outbound network traffic contains any DNS lookups that your DNS server can't handle, your DNS server must have appropriate upstream DNS providers configured.
- Configure your DNS server to produce User Datagram Protocol (UDP) responses of 512 bytes or less.
- Configure your DNS server to produce Transmission Control Protocol (TCP) responses of 1024 bytes or less.
- Configure your DNS server to allow inbound traffic from your Amazon RDS DB instances over port 53. If your DNS server is in an Amazon VPC, the VPC must have a security group that contains inbound rules that allow UDP and TCP traffic on port 53. If your DNS server is not in an Amazon VPC, it must have appropriate firewall whitelisting to allow UDP and TCP inbound traffic on port 53.

  For more information, see Security Groups for Your VPC and Adding and Removing Rules.
- Configure the VPC of your Amazon RDS DB instance to allow outbound traffic over port 53. Your VPC must have a security group that contains outbound rules that allow UDP and TCP traffic on port 53.

  For more information, see Security Groups for Your VPC and Adding and Removing Rules.
- The routing path between the Amazon RDS DB instance and the DNS server has to be configured correctly to allow DNS traffic.
  - If the Amazon RDS DB instance and the DNS server are not in the same VPC, a peering connection has to be setup between them. For more information, see What is VPC Peering?

Related Topics

- Common DBA Database Tasks for Oracle DB Instances (p. 1054)

Common DBA Database Tasks for Oracle DB Instances

This section describes how you can perform common DBA tasks related to databases on your Amazon RDS DB instances running Oracle. To deliver a managed service experience, Amazon RDS doesn't provide shell access to DB instances, and restricts access to certain system procedures and tables that require advanced privileges.

Changing the Global Name of a Database

You can use the Amazon RDS procedure `rdsadmin.rdsadmin_util.rename_global_name` to change the global name of a database. The `rename_global_name` procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
`p_new_global_name`	varchar2	—	required	The new global name for the database.

The database must be open for the name change to occur. For more information about changing the global name of a database, see ALTER DATABASE in the Oracle documentation.

The following example changes the global name of a database to `new_global_name`.

```
exec rdsadmin.rdsadmin_util.rename_global_name(p_new_global_name => 'new_global_name');
```

Creating and Sizing Tablespaces

Amazon RDS only supports Oracle Managed Files (OMF) for data files, log files and control files. When you create data files and log files, you can't specify the physical file names.

By default, tablespaces are created with auto-extend enabled, and no maximum size. Because of these default settings, tablespaces can grow to consume all allocated storage. We recommend that you specify an appropriate maximum size on permanent and temporary tablespaces, and that you carefully monitor space usage.

The following example creates a tablespace named `users2` with a starting size of 1 gigabyte and a maximum size of 10 gigabytes:

```
create tablespace users2 datafile size 1G autoextend on maxsize 10G;
```

The following example creates temporary tablespace named `temp01`:

```
create temporary tablespace temp01;
```

The Oracle `ALTER DATABASE` system privilege is not available on Amazon RDS. We recommend that you don't use smallfile tablespaces, because you can only perform some operations, such as resizing existing datafiles, by using the `ALTER DATABASE` statement.

You can resize a bigfile tablespace by using `ALTER TABLESPACE`. You can specify the size in kilobytes (K), megabytes (M), gigabytes (G), or terabytes (T).

The following example resizes a bigfile tablespace named `users2` to 200 MB:

```
alter tablespace users2 resize 200M;
```

The following example adds an additional datafile to a smallfile tablespace named users2:

```
alter tablespace users3 add datafile size 100000M autoextend on next 250m
 maxsize UNLIMITED;
```

Setting the Default Tablespace

You can use the Amazon RDS procedure `rdsadmin.rdsadmin_util.alter_default_tablespace` to set the default tablespace. The `alter_default_tablespace` procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
`tablespace_name`	varchar	—	required	The name of the default tablespace.

The following example sets the default tablespace to *`users2`*:

```
exec rdsadmin.rdsadmin_util.alter_default_tablespace(tablespace_name => 'users2');
```

Setting the Default Temporary Tablespace

You can use the Amazon RDS procedure `rdsadmin.rdsadmin_util.alter_default_temp_tablespace` to set the default temporary tablespace. The `alter_default_temp_tablespace` procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
`tablespace_name`	varchar	—	required	The name of the default temporary tablespace.

The following example sets the default tepmorary tablespace to *`temp01`*:

```
exec rdsadmin.rdsadmin_util.alter_default_temp_tablespace(tablespace_name => 'temp01');
```

Checkpointing the Database

You can use the Amazon RDS procedure `rdsadmin.rdsadmin_util.checkpoint` to checkpoint the database. The `checkpoint` procedure has no parameters.

The following example checkpoints the database:

```
exec rdsadmin.rdsadmin_util.checkpoint;
```

Setting Distributed Recovery

You can use the Amazon RDS procedures `rdsadmin.rdsadmin_util.enable_distr_recovery` and `disable_distr_recovery` to set distributed recovery. The procedures have no parameters.

The following example enables distributed recovery:

```
exec rdsadmin.rdsadmin_util.enable_distr_recovery;
```

The following example disables distributed recovery:

```
exec rdsadmin.rdsadmin_util.disable_distr_recovery;
```

Setting the Database Time Zone

There are two different ways that you can set the time zone of your Amazon RDS Oracle database:

- You can use the `Timezone` option.

  The `Timezone` option changes the time zone at the host level and impacts all date columns and values such as `SYSDATE`. For more information about the `Timezone` option, see Oracle Time Zone (p. 1033).

- You can use the Amazon RDS procedure `rdsadmin.rdsadmin_util.alter_db_time_zone`.

  The `alter_db_time_zone` procedure changes the time zone for only certain data types, and doesn't change `SYSDATE`. There are additional restrictions on setting the time zone listed in the Oracle documentation.

The `alter_db_time_zone` procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
`p_new_tz`	varchar2	—	required	The new time zone as an named region or an absolute offset from Coordinated Universal Time (UTC). Valid offsets range from -12:00 to +14:00.

The following example changes the time zone to UTC plus 3 hours:

```
exec rdsadmin.rdsadmin_util.alter_db_time_zone(p_new_tz => '+3:00');
```

The following example changes the time zone to the time zone of the Africa/Algiers region:

```
exec rdsadmin.rdsadmin_util.alter_db_time_zone(p_new_tz => 'Africa/Algiers');
```

After you alter the time zone by using the `alter_db_time_zone` procedure, you must reboot the DB instance for the change to take effect. For more information, see Rebooting a DB Instance (p. 119).

Working with Oracle External Tables

Oracle external tables are tables with data that is not in the database. Instead, the data is in external files that the database can access. By using external tables, you can access data without loading it into the database. For more information about external tables, see Managing External Tables in the Oracle documentation.

With Amazon RDS, you can store external table files in directory objects. You can create a directory object, or you can use one that is predefined in the Oracle database, such as the DATA_PUMP_DIR directory. For information about creating directory objects, see Creating New Directories in the Main Data Storage Space (p. 1072). You can query the ALL_DIRECTORIES view to list the directory objects for your Amazon RDS Oracle DB instance.

> **Note**
> Directory objects point to the main data storage space (Amazon EBS volume) used by your instance. The space used—along with data files, redo logs, audit, trace, and other files—counts against allocated storage.

You can move an external data file from one Oracle database to another by using the DBMS_FILE_TRANSFER package or the UTL_FILE package. The external data file is moved from a directory on the source database to the specified directory on the destination database. For information about using DBMS_FILE_TRANSFER, see Oracle Data Pump (p. 983).

After you move the external data file, you can create an external table with it. The following example creates an external table that uses the `emp_xt_file1.txt` file in the USER_DIR1 directory:

```
CREATE TABLE emp_xt (
  emp_id      NUMBER,
  first_name  VARCHAR2(50),
  last_name   VARCHAR2(50),
  user_name   VARCHAR2(20)
)
ORGANIZATION EXTERNAL (
  TYPE ORACLE_LOADER
  DEFAULT DIRECTORY USER_DIR1
  ACCESS PARAMETERS (
    RECORDS DELIMITED BY NEWLINE
    FIELDS TERMINATED BY ','
    MISSING FIELD VALUES ARE NULL
    (emp_id,first_name,last_name,user_name)
  )
  LOCATION ('emp_xt_file1.txt')
)
PARALLEL
REJECT LIMIT UNLIMITED;
```

Suppose that you want to move data that is in an Amazon RDS Oracle DB instance into an external data file. In this case, you can populate the external data file by creating an external table and selecting the data from the table in the database. For example, the following SQL statement creates the `orders_xt` external table by querying the `orders` table in the database.

```
CREATE TABLE orders_xt
  ORGANIZATION EXTERNAL
   (
     TYPE ORACLE_DATAPUMP
     DEFAULT DIRECTORY DATA_PUMP_DIR
     LOCATION ('orders_xt.dmp')
   )
```

```
  AS SELECT * FROM orders;
```

In this example, the data is populated in the `orders_xt.dmp` file in the DATA_PUMP_DIR directory.

Working with Automatic Workload Repository (AWR)

If you use Oracle Database Enterprise Edition and want to use Automatic Workload Repository (AWR), you can enable AWR by changing the `CONTROL_MANAGEMENT_PACK_ACCESS` parameter.

Oracle AWR includes several report generation scripts, such as awrrpt.sql, that are installed on the host server. You do not have direct access to the host, but you can copy the scripts from another installation of Oracle Database.

Adjusting Database Links for Use with DB Instances in a VPC

To use Oracle database links with Amazon RDS DB instances inside the same VPC or peered VPCs, the two DB instances should have a valid route between them. Verify the valid route between the DB instances by using your VPC routing tables and network access control list (ACL).

The security group of each DB instance must allow ingress to and egress from the other DB instance. The inbound and outbound rules can refer to security groups from the same VPC or a peered VPC. For more information, see Updating Your Security Groups to Reference Peered VPC Security Groups.

If you have configured a custom DNS server using the DHCP Option Sets in your VPC, your custom DNS server must be able to resolve the name of the database link target. For more information, see Setting Up a Custom DNS Server (p. 1053).

For more information about using database links with Oracle Data Pump, see Oracle Data Pump (p. 983).

Setting the Default Edition for a DB Instance

You can redefine database objects in a private environment called an edition. You can use edition-based redefinition to upgrade an application's database objects with minimal downtime.

You can set the default edition of an Amazon RDS Oracle DB instance using the Amazon RDS procedure `rdsadmin.rdsadmin_util.alter_default_edition`.

The following example sets the default edition for the Amazon RDS Oracle DB instance to `RELEASE_V1`.

```
exec rdsadmin.rdsadmin_util.alter_default_edition('RELEASE_V1');
```

The following example sets the default edition for the Amazon RDS Oracle DB instance back to the Oracle default.

```
exec rdsadmin.rdsadmin_util.alter_default_edition('ORA$BASE');
```

For more information about Oracle edition-based redefinition, see About Editions and Edition-Based Redefinition in the Oracle documentation.

Validating DB Instance Files

You can use the Amazon RDS package `rdsadmin.rdsadmin_rman_util` to validate Amazon RDS Oracle DB instance files, such as data files, server parameter files (SPFILEs), and control files.

Note
The `rdsadmin.rdsadmin_rman_util` package provides capabilities that are available with Oracle Recovery Manager (RMAN) validation. While Amazon RDS does not use RMAN for backups, you can use the package to execute RMAN validation commands against the database,

control file, SPFILE, tablespaces, or data files. For more information about RMAN validation, see Validating Database Files and Backups and VALIDATE in the Oracle documentation.

Validating a DB Instance

You can use the Amazon RDS procedure `rdsadmin.rdsadmin_rman_util.validate_database` to validate all of the relevant files used by an Amazon RDS Oracle DB instance.

Parameter Name	Data Type	Valid Values	Default	Required	Description
`p_validation_type`	varchar2	`'PHYSICAL'`,`'PHYSICAL +LOGICAL'`	`'PHYSICAL'`	Optional	The level of corruption detection. Specify `'PHYSICAL'` to check for physical corruption. An example of physical corruption is a block with a mismatch in the header and footer. Specify `'PHYSICAL +LOGICAL'` to check for logical inconsistencies in addition to physical corruption. An example of logical corruption is a corrupt block.
`p_parallel`	number	A valid integer between `1` and `254` for Oracle Database Enterprise Edition (EE) `1` for other Oracle Database editions	`1`	Optional	Number of channels.
`p_section_size_mb`	number	A valid integer	`NULL`	Optional	The section size in megabytes (MB). Validates in parallel by dividing each file into the specified section size. When `NULL`, the parameter is ignored.
`p_rman_to_dbms_output`	boolean	`TRUE`, `FALSE`	`FALSE`	Optional	When `TRUE`, the RMAN output is sent to

Parameter Name	Data Type	Valid Values	Default	Required	Description
					the DBMS_OUTPUT package in addition to a file in the BDUMP directory. When using SQL*Plus, execute SET SERVEROUTPUT ON to see the output. When FALSE, the RMAN output is only sent to a file in the BDUMP directory.

The following example validates the DB instance using the default values for the parameters:

```
exec rdsadmin.rdsadmin_rman_util.validate_database;
```

The following example validates the DB instance using the specified values for the parameters:

```
BEGIN
    rdsadmin.rdsadmin_rman_util.validate_database(
        p_validation_type     => 'PHYSICAL+LOGICAL',
        p_parallel            => 4,
        p_section_size_mb     => 10,
        p_rman_to_dbms_output => FALSE);
END;
/
```

When the p_rman_to_dbms_output parameter is set to FALSE, the RMAN output is written to a file in the BDUMP directory.

To view the files in the BDUMP directory, run the following SELECT statement:

```
SELECT * FROM table(rdsadmin.rds_file_util.listdir('BDUMP')) order by mtime;
```

To view the contents of a file in the BDUMP directory, run the following SELECT statement:

```
SELECT text FROM table(rdsadmin.rds_file_util.read_text_file('BDUMP','rds-rman-
validate-nnn.txt'));
```

Replace the file name with the name of the file you want to view.

Validating a Tablespace

You can use the Amazon RDS procedure rdsadmin.rdsadmin_rman_util.validate_tablespace to validate the files associated with a tablespace.

Parameter Name	Data Type	Valid Values	Default	Required	Description
p_tablespace_name	varchar2	A valid tablespace name	—	Required	The name of the tablespace.
p_validation_type	varchar2	'PHYSICAL','PHYSICAL' 'PHYSICAL +LOGICAL'	'PHYSICAL'	Optional	The level of corruption detection. Specify 'PHYSICAL' to check for physical corruption. An example of physical corruption is a block with a mismatch in the header and footer. Specify 'PHYSICAL +LOGICAL' to check for logical inconsistencies in addition to physical corruption. An example of logical corruption is a corrupt block.
p_parallel	number	A valid integer between 1 and 254 for Oracle Database Enterprise Edition (EE) 1 for other Oracle Database editions	1	Optional	Number of channels.
p_section_size_mb	number	A valid integer	NULL	Optional	The section size in megabytes (MB). Validates in parallel by dividing each file into the specified section size. When NULL, the parameter is ignored.
p_rman_to_dbms_output	boolean	TRUE, FALSE	FALSE	Optional	When TRUE, the RMAN output is sent to the DBMS_OUTPUT package in addition to a file in the BDUMP

Parameter Name	Data Type	Valid Values	Default	Required	Description
					directory. When using SQL*Plus, execute `SET SERVEROUTPUT ON` to see the output. When `FALSE`, the RMAN output is only sent to a file in the `BDUMP` directory.

Validating a Control File

You can use the Amazon RDS procedure `rdsadmin.rdsadmin_rman_util.validate_current_controlfile` to validate only the control file used by an Amazon RDS Oracle DB instance.

Parameter Name	Data Type	Valid Values	Default	Required	Description
`p_validation_type`	varchar2	`'PHYSICAL'` `'PHYSICAL +LOGICAL'`	`'PHYSICAL'`	Optional	The level of corruption detection. Specify `'PHYSICAL'` to check for physical corruption. An example of physical corruption is a block with a mismatch in the header and footer. Specify `'PHYSICAL +LOGICAL'` to check for logical inconsistencies in addition to physical corruption. An example of logical corruption is a corrupt block.
`p_rman_to_dbms_output`	boolean	`TRUE`, `FALSE`	`FALSE`	Optional	When `TRUE`, the RMAN output is sent to the `DBMS_OUTPUT` package in addition to a file in the `BDUMP` directory. When using SQL*Plus, execute `SET SERVEROUTPUT ON` to see the output. When `FALSE`, the RMAN output is only sent to a file in the `BDUMP` directory.

Validating an SPFILE

You can use the Amazon RDS procedure `rdsadmin.rdsadmin_rman_util.validate_spfile` to validate only the server parameter file (SPFILE) used by an Amazon RDS Oracle DB instance.

Parameter Name	Data Type	Valid Values	Default	Required	Description
`p_validation_type`	varchar2	`'PHYSICAL'`, `'PHYSICAL +LOGICAL'`	`'PHYSICAL'`	Optional	The level of corruption detection. Specify `'PHYSICAL'` to check for physical corruption. An example of physical corruption is a block with a mismatch in the header and footer. Specify `'PHYSICAL +LOGICAL'` to check for logical inconsistencies in addition to physical corruption. An example of logical corruption is a corrupt block.
`p_rman_to_dbms_output`	boolean	`TRUE`, `FALSE`	`FALSE`	Optional	When `TRUE`, the RMAN output is sent to the `DBMS_OUTPUT` package in addition to a file in the `BDUMP` directory. When using SQL*Plus, execute `SET SERVEROUTPUT ON` to see the output. When `FALSE`, the RMAN output is only sent to a file in the `BDUMP` directory.

Validating a Data File

You can use the Amazon RDS procedure `rdsadmin.rdsadmin_rman_util.validate_datafile` to validate a data file.

Parameter Name	Data Type	Valid Values	Default	Required	Description
`p_datafile`	varchar2	A valid tablespace name	—	Required	The name of the data file.

Parameter Name	Data Type	Valid Values	Default	Required	Description
p_from_block	number	A valid integer	NULL	Optional	The number of the block where the validation starts within the data file. When NULL, 1 is used.
p_to_block	number	A valid integer	NULL	Optional	The number of the block where the validation ends within the data file. When NULL, the max block in the data file is used.
p_validation_type	varchar2	'PHYSICAL','PHYSICAL' 'PHYSICAL +LOGICAL'	'PHYSICAL'	Optional	The level of corruption detection. Specify 'PHYSICAL' to check for physical corruption. An example of physical corruption is a block with a mismatch in the header and footer. Specify 'PHYSICAL +LOGICAL' to check for logical inconsistencies in addition to physical corruption. An example of logical corruption is a corrupt block.
p_parallel	number	A valid integer between 1 and 254 for Oracle Database Enterprise Edition (EE) 1 for other Oracle Database editions	1	Optional	Number of channels.

Parameter Name	Data Type	Valid Values	Default	Required	Description
p_section_size_mb	number	A valid integer	NULL	Optional	The section size in megabytes (MB). Validates in parallel by dividing each file into the specified section size. When NULL, the parameter is ignored.
p_rman_to_dbms_output	boolean	TRUE, FALSE	FALSE	Optional	When TRUE, the RMAN output is sent to the DBMS_OUTPUT package in addition to a file in the BDUMP directory. When using SQL*Plus, execute SET SERVEROUTPUT ON to see the output. When FALSE, the RMAN output is only sent to a file in the BDUMP directory.

Related Topics

- Common DBA System Tasks for Oracle DB Instances (p. 1045)
- Common DBA Log Tasks for Oracle DB Instances (p. 1065)
- Common DBA Miscellaneous Tasks for Oracle DB Instances (p. 1072)

Common DBA Log Tasks for Oracle DB Instances

This section describes how you can perform common DBA tasks related to logging on your Amazon RDS DB instances running Oracle. To deliver a managed service experience, Amazon RDS doesn't provide shell access to DB instances, and restricts access to certain system procedures and tables that require advanced privileges.

For more information, see Oracle Database Log Files (p. 318).

Setting Force Logging

In force logging mode, Oracle logs all changes to the database except changes in temporary tablespaces and temporary segments (NOLOGGING clauses are ignored). For more information, see Specifying FORCE LOGGING Mode in the Oracle documentation.

You can use the Amazon RDS procedure rdsadmin.rdsadmin_util.force_logging to set force logging. The force_logging procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
p_enable	boolean	true	optional	Set to true to put the database in force logging mode, false to remove the database from force logging mode.

The following example puts the database in force logging mode.

```
exec rdsadmin.rdsadmin_util.force_logging(p_enable => true);
```

Setting Supplemental Logging

Supplemental logging ensures that LogMiner and products that use LogMiner technology have sufficient information to support chained rows and storage arrangements such as cluster tables. For more information, see Supplemental Logging in the Oracle documentation.

Oracle Database doesn't enable supplemental logging by default. You can use the Amazon RDS procedure rdsadmin.rdsadmin_util.alter_supplemental_logging to enable and disable supplemental logging. For more information about how Amazon RDS manages the retention of archived redo logs for Oracle DB instances, see Retaining Archived Redo Logs (p. 1070).

The alter_supplemental_logging procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
p_action	varchar2	—	required	'ADD' to add supplemental logging, 'DROP' to drop supplemental logging.
p_type	varchar2	null	optional	The type of supplemental logging. Valid values are 'ALL', 'FOREIGN KEY', 'PRIMARY KEY', or 'UNIQUE'.

The following example enables supplemental logging:

```
begin
    rdsadmin.rdsadmin_util.alter_supplemental_logging(
        p_action => 'ADD');
end;
/
```

The following example enables supplemental logging for all fixed-length maximum size columns:

```
begin
    rdsadmin.rdsadmin_util.alter_supplemental_logging(
        p_action => 'ADD',
        p_type   => 'ALL');
end;
/
```

The following example enables supplemental logging for primary key columns:

```
begin
    rdsadmin.rdsadmin_util.alter_supplemental_logging(
        p_action => 'ADD',
        p_type   => 'PRIMARY KEY');
end;
/
```

Switching Online Log Files

You can use the Amazon RDS procedure `rdsadmin.rdsadmin_util.switch_logfile` to switch log files. The `switch_logfile` procedure has no parameters.

The following example switches log files.

```
exec rdsadmin.rdsadmin_util.switch_logfile;
```

Adding Online Redo Logs

An Amazon RDS DB instance running Oracle starts with four online redo logs, 128 MB each. You can use the Amazon RDS procedure `rdsadmin.rdsadmin_util.add_logfile` to add additional redo logs.

For any version of Oracle, the `add_logfile` procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
`bytes`	positive	null	optional	The size of the log file in bytes.

The `add_logfile` procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
`p_size`	varchar2	—	required	The size of the log file. You can specify the size in kilobytes (K), megabytes (M), or gigabytes (G).

The following command adds a 100 MB log file:

```
exec rdsadmin.rdsadmin_util.add_logfile(p_size => '100M');
```

Dropping Online Redo Logs

You can use the Amazon RDS procedure `rdsadmin.rdsadmin_util.drop_logfile` to drop redo logs. The `drop_logfile` procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
`grp`	positive	—	requried	The group number of the log.

The following example drops the log with group number 3:

```
exec rdsadmin.rdsadmin_util.drop_logfile(grp => 3);
```

You can only drop logs that have a status of unused or inactive. The following example gets the statuses of the logs:

```
select GROUP#, STATUS from V$LOG;

GROUP#     STATUS
---------- ----------------
1          CURRENT
2          INACTIVE
3          INACTIVE
4          UNUSED
```

Resizing Online Redo Logs

An Amazon RDS DB instance running Oracle starts with four online redo logs, 128 MB each. The following example shows how you can use Amazon RDS procedures to resize your logs from 128 MB each to 512 MB each.

```
/* Query V$LOG to see the logs.          */
/* You start with 4 logs of 128 MB each. */

select GROUP#, BYTES, STATUS from V$LOG;

GROUP#     BYTES      STATUS
---------- ---------- ----------------
1          134217728  INACTIVE
2          134217728  CURRENT
3          134217728  INACTIVE
4          134217728  INACTIVE


/* Add four new logs that are each 512 MB */

exec rdsadmin.rdsadmin_util.add_logfile(bytes => 536870912);
exec rdsadmin.rdsadmin_util.add_logfile(bytes => 536870912);
exec rdsadmin.rdsadmin_util.add_logfile(bytes => 536870912);
exec rdsadmin.rdsadmin_util.add_logfile(bytes => 536870912);


/* Query V$LOG to see the logs. */
/* Now there are 8 logs.        */

select GROUP#, BYTES, STATUS from V$LOG;

GROUP#     BYTES      STATUS
---------- ---------- ----------------
1          134217728  INACTIVE
2          134217728  CURRENT
3          134217728  INACTIVE
4          134217728  INACTIVE
5          536870912  UNUSED
6          536870912  UNUSED
7          536870912  UNUSED
8          536870912  UNUSED


/* Drop each inactive log using the group number. */
```

```
exec rdsadmin.rdsadmin_util.drop_logfile(grp => 1);
exec rdsadmin.rdsadmin_util.drop_logfile(grp => 3);
exec rdsadmin.rdsadmin_util.drop_logfile(grp => 4);


/* Query V$LOG to see the logs. */
/* Now there are 5 logs.        */

select GROUP#, BYTES, STATUS from V$LOG;

GROUP#     BYTES      STATUS
---------- ---------- ----------------
2          134217728  CURRENT
5          536870912  UNUSED
6          536870912  UNUSED
7          536870912  UNUSED
8          536870912  UNUSED


/* Switch logs so that group 2 is no longer current. */

exec rdsadmin.rdsadmin_util.switch_logfile;


/* Query V$LOG to see the logs.        */
/* Now one of the new logs is current. */

SQL>select GROUP#, BYTES, STATUS from V$LOG;

GROUP#     BYTES      STATUS
---------- ---------- ----------------
2          134217728  ACTIVE
5          536870912  CURRENT
6          536870912  UNUSED
7          536870912  UNUSED
8          536870912  UNUSED


/* Issue a checkpoint to clear log 2. */

exec rdsadmin.rdsadmin_util.checkpoint;


/* Query V$LOG to see the logs.            */
/* Now the final original log is inactive. */

select GROUP#, BYTES, STATUS from V$LOG;

GROUP#     BYTES      STATUS
---------- ---------- ----------------
2          134217728  INACTIVE
5          536870912  CURRENT
6          536870912  UNUSED
7          536870912  UNUSED
8          536870912  UNUSED


# Drop the final inactive log.

exec rdsadmin.rdsadmin_util.drop_logfile(grp => 2);


/* Query V$LOG to see the logs.    */
/* Now there are four 512 MB logs. */

select GROUP#, BYTES, STATUS from V$LOG;
```

```
GROUP#     BYTES      STATUS
---------- ---------- ----------------
5          536870912  CURRENT
6          536870912  UNUSED
7          536870912  UNUSED
8          536870912  UNUSED
```

Retaining Archived Redo Logs

You can retain archived redo logs locally on your DB instance for use with products like Oracle LogMiner (DBMS_LOGMNR). After you have retained the redo logs, you can use LogMiner to analyze the logs. For more information, see Using LogMiner to Analyze Redo Log Files in the Oracle documentation.

You can use the Amazon RDS procedure `rdsadmin.rdsadmin_util.set_configuration` to retain archived redo logs. The `set_configuration` procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
`name`	varchar	—	required	The name of the configuration to update.
`value`	varchar	—	required	The value for the configuration.

The following example retains 24 hours of redo logs:

```
begin
    rdsadmin.rdsadmin_util.set_configuration(
        name  => 'archivelog retention hours',
        value => '24');
end;
/
```

You can use the Amazon RDS procedure `rdsadmin.rdsadmin_util.show_configuration` to view how long archived redo logs are retained for your DB instance.

The following example shows the log retention time:

```
set serveroutput on
exec rdsadmin.rdsadmin_util.show_configuration;
```

The output shows the current setting for `archivelog retention hours`. The following output shows that archived redo logs are retained for 48 hours:

```
NAME:archivelog retention hours
VALUE:48
DESCRIPTION:ArchiveLog expiration specifies the duration in hours before archive/redo log
 files are automatically deleted.
```

Because the archived redo logs are retained on your DB instance, ensure that your DB instance has enough allocated storage for the retained logs. To determine how much space your DB instance has used in the last X hours, you can run the following query, replacing X with the number of hours:

```
select sum(BLOCKS * BLOCK_SIZE) bytes
  from V$ARCHIVED_LOG
 where FIRST_TIME >= SYSDATE-(X/24) and DEST_ID=1;
```

Archived redo logs are only generated if the backup retention period of your DB instance is greater than zero. By default the backup retention period is greater than zero, so unless you explicitly set yours to zero, archived redo logs are generated for your DB instance. To modify the backup retention period for your DB instance, see Modifying a DB Instance Running the Oracle Database Engine (p. 967).

After the archived redo logs are removed from your DB instance, you can't download them again to your DB instance. Amazon RDS retains the archived redo logs outside of your DB instance to support restoring your DB instance to a point in time. Amazon RDS retains the archived redo logs outside of your DB instance based on the backup retention period configured for your DB instance. To modify the backup retention period for your DB instance, see Modifying a DB Instance Running the Oracle Database Engine (p. 967).

Accessing Transaction Logs

Accessing transaction logs is supported for Oracle version 11.2.0.4.v11 and later, and 12.1.0.2.v7 and later.

You might want to access your online and archived redo log files for mining with external tools such as GoldenGate, Attunity, Informatica, and others. If you want to access your online and archived redo log files, you must first create directory objects that provide read-only access to the physical file paths.

The following code creates directories that provide read-only access to your online and archived redo log files:

Important
This code also revokes the `DROP ANY DIRECTORY` privilege.

```
exec rdsadmin.rdsadmin_master_util.create_archivelog_dir;
exec rdsadmin.rdsadmin_master_util.create_onlinelog_dir;
```

After you create directory objects for your online and archived redo log files, you can read the files by using PL/SQL. For more information about reading files from directory objects, see Listing Files in a DB Instance Directory (p. 1072) and Reading Files in a DB Instance Directory (p. 1073).

The following code drops the directories for your online and archived redo log files:

```
exec rdsadmin.rdsadmin_master_util.drop_archivelog_dir;
exec rdsadmin.rdsadmin_master_util.drop_onlinelog_dir;
```

The following code grants and revokes the `DROP ANY DIRECTORY` privilege:

```
exec rdsadmin.rdsadmin_master_util.revoke_drop_any_directory;
exec rdsadmin.rdsadmin_master_util.grant_drop_any_directory;
```

Related Topics

- Common DBA System Tasks for Oracle DB Instances (p. 1045)
- Common DBA Database Tasks for Oracle DB Instances (p. 1054)
- Common DBA Miscellaneous Tasks for Oracle DB Instances (p. 1072)

Common DBA Miscellaneous Tasks for Oracle DB Instances

This section describes how you can perform miscellaneous DBA tasks on your Amazon RDS DB instances running Oracle. To deliver a managed service experience, Amazon RDS doesn't provide shell access to DB instances, and restricts access to certain system procedures and tables that require advanced privileges.

Creating New Directories in the Main Data Storage Space

You can use the Amazon RDS procedure `rdsadmin.rdsadmin_util.create_directory` to create directories. You can create up to 10,000 directories, all located in your main data storage space.

The `create_directory` procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
`p_directory_name`	varchar2	—	required	The name of the new directory.

The following example creates a new directory named `product_descriptions`:

```
exec rdsadmin.rdsadmin_util.create_directory(p_directory_name => 'product_descriptions');
```

You can list the directories by querying `DBA_DIRECTORIES`. The system chooses the actual host pathname automatically. The following example gets the directory path for the directory named `product_descriptions`:

```
select DIRECTORY_PATH
  from DBA_DIRECTORIES
 where DIRECTORY_NAME='product_descriptions';

DIRECTORY_PATH
----------------------------------------
/rdsdbdata/userdirs/01
```

The master user name for the DB instance has read and write privileges in the new directory, and can grant access to other users. Execute privileges are not available for directories on a DB instance. Directories are created in your main data storage space and will consume space and I/O bandwidth.

You can drop a directory that you created by using the Oracle `drop directory` command. Dropping a directory doesn't remove its contents. Because the `create_directory()` method can reuse pathnames, files in dropped directories can appear in a newly created directory. Before you drop a directory, you should use UTL_FILE.FREMOVE to remove files from the directory.

Listing Files in a DB Instance Directory

You can use the Amazon RDS procedure `rdsadmin.rds_file_util.listdir` to list the files in a directory. The `listdir` procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
`p_directory`	varchar2	—	required	The name of the directory to list.

The following example lists the files in the directory named `product_descriptions`:

```
select * from table
    (rdsadmin.rds_file_util.listdir(p_directory => 'product_descriptions'));
```

Reading Files in a DB Instance Directory

You can use the Amazon RDS procedure `rdsadmin.rds_file_util.read_text_file` to read a text file. The `read_text_file` procedure has the following parameters.

Parameter Name	Data Type	Default	Required	Description
`p_directory`	varchar2	—	required	The name of the directory that contains the file.
`p_filename`	varchar2	—	required	The name of the file to read.

The following example reads the file `rice.txt` from the directory `product_descriptions`:

```
select * from table
    (rdsadmin.rds_file_util.read_text_file(
        p_directory => 'product_descriptions',
        p_filename  => 'rice.txt'));
```

Related Topics

- Common DBA System Tasks for Oracle DB Instances (p. 1045)
- Common DBA Database Tasks for Oracle DB Instances (p. 1054)
- Common DBA Log Tasks for Oracle DB Instances (p. 1065)

Related Topics

- Oracle Database Log Files (p. 318)
- Options for Oracle DB Instances (p. 993)
- Tools and Third-Party Software for Oracle DB Instances (p. 1074)

Tools and Third-Party Software for Oracle DB Instances

This section provides information about tools and third-party software for Oracle DB instances on Amazon RDS.

Topics

Setting Up Amazon RDS to Host Tools and Third-Party Software for Oracle

You can use Amazon RDS to host an Oracle DB instance that supports software and components such as the following:

- Siebel Customer Relationship Management (CRM)
- Oracle Fusion Middleware Metadata — installed by the Repository Creation Utility (RCU)

The following procedures help you create an Oracle DB instance on Amazon RDS that you can use to host additional software and components for Oracle.

Creating an Amazon VPC for Use with an Oracle Database

In the following procedure, you create an Amazon VPC, a private subnet, and a security group. Because your Amazon RDS DB instance needs to be available only to your middle-tier components, and not to the public Internet, your Amazon RDS DB instance is hosted in a private subnet, providing greater security.

To create an Amazon VPC

1. Sign in to the AWS Management Console and open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the top-right corner of the AWS Management Console, choose the AWS Region for your VPC. This example uses the US West (Oregon) region.
3. In the upper-left corner, choose **VPC Dashboard** and then choose **Start VPC Wizard**.
4. On the page **Step 1: Select a VPC Configuration**, choose **VPC with Public and Private Subnets**, and then choose **Select**.
5. On the page **Step 2: VPC with Public and Private Subnets**, shown following, set these values:

Option	Value
IP CIDR block	`10.0.0.0/16` For more information about selecting CIDR blocks for your VPC, see VPC Sizing.

Option	Value
VPC name	The name for your VPC, for example **`vpc-1`**.
Public subnet	`10.0.0.0/24` For more information about subnet sizing, see Subnet Sizing.
Availability Zone	An Availability Zone for your AWS Region.
Public subnet name	The name for your public subnet, for example **`subnet-public-1`**.
Private subnet	`10.0.1.0/24` For more information about subnet sizing, see Subnet Sizing.
Availability Zone	An Availability Zone for your AWS Region.
Private subnet name	The name for your private subnet, for example **`subnet-private-1`**.
Instance type	An instance type for your NAT instance, for example `t2.small`. **Note** If you don't see **Instance type** in the console, choose **Use a NAT instance instead**.
Key pair name	`No key pair`
Subnet	`None`
Enable DNS hostnames	`Yes`
Hardware tenancy	`Default`

6. Choose **Create VPC**.

An Amazon RDS DB instance in a VPC requires at least two private subnets or at least two public subnets, to support Multi-AZ deployment. For more information about working with multiple Availability Zones, see Regions and Availability Zones (p. 97). Because your database is private, add a second private subnet to your VPC.

To create an additional subnet

1. Sign in to the AWS Management Console and open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the top-right corner of the AWS Management Console, confirm that you are in the correct AWS Region for your VPC.

3. In the upper-left corner, choose **VPC Dashboard**, choose **Subnets**, and then choose **Create Subnet**.
4. On the **Create Subnet** page, set these values:

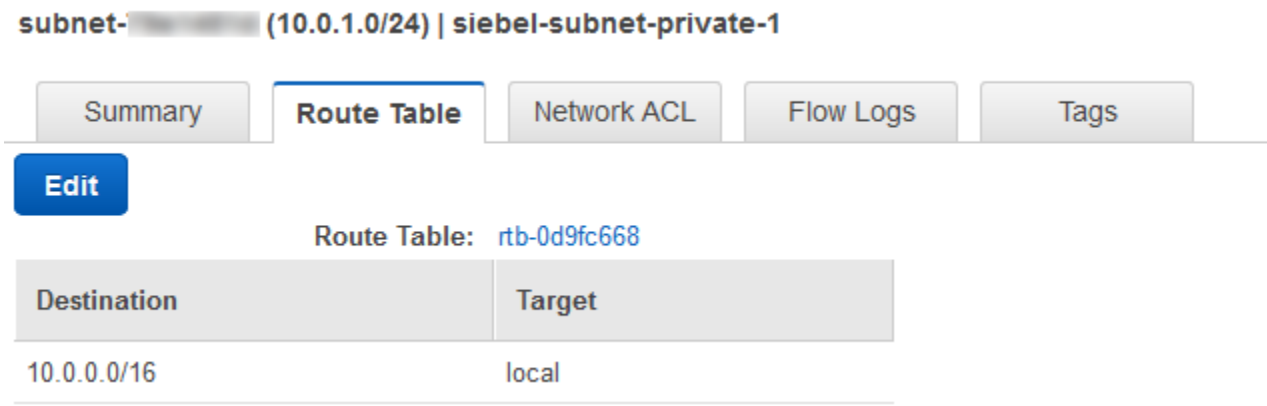
Option	Value
Name tag	The name for your second private subnet, for example **`subnet-private-2`**.
VPC	Your VPC, for example **`vpc`**.
Availability Zone	An Availability Zone for your AWS Region. **Note** Choose an Availability Zone different from the one that you chose for the first private subnet.
CIDR block	`10.0.2.0/24`

5. Choose **Yes, Create**.

Both private subnets must use the same route table. In the following procedure, you check to make sure the route tables match, and if not you edit one of them.

To ensure the subnets use the same route table.

1. Sign in to the AWS Management Console and open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the top-right corner of the AWS Management Console, confirm that you are in the correct AWS Region for your VPC.
3. In the upper-left corner, choose **VPC Dashboard**, choose **Subnets**, and then choose your first private subnet, for example **`subnet-private-1`**.
4. At the bottom of the console, choose the **Route Table** tab, shown following.



5. Make a note of the route table, for example `rtb-0d9fc668`.
6. In the list of subnets, choose the second private subnet, for example **`subnet-private-2`**.
7. At the bottom of the console, choose the **Route Table** tab.
8. If the route table for the second subnet is not the same as the route table for the first subnet, edit it to match:
   a. Choose **Edit**.
   b. For **Change to**, select the route table that matches your first subnet.

c. Choose **Save**.

A security group acts as a virtual firewall for your DB instance to control inbound and outbound traffic. In the following procedure, you create a security group for your DB instance. For more information about security groups, see Security Groups for Your VPC.

To create a VPC security group for a Private Amazon RDS DB Instance

1. Sign in to the AWS Management Console and open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the top-right corner of the AWS Management Console, confirm that you are in the correct AWS Region for your VPC.
3. In the upper-left corner, choose **VPC Dashboard**, choose **Security Groups**, and then choose **Create Security Group**.
4. On the page **Create Security Group**, set these values:

Option	Value
Name tag	The name for your security group, for example **`sg-db-1`**.
Group name	The name for your security group, for example **`sg-db-1`**.
Description	A description for your security group.
VPC	Your VPC, for example **`vpc-1`**.

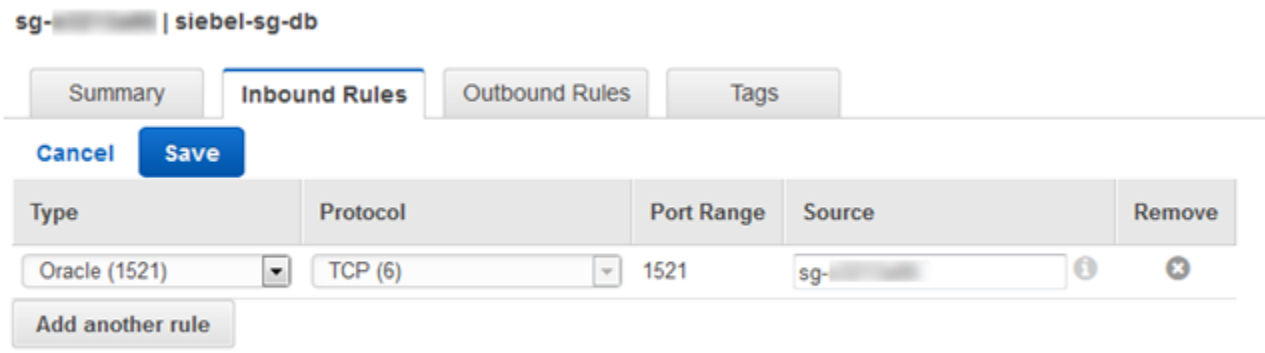
5. Choose **Yes, Create**.

In the following procedure, you add rules to your security group to control inbound traffic to your DB instance. For more information about inbound rules, see Security Group Rules.

To add inbound rules to the security group

1. Sign in to the AWS Management Console and open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the top-right corner of the AWS Management Console, confirm that you are in the correct AWS Region for your VPC.
3. In the upper-left corner, choose **VPC Dashboard**, choose **Security Groups**, and then choose your security group, for example **`sg-db-1`**.
4. At the bottom of the console, choose the **Inbound Rules** tab, and then choose **Edit**.
5. Set these values, as shown following:

Option	Value
Type	`Oracle (1521)`
Protocol	`TCP (6)`
Port Range	1521
Source	The identifier of your security group. When you choose the box, you see the name of your vpc, for example **`vpc-1`**.



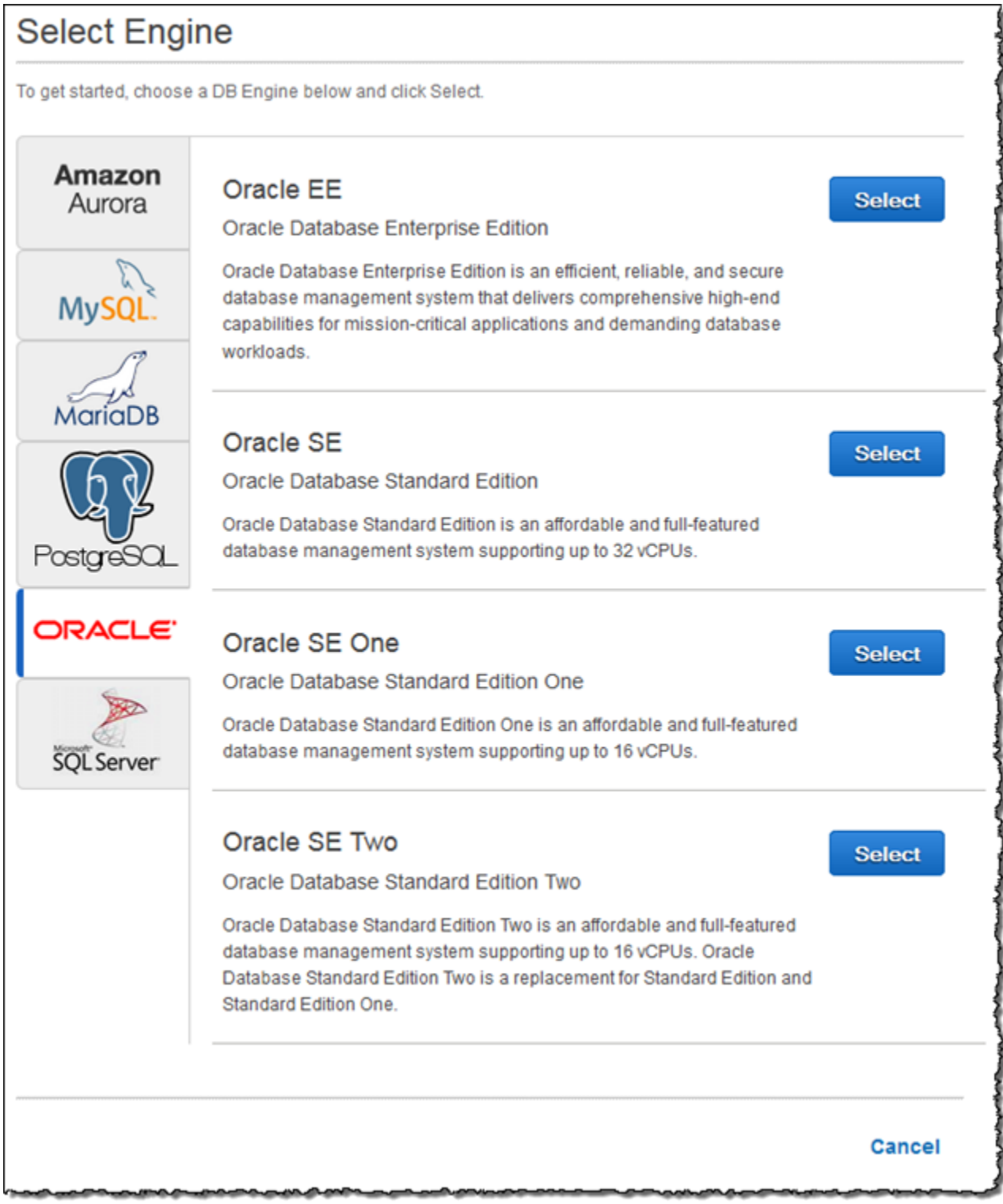
6. Choose **Save**.

Creating an Oracle DB Instance

You can use Amazon RDS to host an Oracle DB instance. In the following procedure, you create the Oracle DB instance.

To launch an Oracle DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the top-right corner of the AWS Management Console, choose the AWS Region for your DB instance. Choose the same AWS Region as your VPC.
3. In the upper-left corner, choose **RDS Dashboard** and then choose **Launch a DB Instance**.
4. On the page **Step 1: Select Engine**, choose **Oracle**, and then choose the **Select** button for the Oracle Database Enterprise Edition.

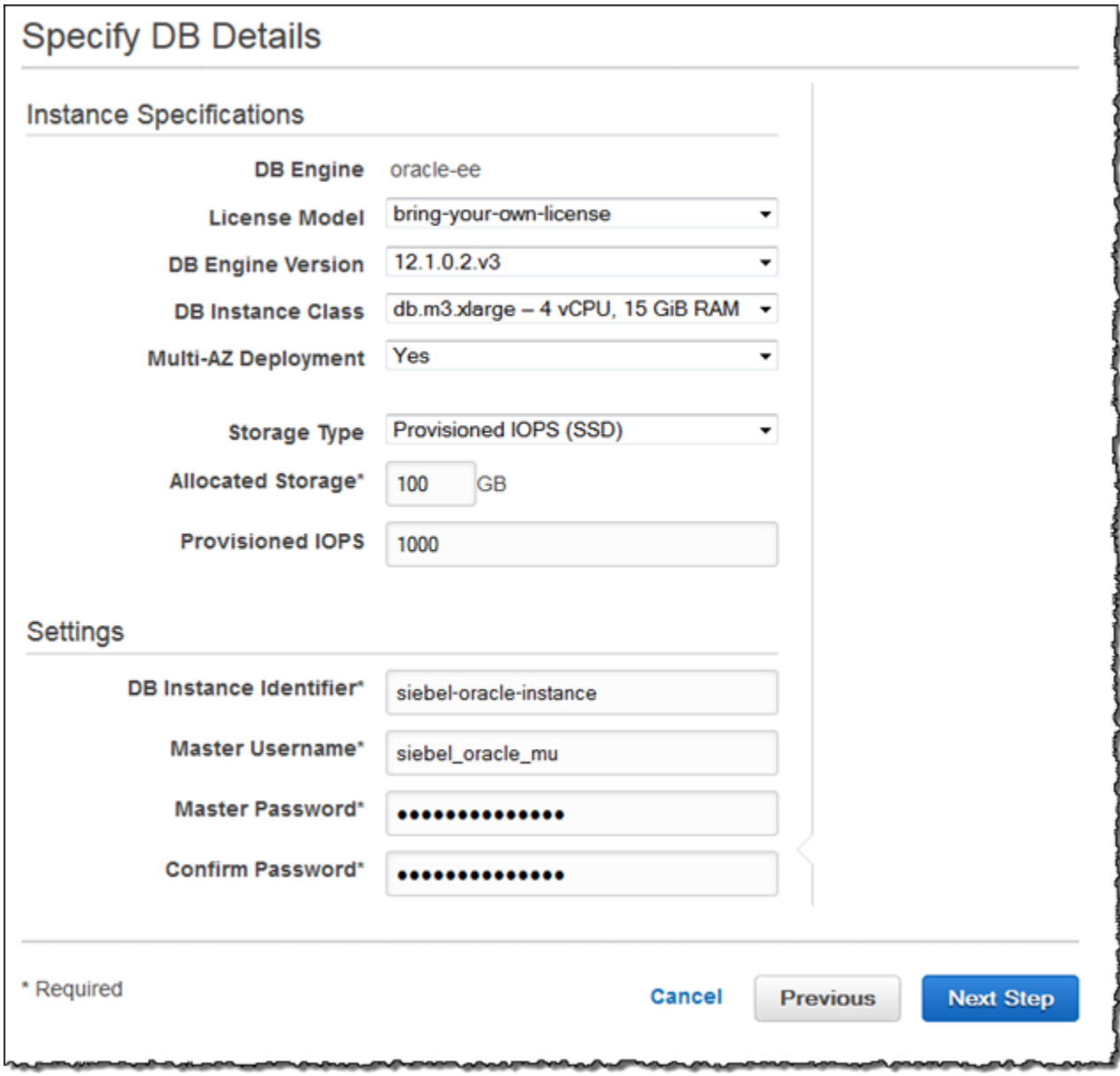


5. On the page **Step 2: Production?**, choose **Production**, and then choose **Next Step**.

   Note
   For a DB instance for development and testing you can choose **Dev/Test**.

6. On the page **Step 3: Specify DB Details**, shown following, set these values:

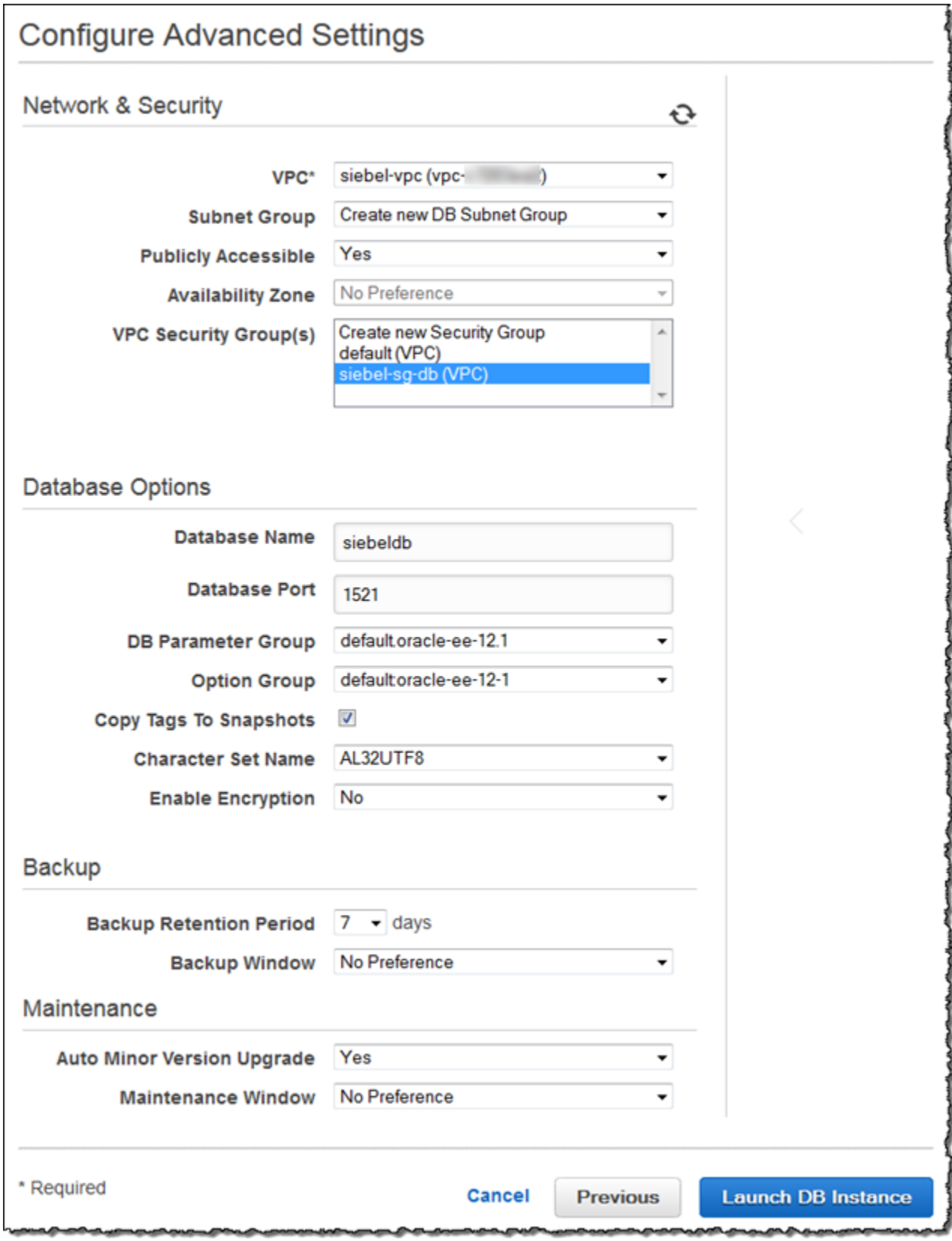
Option	Value
DB Engine	`oracle-ee`
License Model	`bring-your-own-license`
DB Engine Version	The Oracle version you want to use. Use Oracle 12c, version 12.1.0.2.0.
DB Instance Class	The DB instance class you want to use. For more information, see DB Instance Class (p. 92).
Multi-AZ Deployment	`Yes`. Multi-AZ deployment creates a standby replica of your DB instance in another Availability Zone for failover support. Multi-AZ is recommended for production workloads. For more information about multiple Availability Zones, see Regions and Availability Zones (p. 97). **Note** For development and testing, you can choose **`No`**.
Storage Type	`Provisioned IOPS (SSD)`. Provisioned IOPS (input/output operations per second) is recommended for production workloads. For more information about storage, see Storage for Amazon RDS (p. 410). **Note** For development and testing, you can choose **`General Purpose (SSD)`**.
Allocated Storage	The storage to allocate for your database. Allocate at least 20 GB of storage for your database. In some cases, allocating a higher amount of storage for your DB instance than the size of your database can improve I/O performance. For more information about storage allocation, see Amazon RDS Storage Types (p. 410) and Guidelines for Creating Oracle Database Tablespaces.
Provisioned IOPS	The amount of Provisioned IOPS to be initially allocated for the DB instance. This value must be a multiple between 3 and 10 of the storage amount for the DB instance. This value must also be an integer multiple of 1,000. **Note** For development and testing, you do not need Provisioned IOPS.
DB Instance Identifier	The name for DB instance, for example **`oracle-instance`**.
Master User Name	The master user name for the DB instance, for example **`oracle_mu`**.
Master User Password and **Confirm Password**	A password that contains from 8 to 30 printable ASCII characters (excluding /,", and @) for your master user password. Retype the password in the **Confirm Password** box.



7. Choose **Next Step**.
8. On the page **Step 4: Configure Advanced Settings**, shown following, set these values:

Option	Value
VPC	Your VPC, for example **`vpc-1`**.
Subnet Group	`Create new DB Subnet Group`
Publicly Accessible	`No`
Availability Zone	`No Preference`

Option	Value
VPC Security Group	Your VPC security group, for example **`sg-db-1`**.
Database Name	The name for your database, for example **`db1`**.
Database Port	`1521`
Parameter Group	The default parameter group.
Option Group	The default option group.
Copy Tags To Snapshots	This option, when chosen, specifies to have any DB instance tags copied to a DB snapshot when you create a snapshot. For more information, see Tagging Amazon RDS Resources (p. 129).
Character Set Name	A character set for your DB instance. The default value of **`AL32UTF8`** is for the Unicode 5.0 UTF-8 Universal character set. You can't change the character set after the DB instance is created.
Enable Encryption	**`Yes`** or **`No`**. A value of `Yes` enables encryption at rest for this DB instance. For more information, see Encrypting Amazon RDS Resources (p. 355).
Backup Retention Period	The number of days you want to retain automatic backups of your database. For most DB instances, you should set this value to **`1`** or greater.
Backup Window	Unless you have a specific time that you want to have your database backup, use the default of **`No Preference`**.
Auto Minor Version Upgrade	Select **`Yes`** to enable your DB instance to receive minor DB engine version upgrades automatically when they become available.
Maintenance Window	Select the 30 minute window in which pending modifications to your DB instance are applied. If you the time period doesn't matter, select **`No Preference`**.


Configure Advanced Settings
Network & Security
VPC*
siebel-vpc (vpc-
)
Subnet Group
Create new DB Subnet Group
Publicly Accessible
Yes
Availability Zone
No Preference
VPC Security Group(s)
Create new Security Group
default (VPC)
siebel-sg-db (VPC)
Database Options
Database Name
siebeldb
Database Port
1521
DB Parameter Group
default.oracle-ee-12.1
Option Group
default:oracle-ee-12-1
Copy Tags To Snapshots
Character Set Name
AL32UTF8
Enable Encryption
No
Backup
Backup Retention Period
7
days
Backup Window
No Preference
Maintenance
Auto Minor Version Upgrade
Yes
Maintenance Window
No Preference
* Required
Cancel
Previous
Launch DB Instance

9. Choose **Launch DB Instance**.

   On the RDS console, the new DB instance appears in the list of DB instances. The DB instance has a status of **creating** until it's ready to use. When the status of the DB instance changes to **available**, you can connect to it. Depending on the DB instance configuration, it can take several minutes for the new DB instance to become available.

Additional Amazon RDS Interfaces

In the preceding procedures, we use the AWS Management Console to perform tasks. Amazon Web Services also provides the AWS Command Line Interface (AWS CLI), and an application programming interface (API). You can use the AWS CLI or the API to automate many of the tasks for managing Amazon RDS, including tasks to manage an Oracle DB instance with Amazon RDS.

For more information, see AWS Command Line Interface Reference for Amazon RDS and Amazon Relational Database Service API Reference.

Related Topics

- Setting Up for Amazon RDS (p. 5)
- Using the Oracle Repository Creation Utility on Amazon RDS for Oracle (p. 1112)
- Installing a Siebel Database on Oracle on Amazon RDS (p. 1117)
- Scenarios for Accessing a DB Instance in a VPC (p. 392)
- Connecting to a DB Instance Running the Oracle Database Engine (p. 959)

Using AWS CloudHSM Classic to Store Amazon RDS Oracle TDE Keys

You can use AWS CloudHSM Classic with an Amazon RDS DB instance running Oracle Enterprise Edition to store keys when you use Oracle Transparent Data Encryption (TDE). AWS CloudHSM Classic is a service that provides a hardware appliance called a hardware security module (HSM) that performs secure key storage and cryptographic operations. You enable an Amazon RDS DB instance to use AWS CloudHSM Classic by setting up an HSM appliance, setting the proper permissions for cross-service access, and then setting up Amazon RDS and the DB instance that will use AWS CloudHSM Classic.

Important
Review the following availability and pricing information before you setup AWS CloudHSM Classic:

- Amazon RDS supports AWS CloudHSM Classic for Oracle DB instances in the following regions: US East (N. Virginia), US West (Oregon), Asia Pacific (Seoul), Asia Pacific (Singapore), Asia Pacific (Sydney), Asia Pacific (Tokyo), EU (Frankfurt), EU (Ireland).
- AWS CloudHSM Classic pricing:

  AWS CloudHSM Classic pricing information is available on the AWS CloudHSM Classic pricing page.
- AWS CloudHSM Classic upfront fee refund (API and CLI Tools):

  You are charged an upfront fee for each new AWS CloudHSM Classic instance that you create by using the CreateHsm API operation or the create-hsm AWS CLI command. If you accidentally provision an HSM instance that you don't need, first delete the HSM instance by using the DeleteHsm API operation or the delete-hsm AWS CLI command. You can then request a refund of the upfront fee at the AWS Support Center, by creating a new case and choosing **Account and Billing Support**.

The number of Oracle databases you can support on a single AWS CloudHSM Classic partition will depend on the rotation schedule you choose for your data. You should rotate your keys as often as your data needs require. The PCI-DSS documentation and the National Institute of Standards and Technology (NIST) provide guidance on appropriate key rotation frequency. You can maintain approximately 10,000 symmetric master keys per AWS CloudHSM Classic device. Note that after key rotation the old master key remains on the partition and is still counted against the per-partition maximum.

AWS CloudHSM Classic works with Amazon Virtual Private Cloud (Amazon VPC). An appliance is provisioned inside your VPC with a private IP address that you specify, providing simple and private network connectivity to your Amazon RDS DB instance. Your HSM appliances are dedicated exclusively to you and are isolated from other AWS customers. For more information, see Amazon Virtual Private Cloud (VPCs) and Amazon RDS (p. 390) and Creating a DB Instance in a VPC (p. 402).

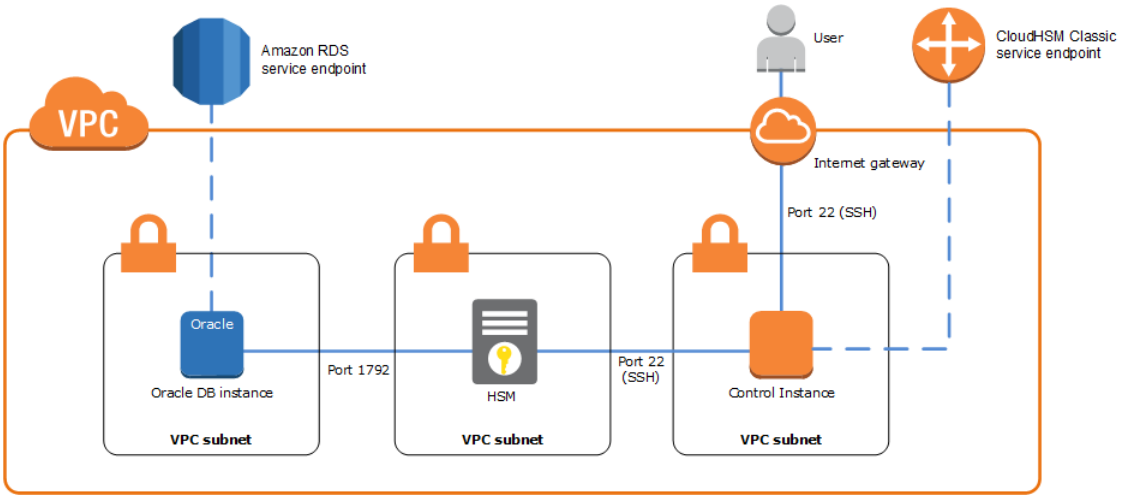
To use AWS CloudHSM Classic with an Amazon RDS Oracle DB instance, you must complete the following tasks, which are explained in detail in the following sections:

- Setting Up AWS CloudHSM Classic to Work with Amazon RDS (p. 1088)
- Setting Up Amazon RDS to Work with AWS CloudHSM Classic (p. 1091)

When you complete the entire setup, you should have the following AWS components.

- An AWS CloudHSM Classic control instance that will communicate with the HSM appliance using port 22, and the AWS CloudHSM Classic endpoint. The AWS CloudHSM Classic control instance is an Amazon EC2 instance that is in the same VPC as the HSMs and is used to manage the HSMs.

- An Amazon RDS Oracle DB instance that will communicate with the Amazon RDS service endpoint, as well as the HSM appliance using port 1792.



Topics

- Setting Up AWS CloudHSM Classic to Work with Amazon RDS (p. 1088)
- Setting Up Amazon RDS to Work with AWS CloudHSM Classic (p. 1091)
- Verifying the HSM Connection, the Oracle Keys in the HSM, and the TDE Key (p. 1098)
- Restoring Encrypted DB Instances (p. 1099)
- Managing a Multi-AZ Failover (p. 1100)

Setting Up AWS CloudHSM Classic to Work with Amazon RDS

To use AWS CloudHSM Classic with an Oracle DB instance using TDE, you must first complete the tasks required to setup AWS CloudHSM Classic. The tasks are explained in detail in the following sections.

Amazon RDS supports AWS CloudHSM Classic for Oracle DB instances in the following regions: US East (N. Virginia), US West (Oregon), Asia Pacific (Seoul), Asia Pacific (Singapore), Asia Pacific (Sydney), Asia Pacific (Tokyo), EU (Frankfurt), EU (Ireland).

Completing the AWS CloudHSM Classic Prerequisites

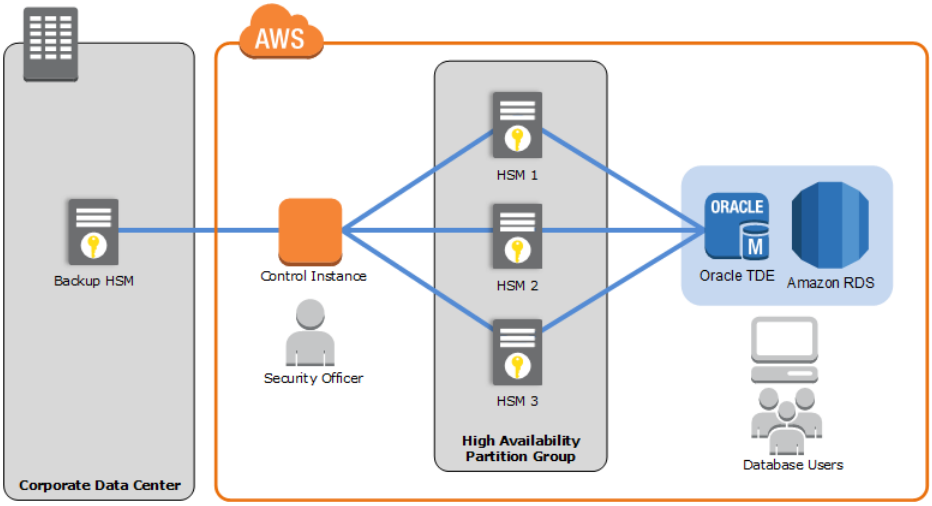
Follow the procedure in the Setting Up AWS CloudHSM section in the *AWS CloudHSM Classic User Guide* to setup an AWS CloudHSM Classic environment.

Installing the AWS CloudHSM Classic Command Line Interface Tools

Follow the instructions in the Setting Up the AWS CloudHSM CLI Tools section in the *AWS CloudHSM Classic User Guide* to install the AWS CloudHSM Classic command line interface tools on your AWS CloudHSM Classic control instance.

Configuring Your HSMs

The recommended configuration for using AWS CloudHSM Classic with Amazon RDS is to use three AWS CloudHSM Classic appliances configured into a high-availability (HA) partition group. A minimum of three HSMs are suggested for HA purposes. Even if two of your HSMs are unavailable, your keys will still be available to Amazon RDS.



Important
Initializing an HSM sets the password for the HSM security officer account (also known as the HSM administrator). Record the security officer password on your Password Worksheet (p. 1090) and do not lose it. We recommend that you print out a copy of the Password Worksheet (p. 1090), use it to record your AWS CloudHSM Classic passwords, and store it in a secure place. We also recommended that you store at least one copy of this worksheet in secure off-site storage. AWS does not have the ability to recover your key material from an HSM for which you do not have the proper HSM security officer credentials.

To provision and initialize your HSMs using the AWS CloudHSM Classic CLI tools, perform the following steps from your control instance:

1. Following the instructions in Creating Your HSMs with the CLI, provision the number of HSMs you need for your configuration. When you provision your HSMs, make note of the ARN of each HSM because you will need these to initialize your HSMs and create your high-availability partition group.

2. Following the instructions in Initializing Your HSMs, initialize each of your HSMs.

Creating Your High-Availability Partition Group

After your HSMs are initialized, create an HA partition group with the initialized HSMs. Creating an HA partition group is a three-step process. You create the HA partition group, add your HSMs to the HA partition group, and register the clients for use with the HA partition group.

To create and initialize an HA partition group

1. Following the instructions in the Create the HA Partition Group section in the *AWS CloudHSM Classic User Guide*, create your HA partition group. Save the HA partition group ARN returned from the `create-hapg` command for later use.

   Save the partition password on your Password Worksheet (p. 1090).
2. Following the instructions in Registering a Client with a High-Availability Partition Group, create, register, and assign the clients to use with your HA partition group.

Repeat this process to add additional partitions if necessary. One partition can support multiple Oracle databases.

Password Worksheet

Use the following worksheet to compile information for your AWS CloudHSM Classic appliances. Print this page and use it to record your AWS CloudHSM Classic passwords, and store it in a secure place. We also recommended that you store at least one copy of this worksheet in secure off-site storage.

Security Officer Password

This password was set when you initialized the AWS CloudHSM Classic appliance.

_____

Manager Password (Optional)

This password was optionally set with the `user password manager` command on the AWS CloudHSM Classic appliance.

_____

Partition Passwords

Partition Label	Partition Password	Cloning Domain

Setting Up Amazon RDS to Work with AWS CloudHSM Classic

To use AWS CloudHSM Classic with an Oracle DB instance using Oracle TDE, you must do the following tasks:

- Ensure that the security group associated with the Oracle DB instance allows access to the HSM port 1792.
- Create a DB subnet group that uses the same subnets as those in the VPC used by your HSMs, and then assign that DB subnet group to your Oracle DB instance.
- Set up the Amazon RDS CLI.
- Add IAM permissions for Amazon RDS to use when accessing AWS CloudHSM Classic.
- Add the **TDE_HSM** option to the option group associated with your Oracle DB instance using the Amazon RDS CLI.
- Add two new DB instance parameters to the Oracle DB instance that will use AWS CloudHSM Classic. The `tde-credential-arn` parameter is the Amazon Resource Number (ARN) of the high-availability (HA) partition group returned from the `create-hapg` command. The `tde-credential-password` is the partition password you used when you initialized the HA partition group.

The Amazon RDS CLI documentation can be found at What Is the AWS Command Line Interface? and the section Getting Set Up with the AWS Command Line Interface. General instructions on using the AWS CLI can be found at Using the AWS Command Line Interface.

The following sections show you how to set up the Amazon RDS CLI, add the required permissions for RDS to access your HSMs, create an option group with the **TDE_HSM** option, and how to create or modify a DB instance that will use the **TDE_HSM** option.

Security Group

To allow the RDS instance to communicate with the HSM, the security group ENI assigned to the HSM appliance must authorize ingress connectivity on TCP port 1792 from the DB instance. Additionally, the Network ACL associated with the HSM's ENI must permit ingress TCP port 1792 from the RDS instance, and egress connections from the HSM to the Dynamic Port range on the RDS instance. For more information about the Dynamic TCP Port range, please see the Amazon VPC documentation.

If you used the AWS CloudFormation template to create your AWS CloudHSM Classic environment, modify the security group that allows SSH and NTLS from the public subnet. If you didn't use the AWS CloudFormation template, modify the security group associated with the ENI assigned to the HSM appliance.

DB Subnet Group

The DB subnet group that you assign to your Oracle DB instance must have the same subnets as those in the VPC used by the AWS CloudHSM Classic. For information about how to create a DB subnet group, see Creating a DB Subnet Group, or you can use the AWS CLI create-db-subnet-group command to create the DB subnet group.

Setting Up the Amazon RDS CLI

The Amazon RDS CLI can be installed on a computer running the Linux or Windows operating system and that has Java version 1.6 or higher installed.

The following steps install and configure the Amazon RDS CLI:

1. Download the Amazon RDS CLI from here. Unzip the file.

2. Set the following environment variables:

```
AWS_RDS_HOME - <The directory where the deployment files were copied to>
JAVA_HOME - <Java Installation home directory>
```

   You can check that the environment variables are set correctly by running the following command for Linux or Windows should list `describe-db-instances` and other AWS CLI commands.

   For Linux, OS X, or Unix:

```
ls ${AWS_RDS_HOME}/bin
```

   For Windows:

```
dir %AWS_RDS_HOME%\bin
```

3. Add `${AWS_RDS_HOME}/bin` (Linux) or `%AWS_RDS_HOME%\bin` (Windows) to your path
4. Add the RDS service URL information for your AWS region to your shell configuration. For example:

```
export RDS_URL=https://rds.us-east-1.amazonaws.com
export SERVICE_SIG_NAME=rds
```

5. If you are on a Linux system, set execute permissions on all files in the bin directory using the following command:

```
chmod +x ${AWS_RDS_HOME}/bin/*
```

6. Provide the Amazon RDS CLI with your AWS user credentials. There are two ways you can provide credentials: AWS keys, or using X.509 certificates.

   If you are using AWS keys, do the following:

   a. Edit the credential file included in the zip file, ${AWS_RDS_HOME}/credential-file-path.template, to add your AWS credentials. If you are on a Linux system, limit permissions to the owner of the credential file:

```
$ chmod 600 <credential file>
```

   b. Alternatively, you can provide the following option with every command:

```
aws rds <AWSCLIcommand>  --aws-credential-file <credential file>
```

   c. Or you can explicitly specify credentials on the command line: --I ACCESS_KEY --S SECRET_KEY

   If you are using X.509 certifications, do the following:

   a. Save your certificate and private keys to files: e.g. my-cert.pem and my-pk.pem.

   b. Set the following environment variables:

```
EC2_CERT=<path_to_my_cert>
EC2_PRIVATE_KEY=<path_to_my_private_key>
```

   c. Or you can specify the files directly on command-line for every command:

   For Linux, OS X, or Unix:

```
aws rds <AWSCLIcommand> \
    --ec2-cert-file-path <path_to_my_cert> \
```

```
    --ec2-private-key-file-path <path_to_my_private_key>
```

For Windows:

```
aws rds <AWSCLIcommand> ^
    --ec2-cert-file-path <path_to_my_cert> ^
    --ec2-private-key-file-path <path_to_my_private_key>
```

You can test that you have set up the AWS CLI correctly by running the following commands. The first command should output the usage page for all Amazon RDS commands. The second command should output information on all DB instances for the account you are using.

```
aws rds --help
aws rds describe-db-instances --headers
```

Adding IAM Permissions for Amazon RDS to Access the AWS CloudHSM Classic

You can use a single AWS account to work with Amazon RDS and AWS CloudHSM Classic or you can use two separate accounts, one for Amazon RDS and one for AWS CloudHSM Classic. This section provides information on both processes.

Topics

Adding IAM Permissions for a Single Account for Amazon RDS to Access the AWS CloudHSM Classic API

To create a IAM role that Amazon RDS uses to access the AWS CloudHSM Classic API, use the following procedure. Amazon RDS checks for the presence of this IAM role when you create or modify a DB instance that uses AWS CloudHSM Classic.

To create a IAM role for Amazon RDS to access the AWS CloudHSM Classic API

1. Open the IAM Console at https://console.aws.amazon.com.
2. In the left navigation pane, click **Roles**.
3. Click **Create New Role**.
4. In the **Role Name** text box, type **`RDSCloudHsmAuthorization`**. Currently, you must use this name. Click **Next Step**.
5. Click **AWS Service Roles**, scroll to **Amazon RDS**, choose **Select**.
6. On the **Attach Policy** page, click **Next Step**. The correct policy is already attached to this role.
7. Review the information and then click **Create Role**.

Using Separate AWS CloudHSM Classic and Amazon RDS Accounts for Amazon RDS to Access AWS CloudHSM Classic

If you want to separately manage your AWS CloudHSM Classic and Amazon RDS resources, you can use the two services with separate accounts. To use two different accounts, you must set up each account as described in the following section.

To use two accounts, you must have the following:

- An account that is enabled for the AWS CloudHSM Classic service and that is the owner of your hardware security module (HSM) devices. Generally, this account is your AWS CloudHSM Classic account, with a customer ID of HSM_ACCOUNT_ID.
- An account for Amazon RDS that you can use to create and manage a DB instance that uses Oracle TDE. Generally, this account is your DB account, with a customer ID DB_ACCOUNT_ID.

To add DB account permission to access AWS CloudHSM Classic resources under the AWS CloudHSM Classic account

1. Open the IAM Console at https://console.aws.amazon.com/.
2. Log in using your DB account.
3. In the left navigation pane, choose **Roles**.
4. Choose **Create New Role**.
5. For **Role Name**, type **`RDSCloudHsmAssumeAuthorization`**. Currently, you must use this role name for this approach to work. Choose **Next Step**.
6. Choose **AWS Service Roles**, scroll to **Amazon RDS**, choose **Select**.
7. On the **Attach Policy** page, do not attach a policy. Choose **Next Step**.
8. Review the information, and then choose **Create Role**.
9. For Roles, choose the **RDSCloudHsmAssumeAuthorization** role.
10. For Permissions, choose **Inline Policies**. Text appears that provides a link; click **click here**.
11. On the **Set Permissions** page, choose **Custom Policy**, then choose **Select**.
12. For **Policy Name**, type **`AssumeRole`**.
13. For **Policy Document**, type the following policy information:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Resource": "*"
        }
    ]
}
```

14. Choose **Apply Policy**, and then log out of your DB account.

To revise the AWS CloudHSM Classic account to trust permission to access AWS CloudHSM Classic resources under the AWS CloudHSM Classic account

1. Open the IAM Console at https://console.aws.amazon.com/.
2. Log in using your AWS CloudHSM Classic account.
3. In the left navigation pane, choose **Roles**.
4. Choose the **RDSCloudHsmAuthorization** role. This role is the one created for a single account CloudHSM-RDS.
5. Choose **Edit Trust Relationship**.
6. Add your DB account as a trusted account. The policy document should look like the following, with your DB account replacing the *<DB_ACCOUNT_ID>* placeholder:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "rds.amazonaws.com",
        "AWS":[   "arn:aws:iam::$<DB_ACCOUNT_ID>$:role/RDSCloudHsmAssumeAuthorization"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

7. Choose **Update Trust Policy**.

Creating an Amazon VPC Using the DB Account That Can Connect to Your HSM

HSM appliances are provisioned into an HSM-specific Amazon VPC. By default, only hosts inside the HSM VPC can see the HSM devices. Thus, all DB instances need to be created inside the HSM VPC or in a VPC that can be linked to the HSM VPC using VPC peering.

To use AWS CloudHSM Classic with an Amazon RDS DB instance in a different VPC (which you create under your DB account, as described in Creating a DB Instance in a VPC (p. 402)), you set up VPC peering from the VPC containing the DB instance to the HSM-specific VPC that contains your HSM appliances.

To set up VPC peering between the two VPCs

1. Use an existing VPC created under your DB account, or create a new VPC using your DB account. The VPC should not have any CIDR ranges that overlap with the CIDR ranges of the HSM-specific VPC.
2. Perform VPC peering between the DB VPC and the HSM VPC. For instructions, go to VPC Peering in the *Amazon Virtual Private Cloud User Guide.*
3. Ensure that the VPC routing table is correctly associated with the VPC subnet and the VPC security group on the HSM network interface.

Note that you must configure both VPCs' routing tables so that network traffic goes to the correct VPC (from the DB VPC to the HSM VPC, and from the HSM VPC to the DB VPC). The two VPCs don't need to share the same security group, though the security groups must not prevent network traffic between the two VPCs.

Creating an Option Group with the TDE_HSM Option

The **TDE_HSM** option can be added to an existing option group just like other Oracle options, or you can create a new option group and add the **TDE_HSM** option. The following Amazon RDS CLI example creates an option group for Oracle Enterprise Edition 11.2 named *tdehsm-option-group*.

For Linux, OS X, or Unix:

```
aws rds create-option-group \
    --option-group-name tdehsm-option-group \
    --option-group-description "Option Group with TDE_HSM" \
    --engine-name oracle-ee \
    --major-engine-version 11.2
```

For Windows:

```
aws rds create-option-group ^
    --option-group-name tdehsm-option-group ^
    --option-group-description "Option Group with TDE_HSM" ^
    --engine-name oracle-ee ^
    --major-engine-version 11.2
```

The output of the command should appear similar to the following example:

```
OPTIONGROUP  tdehsm-option-group  oracle-ee  11.2  Option Group with TDE_HSM  n
```

Once the option group has been created, you can use the following command to add the **TDE_HSM** option to the option group.

For Linux, OS X, or Unix:

```
aws rds add-option-to-option-group \
    --option-group-name tdehsm-option-group \
    --option-name TDE_HSM
```

For Windows:

```
aws rds add-option-to-option-group ^
    --option-group-name tdehsm-option-group ^
    --option-name TDE_HSM
```

The output of the command should appear similar to the following example:

```
OPTION  TDE_HSM  y  n  Oracle Advanced Security - TDE with HSM
```

Adding the AWS CloudHSM Classic Parameters to an Oracle DB Instance

An Oracle Enterprise Edition DB instance that uses AWS CloudHSM Classic must have two new parameters added to the DB instance. The `tde-credential-arn` and `tde-credential-password` parameters are new parameters you must include when creating a new DB instance or when modifying an existing DB instance to use AWS CloudHSM Classic.

Creating a New Oracle DB Instance with Additional Parameters for AWS CloudHSM Classic

When creating a new DB instance to use with AWS CloudHSM Classic, there are several requirements:

- You must include the option group that contains the **TDE_HSM** option
- You must provide values for the `tde-credential-arn` and `tde-credential-password` parameters. The `tde-credential-arn` parameter value is the Amazon Resource Number (ARN) of the HA partition group returned from the `create-hapg` command. You can also retrieve the ARNs of all of your high-availability partition groups with the `list-hapgs` command.

  The `tde-credential-password` is the partition password you used when you initialized the HA partition group.
- The IAM Role that provides cross-service access must be created.
- You must create an Oracle Enterprise Edition DB instance.

The following command creates a new Oracle Enterprise Edition DB instance called *HsmInstance-test01* that includes the two parameters that provide AWS CloudHSM Classic access and uses an option group called *tdehsm-option-group*.

For Linux, OS X, or Unix:

```
aws rds create-db-instance \
    --db-instance-identifier HsmInstance-test01 \
    --db-instance-class <instance class> \
    --engine oracle-ee \
    --tde-credential-arn <ha partition group ARN> \
    --tde-credential-password <partition password> \
    --db-name <Oracle DB instance name> \
    --db-subnet-group-name <subnet group name> \
    --connection-timeout <connection timeout value> \
    --master-user-password <master user password> \
    --master-username <master user name> \
    --allocated-storage <storage value> \
    --option-group-name <TDE option group>
```

For Windows:

```
aws rds create-db-instance ^
    --db-instance-identifier HsmInstance-test01 ^
    --db-instance-class <instance class> ^
    --engine oracle-ee ^
    --tde-credential-arn <ha partition group ARN> ^
    --tde-credential-password <partition password> ^
    --db-name <Oracle DB instance name> ^
    --db-subnet-group-name <subnet group name> ^
    --connection-timeout <connection timeout value> ^
    --master-user-password <master user password> ^
    --master-username <master user name> ^
    --allocated-storage <storage value> ^
    --option-group-name <TDE option group>
```

The output of the command should appear similar to the following example:

```
DBINSTANCE  hsminstance-test01  db.m1.medium  oracle-ee  40  fooooo  creating
1  ****  n  11.2.0.4.v7  bring-your-own-license  AL52UTF8  n
      VPCSECGROUP  sg-922xvc2fd  active
SUBNETGROUP  dev-test  test group  Complete  vpc-3facfe54
      SUBNET  subnet-1fd6a337  us-east-1e  Active
      SUBNET  subnet-28aeff43  us-east-1c  Active
      SUBNET  subnet-5daeff36  us-east-1b  Active
      SUBNET  subnet-2caeff47  us-east-1d  Active
      PARAMGRP  default.oracle-ee-11.2  in-sync
      OPTIONGROUP  tdehsm-option-group  pending-apply
```

Modifying an Existing DB Instance to Add Parameters for AWS CloudHSM Classic

The following command modifies an existing Oracle Enterprise Edition DB instance and adds the `tde-credential-arn` and `tde-credential-password` parameters. Note that you must also include in the command the option group that contains the **TDE_HSM** option.

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier hsm03 \
    --tde-credential-arn <ha partition group ARN> \
    --tde-credential-password <partition password> \
    --option-group <tde hsm option group> \
    --apply-immediately
```

For Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier hsm03 ^
    --tde-credential-arn <ha partition group ARN> ^
    --tde-credential-password <partition password> ^
    --option-group <tde hsm option group> ^
    --apply-immediately
```

The output of the command should appear similar to the following example:

```
DBINSTANCE  hsm03  2014-04-03T18:48:53.106Z  db.m1.medium  oracle-ee  40  fooooo  available

hsm03.c1iibpgwvdfo.us-east-1.rds.amazonaws.com  1521  us-east-1e  1
n  11.2.0.4.v7  bring-your-own-license  AL32UTF8  n
      VPCSECGROUP  sg-922dc2fd  active
SUBNETGROUP  dev-test  test group  Complete  vpc-3faffe54
      SUBNET  subnet-1fd6a337  us-east-1e  Active
      SUBNET  subnet-28aeff43  us-east-1c  Active
      SUBNET  subnet-5daeff36  us-east-1b  Active
      SUBNET  subnet-2caeff47  us-east-1d  Active
      PARAMGRP  default.oracle-ee-11.2  in-sync
      OPTIONGROUP  tdehsm-option-group              pending-apply
      OPTIONGROUP  default:oracle-ee-11-2  pending-removal
```

Verifying the HSM Connection, the Oracle Keys in the HSM, and the TDE Key

Once you have completed all the set up steps, you can verify the HSM is working properly for TDE key storage. Connect to the Oracle DB instance using a SQL utility such as *sqlplus* on a client computer or from the Amazon EC2 control instance if it has *sqlplus* installed. For more information on connecting to an Oracle DB instance, see Connecting to a DB Instance Running the Oracle Database Engine.

> **Note**
> Before you continue, you must verify that the option group that you created for your Oracle instance returns a status of `in-sync`. You can verify this passing the DB instance identifier to the `describe-db-instances` command.

Verifying the HSM Connection

You can verify the connection between an Oracle DB instance and the HSM. Connect to the Oracle DB instance and use the following command:

```
$ select * from v$encryption_wallet;
```

If the HSM connection is working, the command should return a status of *OPEN*. The output of the command is similar to the following example:

```
WRL_TYPE
--------------------
WRL_PARAMETER
--------------------
STATUS
------------------
HSM
OPEN

1 row selected.
```

Verifying the Oracle Keys in the HSM

Once Amazon RDS starts and Oracle is running, Oracle creates two master keys on the HSM. Do the following steps to confirm the existence of the master keys in the HSM. You can run these commands from the prompt on the Amazon EC2 control instance or from the Amazon RDS Oracle DB instance.

1. Use SSH to connect to the HSM appliance. The following command

```
$ ssh manager@10.0.203.58
```

2. Log in to the HSM as the HSM manager

```
$ hsm login
```

3. Once you have successfully logged in, the Luna Shell prompt appears ([hostname]lunash:>). Display the contents of the HSM partition that corresponds to the Oracle DB instance using TDE. Look for two symmetric key objects that begin with "ORACLE.TDE.HSM."

```
lunash:>part showContents -par <hapg_label> -password <partition_password>
```

   The following output is an example of the information returned from the command:

```
Partition Name:  hapg_label
Partition SN:    154749011
Storage (Bytes): Total=102701, Used=348, Free=102353
Number objects:  2

Object Label:  ORACLE.TDE.HSM.MK.0699468E1DC88E4F27BF426176B94D4907
Object Type:   Symmetric Key

Object Label:  ORACLE.TSE.HSM.MK.0784B1918AB6C19483189B2296FAE261C70203
Object Type:   Symmetric Key

Command Result : 0 (Success)
```

Verifying the TDE Key

The final step to verifying that the TDE key is correctly stored in the HSM is to create an encrypted tablespace. The following commands creates an encrypted tablespace and shows that it is encrypted.

```
SQL> create tablespace encrypted_ts
datafile size 50M encryption using 'AES128'
default storage (encrypt)
/
SQL> select tablespace_Name, encrypted from dba_tablespaces where encrypted='YES'
```

The following sample output shows that the tablespace was encrypted:

```
TABLESPACE_NAME                ENC
------------------------------ ---
ENCRYPTED_TS                   YES
```

Restoring Encrypted DB Instances

To restore an encrypted Oracle DB instance, you can use your existing AWS CloudHSM Classic HA partition group or create a new HA partition group and copy the contents from the original partition

group to the new partition group. Please update the SafeNet client on your HSM control instance if you would like to use your existing HA partition group. Then use the `restore-db-instance-from-db-snapshot` command to restore the DB instance.

To restore the instance, perform the following procedure:

1. On your AWS CloudHSM Classic control instance, create a new HA partition group as shown in Creating Your High-Availability Partition Group (p. 1089). When you create the new HA partition group, you must specify the same partition password as the original HA partition group. Make a note of the ARN of the new HA partition group, which you will need in the next two steps.
2. On your AWS CloudHSM Classic control instance, clone the contents of the existing HA partition group to the new HA partition group with the `clone-hapg` command.

   For Linux, OS X, or Unix:

   ```
   cloudhsm clone-hapg --conf_file ~/cloudhsm.conf \
      --src-hapg-arn <src_arn> \
      --dest-hapg-arn <dest_arn> \
      --client-arn <client_arn> \
      --partition-password <partition_password>
   ```

   For Windows:

   ```
   cloudhsm clone-hapg --conf_file ~/cloudhsm.conf ^
      --src-hapg-arn <src_arn> ^
      --dest-hapg-arn <dest_arn> ^
      --client-arn <client_arn> ^
      --partition-password <partition_password>
   ```

   The parameters are as follows:

   <src_arn>

   The identifier of the existing HA partition group.

   <dest_arn>

   The identifier of the new HA partition group created in the previous step.

   <client_arn>

   The identifier of the HSM client.

   <partition_password>

   The password for the member partitions. Both HA partition groups must have the same partition password.
3. To restore the DB instance, use the AWS CLI restore-db-instance-from-db-snapshot command. For the parameter `tde-credential-arn`, specify the ARN of the new HA partition group in. For the parameter `tde-credential-password`, specify the partition password for the HA partition group.

Managing a Multi-AZ Failover

You do not need to set up a AWS CloudHSM Classic HA partition group for your standby DB instance if you are using a Multi-AZ deployment. In fact, the details of a failover are handled automatically for you. During a failover, the standby instance becomes the new primary instance and the HSM continues to work with the new primary instance.

Using Oracle GoldenGate with Amazon RDS

Oracle GoldenGate is used to collect, replicate, and manage transactional data between databases. It is a log-based change data capture (CDC) and replication software package used with Oracle databases for online transaction processing (OLTP) systems. GoldenGate creates trail files that contain the most recent changed data from the source database and then pushes these files to the target database. You can use Oracle GoldenGate with Amazon RDS for Active-Active database replication, zero-downtime migration and upgrades, disaster recovery, data protection, and in-region and cross-region replication.

The following are important points to know when working with Oracle GoldenGate on Amazon RDS:

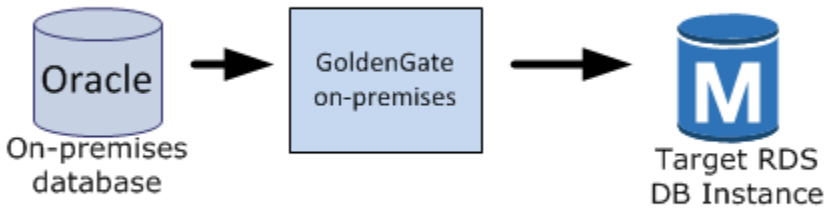
- You are responsible for setting up and managing GoldenGate on Amazon RDS.
- Amazon RDS supports Oracle GoldenGate under the bring-your-own-license model in all AWS regions. For more information, see Oracle Licensing (p. 933).
- Amazon RDS supports Oracle GoldenGate for Oracle Database Standard Edition Two (SE2), Standard Edition One (SE1), Standard Edition (SE), and Enterprise Edition (EE).
- Amazon RDS supports Oracle GoldenGate for database version 11.2.0.4 or 12.1.0.2.
- Amazon RDS supports Oracle GoldenGate version 11.2.1 and 12.1.x.
- Amazon RDS supports migration and replication across Oracle databases using Oracle GoldenGate. We do not support nor prevent customers from migrating or replicating across heterogeneous databases.
- You can use GoldenGate on Amazon RDS Oracle DB instances that use Oracle Transparent Data Encryption (TDE). Since trail files save data unencrypted by default, you should encrypt the pipeline between the source instance, the GoldenGate hub, and the target instance using `sqlnet.ora` encryption. For more information on `sqlnet.ora` encryption, see the Oracle documentation.
- Oracle GoldenGate DDL is not currently supported.

Overview

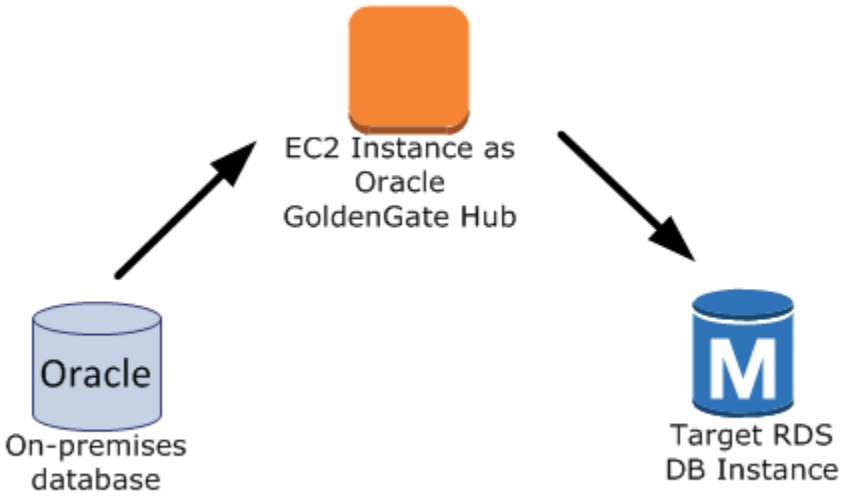
The Oracle GoldenGate architecture for use with Amazon RDS consists of three decoupled modules. The source database can be either an on-premises Oracle database, an Oracle database on an EC2 instance, or an Oracle database on an Amazon RDS DB instance. Next, the GoldenGate hub, which moves transaction information from the source database to the target database, can be either an EC2 instance with Oracle Database 11.2.0.4 and with GoldenGate 11.2.1 installed, or an on-premises Oracle installation. You can have more than one EC2 hub, and we recommend that you use two hubs if you are using GoldenGate for cross-region replication. Finally, the target database can be either on an Amazon RDS DB instance, on an EC2 instance, or on an on-premises location.

Oracle GoldenGate on Amazon RDS supports the following common scenarios:

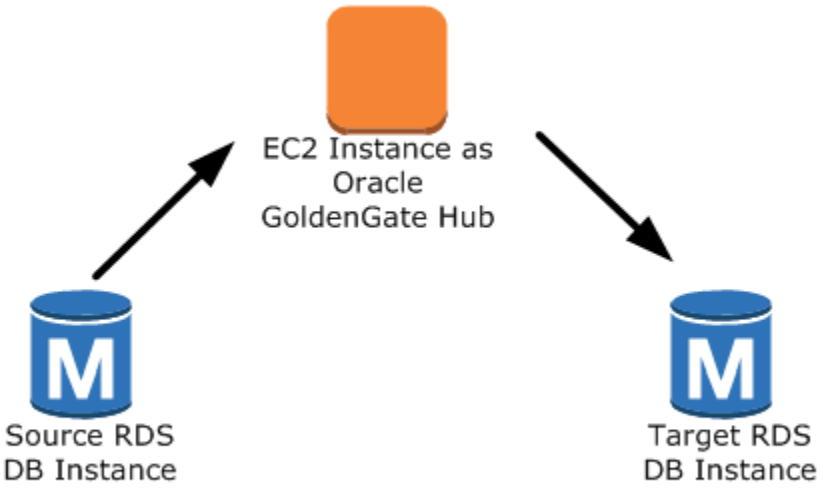
Scenario 1: An on-premises Oracle source database and on-premises Oracle GoldenGate hub, that provides data to a target Amazon RDS DB instance.



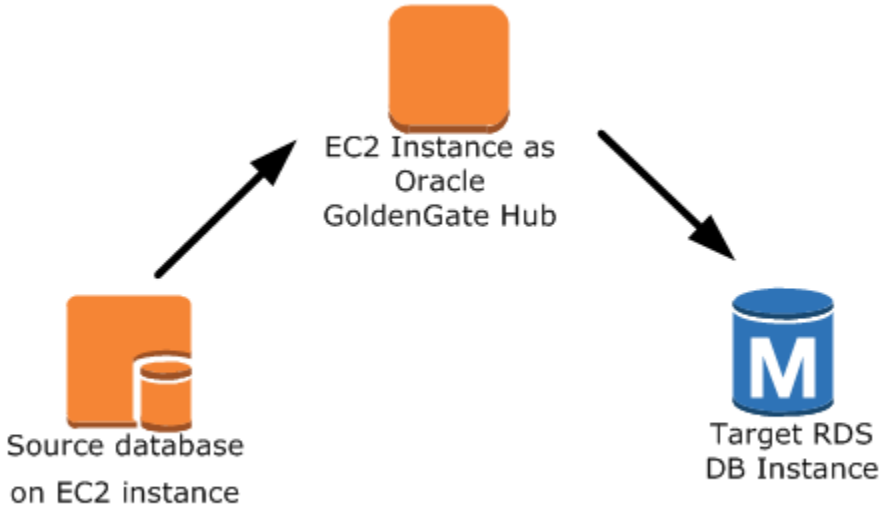
Scenario 2: An on-premises Oracle database that acts as the source database, connected to an Amazon EC2 instance hub that provides data to a target Amazon RDS DB instance.



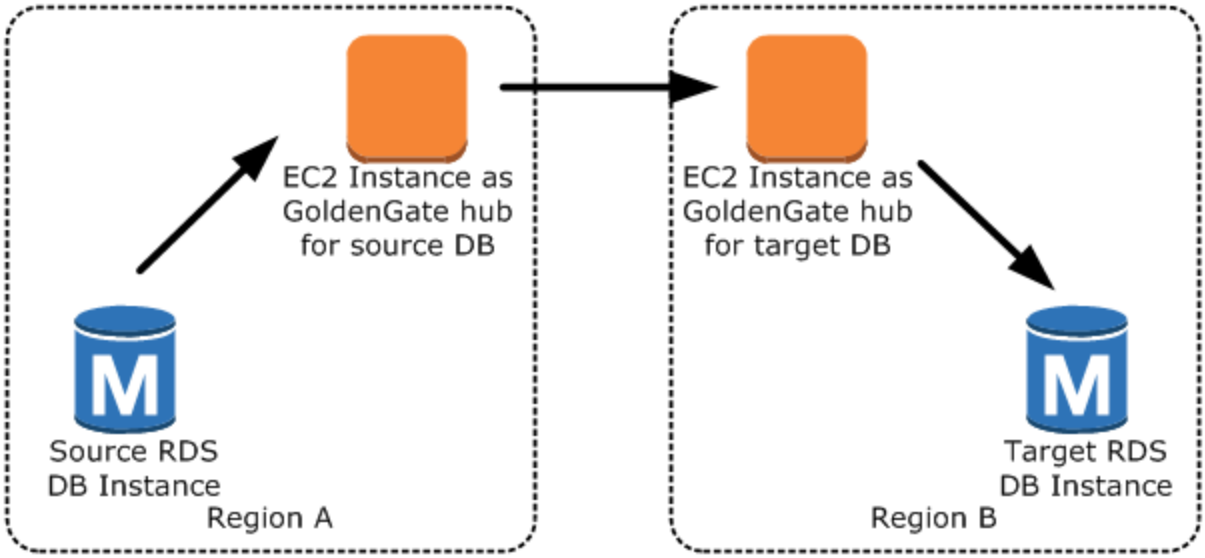
Scenario 3: An Oracle database on an Amazon RDS DB instance that acts as the source database, connected to an Amazon EC2 instance hub that provides data to a target Amazon RDS DB instance.



Scenario 4: An Oracle database on an Amazon EC2 instance that acts as the source database, connected to an Amazon EC2 instance hub that provides data to a target Amazon RDS DB instance.



Scenario 5: An Oracle database on an Amazon RDS DB instance connected to an Amazon EC2 instance hub in the same region, connected to an Amazon EC2 instance hub in a different region that provides data to the target Amazon RDS DB instance in the same region as the second EC2 instance hub.



Note
Any issues that impact running Oracle GoldenGate on an on-premises environment will also impact running GoldenGate on AWS. We strongly recommend that you monitor the GoldenGate hub to ensure that `EXTRACT` and `REPLICAT` are resumed if a failover occurs. Since the GoldenGate hub is run on an Amazon EC2 instance, Amazon RDS does not manage the GoldenGate hub and cannot ensure that it is running.

You can use GoldenGate using Amazon RDS to upgrade to major versions of Oracle. For example, you can use GoldenGate using Amazon RDS to upgrade from an Oracle version 8 on-premises database to an Oracle database running version 11.2.0.4 on an Amazon RDS DB instance.

To set up Oracle GoldenGate using Amazon RDS, you configure the hub on the EC2 instance, and then configure the source and target databases. The following steps show how to set up GoldenGate for use with Amazon RDS. Each step is explained in detail in the following sections:

- Setting Up an Oracle GoldenGate Hub on EC2 (p. 1104)
- Setting Up a Source Database for Use with GoldenGate on Amazon RDS (p. 1105)
- Setting Up a Target Database for Use with GoldenGate on Amazon RDS (p. 1107)
- Working with the EXTRACT and REPLICAT Utilities of Oracle GoldenGate (p. 1108)

Setting Up an Oracle GoldenGate Hub on EC2

There are several steps to creating an Oracle GoldenGate hub on an Amazon EC2 instance. First, you create an EC2 instance with a full installation of Oracle DBMS 11g version 11.2.0.4. The EC2 instance must also have Oracle GoldenGate 11.2.1 software installed, and you must have Oracle patch 13328193 installed. For more information about installing GoldenGate, see the Oracle documentation.

Since the EC2 instance that is serving as the GoldenGate hub stores and processes the transaction information from the source database into trail files, you must have enough allocated storage to store the trail files. You must also ensure that the EC2 instance has enough processing power to manage the amount of data being processed and enough memory to store the transaction information before it is written to the trail file.

The following tasks set up a GoldenGate hub on an Amazon EC2 instance; each task is explained in detail in this section. The tasks include:

- Add an alias to the tnsname.ora file
- Create the GoldenGate subdirectories
- Update the GLOBALS parameter file
- Configure the mgr.prm file and start the *manager*

Add the following entry to the tnsname.ora file to create an alias. For more information on the tnsname.ora file, see the Oracle documentation.

```
$ cat /example/config/tnsnames.ora
TEST=
(DESCRIPTION=
 (ENABLE=BROKEN)
 (ADDRESS_LIST=
   (ADDRESS=(PROTOCOL=TCP)(HOST=goldengate-test.abcdef12345.us-west-2.rds.amazonaws.com)
(PORT=8200))
 )
 (CONNECT_DATA=
   (SID=ORCL)
 )
)
```

Next, create subdirectories in the GoldenGate directory using the EC2 command line shell and *ggsci*, the GoldenGate command interpreter. The subdirectories are created under the gg directory and include directories for parameter, report, and checkpoint files.

```
prompt$ cd /gg
prompt$ ./ggsci
  GGSCI> CREATE SUBDIRS
```

Create a GLOBALS parameter file using the EC2 command line shell. Parameters that affect all GoldenGate processes are defined in the GLOBALS parameter file. The following example creates the necessary file:

```
prompt$ cd $GGHOME
prompt$ vi GLOBALS
CheckpointTable oggadm1.oggchkpt
```

The last step in setting up and configuring the GoldenGate hub is to configure the *manager*. Add the following lines to the *mgr.prm* file, then start the *manager* using *ggsci*:

```
PORT 8199
PurgeOldExtracts ./dirdat/*, UseCheckpoints, MINKEEPDAYS 5
```

```
GGSCI>  start mgr
```

Once you have completed these steps, the GoldenGate hub is ready for use. Next, you set up the source and target databases.

Setting Up a Source Database for Use with GoldenGate on Amazon RDS

When your source database is running version 11.2.0.4 or later, there are three tasks you need to accomplish to set up a source database for use with GoldenGate:

- Set the `compatible` parameter to 11.2.0.4 or later.
- Set the `ENABLE_GOLDENGATE_REPLICATION` parameter to *True*. This parameter turns on supplemental logging for the source database. If your source database is on an Amazon RDS DB instance, you must have a parameter group assigned to the DB instance with the `ENABLE_GOLDENGATE_REPLICATION` parameter set to *true*. For more information about the `ENABLE_GOLDENGATE_REPLICATION` parameter, see the Oracle documentation.
- Set the retention period for archived redo logs for the GoldenGate source database.
- Create a GoldenGate user account on the source database.
- Grant the necessary privileges to the GoldenGate user.

The source database must have the `compatible` parameter set to 11.2.0.4 or later. If you are using an Oracle database on an Amazon RDS DB instance as the source database, you must have a parameter group with the `compatible` parameter set to 11.2.0.4 or later associated with the DB instance. If you change the `compatible` parameter in a parameter group associated with the DB instance, the change requires an instance reboot. You can use the following Amazon RDS CLI commands to create a new parameter group and set the `compatible` parameter. Note that you must associate the new parameter group with the source DB instance:

For Linux, OS X, or Unix:

```
aws rds create-db-parameter-group \
    --db-parameter-group-name example-goldengate \
    --description "Parameters to allow GoldenGate" \
    --db-parameter-group-family oracle-ee-11.2

aws rds modify-db-parameter-group \
    --db-parameter-group-name example-goldengate \
    --parameters "ParameterName=compatible, ParameterValue=11.2.0.4, ApplyMethod=pending-
reboot"
```

```
aws rds modify-db-instance \
    --db-instance-identifier example-test \
    --db-parameter-group-name example-goldengate \
    --apply-immediately

aws rds reboot-db-instance \
    --db-instance-identifier example-test
```

For Windows:

```
aws rds create-db-parameter-group ^
    --db-parameter-group-name example-goldengate ^
    --description "Parameters to allow GoldenGate" ^
    --db-parameter-group-family oracle-ee-11.2

aws rds modify-db-parameter-group ^
    --db-parameter-group-name example-goldengate ^
    --parameters "ParameterName=compatible, ParameterValue=11.2.0.4, ApplyMethod=pending-
reboot"

aws rds modify-db-instance ^
    --db-instance-identifier example-test ^
    --db-parameter-group-name example-goldengate ^
    --apply-immediately

aws rds reboot-db-instance ^
    --db-instance-identifier example-test
```

Always retain the parameter group with the `compatible` parameter. If you restore an instance from a DB snapshot, you must modify the restored instance to use the parameter group that has a matching or greater `compatible` parameter value. This should be done as soon as possible after the restore action and will require a reboot of the instance.

The `ENABLE_GOLDENGATE_REPLICATION` parameter, when set to *True*, turns on supplemental logging for the source database and configures the required GoldenGate permissions. If your source database is on an Amazon RDS DB instance, you must have a parameter group assigned to the DB instance with the `ENABLE_GOLDENGATE_REPLICATION` parameter set to *true*. For more information about the `ENABLE_GOLDENGATE_REPLICATION` parameter, see the Oracle documentation.

The source database must also retain archived redo logs. For example, the following command sets the retention period for archived redo logs to 24 hours:

```
exec rdsadmin.rdsadmin_util.set_configuration('archivelog retention hours',24);
```

The duration for log retention is specified in hours. The duration should exceed any potential downtime of the source instance or any potential communication/networking issues to the source instance, so that Oracle GoldenGate can recover logs from the source instance as needed. The absolute minimum value required is one (1) hour of logs retained.

A log retention setting that is too small will result in the following message:

```
ERROR OGG-02028  Failed to attach to logmining server OGG$<extract_name> error 26927 -
 ORA-26927: altering an outbound server with a remote capture is not allowed.
```

Because these logs are retained on your DB instance, you need to ensure that you have enough storage available on your instance to accommodate the log files. To see how much space you have used in the last "X" hours, use the following query, replacing "X" with the number of hours.

```
select sum(blocks * block_size) bytes from v$archived_log
```

```
   where next_time>=sysdate-X/24 and dest_id=1;
```

GoldenGate runs as a database user and must have the appropriate database privileges to access the redo and archive logs for the source database, so you must create a GoldenGate user account on the source database. For more information about the permissions for a GoldenGate user account, see the sections 4, section 4.4, and table 4.1 in the Oracle documentation.

The following statements create a user account named *oggadm1*:

```
CREATE tablespace administrator;
CREATE USER oggadm1  IDENTIFIED BY "XXXXXX"
   default tablespace ADMINISTRATOR temporary tablespace TEMP;
```

Finally, grant the necessary privileges to the GoldenGate user account. The following statements grant privileges to a user named *oggadm1*:

```
grant create session, alter session to oggadm1;
grant resource to oggadm1;
grant select any dictionary to oggadm1;
grant flashback any table to oggadm1;
grant select any table to oggadm1;
grant select_catalog_role to <RDS instance master username> with admin option;
exec RDSADMIN.RDSADMIN_UTIL.GRANT_SYS_OBJECT ('DBA_CLUSTERS', 'OGGADM1');
grant execute on dbms_flashback to oggadm1;
grant select on SYS.v_$database to oggadm1;
grant alter any table to oggadm1;

EXEC DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE (grantee=>'OGGADM1',
   privilege_type=>'capture',
   grant_select_privileges=>true,
   do_grants=>TRUE);
```

Setting Up a Target Database for Use with GoldenGate on Amazon RDS

The following tasks set up a target DB instance for use with GoldenGate:

- Set the `compatible` parameter to 11.2.0.4 or later
- Set the ENABLE_GOLDENGATE_REPLICATION parameter to *True*. If your target database is on an Amazon RDS DB instance, you must have a parameter group assigned to the DB instance with the ENABLE_GOLDENGATE_REPLICATION parameter set to *true*. For more information about the ENABLE_GOLDENGATE_REPLICATION parameter, see the Oracle documentation .
- Create and manage a GoldenGate user account on the target database
- Grant the necessary privileges to the GoldenGate user

GoldenGate runs as a database user and must have the appropriate database privileges, so you must create a GoldenGate user account on the target database. The following statements create a user named *oggadm1*:

```
create tablespace administrator;
create tablespace administrator_idx;
CREATE USER oggadm1  IDENTIFIED BY "XXXXXX"
   default tablespace ADMINISTRATOR
   temporary tablespace TEMP;
alter user oggadm1 quota unlimited on ADMINISTRATOR;
alter user oggadm1 quota unlimited on ADMINISTRATOR_IDX;
```

Finally, grant the necessary privileges to the GoldenGate user account. The following statements grant privileges to a user named *oggadm1*:

```
grant create session        to oggadm1;
grant alter session         to oggadm1;
grant CREATE CLUSTER        to oggadm1;
grant CREATE INDEXTYPE      to oggadm1;
grant CREATE OPERATOR       to oggadm1;
grant CREATE PROCEDURE      to oggadm1;
grant CREATE SEQUENCE       to oggadm1;
grant CREATE TABLE          to oggadm1;
grant CREATE TRIGGER        to oggadm1;
grant CREATE TYPE           to oggadm1;
grant select any dictionary to oggadm1;
grant create any table      to oggadm1;
grant alter any table       to oggadm1;
grant lock any table        to oggadm1;
grant select any table      to oggadm1;
grant insert any table      to oggadm1;
grant update any table      to oggadm1;
grant delete any table      to oggadm1;

EXEC DBMS_GOLDENGATE_AUTH.GRANT_ADMIN_PRIVILEGE
   (grantee=>'OGGADM1',privilege_type=>'apply',
   grant_select_privileges=>true, do_grants=>TRUE);
```

Working with the EXTRACT and REPLICAT Utilities of Oracle GoldenGate

The Oracle GoldenGate utilities `EXTRACT` and `REPLICAT` work together to keep the source and target databases in sync via incremental transaction replication using trail files. All changes that occur on the source database are automatically detected by `EXTRACT`, then formatted and transferred to trail files on the GoldenGate on-premises or EC2-instance hub. After initial load is completed, the data is read from these files and replicated to the target database by the `REPLICAT` utility.

Running Oracle GoldenGate's `EXTRACT` Utility

The `EXTRACT` utility retrieves, converts, and outputs data from the source database to trail files. `EXTRACT` queues transaction details to memory or to temporary disk storage. When the transaction is committed to the source database, `EXTRACT` flushes all of the transaction details to a trail file for routing to the GoldenGate on-premises or EC2-instance hub and then to the target database.

The following tasks enable and start the `EXTRACT` utility:

- Configure the `EXTRACT` parameter file on the GoldenGate hub (on-premises or EC2 instance). The following listing shows an example `EXTRACT` parameter file.

```
EXTRACT EABC
SETENV (ORACLE_SID=ORCL)
SETENV (NLSLANG=AL32UTF8)

USERID oggadm1@TEST, PASSWORD XXXXXX
EXTTRAIL /path/to/goldengate/dirdat/ab

IGNOREREPLICATES
GETAPPLOPS
TRANLOGOPTIONS EXCLUDEUSER OGGADM1

TABLE EXAMPLE.TABLE;
```

- On the GoldenGate hub, launch the GoldenGate command line interface (*ggsci*). Log into the source database. The following example shows the format for logging in:

```
 dblogin userid <user>@<db tnsname>
```

- Add a checkpoint table for the database:

```
add checkpointtable
```

- Add transdata to turn on supplemental logging for the database table:

```
add trandata <user>.<table>
```

  Alternatively, you can add transdata to turn on supplemental logging for all tables in the database:

```
add trandata <user>.*
```

- Using the `ggsci` command line, enable the `EXTRACT` utility using the following commands:

```
add extract <extract name> tranlog, INTEGRATED tranlog, begin now
add exttrail <path-to-trail-from-the param-file>
   extract <extractname-from-paramfile>,
   MEGABYTES Xm
```

- Register the `EXTRACT` utility with the database so that the archive logs are not deleted. This allows you to recover old, uncommitted transactions if necessary. To register the `EXTRACT` utility with the database, use the following command:

```
 register EXTRACT <extract process name>, DATABASE
```

- To start the `EXTRACT` utility, use the following command:

```
start <extract process name>
```

Running Oracle GoldenGate's `REPLICAT` Utility

The `REPLICAT` utility is used to "push" transaction information in the trail files to the target database.

The following tasks enable and start the `REPLICAT` utility:

- Configure the `REPLICAT` parameter file on the GoldenGate hub (on-premises or EC2 instance). The following listing shows an example `REPLICAT` parameter file.

```
REPLICAT RABC
SETENV (ORACLE_SID=ORCL)
SETENV (NLSLANG=AL32UTF8)

USERID oggadm1@TARGET, password XXXXXX

ASSUMETARGETDEFS
MAP EXAMPLE.TABLE, TARGET EXAMPLE.TABLE;
```

- Launch the GoldenGate command line interface (ggsci). Log into the target database. The following example shows the format for logging in:

```
 dblogin userid <user>@<db tnsname>
```

- Using the *ggsci* command line, add a checkpoint table. Note that the user indicated should be the GoldenGate user account, not the target table schema owner. The following example creates a checkpoint table named *gg_checkpoint*.

```
add checkpointtable <user>.gg_checkpoint
```

- To enable the `REPLICAT` utility, use the following command:

```
add replicat <replicat name> EXTTRAIL <extract trail file> CHECKPOINTTABLE
 <user>.gg_checkpoint
```

- To start the `REPLICAT` utility, use the following command:

```
start <replicat name>
```

Troubleshooting Issues When Using Oracle GoldenGate with Amazon RDS

This section explains the most common issues when using GoldenGate with Amazon RDS.

Topics

Log Retention

You must have log retention enabled. If you do not, or if the retention value is too small, you will see the following message:

```
2014-03-06 06:17:27  ERROR   OGG-00446  error 2 (No such file or directory)
opening redo log /rdsdbdata/db/GGTEST3_A/onlinelog/o1_mf_2_9k4bp1n6_.log
for sequence 1306Not able to establish initial position for begin time 2014-03-06
 06:16:55.
```

GoldenGate appears to be properly configured but replication is not working

For pre-existing tables, GoldenGate needs to be told which SCN it should work from. Take the following steps to fix this issue:

- Launch the GoldenGate command line interface (ggsci). Log into the source database. The following example shows the format for logging in:

```
dblogin userid <user>@<db tnsname>
```

- Using the ggsci command line, set up the start SCN for the `EXTRACT` process. The following example sets the SCN to 223274 for the extract:

```
ALTER EXTRACT <extract process name> SCN 223274
start <extract process name>
```

- Log into the target database. The following example shows the format for logging in:

```
dblogin userid <user>@<db tnsname>
```

- Using the ggsci command line, set up the start SCN for the `REPLICAT` process. The following example sets the SCN to 223274 for the `REPLICAT`:

```
start <replicat process name> atcsn 223274
```

Using the Oracle Repository Creation Utility on Amazon RDS for Oracle

You can use Amazon RDS to host an Oracle DB instance that holds the schemas to support your Fusion Middleware components. Before you can use Fusion Middleware components, you must create and populate schemas for them in your database. You create and populate the schemas by using the Oracle Repository Creation Utility (RCU).

You can store the schemas for any Fusion Middleware components in your Amazon RDS DB instance. The following is a list of schemas that have been verified to install correctly:

- Analytics (ACTIVITIES)
- Audit Services (IAU)
- Audit Services Append (IAU_APPEND)
- Audit Services Viewer (IAU_VIEWER)
- Discussions (DISCUSSIONS)
- Metadata Services (MDS)
- Oracle Business Intelligence (BIPLATFORM)
- Oracle Platform Security Services (OPSS)
- Portal and Services (WEBCENTER)
- Portlet Producers (PORTLET)
- Service Table (STB)
- SOA Infrastructure (SOAINFRA)
- User Messaging Service (UCSUMS)
- WebLogic Services (WLS)

Licensing and Versions

Amazon RDS supports Oracle Repository Creation Utility (RCU) version 12c only. You can use the RCU in the following configurations:

- RCU 12c with Oracle database 12.1.0.2.v4 or later
- RCU 12c with Oracle database 11.2.0.4.v8 or later

Before you can use RCU, you need a license for Oracle Fusion Middleware. You also need to follow the Oracle licensing guidelines for the Oracle database that hosts the repository. For more information, see Oracle Fusion Middleware Licensing Information User Manual in the Oracle documentation.

Fusion MiddleWare supports repositories on Oracle Database Enterprise Edition and Standard Editions (SE, SE One, or SE Two). Oracle recommends Enterprise Edition for production installations that require partitioning and installations that require online index rebuild.

Before you create your Oracle DB instance, confirm the Oracle database version that you need to support the components that you want to deploy. You can use the Certification Matrix to find the requirements for the Fusion Middleware components and versions you want to deploy. For more information, see Oracle Fusion Middleware Supported System Configurations in the Oracle documentation.

Amazon RDS supports Oracle database version upgrades as needed. For more information, see Upgrading a DB Instance Engine Version (p. 115).

Before You Begin

Before you begin, you need an Amazon VPC. Because your Amazon RDS DB instance needs to be available only to your Fusion Middleware components, and not to the public Internet, your Amazon RDS DB instance is hosted in a private subnet, providing greater security. For information about how to create an Amazon VPC for use with an Oracle DB instance, see Creating an Amazon VPC for Use with an Oracle Database (p. 1074).

Before you begin, you also need an Oracle DB instance. For information about how to create an Oracle DB instance for use with Fusion Middleware metadata, see Creating an Oracle DB Instance (p. 1079).

Recommendations

The following are some recommendations for working with your DB instance in this scenario:

- We recommend that you use Multi-AZ for production workloads. For more information about working with multiple Availability Zones, see Regions and Availability Zones (p. 97).
- For additional security, Oracle recommends that you use Transparent Data Encryption (TDE) to encrypt your data at rest. If you have an Enterprise Edition license that includes the Advanced Security Option, you can enable encryption at rest by using the TDE option. For more information, see Oracle Transparent Data Encryption (p. 1036).

  Amazon RDS also provides an encryption at rest option for all database editions. For more information, see Encrypting Amazon RDS Resources (p. 355).
- Configure your VPC Security Groups to allow communication between your application servers and your Amazon RDS DB instance. The application servers that host the Fusion Middleware components can be on Amazon EC2 or on-premises.

Using the Oracle Repository Creation Utility

You use the Oracle Repository Creation Utility (RCU) to create and populate the schemas to support your Fusion Middleware components.

Running RCU Using the Command Line in One Step

If you don't need to edit any of your schemas before populating them, you can run RCU in a single step. Otherwise, see the following section for running RCU in multiple steps.

You can run the RCU in silent mode by using the command-line parameter `-silent`. When you run RCU in silent mode, you can avoid typing passwords on the command line by creating a text file containing the passwords. Create a text file with the password for `dbUser` on the first line, and the password for each component on subsequent lines. You specify the name of the password file as the last parameter to the RCU command.

Example

The following example creates and populates schemas for the SOA Infrastructure component (and its dependencies) in a single step.

For Linux, OS X, or Unix:

```
export ORACLE_HOME=/u01/app/oracle/product/12.2.1.0/fmw
export JAVA_HOME=/usr/java/jdk1.8.0_65
${ORACLE_HOME}/oracle_common/bin/rcu \
-silent \
-createRepository \
-connectString ${dbhost}:${dbport}:${dbname} \
```

```
-dbUser ${dbuser} \
-dbRole Normal \
-honorOMF \
-schemaPrefix ${SCHEMA_PREFIX} \
-component MDS \
-component STB \
-component OPSS \
-component IAU \
-component IAU_APPEND \
-component IAU_VIEWER \
-component UCSUMS \
-component WLS \
-component SOAINFRA \
-f < /tmp/passwordfile.txt
```

For more information, see Running Repository Creation Utility from the Command Line in the Oracle documentation.

Running RCU Using the Command Line in Multiple Steps

If you need to manually edit your schema scripts, you can run the RCU in multiple steps:

1. Run RCU in **Prepare Scripts for System Load** mode by using the `-generateScript` command-line parameter to create the scripts for your schemas.
2. Manually edit and run the generated script `script_systemLoad.sql`.
3. Run RCU again in **Perform Product Load** mode by using the `-dataLoad` command-line parameter to populate the schemas.
4. Run the generated clean-up script `script_postDataLoad.sql`.

You can run the RCU in silent mode by using the command-line parameter `-silent`. When you run RCU in silent mode, you can avoid typing passwords on the command line by creating a text file containing the passwords. Create a text file with the password for `dbUser` on the first line, and the password for each component on subsequent lines. You specify the name of the password file as the last parameter to the RCU command.

Example

The following example creates schema scripts for the SOA Infrastructure component (and its dependencies).

For Linux, OS X, or Unix:

```
export ORACLE_HOME=/u01/app/oracle/product/12.2.1.0/fmw
export JAVA_HOME=/usr/java/jdk1.8.0_65
${ORACLE_HOME}/oracle_common/bin/rcu \
-silent \
-generateScript \
-connectString ${dbhost}:${dbport}:${dbname} \
-dbUser ${dbuser} \
-dbRole Normal \
-honorOMF \
[-encryptTablespace true] \
-schemaPrefix ${SCHEMA_PREFIX} \
-component MDS \
-component STB \
-component OPSS \
-component IAU \
-component IAU_APPEND \
-component IAU_VIEWER \
-component UCSUMS \
```

```
-component WLS \
-component SOAINFRA \
-scriptLocation /tmp/rcuscripts \
-f < /tmp/passwordfile.txt
```

Now you can edit the generated script, connect to your Oracle DB instance, and run the script. The generated script is named `script_systemLoad.sql`. For information about connecting to your Oracle DB instance, see Connecting to Your Sample Oracle DB Instance (p. 50).

The following example populates the schemas for the SOA Infrastructure component (and its dependencies).

For Linux, OS X, or Unix:

```
export JAVA_HOME=/usr/java/jdk1.8.0_65
${ORACLE_HOME}/oracle_common/bin/rcu \
-silent \
-dataLoad \
-connectString ${dbhost}:${dbport}:${dbname} \
-dbUser ${dbuser} \
-dbRole Normal \
-honorOMF \
-schemaPrefix ${SCHEMA_PREFIX} \
-component MDS \
-component STB \
-component OPSS \
-component IAU \
-component IAU_APPEND \
-component IAU_VIEWER \
-component UCSUMS \
-component WLS \
-component SOAINFRA \
-f < /tmp/passwordfile.txt
```

To finish, you connect to your Oracle DB instance, and run the clean-up script. The script is named `script_postDataLoad.sql`.

For more information, see Running Repository Creation Utility from the Command Line in the Oracle documentation.

Running RCU in Interactive Mode

To use the RCU graphical user interface, you can run RCU in interactive mode. To run RCU in interactive mode, include the `-interactive` parameter and omit the `-silent` parameter. For more information, see Understanding Repository Creation Utility Screens in the Oracle documentation.

Example

The following example starts RCU in interactive mode and pre-populates the connection information.

For Linux, OS X, or Unix:

```
export ORACLE_HOME=/u01/app/oracle/product/12.2.1.0/fmw
export JAVA_HOME=/usr/java/jdk1.8.0_65
${ORACLE_HOME}/oracle_common/bin/rcu \
-interactive \
-createRepository \
-connectString ${dbhost}:${dbport}:${dbname} \
-dbUser ${dbuser} \
-dbRole Normal
```

Known Issues

The following are some known issues for working with RCU, with some troubleshooting suggestions:

- Oracle Managed Files (OMF) — Amazon RDS uses OMF data files to simplify storage management. You can customize tablespace attributes, such as size and extent management. However, specifying a data file name when you run RCU causes tablespace code to fail with `ORA-20900`. The RCU can be used with OMF in the following ways:
  - In RCU 12.2.1.0 and later, use the `-honorOMF` command-line parameter.
  - In RCU 12.1.0.3 and later, use multiple steps and edit the generated script. For more information, see Running RCU Using the Command Line in Multiple Steps (p. 1114).
- SYSDBA — Because Amazon RDS is a managed service, you don't have full SYSDBA access to your Oracle DB instance. However, RCU 12c supports users with lower privileges. In most cases, the master user privilege is sufficient to create repositories. In some cases, the RCU might fail with `ORA-01031` when attempting to grant SYS object privileges. You can retry and run the RDSADMIN_UTIL.GRANT_SYS_OBJECT() stored procedure, or contact AWS Support.
- Dropping Enterprise Scheduler Service — When you use the RCU to drop an Enterprise Scheduler Service repository, the RCU might fail with `Error: Component drop check failed`.

Related Topics

- Oracle Licensing (p. 933)

Installing a Siebel Database on Oracle on Amazon RDS

You can use Amazon RDS to host a Siebel Database on an Oracle DB instance. The Siebel Database is part of the Siebel Customer Relationship Management (CRM) application architecture. For an illustration, see Generic Architecture of Siebel Business Application.

This topic helps you set up a Sibel Database on an Oracle DB instance on Amazon RDS. You can also find out how to use Amazon Web Services to support the other components required by the Siebel CRM application architecture.

Note
To install a Siebel Database on Oracle on Amazon RDS, you need to use the master user account. You don't need `SYSDBA` privilege; master user privilege is sufficient. For more information, see Master User Account Privileges (p. 388).

Licensing and Versions

To install a Siebel Database on Amazon RDS, you must use your own Oracle Database license, and your own Siebel license. You must have the appropriate Oracle Database license (with Software Update License and Support) for the DB instance class and Oracle Database edition. For more information, see Oracle Licensing (p. 933).

Oracle Database Enterprise Edition is the only edition certified by Siebel for this scenario. Amazon RDS supports Siebel CRM version 15.0 or 16.0. Use Oracle 12c, version 12.1.0.2.0. For the procedures following, we use Siebel CRM version 15.0 and Oracle 12.1.0.2.0. For more information, see Oracle 12c with Amazon RDS (p. 936).

Amazon RDS supports database version upgrades. For more information, see Upgrading a DB Instance Engine Version (p. 115).

Before You Begin

Before you begin, you need an Amazon VPC. Because your Amazon RDS DB instance needs to be available only to your Siebel Enterprise Server, and not to the public Internet, your Amazon RDS DB instance is hosted in a private subnet, providing greater security. For information about how to create an Amazon VPC for use with Siebel CRM, see Creating an Amazon VPC for Use with an Oracle Database (p. 1074).

Before you begin, you also need an Oracle DB instance. For information about how to create an Oracle DB instance for use with Siebel CRM, see Creating an Oracle DB Instance (p. 1079).

Installing and Configuring a Siebel Database

After you create your Oracle DB instance, you can install your Siebel Database. You install the database by creating table owner and administrator accounts, installing stored procedures and functions, and then running the Siebel Database Configuration Wizard. For more information, see Installing the Siebel Database on the RDBMS.

To run the Siebel Database Configuration Wizard, you need to use the master user account. You don't need `SYSDBA` privilege; master user privilege is sufficient. For more information, see Master User Account Privileges (p. 388).

Using Other Amazon RDS Features with a Siebel Database

After you create your Oracle DB instance, you can use additional Amazon RDS features to help you customize your Siebel Database.

Collecting Statistics with the Oracle Statspack Option

You can add features to your DB instance through the use of options in DB option groups. When you created your Oracle DB instance, you used the default DB option group. If you want to add features to your database, you can create a new option group for your DB instance.

If you want to collect performance statistics on your Siebel Database, you can add the Oracle Statspack feature. For more information, see Oracle Statspack (p. 1029).

Some option changes are applied immediately, and some option changes are applied during the next maintenance window for the DB instance. For more information, see Working with Option Groups (p. 153). After you create a customized option group, modify your DB instance to attach it. For more information, see Modifying a DB Instance Running the Oracle Database Engine (p. 967).

Performance Tuning with Parameters

You manage your DB engine configuration through the use of parameters in a DB parameter group. When you created your Oracle DB instance, you used the default DB parameter group. If you want to customize your database configuration, you can create a new parameter group for your DB instance.

When you change a parameter, depending on the type of the parameter, the changes are applied either immediately or after you manually reboot the DB instance. For more information, see Working with DB Parameter Groups (p. 170). After you create a customized parameter group, modify your DB instance to attach it. For more information, see Modifying a DB Instance Running the Oracle Database Engine (p. 967).

To optimize your Oracle DB instance for Siebel CRM, you can customize certain parameters. The following table shows some recommended parameter settings. For more information about performance tuning Siebel CRM, see Siebel CRM Performance Tuning Guide.

Parameter Name	Default Value	Guidance for Optimal Siebel CRM Performance
_always_semi_join	CHOOSE	OFF
_b_tree_bitmap_plans	TRUE	FALSE
_like_with_bind_as_equality	FALSE	TRUE
_no_or_expansion	FALSE	FALSE
_optimizer_join_sel_sanity_check	TRUE	TRUE
_optimizer_max_permutations	2000	100
_optimizer_sortmerge_join_enabled	TRUE	FALSE
_partition_view_enabled	TRUE	FALSE
open_cursors	300	At least **2000**.

Creating Snapshots

After you create your Siebel Database, you can copy the database by using the snapshot features of Amazon RDS. For more information, see Creating a DB Snapshot (p. 207) and Restoring from a DB Snapshot (p. 209).

Support for Other Siebel CRM Components

In addition to your Siebel Database, you can also use Amazon Web Services to support the other components of your Siebel CRM application architecture. You can find more information about the support provided by Amazon AWS for additional Siebel CRM components in the following table.

Siebel CRM Component	Amazon AWS Support
Siebel Enterprise (with one or more Siebel Servers)	You can host your Siebel Servers on Amazon Elastic Compute Cloud (Amazon EC2) instances. You can use Amazon EC2 to launch as many or as few virtual servers as you need. Using Amazon EC2, you can scale up or down easily to handle changes in requirements. For more information, see What Is Amazon EC2? You can put your servers in the same VPC with your DB instance and use the VPC security group to access the database. For more information, see Working with an Amazon RDS DB Instance in a VPC (p. 399).
Web Servers (with Siebel Web Server Extensions)	You can install multiple Web Servers on multiple EC2 instances. You can then use Elastic Load Balancing to distribute incoming traffic among the instances. For more information, see What Is Elastic Load Balancing?
Siebel Gateway Name Server	You can host your Siebel Gateway Name Server on an EC2 instance. You can then put your server in the same VPC with the DB instance and use the VPC security group to access the database. For more information, see Working with an Amazon RDS DB Instance in a VPC (p. 399).

Related Topics

- Connecting to a DB Instance Running the Oracle Database Engine (p. 959)

Appendix: Oracle Database Engine Release Notes

Amazon RDS incorporates bug fixes from Oracle via their quarterly Database Patch Set Updates (PSU). You can be confident that your DB instance is running a stable, common version of the database software that has been regression tested by both Oracle and Amazon. We do not support applying one-off patches to individual DB instances.

The following table shows what Oracle PSUs are applied to the Oracle versions in Amazon RDS:

PSU	Version 12.1.0.2	Version 11.2.0.4
2017 July	12.1.0.2.v9 (p. 1121)	11.2.0.4.v13 (p. 1131)
2017 April	12.1.0.2.v8 (p. 1122)	11.2.0.4.v12 (p. 1132)
2017 January	12.1.0.2.v7 (p. 1124)	11.2.0.4.v11 (p. 1133)
2016 October	12.1.0.2.v6 (p. 1125)	11.2.0.4.v10 (p. 1134)
2016 July	12.1.0.2.v5 (p. 1126)	11.2.0.4.v9 (p. 1136)
2016 April	12.1.0.2.v4 (p. 1127)	11.2.0.4.v8 (p. 1137)
2016 January	12.1.0.2.v3 (p. 1128)	11.2.0.4.v7 (p. 1138)
2015 October	12.1.0.2.v2 (p. 1129)	11.2.0.4.v6 (p. 1139) 11.2.0.4.v5 (p. 1139)
2015 April	12.1.0.2.v1 (p. 1129)	11.2.0.4.v4 (p. 1140)
2014 October	—	11.2.0.4.v3 (p. 1141)
2014 July	—	11.2.0.4.v2 (p. 1142) (Deprecated)
2014 January	—	11.2.0.4.v1 (p. 1143)

Topics

Database Engine: 12.1.0.2

The following versions are available for database engine 12.1.0.2:

- Version 12.1.0.2.v2 (p. 1129)
- Version 12.1.0.2.v1 (p. 1129)

Version 12.1.0.2.v9

Version 12.1.0.2.v9 adds support for the following:

- Oracle July 2017 PSU, a combination of database PSU (patch 26609783) + OJVM component PSU (patch 26027162)
- Oracle recommended RDBMS patches for Oracle GoldenGate (patch 17969866)
- DBMS_STATS AUTO DOP COMPUTES A HIGH DOP UNNECESSARILY (patch 21171382)
- JSON bundle patch (patch 26083365)
- KGL heap size patch (patch 20033733 for 12.1.0.2)
- Timezone file DSTv30 (patch 25881255, OJVM patch 25881271)
- Adds support for Validating DB Instance Files (p. 1058) with the `RMAN` logical validation utility
- Adds support for Setting the Default Edition for a DB Instance (p. 1058)

Oracle patch 26609783, released July 2017

Bugs fixed: 21099555, 22175564, 19141838, 22083366, 20842388, 19865345, 20117253 19791273, 20671094, 21542577, 20951038, 19243521, 22165897, 19238590 21281532, 17008068, 19908836, 24577566, 21184223, 25427662, 19134173 20569094, 20031873, 20387265, 20322560, 21575362, 19149990, 21263635 17551063, 18886413, 22160989, 22507210, 19703301, 19366375, 18007682 19001390, 18202441, 24285405, 25655390, 20267166, 19358317, 19706965 19068970, 24739928, 18549238, 22148226, 18797519, 26544823, 20825533 21196809, 18940497, 19670108, 19649152, 18866977, 18948177, 22496904 19404068, 18964978, 19176326, 19035573, 20413820, 20717081, 19176223 21106027, 20904530, 20134339, 19074147, 20868862, 18411216, 21072646 25475853, 21322887, 22507234, 20425790, 20862087, 18966843, 21329301 20562898, 19333670, 19468991, 20124446, 19883092, 20878790, 18510194 19658708, 19591608, 19402853, 20618595, 21787056, 22380919, 21266085 19469538, 17835294, 19721304, 19068610, 19791377, 22178855, 16777441 22173980, 20746251, 20048359, 21896069, 19185876, 20898391, 20281121 20907061, 6599380, 19577410, 22092979, 19001359, 20603378, 23089357 21387964, 19490948, 22294260, 20832516, 17532734, 22351572, 19309466 19081128, 20627866, 20844426, 24908321, 21188532, 18791688, 21442094 20890311, 20596234, 20368850, 18973548, 19303936, 21296029, 20882568 21479753, 19461270, 20235511, 22077517, 20936905, 21220620, 18964939 19430401, 22296366, 21153266, 19409212, 22657942, 20703000, 20657441 19879746, 20557786, 19684504, 21294938, 19024808, 24693382, 20528052 20977794, 18799993, 20466322, 18740837, 19662635, 18440095, 20228093 19065556, 20212067, 25547060, 21868720, 22905130, 19524384, 25459958 24350831, 17722075, 20446883, 25056052, 18952989, 24523374, 16870214 19928926, 19835133, 21629064, 21354456, 20466628, 24386767, 25490238 19931709, 19730508, 18819908, 20250147, 23124895, 25643931, 23220453 19188927, 20074391, 18307021, 23533807, 20356733, 14643995, 18090142 19065677, 19547370, 21225209, 21960504, 26575788, 20397490, 20172151 18967382, 19174430, 21241829, 19536415, 26546664, 19171086, 21132297 21889720, 22465352, 22168163, 19335438, 24397438, 20076781, 20447445 18856999, 20471920, 19869255, 21620471, 18990693, 23096938, 19124336 17890099, 24812585, 18990023, 21300341, 20101006, 20848335, 21744290 20897759, 21668627, 19304354, 19052488, 20543011, 20794034, 23025340 25606091, 23260854, 18681056, 19562381, 20952966, 19896336, 20828947 25539063, 18618122, 20328248, 20440930, 18456643, 19699191, 22865673 19201867, 22022760, 21514877, 18743542, 20798891, 20347562, 25161298 23294548, 24560906, 22551446, 19777862, 19687159, 21373076, 19174942 20424899, 21899588, 18899974, 21476308, 20598042, 24308635, 21297872 19058490, 19032777, 20171986, 22815955, 19399918, 19434529, 19018447 18051556, 21273804, 22757364, 18851894, 19022470, 19284031, 18043064 20173897, 22062026, 20475845, 17274537, 19440586, 24825843, 18974476 22374754, 16887946, 17319928, 20401975, 20708701, 24674955, 22062517 22809871, 17655240, 19805359, 16439813,

19155797, 20859910, 19393542 17210525, 22024071, 19189525, 21847223, 21649497, 19075256, 25823754 25079710, 20315311, 22762046, 22075064, 20936731, 20437153, 18845653 19280225, 19248799, 20560611, 18988834, 21756699, 18921743, 20245930 18799063, 20373598, 20476175, 19571367, 20925795, 19018206, 25264559 20711718, 20509482, 20181030, 20588502, 21911701, 18849537, 23501901 19183343, 21917884, 21142837, 20603431, 19189317, 19644859, 19390567 26546754, 19279273, 20669434, 16863642, 22528741, 25546608, 19619732 20348653, 18607546, 19315691, 19676905, 20165574, 17867700, 20558005 20734332, 19532017, 20922010, 19818513, 19450314, 22353346, 16941434 20361671, 25423453, 20009833, 22366558, 20294666, 23197103, 18191823 19195895, 19371175, 19307662, 19154375, 20043616, 21977392, 18914624 22529728, 20139391, 25330273, 19593445, 21291274, 19382851, 19520602 19174521, 21875360, 19676012, 19326908, 20217801, 20093776, 18840932 21097043, 21246723, 20803014, 21665897, 19143550, 23026585, 20428621 19627012, 14283239, 21422580, 19213447, 19518079, 18610915, 18674024 24413809, 18306996, 19915271, 21626377, 19524158, 20122715, 20513399 20284155, 25091141, 21080143, 20017509, 22359063, 19363645, 19597439 21239530, 19383839, 20880215, 21756677, 19888853, 22458049, 19534363 19354335, 19044962, 19639483, 25982666, 19475971, 22353199, 21060755 22243719, 22916353, 20378086, 24808595, 21756661, 21260431, 22923409 19028800, 20877664, 21059919, 20879889, 21380789, 19723336, 19077215 21421886, 19604659, 21285458, 23533524, 23170620, 22365117, 18288842 19048007, 19308965, 19689979, 17409174, 19503821, 21526048, 19197175 19180770, 24573817, 19902195, 24835538, 23324000, 20318889, 19013183 20591183, 19012119, 20464614, 19067244, 21632821, 19841800, 19512341 22695831, 20331945, 19587324, 24316947, 19578350, 19637186, 19054077 18674047, 19708632, 20898997, 21091431, 19289642, 21133343, 20835241 20869721, 21172913, 19258504, 17365043, 21419850, 21644640, 19468347 21373473, 25093739, 16359751, 21164318, 25484507, 22520320, 19769480 19439759, 19272708, 19978542, 19329654, 20402832, 19873610, 23229229 13542050, 21517440, 19291380, 21915719, 25600342, 20879709, 20677396 19076343, 19561643, 19990037, 18909599, 19487147, 25600421, 20831538 19016730, 18250893, 16619249, 18354830, 24411921, 16756406, 18254023 21188584, 19989009, 25766822, 17414008, 20688221, 20441797, 20704450 21780146, 25612095, 25957038, 25483815, 19157754, 19207117, 24437510 18885870, 21785691, 20673810, 21450666, 18893947, 18705806, 22223463 18417036, 16923858, 23314180, 20919320, 20474192, 22046677, 21299490 19501299, 19385656, 20432873, 20920911, 20899461, 21387128, 21315084 18122373, 20581111, 22624709, 19606174, 24690216, 18436647, 19023822 25110233, 19124589, 19178851, 19597583, 18499088, 19050649

Version 12.1.0.2.v8

Version 12.1.0.2.v8 adds support for the following:

- Oracle patch 25433980, a combination of database PSU (patch 25171037) + OJVM component PSU (patch 25437695)
- Oracle recommended RDBMS patches for Oracle GoldenGate (patch 17969866 for 12.1.0.2)
- Oracle Forms patch 18307021 for 12.1.0.2
- DBMS_STATS Patch (patch 21171382 for 12.1.0.2)
- JSON bundle patch (patch 25531469 for 12.1.0.2)
- KGL heap size patch (patch 20033733 for 12.1.0.2)
- Fixed a bug that affected PSU apply after upgrade to 12.1.0.2.v5, v6, and v7
- Timezone file DSTv28 (patch 24701840)
- Adds support for the `DBMS_CHANGE_NOTIFICATION` package
- Adds support for `XSTREAM` packages and views (may require additional licensing)

Oracle patch 25171037, released April 2017

Bugs fixed: 21099555, 22175564, 19141838, 22083366, 20842388, 20117253, 19865345 19791273, 21542577, 20951038, 19243521, 22165897, 17008068, 19908836 21281532, 19238590, 24577566, 21184223, 19134173, 20569094, 20031873 20322560, 20387265, 21575362, 19149990, 21263635,

17551063, 18886413 22160989, 22507210, 19366375, 19703301, 19001390, 24285405, 18202441
20267166, 19358317, 19706965, 19068970, 18549238, 24739928, 18797519 22148226, 20825533,
21196809, 19649152, 19670108, 18940497, 18948177 22496904, 18964978, 19176326, 19035573,
20413820, 19176223, 21106027 20904530, 20134339, 19074147, 20868862, 18411216, 25475853,
21322887 21072646, 22507234, 20425790, 20862087, 18966843, 21329301, 20562898 19333670,
20124446, 19468991, 19883092, 20878790, 18510194, 19658708 19591608, 19402853, 20618595,
21787056, 22380919, 19469538, 21266085 17835294, 19721304, 19068610, 19791377, 22178855,
16777441, 22173980 20048359, 20746251, 21896069, 19185876, 20898391, 20907061, 20281121
6599380, 19577410, 22092979, 19001359, 20603378, 23089357, 21387964 19490948, 22294260,
17532734, 20832516, 22351572, 19309466, 20627866 19081128, 20844426, 21188532, 18791688,
20890311, 21442094, 20596234 20368850, 18973548, 19303936, 21296029, 20882568, 19461270,
21479753 22077517, 20936905, 20235511, 21220620, 18964939, 19430401, 22296366 21153266,
19409212, 20703000, 22657942, 19879746, 20657441, 21294938 19684504, 19024808, 20528052,
24693382, 20977794, 18799993, 20466322 18740837, 19662635, 18440095, 20228093, 19065556,
20212067, 21868720 22905130, 19524384, 24350831, 17722075, 20446883, 25056052, 18952989
24523374, 16870214, 19928926, 19835133, 21629064, 21354456, 20466628 24386767, 25490238,
19931709, 19730508, 18819908, 20250147, 23124895 23220453, 19188927, 20074391, 18307021,
20356733, 14643995, 19065677 19547370, 21960504, 21225209, 20397490, 18967382, 19174430,
21241829 19536415, 19171086, 21889720, 22465352, 22168163, 19335438, 24397438 20447445,
18856999, 19869255, 20471920, 21620471, 23096938, 18990693 19124336, 17890099, 24812585,
18990023, 21300341, 20101006, 20848335 21744290, 20897759, 21668627, 19304354, 20543011,
19052488, 20794034 23025340, 23260854, 18681056, 20952966, 19896336, 25539063, 18618122
20328248, 20440930, 18456643, 19699191, 19201867, 22865673, 22022760 20798891, 18743542,
25161298, 20347562, 22551446, 19777862, 19687159 21373076, 19174942, 20424899, 21899588,
18899974, 21476308, 20598042 21297872, 24308635, 20171986, 19058490, 19032777, 22815955,
19399918 19434529, 21273804, 19018447, 22757364, 18851894, 19022470, 19284031 18043064,
20173897, 22062026, 20475845, 17274537, 19440586, 18974476 24825843, 22374754, 16887946,
17319928, 20401975, 20708701, 22062517 22809871, 17655240, 16439813, 19805359, 19155797,
20859910, 19393542 22024071, 17210525, 19189525, 21847223, 21649497, 25079710, 19075256
20315311, 22762046, 22075064, 20936731, 18845653, 19280225, 19248799 20560611, 18988834,
21756699, 18921743, 20245930, 18799063, 20373598 19571367, 20476175, 20925795, 19018206,
25264559, 20711718, 20509482 20181030, 20588502, 21911701, 18849537, 23501901, 19183343,
21917884 21142837, 19189317, 19644859, 19390567, 19279273, 20669434, 16863642 22528741,
25546608, 19619732, 18607546, 20348653, 19315691, 19676905 20165574, 17867700, 20558005,
20734332, 19532017, 20922010, 19818513 19450314, 22353346, 16941434, 20361671, 20009833,
22366558, 20294666 18191823, 23197103, 19195895, 19371175, 19307662, 19154375, 20043616
21977392, 18914624, 22529728, 25330273, 20139391, 19593445, 21291274 19382851, 19520602,
19174521, 21875360, 19676012, 19326908, 20217801 20093776, 18840932, 21097043, 21246723,
20803014, 21665897, 19143550 20428621, 19627012, 14283239, 21422580, 19213447, 19518079,
18610915 18674024, 24413809, 18306996, 19915271, 19524158, 20122715, 20284155 20017509,
22359063, 19363645, 19597439, 21239530, 19383839, 20880215 21756677, 19888853, 22458049,
19534363, 19354335, 19044962, 19639483 19475971, 22353199, 22243719, 21060755, 22916353,
20378086, 24808595 21756661, 21260431, 22923409, 19028800, 20877664, 21059919, 20879889
21380789, 19723336, 19077215, 19604659, 21421886, 21285458, 23533524 23170620, 22365117,
18288842, 19048007, 19308965, 19689979, 19503821 21526048, 19197175, 19180770, 19902195,
23324000, 20318889, 19013183 20591183, 19012119, 20464614, 19067244, 21632821, 19841800,
19512341 22695831, 20331945, 19587324, 24316947, 19578350, 19637186, 19054077 18674047,
19708632, 20898997, 21091431, 19289642, 21133343, 20869721 21172913, 19258504, 17365043,
21419850, 19468347, 21373473, 25093739 16359751, 21164318, 22520320, 19769480, 19439759,
19272708, 19978542 19329654, 20402832, 19873610, 23229229, 13542050, 21517440, 19291380
21915719, 20879709, 20677396, 19076343, 19561643, 19990037, 19487147 18909599, 20831538,
19016730, 18250893, 16619249, 18354830, 24411921 16756406, 18254023, 21188584, 19989009,
17414008, 20688221, 20704450 20441797, 25483815, 19157754, 24437510, 18885870, 21785691,
20673810 21450666, 18893947, 18705806, 22223463, 16923858, 18417036, 23314180 20919320,
20474192, 22046677, 21299490, 19501299, 19385656, 20920911 20899461, 21387128, 21315084,
18122373, 20581111, 19606174, 24690216 18436647, 19023822, 19124589, 19178851, 19597583,
18499088, 19050649

Version 12.1.0.2.v7

Version 12.1.0.2.v7 adds support for the following:

- Oracle patch 24917069, a combination of database PSU (patch 24732082) + OJVM component PSU (patch 24917972)
- Oracle recommended RDBMS patches for Oracle GoldenGate (patch 17969866 for 12.1.0.2)
- Oracle Forms patch 18307021 for 12.1.0.2
- DBMS_STATS Patch (patch 21171382 for 12.1.0.2)
- JSON bundle patch (patch 25089615 for 12.1.0.2)
- KGL heap size patch (patch 20033733 for 12.1.0.2)

Oracle patch 24917069, released January 2017

Bugs fixed: 24917972, 25067795, 24534298, 25076732, 25076756, 24315824, 21659726 24448240, 24448282, 23177536, 22675136, 23265914, 23265965, 23727148 22674709, 22670413, 22670385, 21188537, 22139226, 22118835, 22118851 21555660, 21811517, 19623450, 21566993, 21566944, 19176885, 21068507 21047803, 21047766, 20415564, 20408829, 20408866, 19877336, 19855285 19909862, 19895362, 19895326, 19153980, 19231857, 19223010, 19245191, 19699946, 21099555, 22175564, 19141838, 22083366, 20842388, 20117253, 19865345 19791273, 21542577, 20951038, 19243521, 22165897, 19908836, 21281532 19238590, 24577566, 21184223, 19134173, 20031873, 20387265, 21575362 19149990, 21263635, 17551063, 18886413, 22160989, 22507210, 19366375 19703301, 19001390, 24285405, 18202441, 20267166, 19358317, 19706965 24739928, 19068970, 18549238, 18797519, 22148226, 20825533, 21196809 19649152, 19670108, 18940497, 18948177, 22496904, 18964978, 19035573 19176326, 20413820, 19176223, 21106027, 20904530, 20134339, 19074147 20868862, 18411216, 21072646, 21322887, 22507234, 20425790, 18966843 21329301, 20562898, 19333670, 20124446, 19468991, 19883092, 18510194 19658708, 19591608, 19402853, 20618595, 21787056, 22380919, 19469538 21266085, 17835294, 19721304, 19791377, 19068610, 22178855, 16777441 22173980, 20048359, 20746251, 21896069, 20898391, 19185876, 20907061 20281121, 6599380, 19577410, 22092979, 19001359, 20603378, 23089357 19490948, 21387964, 22294260, 20832516, 17532734, 19309466, 20627866 19081128, 20844426, 21188532, 18791688, 20890311, 21442094, 20596234 18973548, 21296029, 19303936, 20882568, 19461270, 21479753, 22077517 20936905, 20235511, 21220620, 18964939, 19430401, 22296366, 21153266 19409212, 22657942, 19879746, 20657441, 21294938, 19684504, 24693382 20528052, 19024808, 20977794, 18799993, 20466322, 18740837, 19662635 20228093, 20212067, 19065556, 19524384, 17722075, 20446883, 25056052 24523374, 18952989, 16870214, 19928926, 19835133, 21629064, 21354456 20466628, 24386767, 19931709, 19730508, 18819908, 23124895, 23220453 19188927, 20074391, 18307021, 20356733, 14643995, 19547370, 19065677 21960504, 21225209, 20397490, 18967382, 19174430, 21241829, 19536415 19171086, 22465352, 22168163, 19335438, 24397438, 20447445, 18856999 19869255, 20471920, 21620471, 18990693, 17890099, 24812585, 18990023 21300341, 20101006, 20848335, 21744290, 20897759, 21668627, 19304354 19052488, 20794034, 23025340, 23260854, 18681056, 20952966, 19896336 20328248, 18618122, 20440930, 18456643, 19699191, 19201867, 22865673 22022760, 20798891, 18743542, 25161298, 20347562, 19777862, 22551446 19687159, 21373076, 19174942, 20424899, 21899588, 18899974, 21476308 20598042, 24308635, 19032777, 19058490, 22815955, 19399918, 19434529 21273804, 19018447, 22757364, 18851894, 19022470, 19284031, 18043064 20173897, 22062026, 20475845, 17274537, 19440586, 24825843, 18974476 22374754, 16887946, 17319928, 20401975, 20708701, 22809871, 17655240 16439813, 19805359, 19155797, 20859910, 19393542, 17210525, 22024071 21847223, 19189525, 21649497, 19075256, 20315311, 22762046, 22075064 20936731, 19280225, 18845653, 20560611, 19248799, 21756699, 18988834 20245930, 18921743, 18799063, 20373598, 19571367, 20476175, 20925795 25264559, 19018206, 20711718, 20509482, 20181030, 20588502, 18849537 23501901, 19183343, 21917884, 19189317, 19644859, 19390567, 19279273 20669434, 22528741, 16863642, 19619732, 18607546, 20348653, 19315691 19676905, 20165574, 17867700, 20558005, 20734332, 19532017, 20922010 19818513, 19450314, 22353346, 20361671, 20009833, 22366558, 20294666 23197103,

18191823, 19195895, 19307662, 19371175, 20043616, 19154375 18914624, 22529728, 20139391, 21291274, 19382851, 19520602, 19174521 21875360, 19676012, 19326908, 20217801, 20093776, 18840932, 21097043 21246723, 20803014, 21665897, 19143550, 20428621, 19627012, 14283239 19518079, 18610915, 18674024, 24413809, 18306996, 19524158, 19915271 20122715, 20284155, 20017509, 22359063, 19363645, 19597439, 21239530 19888853, 21756677, 20880215, 22458049, 19534363, 19354335, 19044962 19639483, 19475971, 22353199, 21060755, 22243719, 22916353, 20378086 24808595, 21260431, 21756661, 22923409, 20877664, 19028800, 21059919 20879889, 21380789, 19723336, 19077215, 19604659, 21421886, 21285458 23533524, 23170620, 22365117, 18288842, 19308965, 19048007, 19689979 21526048, 19197175, 19180770, 19902195, 23324000, 20318889, 19013183 20591183, 19012119, 20464614, 19067244, 21632821, 19512341, 19841800 22695831, 20331945, 19587324, 24316947, 19578350, 19637186, 18674047 19054077, 20898997, 19708632, 21091431, 19289642, 21133343, 20869721 21172913, 19258504, 17365043, 19468347, 21373473, 16359751, 19769480 19439759, 19272708, 19978542, 20402832, 19329654, 19873610, 23229229 21517440, 13542050, 19291380, 21915719, 20879709, 20677396, 19076343 19561643, 19990037, 19487147, 18909599, 20831538, 18250893, 19016730 16619249, 18354830, 18254023, 21188584, 19989009, 17414008, 20688221 20704450, 20441797, 19157754, 24437510, 18885870, 21785691, 18893947 21450666, 18705806, 22223463, 16923858, 18417036, 23314180, 20919320 20474192, 22046677, 19385656, 19501299, 20920911, 20899461, 21315084 21387128, 18122373, 20581111, 19606174, 24690216, 18436647, 19023822 19178851, 19124589, 19597583, 18499088, 19050649

Version 12.1.0.2.v6

Version 12.1.0.2.v6 adds support for the following:

- Oracle patch 24433133, a combination of database PSU (patch 24006101) + OJVM component PSU (patch 24315824)
- Oracle recommended RDBMS patches for Oracle GoldenGate (patch 17969866 for 12.1.0.2)
- Oracle Forms patch 18307021 for 12.1.0.2
- DBMS_STATS Patch (patch 21171382 for 12.1.0.2)
- JSON bundle patch (patch 24568656 for 12.1.0.2)
- Fixed a bug that caused 12c upgrade scripts to drop customer directories
- Made DIAG log directory available to customers

Baseline: Oracle Database Patch Set Update 12.1.0.2.161018 (patch 24006101, released October 2016)

Bugs fixed: 21099555, 22175564, 19141838, 22083366, 20842388, 20117253, 19865345 19791273, 19243521, 20951038, 19908836, 21281532, 19238590, 24577566 21184223, 19134173, 20387265, 19149990, 21263635, 18886413, 17551063 22160989, 22507210, 19703301, 19366375, 19001390, 18202441, 20267166 19358317, 19706965, 18549238, 19068970, 18797519, 22148226, 20825533 19649152, 19670108, 18940497, 18948177, 18964978, 19035573, 19176326 20413820, 19176223, 20904530, 20134339, 19074147, 20868862, 18411216 21322887, 22507234, 20425790, 18966843, 21329301, 19333670, 19468991 20124446, 19883092, 19658708, 19591608, 19402853, 20618595, 21787056 22380919, 21266085, 17835294, 19721304, 19791377, 19068610, 22178855 22173980, 20746251, 20048359, 20898391, 19185876, 20281121, 20907061 6599380, 19577410, 22092979, 20603378, 19001359, 19490948, 21387964 20832516, 17532734, 19309466, 19081128, 20627866, 20844426, 21188532 18791688, 21442094, 20890311, 20596234, 18973548, 21296029, 19303936 19461270, 21479753, 20936905, 20235511, 21220620, 18964939, 19430401 22296366, 21153266, 19409212, 22657942, 20657441, 19879746, 19684504 20528052, 19024808, 20977794, 18799993, 20466322, 18740837, 19662635 20228093, 19065556, 20212067, 19524384, 17722075, 20446883, 18952989 16870214, 19928926, 19835133, 21629064, 20466628, 24386767, 19931709 19730508, 18819908, 23124895, 19188927, 20074391, 20356733, 14643995 19547370, 19065677, 21960504, 21225209, 20397490, 18967382, 19174430 21241829, 19536415, 19171086, 22465352, 22168163,

19335438, 20447445 18856999, 20471920, 19869255, 21620471, 18990693, 17890099, 18990023 20101006, 21300341, 20848335, 21744290, 20897759, 21668627, 19304354 19052488, 20794034, 23260854, 18681056, 20952966, 19896336, 18618122 20328248, 20440930, 18456643, 19699191, 19201867, 22865673, 18743542 20798891, 20347562, 22551446, 19777862, 19687159, 21373076, 19174942 20424899, 21899588, 18899974, 20598042, 19032777, 19058490, 22815955 19399918, 19434529, 21273804, 19018447, 22757364, 18851894, 19284031 19022470, 18043064, 20173897, 22062026, 20475845, 17274537, 19440586 16887946, 22374754, 17319928, 20708701, 17655240, 16439813, 19805359 19155797, 20859910, 19393542, 22024071, 17210525, 21847223, 19189525 21649497, 19075256, 22762046, 22075064, 19280225, 18845653, 20560611 19248799, 21756699, 18988834, 20245930, 18921743, 18799063, 20373598 20476175, 19571367, 20925795, 19018206, 20509482, 20711718, 20588502 18849537, 19183343, 21917884, 19189317, 19644859, 19390567, 19279273 20669434, 16863642, 22528741, 19619732, 18607546, 20348653, 19315691 19676905, 20165574, 17867700, 20558005, 20734332, 19532017, 20922010 19450314, 22353346, 20361671, 20009833, 22366558, 20294666, 18191823 19307662, 19371175, 19195895, 20043616, 19154375, 18914624, 20139391 21291274, 19174521, 19520602, 19382851, 21875360, 19676012, 19326908 20217801, 20093776, 21097043, 21246723, 21665897, 19143550, 20428621 19627012, 14283239, 19518079, 18610915, 18674024, 18306996, 19524158 19915271, 20122715, 20284155, 20017509, 19363645, 19597439, 21239530 19888853, 20880215, 21756677, 19534363, 19354335, 19044962, 19639483 22353199, 22243719, 22916353, 20378086, 21756661, 21260431, 22923409 20877664, 19028800, 20879889, 19723336, 19077215, 21421886, 19604659 19308965, 19048007, 18288842, 19689979, 21526048, 19180770, 19197175 19902195, 20318889, 19013183, 19012119, 20464614, 19067244, 21632821 19512341, 19841800, 20331945, 19587324, 24316947, 19578350, 19637186 18674047, 19054077, 20898997, 19708632, 21091431, 19289642, 20869721 19258504, 17365043, 19468347, 21373473, 16359751, 19439759, 19769480 19272708, 19978542, 20402832, 19329654, 19873610, 23229229, 21517440 13542050, 19291380, 21915719, 19076343, 19561643, 19990037, 19487147 18909599, 20831538, 18250893, 19016730, 16619249, 18354830, 21188584 19989009, 17414008, 20688221, 20704450, 20441797, 19157754, 18885870 21785691, 21450666, 18893947, 18705806, 22223463, 16923858, 18417036 20919320, 20474192, 22046677, 19385656, 19501299, 20920911, 20899461 21387128, 21315084, 18122373, 20581111, 19606174, 18436647, 19023822 19178851, 19124589, 19597583, 18499088, 19050649

Version 12.1.0.2.v5

Version 12.1.0.2.v5 adds support for the following:

- Oracle patch 23615289, a combination of database PSU (patch 23054246) + OJVM component PSU (patch 23177536)
- Timezone file DSTv26 (patch 22873635 for 12.1.0.2)
- Oracle recommended RDBMS patches for Oracle GoldenGate (patch 17969866 for 12.1.0.2)
- Oracle Forms patch 18307021 for 12.1.0.2
- Added the ability to create custom password verify functions. For more information, see Creating Custom Functions to Verify Passwords (p. 1049).
- Fixed a bug that prevented implicit recompilation of views owned by SYS

Baseline: Oracle Database Patch Set Update 12.1.0.2.160719 (patch 23054246, released July 2016)

Bugs fixed: 19189525, 21847223, 21099555, 21649497, 19075256, 19141838, 22762046 22075064, 20117253, 19865345, 19791273, 18845653, 19280225, 19248799 19243521, 20951038, 18988834, 21756699, 21281532, 19238590, 21184223 18921743, 20245930, 18799063, 19134173, 20373598, 19571367, 20476175 20925795, 19018206, 20509482, 20711718, 20387265, 20588502, 19149990 21263635, 18849537, 18886413, 17551063, 22507210, 19183343, 19366375 19703301, 21917884, 19001390, 18202441, 19189317, 20267166, 19644859 19390567, 19358317, 19279273, 19706965, 18549238, 16863642, 19068970 22528741, 18797519, 20825533, 19619732, 18607546, 20348653,

19649152 19670108, 18940497, 18948177, 19315691, 19676905, 18964978, 19176326 20165574, 19035573, 20413820, 17867700, 20558005, 19176223, 19532017 20904530, 20134339, 19450314, 19074147, 22353346, 20868862, 18411216 22507234, 20361671, 20425790, 18966843, 20009833, 22366558, 21329301 20294666, 18191823, 19333670, 19195895, 19371175, 19307662, 19154375 20043616, 20124446, 18914624, 19468991, 19883092, 21291274, 19382851 19520602, 19174521, 21875360, 19676012, 19326908, 19658708, 19591608 19402853, 20093776, 20618595, 21787056, 22380919, 21246723, 17835294 19721304, 19068610, 19791377, 21665897, 22178855, 22173980, 20048359 20746251, 19143550, 20898391, 19185876, 19627012, 20281121, 19577410 22092979, 19001359, 14283239, 19518079, 18610915, 19490948, 17532734 18674024, 18306996, 19309466, 19081128, 19524158, 19915271, 20122715 21188532, 18791688, 20284155, 20890311, 21442094, 20596234, 18973548 21296029, 19303936, 19597439, 20936905, 20235511, 21220620, 20880215 18964939, 21756677, 19888853, 19534363, 19430401, 19354335, 19044962 19639483, 22296366, 22353199, 21153266, 19409212, 19879746, 20657441 19684504, 20528052, 19024808, 20977794, 20378086, 18799993, 21756661 21260431, 18740837, 22923409, 19028800, 20877664, 20228093, 20879889 19065556, 19723336, 19077215, 19604659, 21421886, 19524384, 17722075 19308965, 18288842, 19048007, 19689979, 20446883, 18952989, 16870214 19928926, 19835133, 21629064, 21526048, 19197175, 19180770, 20466628 19902195, 19931709, 20318889, 19013183, 19730508, 19012119, 19067244 20074391, 20356733, 14643995, 19512341, 19841800, 20331945, 19587324 19065677, 19547370, 19578350, 21225209, 19637186, 20397490, 18967382 19174430, 21241829, 19054077, 18674047, 20898997, 19708632, 19536415 21091431, 19289642, 20869721, 22168163, 19335438, 19258504, 20447445 17365043, 18856999, 19468347, 19869255, 20471920, 21373473, 21620471 16359751, 18990693, 17890099, 19769480, 19439759, 19272708, 18990023 19978542, 19329654, 20101006, 21300341, 20402832, 19873610, 20848335 23229229, 21744290, 21668627, 21517440, 13542050, 19304354, 19052488 20794034, 19291380, 21915719, 23260854, 18681056, 20952966, 19896336 19076343, 19561643, 18618122, 19990037, 20440930, 18456643, 19699191 19201867, 19487147, 18909599, 20831538, 19016730, 18250893, 20798891 18743542, 20347562, 16619249, 18354830, 22551446, 19777862, 19687159 21373076, 19174942, 20424899, 21188584, 19989009, 17414008, 20688221 21899588, 20441797, 19157754, 19058490, 19032777, 22815955, 19399918 18885870, 19434529, 21273804, 19018447, 21450666, 18893947, 18851894 16923858, 18417036, 20919320, 19022470, 19284031, 20474192, 20173897 22046677, 22062026, 19501299, 19385656, 20920911, 17274537, 20899461 21315084, 19440586, 16887946, 22374754, 17319928, 19606174, 20708701 18436647, 17655240, 19023822, 19124589, 19178851, 16439813, 19805359 19597583, 18499088, 19155797, 19050649, 19393542

Version 12.1.0.2.v4

Version 12.1.0.2.v4 adds support for the following:

- Oracle PSU 12.1.0.2.160419 (22291127)
- Timezone file DSTv25 (patch 22037014)
- Oracle recommended RDBMS patches for Oracle GoldenGate (patch 17969866)
- Adds the ability for the master user to grant the EM_EXPRESS_BASIC and EM_EXPRESS_ALL roles
- Adds the ability for the master user to grant privileges on SYS objects with the grant option using the RDSADMIN.RDSADMIN_UTIL.GRANT_SYS_OBJECT procedure
- Adds master user privileges to support most common schemas created by the Oracle Fusion Middleware Repository Creation Utility (RCU)

Baseline: Oracle Database Patch Set Update 12.1.0.2.160419 (patch 22291127, released April 2016)

Bugs fixed: 21847223, 19189525, 19075256, 19141838, 22762046, 20117253, 19865345 19791273, 19280225, 18845653, 19248799, 20951038, 19243521, 21756699 18988834, 21281532, 19238590, 18921743, 20245930, 18799063, 19134173 20373598, 19571367, 20476175, 20925795, 19018206, 20711718, 20387265 20509482, 20588502, 19149990, 18849537, 17551063, 18886413, 19183343

19703301, 21917884, 19001390, 18202441, 19189317, 19644859, 19358317 19390567, 19279273, 19706965, 22528741, 19068970, 20825533, 19619732 18607546, 20348653, 19649152, 19670108, 18940497, 18948177, 19315691 19676905, 18964978, 19035573, 20165574, 19176326, 20413820, 20558005 19176223, 19532017, 20904530, 20134339, 19450314, 22353346, 19074147 18411216, 20361671, 20425790, 18966843, 21329301, 20294666, 19333670 19195895, 19307662, 19371175, 20043616, 19154375, 20124446, 18914624 19468991, 19883092, 19382851, 19520602, 19174521, 21875360, 19676012 19326908, 19658708, 19591608, 20093776, 20618595, 21787056, 17835294 19721304, 19791377, 19068610, 22173980, 20746251, 20048359, 19143550 19185876, 19627012, 20281121, 19577410, 22092979, 19001359, 19518079 18610915, 19490948, 18674024, 18306996, 19309466, 19081128, 19915271 20122715, 21188532, 18791688, 20284155, 20890311, 21442094, 20596234 18973548, 19303936, 19597439, 20936905, 20235511, 19888853, 21756677 18964939, 19354335, 19430401, 19044962, 19639483, 21153266, 22353199 19409212, 20657441, 19879746, 19684504, 19024808, 21260431, 21756661 18799993, 20877664, 19028800, 20879889, 19065556, 19723336, 19077215 19604659, 21421886, 19524384, 18288842, 19048007, 19689979, 20446883 18952989, 16870214, 19928926, 19835133, 21526048, 20466628, 19197175 19180770, 19902195, 20318889, 19730508, 19012119, 19067244, 20074391 20356733, 14643995, 19512341, 19841800, 20331945, 19587324, 19547370 19065677, 21225209, 19637186, 20397490, 18967382, 19174430, 19054077 18674047, 19536415, 19708632, 21091431, 19289642, 22168163, 20869721 19335438, 19258504, 20447445, 17365043, 18856999, 19468347, 20471920 19869255, 21620471, 16359751, 18990693, 17890099, 19769480, 19439759 19272708, 18990023, 19978542, 20402832, 20101006, 21300341, 19329654 19873610, 21744290, 13542050, 21517440, 21668627, 19304354, 19052488 20794034, 19291380, 21915719, 18681056, 20952966, 19896336, 19076343 19561643, 19990037, 18618122, 20440930, 18456643, 19699191, 19487147 18909599, 20831538, 18250893, 19016730, 18743542, 20347562, 16619249 18354830, 19777862, 19687159, 19174942, 20424899, 19989009, 20688221 21899588, 20441797, 19157754, 19032777, 19058490, 19399918, 18885870 19434529, 21273804, 19018447, 18893947, 16923858, 18417036, 20919320 19022470, 19284031, 20474192, 22046677, 20173897, 22062026, 19385656 19501299, 17274537, 20899461, 21315084, 19440586, 22374754, 16887946 19606174, 18436647, 17655240, 19023822, 19178851, 19124589, 16439813 19805359, 19597583, 18499088, 19155797, 19050649, 19393542

Version 12.1.0.2.v3

Version 12.1.0.2.v3 adds support for the following:

- Oracle PSU 12.1.0.2.160119 (21948354).
- Timezone file DSTv25 (patch 22037014 for 12.1.0.2). 12.1.0.1 includes DSTv24, patch 20875898 (unchanged from 12.1.0.1.v3), because a backport of DSTv25 was unavailable at build time.
- Fixed an issue that prevented customers from creating more than 10 Directory objects in the database.
- Fixed an issue that prevented customers from re-granting read privileges on the ADUMP and BDUMP Directory objects.

Baseline: Oracle Database Patch Set Update 12.1.0.2.160119 (patch 21948354, released January 2016)

Bugs fixed: 19189525, 19075256, 19141838, 19865345, 19791273, 19280225, 18845653 20951038, 19243521, 19248799, 21756699, 18988834, 19238590, 21281532 20245930, 18921743, 18799063, 19134173, 19571367, 20476175, 20925795 19018206, 20509482, 20387265, 20588502, 19149990, 18849537, 18886413 17551063, 19183343, 19703301, 19001390, 18202441, 19189317, 19644859 19358317, 19390567, 19279273, 19706965, 19068970, 19619732, 20348653 18607546, 18940497, 19670108, 19649152, 18948177, 19315691, 19676905 18964978, 19035573, 20165574, 19176326, 20413820, 20558005, 19176223 19532017, 20134339, 19074147, 18411216, 20361671, 20425790, 18966843 20294666, 19307662, 19371175, 19195895, 19154375, 19468991, 19174521 19520602, 19382851, 21875360, 19326908, 19658708, 20093776, 20618595 21787056, 17835294, 19791377, 19068610, 20048359, 20746251, 19143550 19185876, 19627012, 20281121, 19577410, 22092979, 19001359, 19518079 18610915, 19490948, 18674024, 18306996, 19309466, 19081128, 19915271

20122715, 21188532, 20284155, 18791688, 20890311, 21442094, 18973548 19303936, 19597439, 20235511, 18964939, 19430401, 19044962, 19409212 19879746, 20657441, 19684504, 19024808, 18799993, 20877664, 19028800 19065556, 19723336, 19077215, 19604659, 21421886, 19524384, 19048007 18288842, 19689979, 20446883, 18952989, 16870214, 19928926, 21526048 19180770, 19197175, 19902195, 20318889, 19730508, 19012119, 19067244 20074391, 19512341, 19841800, 14643995, 20331945, 19587324, 19547370 19065677, 19637186, 21225209, 20397490, 18967382, 19174430, 18674047 19054077, 19536415, 19708632, 19289642, 20869721, 19335438, 17365043 18856999, 19869255, 20471920, 19468347, 21620471, 16359751, 18990693 17890099, 19439759, 19769480, 19272708, 19978542, 20101006, 21300341 20402832, 19329654, 19873610, 21668627, 21517440, 19304354, 19052488 20794034, 19291380, 18681056, 19896336, 19076343, 19561643, 18618122 20440930, 18456643, 19699191, 18909599, 19487147, 18250893, 19016730 18743542, 20347562, 16619249, 18354830, 19687159, 19174942, 20424899 19989009, 20688221, 20441797, 19157754, 19032777, 19058490, 19399918 18885870, 19434529, 19018447, 18417036, 20919320, 19022470, 19284031 20474192, 20173897, 22062026, 19385656, 19501299, 17274537, 20899461 19440586, 16887946, 19606174, 18436647, 17655240, 19023822, 19178851 19124589, 19805359, 19597583, 19155797, 19393542, 19050649

Version 12.1.0.2.v2

Version 12.1.0.2.v2 adds support for the following:

- Oracle PSU 12.1.0.2.5 (21359755)
- Includes the Daylight Saving Time Patch, patch 20875898: DST-24, that came out after the April 2015 PSU.

Baseline: Oracle Database Patch Set Update 12.1.0.2.5 (patch 21359755, released October 2015)

Bugs fixed: 19189525, 19075256, 19865345, 19791273, 19280225, 18845653, 19248799 19243521, 18988834, 19238590, 21281532, 18921743, 20245930, 19134173 19571367, 20476175, 20925795, 19018206, 20387265, 19149990, 18849537 19183343, 19703301, 19001390, 18202441, 19189317, 19644859, 19390567 19358317, 19279273, 19706965, 19068970, 19619732, 18607546, 20348653 18940497, 19670108, 19649152, 18948177, 19315691, 19676905, 18964978 20165574, 19035573, 19176326, 20413820, 20558005, 19176223, 19532017 20134339, 19074147, 18411216, 20361671, 20425790, 18966843, 20294666 19371175, 19307662, 19195895, 19154375, 19468991, 19174521, 19520602 19382851, 19658708, 20093776, 17835294, 19068610, 19791377, 20746251 20048359, 19143550, 19185876, 19627012, 20281121, 19577410, 19001359 19518079, 18610915, 18674024, 18306996, 19309466, 19081128, 19915271 20122715, 20284155, 18791688, 21442094, 19303936, 19597439, 20235511 18964939, 19430401, 19044962, 19409212, 20657441, 19684504, 19024808 19028800, 19065556, 19723336, 19077215, 21421886, 19524384, 19048007 18288842, 18952989, 16870214, 19928926, 19180770, 19197175, 19730508 19012119, 19067244, 20074391, 19841800, 19512341, 14643995, 20331945 19587324, 19065677, 19547370, 19637186, 21225209, 20397490, 18967382 19174430, 18674047, 19054077, 19708632, 19536415, 19289642, 19335438 17365043, 18856999, 20471920, 19468347, 21620471, 16359751, 18990693 19439759, 19769480, 19272708, 19978542, 19329654, 20402832, 19873610 19304354, 19052488, 19291380, 18681056, 19896336, 19076343, 19561643 18618122, 20440930, 18456643, 19699191, 18909599, 19487147, 18250893 19016730, 18743542, 20347562, 16619249, 18354830, 19687159, 19174942 20424899, 19989009, 20688221, 20441797, 19157754, 19058490, 19032777 19399918, 18885870, 19434529, 19018447, 18417036, 20919320, 19284031 19022470, 20474192, 22062026, 19385656, 19501299, 17274537, 20899461 19440586, 19606174, 18436647, 19023822, 19178851, 19124589, 19805359 19597583, 19155797, 19393542, 19050649

Version 12.1.0.2.v1

Version 12.1.0.2.v1 adds support for the following:

- Oracle PSU 12.1.0.2.3 (20299023)
- The In-Memory option allows storing a subset of data in an in-memory column format optimized for performance.
- Installs additional Oracle Text knowledge bases from Oracle Database. Examples media (English and French)
- Provides access to DBMS_REPAIR through RDSADMIN.RDSADMIN_DBMS_REPAIR
- Grants ALTER DATABASE LINK, ALTER PUBLIC DATABASE LINK, EXEMPT ACCESS POLICY, EXEMPT IDENTITY POLICY, and EXEMPT REDACTION POLICY to master user

Note
Version 12.1.0.2.v1 supports Enterprise Edition only.

Baseline: Oracle Database Patch Set Update 12.1.0.2.3 (patch 20299023, released April 2015)

Bugs fixed: 19189525, 19065556, 19075256, 19723336, 19077215, 19865345, 18845653 19280225, 19524384, 19248799, 18988834, 19048007, 18288842, 19238590 18921743, 18952989, 16870214, 19928926, 19134173, 19180770, 19018206 19197175, 19149990, 18849537, 19730508, 19183343, 19012119, 19001390 18202441, 19067244, 19189317, 19644859, 19358317, 19390567, 20074391 19279273, 19706965, 19068970, 19841800, 19512341, 14643995, 19619732 20348653, 18607546, 18940497, 19670108, 19649152, 19065677, 19547370 18948177, 19315691, 19637186, 19676905, 18964978, 19035573, 19176326 18967382, 19174430, 19176223, 19532017, 18674047, 19074147, 19054077 19536415, 19708632, 19289642, 20425790, 19335438, 18856999, 19371175 19468347, 19195895, 19154375, 16359751, 18990693, 19439759, 19769480 19272708, 19978542, 19329654, 19873610, 19174521, 19520602, 19382851 19658708, 19304354, 19052488, 19291380, 18681056, 19896336, 17835294 19076343, 19791377, 19068610, 19561643, 18618122, 20440930, 18456643 18909599, 19487147, 19143550, 19185876, 19016730, 18250893, 20347562 19627012, 16619249, 18354830, 19577410, 19687159, 19001359, 19174942 19518079, 18610915, 18674024, 18306996, 19309466, 19081128, 19915271 19157754, 19058490, 20284155, 18791688, 18885870, 19303936, 19434529 19018447, 18417036, 19597439, 20235511, 19022470, 18964939, 19430401 19044962, 19385656, 19501299, 17274537, 19409212, 19440586, 19606174 18436647, 19023822, 19684504, 19178851, 19124589, 19805359, 19024808 19597583, 19155797, 19393542, 19050649, 19028800

Related Topics

- Upgrading the Oracle DB Engine (p. 975)
- Oracle on Amazon RDS (p. 931)

Database Engine: 11.2.0.4

The following versions are available for database engine 11.2.0.4:

- Version 11.2.0.4.v13 (p. 1131)
- Version 11.2.0.4.v12 (p. 1132)
- Version 11.2.0.4.v11 (p. 1133)
- Version 11.2.0.4.v10 (p. 1134)
- Version 11.2.0.4.v9 (p. 1136)
- Version 11.2.0.4.v8 (p. 1137)
- Version 11.2.0.4.v7 (p. 1138)
- Version 11.2.0.4.v6 (p. 1139)

Version 11.2.0.4.v13

Version 11.2.0.4.v13 adds support for the following:

- Oracle July 2017 PSU, a combination of database PSU (patch 26609445) + OJVM component PSU (patch 26027154)
- Oracle recommended RDBMS patches for Oracle GoldenGate (patch 26554712)
- RSA Micro-Edition Suite Bundle (patch 26770426)
- Timezone file DSTv30 (patch 25881255, OJVM patch 25881271)
- Adds support for Validating DB Instance Files (p. 1058) with the `RMAN` logical validation utility
- Adds support for Setting the Default Edition for a DB Instance (p. 1058)

Oracle patch 26609445, released July 2017

Bugs fixed: 17288409, 21051852, 24316947, 17811429, 17205719, 18607546, 20506699 17816865, 25957038, 23330119, 17922254, 17754782, 13364795, 16934803 17311728, 20387265, 17284817, 17441661, 24560906, 16992075, 17446237 14015842, 19972569, 21756677, 17375354, 21538558, 20925795, 17449815 26575788, 19463897, 13866822, 17235750, 17982555, 17478514, 18317531 14338435, 18235390, 20803583, 19461270, 13944971, 20142975, 17811789 16929165, 18704244, 20506706, 17546973, 20334344, 14054676, 17088068 17346091, 18264060, 17343514, 21538567, 19680952, 18471685, 19211724 13951456, 21847223, 16315398, 18744139, 16850630, 23177648, 19049453 18673304, 17883081, 19915271, 18641419, 18262334, 25600421, 17006183 16065166, 18277454, 16833527, 10136473, 18051556, 17865671, 17852463 18554871, 17853498, 18334586, 20879889, 17551709, 17588480, 19827973 17344412, 17842825, 18828868, 20509482, 17025461, 11883252, 13609098 17239687, 17602269, 19197175, 18316692, 22195457, 17313525, 12611721 19544839, 18964939, 17600719, 18191164, 19393542, 17571306, 20777150 18482502, 19466309, 22243719, 17040527, 17165204, 18098207, 16785708 17465741, 16180763, 17174582, 12982566, 16777840, 19463893, 22195465 16875449, 22148226, 12816846, 17237521, 6599380, 19358317, 17811438 25505394, 17811447, 17945983, 21983325, 18762750, 16912439, 17184721 18061914, 17282229, 18331850, 18202441, 17082359, 18723434, 21972320 19554106, 25505371, 14034426, 18339044, 19458377, 17752995, 20448824 17891943, 17258090, 17767676, 16668584, 18384391, 17040764, 17381384 15913355, 18356166, 14084247, 20596234, 20506715, 21756661, 13853126 18203837, 14245531, 16043574, 21756699, 22195441, 17848897, 17877323 21453153, 17468141, 20861693, 17786518, 17912217, 17037130, 16956380 18155762, 17478145, 17394950, 18641461, 18189036, 18619917, 17027426 21352646, 16268425, 24476274, 22195492, 19584068, 26544823, 18436307 22507210, 17265217, 17634921, 13498382, 19469538, 21526048, 19258504 18043064, 20004087, 17443671, 22195485, 18000422, 20004021, 22321756 17571039, 21067387, 16832076, 22905130, 16344544, 18009564, 14354737 21286665, 18135678, 14521849, 18614015, 20441797, 18362222, 25655390 16472716, 17835048, 17050888, 17936109, 14010183, 17325413, 18747196 17761775, 16721594, 17082983, 20067212, 21179898, 17302277, 18084625 15990359, 24842886, 18203835, 17297939, 17811456, 16731148, 22380919 21168487, 14133975, 13829543, 17215560, 17694209, 17385178, 18091059 8322815, 18259031, 19689979, 17586955, 17201159, 17655634, 18331812 19730508, 18868646, 17648596, 16220077, 16069901, 17348614, 17393915 17957017, 17274537, 18096714, 17308789, 18436647, 14285317, 19289642 14764829, 17622427, 18328509, 16943711, 22195477, 14368995, 22502493 17346671, 18996843, 17783588, 21343838, 16618694, 17672719, 18856999 18783224, 17851160, 17546761, 22168163, 17798953, 18273830, 22092979

16596890, 19972566, 13871092, 17726838, 16384983, 22296366, 17360606 22321741, 13645875, 25879656, 18199537, 16542886, 21787056, 17889549 14565184, 17071721, 17610798, 20299015, 21343897, 22893153, 20657441 17397545, 18230522, 16360112, 19769489, 12905058, 18641451, 12747740 18430495, 25423453, 17016369, 17042658, 14602788, 17551063, 19972568 21517440, 19788842, 18508861, 14657740, 17332800, 13837378, 17186905 19972564, 19699191, 18315328, 17437634, 22353199, 18093615, 19006849 19013183, 17296856, 18674024, 17232014, 16855292, 17762296, 14692762 21051840, 17705023, 22507234, 19121551, 21330264, 19854503, 26030218 21868720, 19309466, 18681862, 17365043, 20558005, 18554763, 17390160 18456514, 16306373, 13955826, 18139690, 17501491, 17752121, 21668627 17299889, 17889583, 18673325, 19721304, 18293054, 17242746, 19888853 17951233, 18094246, 17649265, 19615136, 17011832, 16870214, 17477958 18522509, 20631274, 16091637, 17323222, 16595641, 16524926, 18228645 18282562, 17596908, 18031668, 17156148, 16494615, 22683225, 17545847 25093656, 17655240, 24528741, 17614134, 25427662, 13558557, 17341326 17891946, 17716305, 22657942, 18440095, 16392068, 19271443, 21351877 18092127, 17614227, 18440047, 16903536, 14106803, 18973907, 18673342 17389192, 25505382, 19032867, 17612828, 16194160, 17006570, 25369547 25505407, 16685417, 17721717, 17390431, 17570240, 16863422, 18325460 19727057, 16422541, 19972570, 17267114, 18244962, 21538485, 18203838 18765602, 16198143, 17246576, 14829250, 17835627, 18247991, 14458214 21051862, 16692232, 17786278, 17227277, 24476265, 16042673, 16314254 16228604, 16837842, 17393683, 23536835, 25823754, 18899974, 17787259 20331945, 20074391, 15861775, 16399083, 18018515, 22683212, 21051858 18260550, 17080436, 16613964, 17036973, 16579084, 24433711, 18384537 18280813, 20296213, 16901385, 15979965, 23330124, 18441944, 16450169 9756271, 17892268, 11733603, 16285691, 17587063, 21343775, 18180390 16538760, 18193833, 21387964, 21051833, 17238511, 19777862, 17824637 16571443, 18306996, 19578350, 14852021, 17853456, 18674047, 12364061 24411921, 19207117, 22195448

Version 11.2.0.4.v12

Version 11.2.0.4.v12 adds support for the following:

- Oracle patch 25440428, a combination of database PSU (patch 24732075) + OJVM component PSU (patch 25434033)
- Oracle recommended RDBMS patches for Oracle GoldenGate (patch 25734992)
- MES Bundle (patch 24975421 for 11.2.0.4)
- Timezone file DSTv28 (patch 24701840)
- Adds support for the `DBMS_CHANGE_NOTIFICATION` package
- Adds support for `XSTREAM` packages and views (may require additional licensing)

Oracle patch 24732075, released April 2017

Bugs fixed: 17288409, 21051852, 24316947, 17811429, 17205719, 18607546, 20506699 17816865, 17922254, 23330119, 17754782, 16934803, 13364795, 17311728 17284817, 17441661, 24560906, 16992075, 17446237, 14015842, 19972569 21756677, 17375354, 20925795, 21538558, 17449815, 19463897, 13866822 17235750, 17982555, 17478514, 18317531, 14338435, 18235390, 20803583 13944971, 20142975, 17811789, 16929165, 18704244, 20506706, 17546973 20334344, 14054676, 17088068, 17346091, 18264060, 17343514, 21538567 19680952, 18471685, 19211724, 13951456, 21847223, 16315398, 18744139 16850630, 23177648, 19049453, 18673304, 17883081, 19915271, 18641419 18262334, 17006183, 16065166, 18277454, 16833527, 10136473, 18051556 17865671, 17852463, 18554871, 17853498, 18334586, 17551709, 17588480 19827973, 17344412, 17842825, 18828868, 17025461, 11883252, 13609098 17239687, 17602269, 19197175, 18316692, 22195457, 17313525, 12611721 19544839, 18964939, 17600719, 18191164, 19393542, 17571306, 20777150 18482502, 19466309, 22243719, 17040527, 17165204, 18098207, 16785708 17465741, 17174582, 16180763, 12982566, 16777840, 19463893, 22195465 16875449, 12816846, 22148226, 17237521, 6599380, 19358317, 25505394 17811438, 17811447, 17945983, 21983325, 18762750, 16912439, 17184721 18061914, 17282229, 18331850, 18202441, 17082359, 18723434, 21972320 19554106,

25505371, 14034426, 18339044, 19458377, 17752995, 20448824 17891943, 17258090, 17767676, 16668584, 18384391, 17040764, 17381384 15913355, 18356166, 14084247, 20596234, 20506715, 21756661, 13853126 18203837, 14245531, 16043574, 21756699, 22195441, 17848897, 17877323 21453153, 17468141, 20861693, 17786518, 17912217, 17037130, 16956380 18155762, 17478145, 17394950, 18641461, 18189036, 18619917, 17027426 21352646, 16268425, 24476274, 22195492, 19584068, 18436307, 22507210 17265217, 17634921, 13498382, 21526048, 19258504, 20004087, 17443671 22195485, 18000422, 22321756, 20004021, 17571039, 21067387, 22905130 16344544, 18009564, 14354737, 21286665, 18135678, 18614015, 20441797 18362222, 17835048, 16472716, 17936109, 17050888, 14010183, 17325413 18747196, 17761775, 16721594, 17082983, 20067212, 21179898, 17302277 18084625, 15990359, 24842886, 18203835, 17297939, 17811456, 22380919 16731148, 21168487, 14133975, 13829543, 17215560, 17694209, 17385178 18091059, 8322815, 17586955, 17201159, 17655634, 18331812, 19730508 18868646, 17648596, 16220077, 16069901, 17348614, 17393915, 17274537 17957017, 18096714, 17308789, 18436647, 14285317, 19289642, 14764829 17622427, 18328509, 16943711, 22195477, 14368995, 22502493, 17346671 18996843, 17783588, 21343838, 16618694, 17672719, 18856999, 18783224 17851160, 17546761, 17798953, 18273830, 22092979, 16596890, 19972566 16384983, 17726838, 22296366, 17360606, 22321741, 13645875, 18199537 16542886, 21787056, 17889549, 14565184, 17071721, 17610798, 20299015 21343897, 22893153, 20657441, 17397545, 18230522, 16360112, 19769489 12905058, 18641451, 12747740, 18430495, 17016369, 17042658, 14602788 17551063, 19972568, 21517440, 18508861, 19788842, 14657740, 17332800 13837378, 19972564, 17186905, 18315328, 19699191, 17437634, 22353199 18093615, 19006849, 19013183, 17296856, 18674024, 17232014, 16855292 17762296, 14692762, 21051840, 17705023, 22507234, 19121551, 21330264 19854503, 21868720, 19309466, 18681862, 20558005, 18554763, 17390160 18456514, 16306373, 13955826, 18139690, 17501491, 17752121, 21668627 17299889, 17889583, 18673325, 19721304, 18293054, 17242746, 17951233 18094246, 17649265, 19615136, 17011832, 16870214, 17477958, 18522509 20631274, 16091637, 17323222, 16595641, 16524926, 18228645, 18282562 17596908, 18031668, 17156148, 16494615, 22683225, 17545847, 25093656 17655240, 24528741, 17614134, 13558557, 17341326, 17891946, 17716305 22657942, 18440095, 16392068, 19271443, 21351877, 18092127, 17614227 18440047, 16903536, 14106803, 18973907, 18673342, 25505382, 19032867 17389192, 17612828, 16194160, 17006570, 25369547, 25505407, 17721717 17390431, 17570240, 16863422, 18325460, 19727057, 16422541, 19972570 17267114, 18244962, 21538485, 18765602, 18203838, 16198143, 17246576 14829250, 17835627, 18247991, 14458214, 21051862, 16692232, 17786278 17227277, 24476265, 16042673, 16314254, 16228604, 16837842, 17393683 23536835, 17787259, 20331945, 20074391, 15861775, 16399083, 18018515 22683212, 18260550, 21051858, 17080436, 16613964, 17036973, 16579084 24433711, 18384537, 18280813, 20296213, 16901385, 15979965, 23330124 18441944, 16450169, 9756271, 17892268, 11733603, 16285691, 17587063 21343775, 18180390, 16538760, 18193833, 21387964, 21051833, 17238511 17824637, 16571443, 18306996, 14852021, 17853456, 18674047, 12364061 24411921, 22195448

Version 11.2.0.4.v11

Version 11.2.0.4.v11 adds support for the following:

- Oracle patch 24918033, a combination of database PSU (patch 24006111) + OJVM component PSU (patch 24917954)
- Oracle recommended RDBMS patches for Oracle GoldenGate (patch 24491261)
- MES Bundle (patch 24975421 for 11.2.0.4)

Oracle patch 24918033, released January 2017

Bugs fixed: 18933818, 19176885, 17201047, 25067795, 14774730, 19153980, 21911849 23727132, 18166577, 24448240, 17056813, 21811517, 19909862, 22675136 24534298, 19895326, 22253904, 17804361, 19231857, 17528315, 19058059 19554117, 19007266, 17285560, 22670385, 18458318, 19187988, 23265914 19006757, 19374518, 19223010, 25076732, 22118835, 19852360, 20408829 21047766, 21566944, 17288409, 21051852, 24316947, 17811429, 18607546, 17205719, 20506699 17816865, 17922254, 23330119, 17754782, 16934803, 13364795, 17311728 17441661, 17284817,

16992075, 17446237, 14015842, 19972569, 21756677 17375354, 20925795, 21538558, 17449815, 19463897, 13866822, 17235750 17982555, 17478514, 18317531, 14338435, 18235390, 20803583, 13944971 20142975, 17811789, 16929165, 18704244, 20506706, 17546973, 20334344 14054676, 17088068, 17346091, 18264060, 17343514, 21538567, 19680952 18471685, 19211724, 13951456, 21847223, 16315398, 18744139, 16850630 23177648, 19049453, 18673304, 17883081, 19915271, 18641419, 18262334 17006183, 16065166, 18277454, 16833527, 10136473, 18051556, 17865671 17852463, 18554871, 17853498, 18334586, 17551709, 17588480, 19827973 17344412, 17842825, 18828868, 17025461, 11883252, 13609098, 17239687 17602269, 19197175, 22195457, 18316692, 17313525, 12611721, 19544839 18964939, 17600719, 18191164, 19393542, 17571306, 20777150, 18482502 19466309, 22243719, 17040527, 17165204, 18098207, 16785708, 17465741 17174582, 16180763, 16777840, 12982566, 19463893, 22195465, 22148226 16875449, 12816846, 17237521, 6599380, 19358317, 17811438, 17811447 17945983, 21983325, 18762750, 16912439, 17184721, 18061914, 17282229 18331850, 18202441, 17082359, 18723434, 21972320, 19554106, 14034426 18339044, 19458377, 17752995, 20448824, 17891943, 17258090, 17767676 16668584, 18384391, 17040764, 17381384, 15913355, 18356166, 14084247 20596234, 20506715, 21756661, 13853126, 18203837, 14245531, 16043574 21756699, 22195441, 17848897, 17877323, 21453153, 17468141, 20861693 17786518, 17912217, 17037130, 16956380, 18155762, 17478145, 17394950 18641461, 18189036, 18619917, 17027426, 21352646, 16268425, 24476274 22195492, 19584068, 18436307, 22507210, 17265217, 17634921, 13498382 21526048, 19258504, 20004087, 17443671, 22195485, 18000422, 22321756 20004021, 17571039, 21067387, 16344544, 18009564, 14354737, 21286665 18135678, 18614015, 20441797, 18362222, 17835048, 16472716, 17936109 17050888, 17325413, 14010183, 18747196, 17761775, 16721594, 17082983 20067212, 21179898, 17302277, 18084625, 15990359, 18203835, 17297939 17811456, 22380919, 16731148, 21168487, 14133975, 13829543, 17215560 17694209, 17385178, 18091059, 8322815, 17586955, 17201159, 17655634 18331812, 19730508, 18868646, 17648596, 16220077, 16069901, 17348614 17393915, 17274537, 17957017, 18096714, 17308789, 18436647, 14285317 19289642, 14764829, 18328509, 17622427, 16943711, 22195477, 14368995 22502493, 17346671, 18996843, 17783588, 21343838, 16618694, 17672719 18856999, 18783224, 17851160, 17546761, 17798953, 18273830, 22092979 16596890, 19972566, 16384983, 17726838, 22296366, 17360606, 22321741 13645875, 18199537, 16542886, 21787056, 17889549, 14565184, 17071721 17610798, 20299015, 21343897, 22893153, 20657441, 17397545, 18230522 16360112, 19769489, 12905058, 18641451, 12747740, 18430495, 17016369 17042658, 14602788, 17551063, 19972568, 21517440, 18508861, 19788842 14657740, 17332800, 13837378, 19972564, 17186905, 18315328, 19699191 17437634, 22353199, 18093615, 19006849, 19013183, 17296856, 18674024 17232014, 16855292, 17762296, 14692762, 21051840, 17705023, 22507234 19121551, 21330264, 19854503, 21868720, 19309466, 18681862, 20558005 18554763, 17390160, 18456514, 16306373, 13955826, 18139690, 17501491 17752121, 21668627, 17299889, 17889583, 18673325, 19721304, 18293054 17242746, 17951233, 18094246, 17649265, 19615136, 17011832, 16870214 17477958, 18522509, 20631274, 16091637, 17323222, 16595641, 16524926 18228645, 18282562, 17596908, 18031668, 17156148, 16494615, 22683225 17545847, 17655240, 24528741, 17614134, 13558557, 17341326, 17891946 17716305, 22657942, 16392068, 19271443, 21351877, 18092127, 17614227 18440047, 16903536, 14106803, 18973907, 18673342, 19032867, 17389192 17612828, 16194160, 17006570, 17721717, 17390431, 17570240, 16863422 18325460, 19727057, 16422541, 19972570, 17267114, 18244962, 21538485 18765602, 18203838, 16198143, 17246576, 14829250, 17835627, 18247991 14458214, 21051862, 16692232, 17786278, 17227277, 24476265, 16042673 16314254, 16228604, 16837842, 17393683, 23536835, 17787259, 20331945 20074391, 15861775, 16399083, 18018515, 22683212, 18260550, 21051858 17080436, 16613964, 17036973, 16579084, 24433711, 18384537, 18280813 20296213, 16901385, 15979965, 23330124, 18441944, 16450169, 9756271 17892268, 11733603, 16285691, 17587063, 21343775, 18180390, 16538760 18193833, 21387964, 21051833, 17238511, 17824637, 16571443, 18306996 14852021, 17853456, 18674047, 12364061, 22195448

Version 11.2.0.4.v10

Version 11.2.0.4.v10 adds support for the following:

- Oracle patch 24436313, a combination of database PSU (patch 24006111) + OJVM component PSU (patch 24315821)

- Oracle recommended RDBMS patches for Oracle GoldenGate (patch 24491261)
- MES Bundle (patch 24975421 for 11.2.0.4)

Baseline: Oracle Database Patch Set Update 11.2.0.4.161018 (patch 24006111, released October 2016)

Bugs fixed: 17288409, 21051852, 24316947, 17811429, 18607546, 17205719, 20506699 17816865, 17922254, 23330119, 17754782, 16934803, 13364795, 17311728 17441661, 17284817, 16992075, 17446237, 14015842, 19972569, 21756677 17375354, 20925795, 21538558, 17449815, 19463897, 13866822, 17235750 17982555, 17478514, 18317531, 14338435, 18235390, 20803583, 13944971 20142975, 17811789, 16929165, 18704244, 20506706, 17546973, 20334344 14054676, 17088068, 17346091, 18264060, 17343514, 21538567, 19680952 18471685, 19211724, 13951456, 21847223, 16315398, 18744139, 16850630 23177648, 19049453, 18673304, 17883081, 19915271, 18641419, 18262334 17006183, 16065166, 18277454, 16833527, 10136473, 18051556, 17865671 17852463, 18554871, 17853498, 18334586, 17551709, 17588480, 19827973 17344412, 17842825, 18828868, 17025461, 11883252, 13609098, 17239687 17602269, 19197175, 22195457, 18316692, 17313525, 12611721, 19544839 18964939, 17600719, 18191164, 19393542, 17571306, 20777150, 18482502 19466309, 22243719, 17040527, 17165204, 18098207, 16785708, 17465741 17174582, 16180763, 16777840, 12982566, 19463893, 22195465, 22148226 16875449, 12816846, 17237521, 6599380, 19358317, 17811438, 17811447 17945983, 21983325, 18762750, 16912439, 17184721, 18061914, 17282229 18331850, 18202441, 17082359, 18723434, 21972320, 19554106, 14034426 18339044, 19458377, 17752995, 20448824, 17891943, 17258090, 17767676 16668584, 18384391, 17040764, 17381384, 15913355, 18356166, 14084247 20596234, 20506715, 21756661, 13853126, 18203837, 14245531, 16043574 21756699, 22195441, 17848897, 17877323, 21453153, 17468141, 20861693 17786518, 17912217, 17037130, 16956380, 18155762, 17478145, 17394950 18641461, 18189036, 18619917, 17027426, 21352646, 16268425, 24476274 22195492, 19584068, 18436307, 22507210, 17265217, 17634921, 13498382 21526048, 19258504, 20004087, 17443671, 22195485, 18000422, 22321756 20004021, 17571039, 21067387, 16344544, 18009564, 14354737, 21286665 18135678, 18614015, 20441797, 18362222, 17835048, 16472716, 17936109 17050888, 17325413, 14010183, 18747196, 17761775, 16721594, 17082983 20067212, 21179898, 17302277, 18084625, 15990359, 18203835, 17297939 17811456, 22380919, 16731148, 21168487, 14133975, 13829543, 17215560 17694209, 17385178, 18091059, 8322815, 17586955, 17201159, 17655634 18331812, 19730508, 18868646, 17648596, 16220077, 16069901, 17348614 17393915, 17274537, 17957017, 18096714, 17308789, 18436647, 14285317 19289642, 14764829, 18328509, 17622427, 16943711, 22195477, 14368995 22502493, 17346671, 18996843, 17783588, 21343838, 16618694, 17672719 18856999, 18783224, 17851160, 17546761, 17798953, 18273830, 22092979 16596890, 19972566, 16384983, 17726838, 22296366, 17360606, 22321741 13645875, 18199537, 16542886, 21787056, 17889549, 14565184, 17071721 17610798, 20299015, 21343897, 22893153, 20657441, 17397545, 18230522 16360112, 19769489, 12905058, 18641451, 12747740, 18430495, 17016369 17042658, 14602788, 17551063, 19972568, 21517440, 18508861, 19788842 14657740, 17332800, 13837378, 19972564, 17186905, 18315328, 19699191 17437634, 22353199, 18093615, 19006849, 19013183, 17296856, 18674024 17232014, 16855292, 17762296, 14692762, 21051840, 17705023, 22507234 19121551, 21330264, 19854503, 21868720, 19309466, 18681862, 20558005 18554763, 17390160, 18456514, 16306373, 13955826, 18139690, 17501491 17752121, 21668627, 17299889, 17889583, 18673325, 19721304, 18293054 17242746, 17951233, 18094246, 17649265, 19615136, 17011832, 16870214 17477958, 18522509, 20631274, 16091637, 17323222, 16595641, 16524926 18228645, 18282562, 17596908, 18031668, 17156148, 16494615, 22683225 17545847, 17655240, 24528741, 17614134, 13558557, 17341326, 17891946 17716305, 22657942, 16392068, 19271443, 21351877, 18092127, 17614227 18440047, 16903536, 14106803, 18973907, 18673342, 19032867, 17389192 17612828, 16194160, 17006570, 17721717, 17390431, 17570240, 16863422 18325460, 19727057, 16422541, 19972570, 17267114, 18244962, 21538485 18765602, 18203838, 16198143, 17246576, 14829250, 17835627, 18247991 14458214, 21051862, 16692232, 17786278, 17227277, 24476265, 16042673 16314254, 16228604, 16837842, 17393683, 23536835, 17787259, 20331945 20074391, 15861775, 16399083, 18018515, 22683212, 18260550, 21051858 17080436, 16613964, 17036973, 16579084, 24433711, 18384537, 18280813 20296213, 16901385, 15979965, 23330124, 18441944, 16450169, 9756271 17892268, 11733603, 16285691, 17587063, 21343775, 18180390, 16538760 18193833,

21387964, 21051833, 17238511, 17824637, 16571443, 18306996 14852021, 17853456, 18674047, 12364061, 22195448

Version 11.2.0.4.v9

Version 11.2.0.4.v9 adds support for the following:

- Oracle patch 23615392, a combination of database PSU (patch 23054359) + OJVM component PSU (patch 23177551)
- Timezone file DSTv26 (patch 22873635 for 11.2.0.4)
- Oracle recommended RDBMS patches for Oracle GoldenGate (patch 24320398 for 11.2.0.4.160719)
- MES Bundle (patch 22695784 for 11.2.0.4)
- Added the ability to create custom password verify functions. For more information, see Creating Custom Functions to Verify Passwords (p. 1049).
- Fixed a bug that prevented implicit recompilation of views owned by SYS

Baseline: Oracle Database Patch Set Update 11.2.0.4.160719 (patch 23054359, released July 2016)

Bugs fixed: 17288409, 21051852, 17811429, 18607546, 17205719, 20506699, 17816865 23330119, 17922254, 17754782, 16934803, 13364795, 17311728, 17441661 17284817, 16992075, 17446237, 14015842, 19972569, 21756677, 17375354 21538558, 20925795, 17449815, 19463897, 13866822, 17982555, 17235750 17478514, 18317531, 14338435, 18235390, 20803583, 13944971, 20142975 17811789, 16929165, 18704244, 20506706, 17546973, 20334344, 14054676 17088068, 17346091, 18264060, 17343514, 21538567, 19680952, 18471685 19211724, 13951456, 21847223, 16315398, 18744139, 16850630, 23177648 19049453, 18673304, 17883081, 19915271, 18641419, 18262334, 17006183 16065166, 18277454, 16833527, 10136473, 18051556, 17865671, 17852463 18554871, 17853498, 18334586, 17551709, 17588480, 19827973, 17344412 17842825, 18828868, 17025461, 11883252, 13609098, 17239687, 17602269 19197175, 22195457, 18316692, 17313525, 12611721, 19544839, 18964939 17600719, 18191164, 19393542, 17571306, 18482502, 20777150, 19466309 17040527, 17165204, 18098207, 16785708, 17465741, 17174582, 16180763 16777840, 12982566, 19463893, 22195465, 16875449, 12816846, 17237521 19358317, 17811438, 17811447, 17945983, 21983325, 18762750, 16912439 17184721, 18061914, 17282229, 18331850, 18202441, 17082359, 18723434 21972320, 19554106, 14034426, 18339044, 19458377, 17752995, 20448824 17891943, 17258090, 17767676, 16668584, 18384391, 17040764, 17381384 15913355, 18356166, 14084247, 20596234, 20506715, 21756661, 13853126 18203837, 14245531, 16043574, 21756699, 22195441, 17848897, 17877323 21453153, 17468141, 20861693, 17786518, 17912217, 17037130, 16956380 18155762, 17478145, 17394950, 18641461, 18189036, 18619917, 17027426 21352646, 16268425, 22195492, 19584068, 18436307, 22507210, 17265217 17634921, 13498382, 21526048, 19258504, 20004087, 17443671, 22195485 18000422, 22321756, 20004021, 17571039, 21067387, 16344544, 18009564 14354737, 21286665, 18135678, 18614015, 20441797, 18362222, 17835048 16472716, 17936109, 17050888, 17325413, 14010183, 18747196, 17761775 16721594, 17082983, 20067212, 21179898, 17302277, 18084625, 15990359 18203835, 17297939, 22380919, 17811456, 16731148, 21168487, 13829543 17215560, 14133975, 17694209, 17385178, 18091059, 8322815, 17586955 17201159, 17655634, 18331812, 19730508, 18868646, 17648596, 16220077 16069901, 17348614, 17393915, 17274537, 17957017, 18096714, 17308789 18436647, 14285317, 19289642, 14764829, 18328509, 17622427, 16943711 22195477, 14368995, 22502493, 17346671, 18996843, 17783588, 21343838 16618694, 17672719, 18856999, 18783224, 17851160, 17546761, 17798953 18273830, 22092979, 16596890, 19972566, 16384983, 17726838, 22296366 17360606, 22321741, 13645875, 18199537, 16542886, 21787056, 17889549 14565184, 17071721, 17610798, 20299015, 21343897, 22893153, 20657441 17397545, 18230522, 16360112, 19769489, 12905058, 18641451, 12747740 18430495, 17016369, 17042658, 14602788, 17551063, 19972568, 21517440 18508861, 19788842, 14657740, 17332800, 13837378, 19972564, 17186905 18315328, 19699191, 17437634, 22353199, 18093615, 19006849, 19013183 17296856, 18674024, 17232014, 16855292, 17762296, 14692762,

21051840 17705023, 22507234, 19121551, 21330264, 19854503, 21868720, 19309466 18681862, 18554763, 20558005, 17390160, 18456514, 16306373, 13955826 18139690, 17501491, 17752121, 21668627, 17299889, 17889583, 18673325 19721304, 18293054, 17242746, 17951233, 18094246, 17649265, 19615136 17011832, 16870214, 17477958, 18522509, 20631274, 16091637, 17323222 16595641, 16524926, 18228645, 18282562, 17596908, 18031668, 17156148 16494615, 22683225, 17545847, 17655240, 17614134, 13558557, 17341326 17891946, 17716305, 16392068, 19271443, 21351877, 18092127, 17614227 18440047, 16903536, 14106803, 18973907, 18673342, 19032867, 17389192 17612828, 16194160, 17006570, 17721717, 17390431, 17570240, 16863422 18325460, 19727057, 16422541, 19972570, 17267114, 18244962, 21538485 18765602, 18203838, 16198143, 17246576, 14829250, 17835627, 18247991 14458214, 21051862, 16692232, 17786278, 17227277, 16042673, 16314254 16228604, 16837842, 17393683, 23536835, 17787259, 20331945, 20074391 15861775, 16399083, 18018515, 22683212, 18260550, 21051858, 17080436 16613964, 17036973, 16579084, 18384537, 18280813, 20296213, 16901385 15979965, 23330124, 18441944, 16450169, 9756271, 17892268, 11733603 16285691, 17587063, 21343775, 16538760, 18180390, 18193833, 21387964 21051833, 17238511, 17824637, 16571443, 18306996, 14852021, 17853456 18674047, 12364061, 22195448

Version 11.2.0.4.v8

Version 11.2.0.4.v8 adds support for the following:

- Oracle PSU 11.2.0.4.160419 (22502456)
- Timezone file DSTv25 (patch 22037014)
- Oracle recommended RDBMS patches for Oracle GoldenGate (patch 22576728)
- Adds the ability for the master user to grant privileges on SYS objects with the grant option using the RDSADMIN.RDSADMIN_UTIL.GRANT_SYS_OBJECT procedure
- Adds master user privileges to support most common schemas created by the Oracle Fusion Middleware Repository Creation Utility (RCU)

Baseline: Oracle Database Patch Set Update 11.2.0.4.160419 (patch 22502456, released April 2016)

Bugs fixed: 17288409, 21051852, 17811429, 18607546, 17205719, 20506699, 17816865 17922254, 17754782, 16934803, 13364795, 17311728, 17441661, 17284817 16992075, 17446237, 14015842, 19972569, 21756677, 21538558, 20925795 17449815, 17375354, 19463897, 13866822, 17982555, 17235750, 17478514 18317531, 14338435, 18235390, 20803583, 13944971, 20142975, 17811789 16929165, 18704244, 20506706, 17546973, 20334344, 14054676, 17088068 17346091, 18264060, 17343514, 21538567, 19680952, 18471685, 19211724 13951456, 21847223, 16315398, 18744139, 16850630, 19049453, 18673304 17883081, 19915271, 18641419, 18262334, 17006183, 16065166, 18277454 16833527, 10136473, 18051556, 17865671, 17852463, 18554871, 17853498 18334586, 17551709, 17588480, 19827973, 17344412, 17842825, 18828868 17025461, 11883252, 13609098, 17239687, 17602269, 19197175, 22195457 18316692, 17313525, 12611721, 19544839, 18964939, 17600719, 18191164 19393542, 17571306, 18482502, 20777150, 19466309, 17040527, 17165204 18098207, 16785708, 17465741, 17174582, 16180763, 16777840, 12982566 19463893, 22195465, 16875449, 12816846, 17237521, 19358317, 17811438 17811447, 21983325, 17945983, 18762750, 16912439, 17184721, 18061914 17282229, 18331850, 18202441, 17082359, 18723434, 21972320, 19554106 14034426, 18339044, 19458377, 17752995, 20448824, 17891943, 17258090 17767676, 16668584, 18384391, 17040764, 17381384, 15913355, 18356166 14084247, 20596234, 20506715, 21756661, 13853126, 18203837, 14245531 21756699, 16043574, 22195441, 17848897, 17877323, 21453153, 17468141 20861693, 17786518, 17912217, 17037130, 18155762, 16956380, 17478145 17394950, 18641461, 18189036, 18619917, 17027426, 21352646, 16268425 22195492, 19584068, 18436307, 17265217, 17634921, 13498382, 21526048 19258504, 20004087, 17443671, 22195485, 18000422, 20004021, 22321756 17571039, 21067387, 16344544, 18009564, 14354737, 21286665, 18135678 18614015, 20441797, 18362222, 17835048, 16472716, 17936109, 17050888 17325413, 14010183, 18747196, 17761775, 16721594, 17082983, 20067212 21179898, 17302277, 18084625,

15990359, 18203835, 17297939, 17811456 16731148, 21168487, 13829543, 17215560, 14133975, 17694209, 17385178 18091059, 8322815, 17586955, 17201159, 17655634, 18331812, 19730508 18868646, 17648596, 16220077, 16069901, 17348614, 17393915, 17274537 17957017, 18096714, 17308789, 18436647, 14285317, 19289642, 14764829 18328509, 17622427, 22195477, 16943711, 22502493, 14368995, 17346671 18996843, 17783588, 21343838, 16618694, 17672719, 18856999, 18783224 17851160, 17546761, 17798953, 18273830, 22092979, 16596890, 19972566 16384983, 17726838, 17360606, 22321741, 13645875, 18199537, 16542886 21787056, 17889549, 14565184, 17071721, 17610798, 20299015, 21343897 22893153, 20657441, 17397545, 18230522, 16360112, 19769489, 12905058 18641451, 12747740, 18430495, 17016369, 17042658, 14602788, 17551063 19972568, 21517440, 18508861, 19788842, 14657740, 17332800, 13837378 19972564, 17186905, 18315328, 19699191, 17437634, 22353199, 18093615 19006849, 19013183, 17296856, 18674024, 17232014, 16855292, 17762296 14692762, 21051840, 17705023, 19121551, 21330264, 19854503, 21868720 19309466, 18681862, 18554763, 20558005, 17390160, 18456514, 16306373 13955826, 18139690, 17501491, 17752121, 21668627, 17299889, 17889583 18673325, 19721304, 18293054, 17242746, 17951233, 17649265, 18094246 19615136, 17011832, 16870214, 17477958, 18522509, 20631274, 16091637 17323222, 16595641, 16524926, 18228645, 18282562, 17596908, 17156148 18031668, 16494615, 22683225, 17545847, 17655240, 17614134, 13558557 17341326, 17891946, 17716305, 16392068, 19271443, 21351877, 18092127 18440047, 17614227, 14106803, 16903536, 18973907, 18673342, 19032867 17389192, 17612828, 16194160, 17006570, 17721717, 17390431, 17570240 16863422, 18325460, 19727057, 16422541, 19972570, 17267114, 18244962 21538485, 18765602, 18203838, 16198143, 17246576, 14829250, 17835627 18247991, 14458214, 21051862, 16692232, 17786278, 17227277, 16042673 16314254, 16228604, 16837842, 17393683, 17787259, 20331945, 20074391 15861775, 16399083, 18018515, 22683212, 18260550, 21051858, 17036973 16613964, 17080436, 16579084, 18384537, 18280813, 20296213, 16901385 15979965, 18441944, 16450169, 9756271, 17892268, 11733603, 16285691 17587063, 21343775, 16538760, 18180390, 18193833, 21387964, 21051833 17238511, 17824637, 16571443, 18306996, 14852021, 18674047, 17853456 12364061, 22195448

Version 11.2.0.4.v7

Version 11.2.0.4.v7 adds support for the following:

- Oracle PSU 11.2.0.4.160119 (21948347)
- Timezone file DSTv25 - patch 22037014 for 11.2.0.4 and 12.1.0.2 (12.1.0.1 includes DSTv24, patch 20875898 (unchanged from 12.1.0.1.v3), as a backport of DSTv25 was unavailable at build time)
- Fixed an issue that prevented customers from creating more than 10 Directory objects in the database
- Fixed an issue that prevented customers from re-granting read privileges on the ADUMP and BDUMP Directory objects

Baseline: Oracle Database Patch Set Update 11.2.0.4.160119 (patch 21948347, released January 2016)

Bugs fixed: 17288409, 21051852, 18607546, 17205719, 17811429, 17816865, 20506699 17922254, 17754782, 16934803, 13364795, 17311728, 17441661, 17284817 16992075, 17446237, 14015842, 19972569, 17449815, 21538558, 20925795 17375354, 19463897, 17982555, 17235750, 13866822, 17478514, 18317531 18235390, 14338435, 20803583, 13944971, 20142975, 17811789, 16929165 18704244, 20506706, 17546973, 20334344, 14054676, 17088068, 18264060 17346091, 17343514, 21538567, 19680952, 18471685, 19211724, 13951456 21847223, 16315398, 18744139, 16850630, 19049453, 18673304, 17883081 19915271, 18641419, 18262334, 17006183, 16065166, 18277454, 16833527 10136473, 18051556, 17865671, 17852463, 18554871, 17853498, 18334586 17588480, 17551709, 19827973, 17842825, 17344412, 18828868, 17025461 11883252, 13609098, 17239687, 17602269, 19197175, 22195457, 18316692 17313525, 12611721, 19544839, 18964939, 17600719, 18191164, 19393542 17571306, 18482502, 20777150, 19466309, 17040527, 17165204, 18098207 16785708, 17174582, 16180763, 17465741, 16777840, 12982566, 19463893 22195465, 12816846, 16875449, 17237521, 19358317, 17811438, 17811447 17945983, 18762750, 17184721, 16912439,

18061914, 17282229, 18331850 18202441, 17082359, 18723434, 21972320, 19554106, 14034426, 18339044 19458377, 17752995, 20448824, 17891943, 17258090, 17767676, 16668584 18384391, 17040764, 17381384, 15913355, 18356166, 14084247, 20506715 13853126, 18203837, 14245531, 21756699, 16043574, 22195441, 17848897 17877323, 21453153, 17468141, 20861693, 17786518, 17912217, 17037130 18155762, 16956380, 17478145, 17394950, 18189036, 18641461, 18619917 17027426, 21352646, 16268425, 22195492, 19584068, 18436307, 17265217 17634921, 13498382, 21526048, 20004087, 22195485, 17443671, 18000422 22321756, 20004021, 17571039, 21067387, 16344544, 18009564, 14354737 18135678, 18614015, 20441797, 18362222, 17835048, 16472716, 17936109 17050888, 17325413, 14010183, 18747196, 17761775, 16721594, 17082983 20067212, 21179898, 17302277, 18084625, 15990359, 18203835, 17297939 17811456, 16731148, 21168487, 17215560, 13829543, 14133975, 17694209 18091059, 17385178, 8322815, 17586955, 17201159, 17655634, 18331812 19730508, 18868646, 17648596, 16220077, 16069901, 17348614, 17393915 17274537, 17957017, 18096714, 17308789, 18436647, 14285317, 19289642 14764829, 18328509, 17622427, 22195477, 16943711, 14368995, 17346671 18996843, 17783588, 21343838, 16618694, 17672719, 18856999, 18783224 17851160, 17546761, 17798953, 18273830, 22092979, 19972566, 16384983 17726838, 17360606, 22321741, 13645875, 18199537, 16542886, 21787056 17889549, 14565184, 17071721, 17610798, 20299015, 21343897, 20657441 17397545, 18230522, 16360112, 19769489, 12905058, 18641451, 12747740 18430495, 17042658, 17016369, 14602788, 17551063, 19972568, 21517440 18508861, 19788842, 14657740, 17332800, 13837378, 19972564, 17186905 18315328, 19699191, 17437634, 19006849, 19013183, 17296856, 18674024 17232014, 16855292, 21051840, 14692762, 17762296, 17705023, 19121551 21330264, 19854503, 19309466, 18681862, 18554763, 20558005, 17390160 18456514, 16306373, 13955826, 18139690, 17501491, 21668627, 17299889 17752121, 17889583, 18673325, 18293054, 17242746, 17951233, 17649265 18094246, 19615136, 17011832, 16870214, 17477958, 18522509, 20631274 16091637, 17323222, 16595641, 16524926, 18228645, 18282562, 17596908 17156148, 18031668, 16494615, 17545847, 17655240, 17614134, 13558557 17341326, 17891946, 17716305, 16392068, 19271443, 21351877, 18092127 18440047, 17614227, 14106803, 16903536, 18973907, 18673342, 19032867 17389192, 17612828, 16194160, 17006570, 17721717, 17570240, 17390431 16863422, 18325460, 19727057, 16422541, 19972570, 17267114, 18244962 21538485, 18765602, 18203838, 16198143, 17246576, 14829250, 17835627 18247991, 14458214, 21051862, 16692232, 17786278, 17227277, 16042673 16314254, 16228604, 16837842, 17393683, 17787259, 20331945, 20074391 15861775, 16399083, 18018515, 21051858, 18260550, 17036973, 16613964 17080436, 16579084, 18384537, 18280813, 20296213, 16901385, 15979965 18441944, 16450169, 9756271, 17892268, 11733603, 16285691, 17587063 21343775, 16538760, 18180390, 18193833, 21051833, 17238511, 17824637 16571443, 18306996, 14852021, 18674047, 17853456, 12364061, 22195448

Version 11.2.0.4.v6

Version 11.2.0.4.v6 adds support for the following:

- Enable SSL encryption for Standard Edition and Standard Edition One

Version 11.2.0.4.v5

Version 11.2.0.4.v5 adds support for the following:

- Oracle PSU 11.2.0.4.8 (21352635)
- Includes the Daylight Saving Time Patch, patch 20875898: DST-24, that came out after the April 2015 PSU.

Baseline: Oracle Database Patch Set Update 11.2.0.4.8 (patch 21352635, released October 2015)

Bugs fixed: 17288409, 21051852, 18607546, 17205719, 17811429, 17816865, 20506699 17922254, 17754782, 16934803, 13364795, 17311728, 17441661, 17284817 16992075, 17446237, 14015842,

19972569, 21538558, 20925795, 17449815 17375354, 19463897, 17982555, 17235750, 13866822, 18317531, 17478514 18235390, 14338435, 20803583, 13944971, 20142975, 17811789, 16929165 18704244, 20506706, 17546973, 20334344, 14054676, 17088068, 18264060 17346091, 17343514, 21538567, 19680952, 18471685, 19211724, 13951456 16315398, 18744139, 16850630, 19049453, 18673304, 17883081, 19915271 18641419, 18262334, 17006183, 16065166, 18277454, 16833527, 10136473 18051556, 17865671, 17852463, 18554871, 17853498, 18334586, 17588480 17551709, 19827973, 17842825, 17344412, 18828868, 17025461, 11883252 13609098, 17239687, 17602269, 19197175, 18316692, 17313525, 12611721 19544839, 18964939, 17600719, 18191164, 19393542, 17571306, 18482502 20777150, 19466309, 17040527, 17165204, 18098207, 16785708, 17174582 16180763, 17465741, 16777840, 12982566, 19463893, 12816846, 16875449 17237521, 19358317, 17811438, 17811447, 17945983, 18762750, 17184721 16912439, 18061914, 17282229, 18331850, 18202441, 17082359, 18723434 19554106, 14034426, 18339044, 19458377, 17752995, 20448824, 17891943 17258090, 17767676, 16668584, 18384391, 17040764, 17381384, 15913355 18356166, 14084247, 20506715, 13853126, 18203837, 14245531, 16043574 17848897, 17877323, 17468141, 17786518, 17912217, 17037130, 18155762 16956380, 17478145, 17394950, 18189036, 18641461, 18619917, 17027426 21352646, 16268425, 19584068, 18436307, 17265217, 17634921, 13498382 20004087, 17443671, 18000422, 20004021, 17571039, 21067387, 16344544 18009564, 14354737, 18135678, 18614015, 20441797, 18362222, 17835048 16472716, 17936109, 17050888, 17325413, 14010183, 18747196, 17761775 16721594, 17082983, 20067212, 21179898, 17302277, 18084625, 15990359 18203835, 17297939, 17811456, 16731148, 17215560, 13829543, 14133975 17694209, 18091059, 17385178, 8322815, 17586955, 17201159, 17655634 18331812, 19730508, 18868646, 17648596, 16220077, 16069901, 17348614 17393915, 17274537, 17957017, 18096714, 17308789, 18436647, 14285317 19289642, 14764829, 18328509, 17622427, 16943711, 14368995, 17346671 18996843, 17783588, 16618694, 17672719, 18856999, 18783224, 17851160 17546761, 17798953, 18273830, 19972566, 16384983, 17726838, 17360606 13645875, 18199537, 16542886, 17889549, 14565184, 17071721, 20299015 17610798, 20657441, 17397545, 18230522, 16360112, 19769489, 12905058 18641451, 12747740, 18430495, 17042658, 17016369, 14602788, 19972568 18508861, 19788842, 14657740, 17332800, 13837378, 19972564, 17186905 18315328, 19699191, 17437634, 19006849, 19013183, 17296856, 18674024 17232014, 16855292, 21051840, 14692762, 17762296, 17705023, 19121551 19854503, 19309466, 18681862, 18554763, 20558005, 17390160, 18456514 16306373, 13955826, 18139690, 17501491, 17299889, 17752121, 17889583 18673325, 18293054, 17242746, 17951233, 17649265, 18094246, 19615136 17011832, 16870214, 17477958, 18522509, 20631274, 16091637, 17323222 16595641, 16524926, 18228645, 18282562, 17596908, 17156148, 18031668 16494615, 17545847, 17614134, 13558557, 17341326, 17891946, 17716305 16392068, 19271443, 18092127, 18440047, 17614227, 14106803, 16903536 18973907, 18673342, 17389192, 16194160, 17006570, 17612828, 17721717 17570240, 17390431, 16863422, 18325460, 19727057, 16422541, 19972570 17267114, 18244962, 21538485, 18765602, 18203838, 16198143, 17246576 14829250, 17835627, 18247991, 14458214, 21051862, 16692232, 17786278 17227277, 16042673, 16314254, 16228604, 16837842, 17393683, 17787259 20331945, 20074391, 15861775, 16399083, 18018515, 18260550, 21051858 17036973, 16613964, 17080436, 16579084, 18384537, 18280813, 20296213 16901385, 15979965, 18441944, 16450169, 9756271, 17892268, 11733603 16285691, 17587063, 16538760, 18180390, 18193833, 21051833, 17238511 17824637, 16571443, 18306996, 14852021, 18674047, 17853456, 12364061

Version 11.2.0.4.v4

Version 11.2.0.4.v4 adds support for the following:

- Oracle PSU 11.2.0.4.6 (20299013)
- Installs additional Oracle Text knowledge bases from Oracle Database. Examples media (English and French)
- Provides access to DBMS_REPAIR through RDSADMIN.RDSADMIN_DBMS_REPAIR
- Grants ALTER DATABASE LINK, ALTER PUBLIC DATABASE LINK, EXEMPT ACCESS POLICY, EXEMPT IDENTITY POLICY, and EXEMPT REDACTION POLICY to master user

Baseline: Oracle Database Patch Set Update 11.2.0.4.6 (patch 20299013, released April 2015)

Bugs fixed: 17288409, 17798953, 18273830, 18607546, 17811429, 17205719, 20506699 17816865, 19972566, 17922254, 17754782, 16384983, 17726838, 13364795 16934803, 17311728, 17284817, 17441661, 17360606, 13645875, 18199537 16992075, 16542886, 17446237, 14015842, 17889549, 14565184, 19972569 17071721, 20299015, 17610798, 17375354, 17449815, 17397545, 19463897 18230522, 13866822, 17235750, 17982555, 16360112, 18317531, 17478514 19769489, 12905058, 14338435, 18235390, 13944971, 18641451, 20142975 17811789, 16929165, 18704244, 12747740, 18430495, 20506706, 17546973 14054676, 17088068, 17346091, 18264060, 17016369, 17042658, 17343514 14602788, 19972568, 19680952, 18471685, 19788842, 18508861, 14657740 17332800, 19211724, 13837378, 13951456, 16315398, 17186905, 18744139 19972564, 16850630, 18315328, 17437634, 19049453, 18673304, 17883081 19006849, 19915271, 19013183, 18641419, 17296856, 18674024, 18262334 17006183, 18277454, 16833527, 17232014, 16855292, 10136473, 17762296 14692762, 17705023, 18051556, 17865671, 17852463, 18554871, 17853498 19121551, 18334586, 19854503, 17551709, 19309466, 17588480, 19827973 17344412, 17842825, 18828868, 18681862, 18554763, 17390160, 18456514 16306373, 17025461, 13955826, 18139690, 11883252, 13609098, 17501491 17239687, 17752121, 17299889, 17602269, 19197175, 17889583, 18316692 17313525, 18673325, 12611721, 19544839, 18293054, 17242746, 18964939 17600719, 18191164, 19393542, 17571306, 18482502, 19466309, 17951233 17649265, 18094246, 19615136, 17040527, 17011832, 17165204, 18098207 16785708, 16870214, 17465741, 16180763, 17174582, 17477958, 12982566 16777840, 18522509, 20631274, 16091637, 17323222, 19463893, 16595641 16875449, 12816846, 16524926, 17237521, 18228645, 18282562, 17596908 19358317, 17811438, 17811447, 17945983, 18762750, 17156148, 18031668 16912439, 17184721, 16494615, 18061914, 17282229, 17545847, 18331850 18202441, 17082359, 18723434, 19554106, 17614134, 13558557, 17341326 14034426, 17891946, 18339044, 17716305, 19458377, 17752995, 16392068 19271443, 17891943, 18092127, 17258090, 17767676, 16668584, 18384391 17614227, 17040764, 16903536, 17381384, 14106803, 15913355, 18973907 18356166, 18673342, 17389192, 14084247, 16194160, 17612828, 17006570 20506715, 17721717, 13853126, 17390431, 18203837, 17570240, 14245531 16043574, 16863422, 17848897, 17877323, 18325460, 19727057, 17468141 17786518, 17912217, 16422541, 19972570, 17267114, 17037130, 18244962 18765602, 18203838, 18155762, 16956380, 16198143, 17246576, 17478145 17394950, 14829250, 18189036, 18641461, 18619917, 17835627, 17027426 16268425, 18247991, 19584068, 14458214, 18436307, 17265217, 17634921 13498382, 16692232, 17786278, 17227277, 16042673, 16314254, 17443671 18000422, 16228604, 16837842, 17571039, 17393683, 16344544, 17787259 18009564, 20074391, 14354737, 15861775, 18135678, 18614015, 16399083 18362222, 18018515, 16472716, 17835048, 17050888, 17936109, 14010183 17325413, 18747196, 17080436, 16613964, 17036973, 17761775, 16579084 16721594, 17082983, 18384537, 18280813, 20296213, 17302277, 16901385 18084625, 15979965, 15990359, 18203835, 17297939, 17811456, 16731148 13829543, 14133975, 17215560, 17694209, 18091059, 17385178, 8322815 17586955, 18441944, 17201159, 16450169, 9756271, 17655634, 19730508 17892268, 18868646, 17648596, 16220077, 16069901, 11733603, 16285691 17587063, 18180390, 16538760, 18193833, 17348614, 17393915, 17957017 17274537, 18096714, 17308789, 17238511, 18436647, 17824637, 14285317 19289642, 14764829, 17622427, 18328509, 16571443, 16943711, 14368995 18306996, 17346671, 14852021, 18996843, 17783588, 16618694, 17853456 18674047, 17672719, 18856999, 12364061, 18783224, 17851160, 17546761

Version 11.2.0.4.v3

Version 11.2.0.4.v3 adds support for the following:

- Oracle PSU 11.2.0.4.4 (19121551)
- Latest DST file (DSTv23 – patch 19396455, released Oct 2014). This patch is incorporated by default in new instances only.

Baseline: Oracle Database Patch Set Update 11.2.0.4.4 (patch 19121551, released October 2014)

Bugs fixed: 19396455, 18759211, 17432124, 16799735, 17288409, 17205719, 17811429, 17754782, 17726838, 13364795, 17311728 17284817, 17441661, 13645875, 18199537, 16992075, 16542886, 17446237 14565184, 17071721, 17610798, 17375354, 17449815, 17397545, 19463897 18230522, 17235750, 16360112, 13866822, 17982555, 17478514, 12905058 14338435, 13944971, 16929165, 12747740, 17546973, 14054676, 17088068 18264060, 17343514, 17016369, 17042658, 14602788, 14657740, 17332800 19211724, 13951456, 16315398, 17186905, 18744139, 16850630, 17437634 19049453, 18673304, 17883081, 18641419, 17296856, 18262334, 17006183 18277454, 17232014, 16855292, 10136473, 17705023, 17865671, 18554871 19121551, 17588480, 17551709, 17344412, 17842825, 18681862, 17390160 13955826, 13609098, 18139690, 17501491, 17239687, 17752121, 17299889 17602269, 18673325, 17313525, 17242746, 19544839, 17600719, 18191164 17571306, 19466309, 17951233, 18094246, 17165204, 17011832, 17040527 16785708, 16180763, 17477958, 17174582, 17465741, 18522509, 17323222 19463893, 16875449, 16524926, 17237521, 17596908, 17811438, 17811447 18031668, 16912439, 16494615, 18061914, 17545847, 17082359, 19554106 17614134, 17341326, 17891946, 19458377, 17716305, 17752995, 16392068 19271443, 17767676, 17614227, 17040764, 17381384, 18973907, 18673342 14084247, 17389192, 17006570, 17612828, 17721717, 13853126, 18203837 17390431, 17570240, 14245531, 16043574, 16863422, 19727057, 17468141 17786518, 17037130, 17267114, 18203838, 16198143, 16956380, 17478145 14829250, 17394950, 17027426, 16268425, 18247991, 19584068, 14458214 18436307, 17265217, 13498382, 16692232, 17786278, 17227277, 16042673 16314254, 17443671, 16228604, 16837842, 17393683, 17787259, 18009564 15861775, 16399083, 18018515, 16472716, 17050888, 14010183, 17325413 16613964, 17080436, 17036973, 17761775, 16721594, 18280813, 15979965 18203835, 17297939, 16731148, 17811456, 14133975, 17385178, 17586955 16450169, 17655634, 9756271, 17892268, 17648596, 16220077, 16069901 11733603, 16285691, 17587063, 18180390, 17393915, 18096714, 17238511 17824637, 14285317, 19289642, 14764829, 18328509, 17622427, 16943711 17346671, 18996843, 14852021, 17783588, 16618694, 17672719, 17546761

Version 11.2.0.4.v2 (Deprecated)

Version 11.2.0.4.v2 adds support for the following:

- Oracle PSU 11.2.0.4.3 (18522509)
- User access to DBMS_TRANSACTION package to clean-up failed distributed transactions
- Latest DST file (DSTv22 – patch 18759211, released June 2014). This patch is incorporated by default only in new Oracle DB instances.
- Grants DBMS_REPUTIL to DBA role (upgrade to 11.2.0.4 revokes it from public)
- Privileges granted on DBMS_TRANSACTION, `v$pending_xatrans$`, and `v$xatrans$`
- Resolves a problem with DDL commands when user objects have “SYSTEM” in their names
- Installs schema objects to support XA Transactions, allowing transactions to be managed by an external transaction manager
- Permits truncation of temporary SYS and SYSTEM objects, allowing tools like LogMiner to function correctly

Baseline: Oracle Database Patch Set Update 11.2.0.4.3 (patch 18522509, released July 2014)

Bugs fixed: 17432124, 18759211, 18522509, 18031668, 17478514, 17752995, 17288409, 16392068, 17205719, 17811429, 17767676, 17614227 17040764, 17381384, 17754782, 17726838, 13364795, 17311728, 17389192 17006570, 17612828, 17284817, 17441661, 13853126, 17721717, 13645875 18203837, 17390431, 16542886, 16992075, 16043574, 17446237, 16863422 14565184, 17071721, 17610798, 17468141, 17786518, 17375354, 17397545 18203838, 16956380, 17478145, 16360112, 17235750, 17394950, 13866822 17478514, 17027426, 12905058, 14338435, 16268425, 13944971,

18247991 14458214, 16929165, 17265217, 13498382, 17786278, 17227277, 17546973 14054676, 17088068, 16314254, 17016369, 14602788, 17443671, 16228604 16837842, 17332800, 17393683, 13951456, 16315398, 18744139, 17186905 16850630, 17437634, 19049453, 17883081, 15861775, 17296856, 18277454 16399083, 16855292, 18018515, 10136473, 16472716, 17050888, 17865671 17325413, 14010183, 18554871, 17080436, 16613964, 17761775, 16721594 17588480, 17551709, 17344412, 18681862, 15979965, 13609098, 18139690 17501491, 17239687, 17752121, 17602269, 18203835, 17297939, 17313525 16731148, 17811456, 14133975, 17600719, 17385178, 17571306, 16450169 17655634, 18094246, 17892268, 17165204, 17011832, 17648596, 16785708 17477958, 16180763, 16220077, 17465741, 17174582, 18522509, 16069901 16285691, 17323222, 18180390, 17393915, 16875449, 18096714, 17238511

Version 11.2.0.4.v1

Version 11.2.0.4.v1 adds support for the following:

- Oracle PSU 11.2.0.4.1
- Creating New Directories in the Main Data Storage Space (p. 1072)

Baseline: Oracle Database Patch Set Update 11.2.0.4.1 (released January 2014)

Bugs fixed: 17432124, 16850630, 17551709, 13944971, 17811447, 13866822, 17811429, 16069901 16721594, 17443671, 17478514, 17612828, 17610798, 17239687, 17501491 17446237, 16450169, 17811438, 17288409, 17811456, 12905058, 17088068 16285691, 17332800

Related Topics

- Upgrading the Oracle DB Engine (p. 975)
- Oracle on Amazon RDS (p. 931)

PostgreSQL on Amazon RDS

Amazon RDS supports DB instances running several versions of PostgreSQL. You can create DB instances and DB snapshots, point-in-time restores and backups. DB instances running PostgreSQL support Multi-AZ deployments, Read Replicas (version 9.3.5 and later), Provisioned IOPS, and can be created inside a VPC. You can also use Secure Socket Layer (SSL) to connect to a DB instance running PostgreSQL.

Before creating a DB instance, you should complete the steps in the Setting Up for Amazon RDS (p. 5) section of this guide.

You can use any standard SQL client application to run commands for the instance from your client computer. Such applications include *pgAdmin*, a popular Open Source administration and development tool for PostgreSQL, or *psql*, a command line utility that is part of a PostgreSQL installation. In order to deliver a managed service experience, Amazon RDS does not provide host access to DB instances, and it restricts access to certain system procedures and tables that require advanced privileges. Amazon RDS supports access to databases on a DB instance using any standard SQL client application. Amazon RDS does not allow direct host access to a DB instance via Telnet or Secure Shell (SSH).

Amazon RDS for PostgreSQL is compliant with many industry standards. For example, you can use Amazon RDS for PostgreSQL databases to build HIPAA-compliant applications and to store healthcare-related information, including protected health information (PHI) under an executed Business Associate Agreement (BAA) with AWS. Amazon RDS for PostgreSQL also meets Federal Risk and Authorization Management Program (FedRAMP) security requirements. Amazon RDS for PostgreSQL has received a FedRAMP Joint Authorization Board (JAB) Provisional Authority to Operate (P-ATO) at the FedRAMP HIGH Baseline within the AWS GovCloud (US) Region. For more information on supported compliance standards, see AWS Cloud Compliance.

To import PostgreSQL data into a DB instance, follow the information in the Importing Data into PostgreSQL on Amazon RDS (p. 1196) section.

Topics

Common Management Tasks for PostgreSQL on Amazon RDS

The following are the common management tasks you perform with an Amazon RDS PostgreSQL DB instance, with links to relevant documentation for each task.

Task Area	Relevant Documentation
Setting up Amazon RDS for first-time use There are prerequisites you must complete before you create your DB instance. For example, DB instances are created by default with a firewall that prevents access to it. You therefore must create a security group with the correct IP addresses and network configuration to access the DB instance.	Setting Up for Amazon RDS (p. 5)
Understanding Amazon RDS DB instances If you are creating a DB instance for production purposes, you should understand how instance classes, storage types, and Provisioned IOPS work in Amazon RDS.	DB Instance Class (p. 92) Amazon RDS Storage Types (p. 410) Provisioned IOPS Storage (p. 413)
Finding supported PostgreSQL versions Amazon RDS supports several versions of PostgreSQL.	Supported PostgreSQL Database Versions (p. 1147)
Setting up high availability and failover support A production DB instance should use Multi-AZ deployments. Multi-AZ deployments provide increased availability, data durability, and fault tolerance for DB instances.	High Availability (Multi-AZ) (p. 99)
Understanding the Amazon Virtual Private Cloud (VPC) network If your AWS account has a default VPC, then your DB instance is automatically created inside the default VPC. If your account does not have a default VPC, and you want the DB instance in a VPC, you must create the VPC and subnet groups before you create the DB instance.	Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform (p. 391) Working with an Amazon RDS DB Instance in a VPC (p. 399)
Importing data into Amazon RDS PostgreSQL You can use several different tools to import data into your PostgreSQL DB instance on Amazon RDS.	Importing Data into PostgreSQL on Amazon RDS (p. 1196)
Setting up read only Read Replicas (master/standby) PostgreSQL on Amazon RDS supports Read Replicas in both the same AWS Region and in a different AWS Region from the master instance.	Working with PostgreSQL, MySQL, and MariaDB Read Replicas (p. 134) PostgreSQL Read Replicas (Version 9.3.5 and Later) (p. 136) Replicating a Read Replica Across AWS Regions (p. 142)
Understanding security groups By default, DB instances are created with a firewall that prevents access to them. You therefore must create a security group with the correct IP addresses and network configuration to access the DB instance.	Determining Whether You Are Using the EC2-VPC or EC2-Classic Platform (p. 391) Amazon RDS Security Groups (p. 375)

Task Area	Relevant Documentation
In general, if your DB instance is on the *EC2-Classic* platform, you need to create a DB security group. If your DB instance is on the *EC2-VPC* platform, you need to create a VPC security group.	
Setting up parameter groups and features If your DB instance is going to require specific database parameters, you should create a parameter group before you create the DB instance.	Working with DB Parameter Groups (p. 170)
Performing common DBA tasks for PostgreSQL If your DB instance is going to require specific database options, you should create an option group before you create the DB instance. • Creating Roles (p. 1200) • Managing PostgreSQL Database Access (p. 1200) • Working with PostgreSQL Parameters (p. 1201) • Working with PostgreSQL Autovacuum on Amazon RDS (p. 1209) • Audit Logging for a PostgreSQL DB Instance (p. 1216) • Working with PostGIS (p. 1219) • Using pgBadger for Log Analysis with PostgreSQL (p. 1221)	Common DBA Tasks for PostgreSQL (p. 1200)
Connecting to your PostgreSQL DB instance After creating a security group and associating it to a DB instance, you can connect to the DB instance using any standard SQL client application such as pgadmin III.	Connecting to a DB Instance Running the PostgreSQL Database Engine (p. 1179) Using SSL with a PostgreSQL DB Instance (p. 1170)
Backing up and restoring your DB instance You can configure your DB instance to take automated backups, or take manual snapshots, and then restore instances from the backups or snapshots.	Backing Up and Restoring Amazon RDS DB Instances (p. 200)
Monitoring the activity and performance of your DB instance You can monitor a PostgreSQL DB instance by using CloudWatch Amazon RDS metrics, events, and enhanced monitoring.	Viewing DB Instance Metrics (p. 254) Viewing Amazon RDS Events (p. 301)
Upgrading the PostgreSQL database version You can do both major and minor version upgrades for your PostgreSQL DB instance.	Upgrading a PostgreSQL DB Instance (p. 1169) Major Version Upgrades (p. 1191)
Working with log files You can access the log files for your PostgreSQL DB instance.	PostgreSQL Database Log Files (p. 322)
Understanding the best practices for PostgreSQL DB instances Find some of the best practices for working with PostgreSQL on Amazon RDS.	Best Practices for Working with PostgreSQL (p. 87)

Amazon RDS PostgreSQL Planning Information

Amazon RDS supports DB instances running several editions of PostgreSQL. This section shows how you can work with PostgreSQL on Amazon RDS. You should also be aware of the limits for PostgreSQL DB instances.

For information about importing PostgreSQL data into a DB instance, see Importing Data into PostgreSQL on Amazon RDS (p. 1196).

Topics

Using the rds_superuser Role

When you create a DB instance, the master user system account that you create is assigned to the `rds_superuser` role. The `rds_superuser` role is similar to the PostgreSQL superuser role (customarily named postgres in local instances) but with some restrictions. As with the PostgreSQL superuser role, the `rds_superuser` role has the most privileges on your DB instance and you should not assign this role to users unless they need the most access to the DB instance.

The `rds_superuser` role can do the following:

- Add extensions that are available for use with Amazon RDS. For more information, see Supported PostgreSQL Features (p. 1165) and the PostgreSQL documentation.
- Manage tablespaces, including creating and deleting them. For more information, see this section in the PostgreSQL documentation.
- View all users not assigned the `rds_superuser` role using the `pg_stat_activity` command and kill their connections using the `pg_terminate_backend` and `pg_cancel_backend` commands.
- Grant and revoke the replication attribute onto all roles that are not the `rds_superuser` role. For more information, see this section in the PostgreSQL documentation.

Supported PostgreSQL Database Versions

Amazon RDS supports the following PostgreSQL versions:

Topics

PostgreSQL Version 9.6.5 on Amazon RDS

PostgreSQL version 9.6.5 contains several bug fixes for issues in release 9.6.4. For more information on the fixes in 9.6.5, see the PostgreSQL documentation. For information on upgrading the engine version for your PostgreSQL DB instance, see Upgrading a PostgreSQL DB Instance (p. 1169).

This version also includes support for the pgRouting and postgresql-hll extensions, and the decoder_raw optional module.

You can see the complete list of extensions now supported by Amazon RDS PostgreSQL at Supported PostgreSQL Features and Extensions (p. 1153)

PostgreSQL Version 9.6.3 on Amazon RDS

PostgreSQL version 9.6.3 contains several new features and bug fixes. This version includes the following features:

- Supports the extension `pg_repack` version 1.4.0. You can use this extension to remove bloat from tables and indexes. For more information on using `pg_repack` with Amazon RDS, see Working with the `pg_repack` Extension (p. 1218).
- Supports the extension `pgaudit` version 1.1.0. This extension provides detailed session and object audit logging. For more information on using pgaudit with Amazon RDS, see Working with the `pgaudit` Extension (p. 1217).
- Supports `wal2json`, an output plugin for logical decoding.
- Supports the `auto_explain` module. You can use this module to log execution plans of slow statements automatically. The following example shows how to use `auto_explain` from within an Amazon RDS PostgreSQL session:

```
LOAD '$libdir/plugins/auto_explain';
```

  For more information on using `auto_explain`, see the PostgreSQL documentation.

PostgreSQL Version 9.6.2 on Amazon RDS

PostgreSQL version 9.6.2 contains several new features and bug fixes. The new version also includes the following extension versions:

- PostGIS version 2.3.2
- pg_freespacemap version 1.1–Provides a way to examine the free space map (FSM). This extension provides an overloaded function called pg_freespace. The functions show the value recorded in the free space map for a given page, or for all pages in the relation.
- pg_hint_plan version 1.1.3– Provides control of execution plans by using hinting phrases at the beginning of SQL statements.

- log_fdw version 1.0–Using this extension from Amazon RDS, you can load and query your database engine log from within the database. For more information, see Using the `log_fdw` Extension (p. 1163).
- With this version release, you can now edit the `max_worker_processes` parameter in a DB parameter group.

PostgreSQL version 9.6.2 on Amazon RDS also supports altering enum values. For more information, see ALTER ENUM for PostgreSQL (p. 1169).

For more information on the fixes in 9.6.2, see the PostgreSQL documentation. For information on upgrading the engine version for your PostgreSQL DB instance, see Upgrading a PostgreSQL DB Instance (p. 1169).

PostgreSQL Version 9.6.1 on Amazon RDS

PostgreSQL version 9.6.1 contains several new features and improvements. For more information about the fixes and improvements in PostgreSQL 9.6.1, see the PostgreSQL documentation. For information on upgrading the engine version for your PostgreSQL DB instance, see Upgrading a PostgreSQL DB Instance (p. 1169). For information about performing parallel queries and phrase searching using Amazon RDS for PostgreSQL 9.6.1, see the AWS Database Blog.

PostgreSQL version 9.6.1 includes the following changes:

- **Parallel query execution**: Supports parallel execution of large read-only queries, allowing sequential scans, hash joins, nested loops, and aggregates to be run in parallel. By default, parallel query execution is not enabled. To enable parallel query execution, set the parameter `max_parallel_workers_per_gather` to a value larger than zero.
- **Updated postgres_fdw extension**: Supports remote JOINs, SORTs, UPDATEs, and DELETE operations.
- **PL/v8 update**: Provides version 1.5.3 of the PL/v8 language.
- **PostGIS version update**: Supports POSTGIS="2.3.0 r15146" GEOS="3.5.0-CAPI-1.9.0 r4084" PROJ="Rel. 4.9.2, 08 September 2015" GDAL="GDAL 2.1.1, released 2016/07/07" LIBXML="2.9.1" LIBJSON="0.12" RASTER
- **Vacuum improvement**: Avoids scanning pages unnecessarily during vacuum freeze operations.
- **Full-text search support for phrases**: Supports the ability to specify a phrase-search query in tsquery input using the new operators <-> and <N>.
- **Two new extensions are supported**:
  - `bloom`, an index access method based on Bloom filters
  - `pg_visibility`, which provides a means for examining the visibility map and page-level visibility information of a table.
- With the release of version 9.6.2, you can now edit the `max_worker_processes` parameter in a PostgreSQL version 9.6.1 DB parameter group.

You can create a new PostgreSQL 9.6.1 database instance using the AWS Management Console, AWS CLI, or RDS API. You can also upgrade an existing PostgreSQL 9.5 instance to version 9.6.1 using major version upgrade. If you want to upgrade a DB instance from version 9.3 or 9.4 to 9.6, you must perform a point-and-click upgrade to the next major version first. Each upgrade operation involves a short period of unavailability for your DB instance.

PostgreSQL Version 9.5.9 on Amazon RDS

PostgreSQL version 9.5.9 contains several bug fixes for issues in version 9.5.8. For more information on the fixes in 9.5.9, see the PostgreSQL documentation. For information on upgrading the engine version for your PostgreSQL DB instance, see Upgrading a PostgreSQL DB Instance (p. 1169).

PostgreSQL Version 9.5.7 on Amazon RDS

PostgreSQL version 9.5.7 contains several new features and bug fixes. This version includes the following features:

- Supports the extension `pgaudit` version 1.0.5. This extension provides detailed session and object audit logging. For more information on using `pgaudit` with Amazon RDS, see Working with the `pgaudit` Extension (p. 1217).
- Supports `wal2json`, an output plugin for logical decoding.
- Supports the `auto_explain` module. You can use this module to log execution plans of slow statements automatically. The following example shows how to use `auto_explain` from within an Amazon RDS PostgreSQL session.

```
LOAD '$libdir/plugins/auto_explain';
```

  For more information on using `auto_explain`, see the PostgreSQL documentation.

PostgreSQL Version 9.5.6 on Amazon RDS

PostgreSQL version 9.5.6 contains several new features and bug fixes. The new version also includes the following extension versions:

- PostGIS version 2.2.5
- pg_freespacemap version 1.1–Provides a way to examine the free space map (FSM). This extension provides an overloaded function called pg_freespace. This function shows the value recorded in the free space map for a given page, or for all pages in the relation.
- pg_hint_plan version 1.1.3– Provides control of execution plans by using hinting phrases at the beginning of SQL statements.

PostgreSQL version 9.5.6 on Amazon RDS also supports altering enum values. For more information, see ALTER ENUM for PostgreSQL (p. 1169).

For more information on the fixes in 9.5.6, see the PostgreSQL documentation. For information on upgrading the engine version for your PostgreSQL DB instance, see Upgrading a PostgreSQL DB Instance (p. 1169).

PostgreSQL Version 9.5.4 on Amazon RDS

PostgreSQL version 9.5.4 contains several fixes to issue found in previous versions. For more information on the fixes in 9.5.4, see the PostgreSQL documentation. For information on upgrading the engine version for your PostgreSQL DB instance, see Upgrading a PostgreSQL DB Instance (p. 1169).

Beginning with PostgreSQL version 9.4, PostgreSQL supports the streaming of WAL changes using logical replication decoding. Amazon RDS supports logical replication for PostgreSQL version 9.4.9 and higher and 9.5.4 and higher. For more information about PostgreSQL logical replication on Amazon RDS, see Logical Replication for PostgreSQL on Amazon RDS (p. 1165).

Beginning with PostgreSQL version 9.5.4 for Amazon RDS, the command ALTER USER WITH BYPASSRLS is supported.

PostgreSQL versions 9.4.9 and later and version 9.5.4 and later support event triggers, and Amazon RDS supports event triggers for these versions. You can use the master user account can be used to create, modify, rename, and delete event triggers. Event triggers are at the DB instance level, so they can apply to all databases on an instance. For more information about PostgreSQL event triggers on Amazon RDS, see Event Triggers for PostgreSQL on Amazon RDS (p. 1167).

PostgreSQL Version 9.5.2 on Amazon RDS

PostgreSQL version 9.5.2 contains several fixes to issues found in previous versions. For more information on the features in 9.5.2, see the PostgreSQL documentation. For information on upgrading the engine version for your PostgreSQL DB instance, see Upgrading a PostgreSQL DB Instance (p. 1169).

PostgreSQL version 9.5.2 doesn't support the previous generation db.m1 or db.m2 instance classes. If you need to upgrade a DB instance running PostgreSQL version 9.4 to version 9.5.2 to one of these instance classes, you need to scale compute. To do that, you need a comparable current generation db.t2 or db.m3 instance class before you can upgrade a DB instance running PostgreSQL version 9.4 to version 9.5.2. For more information on DB instance classes, see DB Instance Class (p. 92).

Native PostgreSQL version 9.5.2 introduced the command ALTER USER WITH BYPASSRLS.

This release includes updates from previous versions, including the following:

- **CVE-2016-2193:** Fixes an issue where a query plan might be reused for more than one ROLE in the same session. Reusing a query plan can cause the query to use the wrong set of Row Level Security (RLS) policies.
- **CVE-2016-3065:** Fixes a server crash bug triggered by using `pageinspect` with BRIN index pages. Because an attacker might be able to expose a few bytes of server memory, this crash is being treated as a security issue.

Major enhancements in RDS PostgreSQL 9.5 include the following:

- UPSERT: Allow INSERTs that would generate constraint conflicts to be turned into UPDATEs or ignored
- Add the GROUP BY analysis features GROUPING SETS, CUBE, and ROLLUP
- Add row-level security control
- Create mechanisms for tracking the progress of replication, including methods for identifying the origin of individual changes during logical replication
- Add Block Range Indexes (BRIN)
- Add substantial performance improvements for sorting
- Add substantial performance improvements for multi-CPU machines
- PostGIS 2.2.2 - To use this latest version of PostGIS, use the ALTER EXTENSION UPDATE statement to update after you upgrade to version 9.5.2. Example:

  ALTER EXTENSION POSTGIS UPDATE TO '2.2.2'
- Improved visibility of autovacuum sessions by allowing the rds_superuser account to view autovacuum sessions in pg_stat_activity. For example, you can identify and terminate an autovacuum session that is blocking a command from running, or executing slower than a manually issued vacuum command.

RDS PostgreSQL version 9.5.2 includes the following new extensions:

- **address_standardizer** – A single-line address parser that takes an input address and normalizes it based on a set of rules stored in a table, helper lex, and gaz tables.
- hstore_plperl – Provides transforms for the `hstore` type for PL/Perl.
- tsm_system_rows – Provides the table sampling method `SYSTEM_ROWS`, which can be used in the `TABLESAMPLE` clause of a `SELECT` command.

- tsm_system_time – Provides the table sampling method `SYSTEM_TIME`, which can be used in the `TABLESAMPLE` clause of a `SELECT` command.

PostgreSQL Version 9.4.14 on Amazon RDS

PostgreSQL version 9.4.14 contains several bug fixes for issues in release 9.4.12. For more information on the fixes in 9.4.14, see the PostgreSQL documentation. For information on upgrading the engine version for your PostgreSQL DB instance, see Upgrading a PostgreSQL DB Instance (p. 1169).

PostgreSQL Version 9.4.12 on Amazon RDS

PostgreSQL version 9.4.12 contains several fixes to issue found in previous versions.

For more information on the fixes in 9.4.12, see the PostgreSQL documentation. For information on upgrading the engine version for your PostgreSQL DB instance, see Upgrading a PostgreSQL DB Instance (p. 1169).

PostgreSQL Version 9.4.11 on Amazon RDS

PostgreSQL version 9.4.11 contains several fixes to issue found in previous versions.

For more information on the fixes in 9.4.11, see the PostgreSQL documentation. For information on upgrading the engine version for your PostgreSQL DB instance, see Upgrading a PostgreSQL DB Instance (p. 1169).

Beginning with PostgreSQL version 9.4, PostgreSQL supports the streaming of WAL changes using logical replication decoding. Amazon RDS supports logical replication for PostgreSQL version 9.4.9 and higher and 9.5.4 and higher. For more information about PostgreSQL logical replication on Amazon RDS, see Logical Replication for PostgreSQL on Amazon RDS (p. 1165).

PostgreSQL versions 9.4.9 and later and version 9.5.4 and later support event triggers, and Amazon RDS supports event triggers for these versions. The master user account can be used to create, modify, rename, and delete event triggers. Event triggers are at the DB instance level, so they can apply to all databases on an instance. For more information about PostgreSQL event triggers on Amazon RDS, see Event Triggers for PostgreSQL on Amazon RDS (p. 1167).

PostgreSQL Version 9.4.9 on Amazon RDS

PostgreSQL version 9.4.9 contains several fixes to issue found in previous versions. For more information on the fixes in 9.4.9, see the PostgreSQL documentation. For information on upgrading the engine version for your PostgreSQL DB instance, see Upgrading a PostgreSQL DB Instance (p. 1169).

Beginning with PostgreSQL version 9.4, PostgreSQL supports the streaming of WAL changes using logical replication decoding. Amazon RDS supports logical replication for PostgreSQL version 9.4.9 and higher and 9.5.4 and higher. For more information about PostgreSQL logical replication on Amazon RDS, see Logical Replication for PostgreSQL on Amazon RDS (p. 1165).

PostgreSQL versions 9.4.9 and later and version 9.5.4 and later support event triggers, and Amazon RDS supports event triggers for these versions. The master user account can be used to create, modify, rename, and delete event triggers. Event triggers are at the DB instance level, so they can apply to all databases on an instance. For more information about PostgreSQL event triggers on Amazon RDS, see Event Triggers for PostgreSQL on Amazon RDS (p. 1167).

PostgreSQL Version 9.4.7 on Amazon RDS

PostgreSQL version 9.4.7 contains several fixes to issue found in previous versions. For more information on the fixes in 9.4.7, see the PostgreSQL documentation. For information on upgrading the engine version for your PostgreSQL DB instance, see Upgrading a PostgreSQL DB Instance (p. 1169).

PostgreSQL version 9.4.7 includes improved visibility of autovacuum sessions by allowing the rds_superuser account to view autovacuum sessions in pg_stat_activity. For example, you can identify and terminate an autovacuum session that is blocking a command from running, or executing slower than a manually issued vacuum command.

PostgreSQL Version 9.3.19 on Amazon RDS

PostgreSQL version 9.3.19 contains several bug fixes for issues in version 9.3.18. For more information on the fixes in 9.3.19, see the PostgreSQL documentation. For information on upgrading the engine version for your PostgreSQL DB instance, see Upgrading a PostgreSQL DB Instance (p. 1169).

PostgreSQL Version 9.3.17 on Amazon RDS

PostgreSQL version 9.3.17 contains several fixes for bugs found in previous versions. This version contains the same extension components as version 9.3.16. For a list of fixes in version 9.3.17, see the PostgreSQL documentation. For information on upgrading the engine version for your PostgreSQL DB instance, see Upgrading a PostgreSQL DB Instance (p. 1169).

PostgreSQL Version 9.3.16 on Amazon RDS

PostgreSQL version 9.3.16 contains several fixes for bugs found in previous versions. This version contains the same extension components as version 9.3.14. For a list of fixes in version 9.3.16, see the PostgreSQL documentation. For information on upgrading the engine version for your PostgreSQL DB instance, see Upgrading a PostgreSQL DB Instance (p. 1169).

PostgreSQL Version 9.3.14 on Amazon RDS

PostgreSQL version 9.3.14 contains several fixes for bugs found in previous versions. For a list of fixes in version 9.3.14, see the PostgreSQL documentation. For information on upgrading the engine version for your PostgreSQL DB instance, see Upgrading a PostgreSQL DB Instance (p. 1169).

PostgreSQL Version 9.3.12 on Amazon RDS

PostgreSQL version 9.3.12 contains several fixes for bugs found in previous versions. For a list of fixes in version 9.3.12, see the PostgreSQL documentation. For information on upgrading the engine version for your PostgreSQL DB instance, see Upgrading a PostgreSQL DB Instance (p. 1169).

PostgreSQL version 9.3.12 includes improved visibility of autovacuum sessions by allowing the rds_superuser account to view autovacuum sessions in pg_stat_activity. For example, you can identify and terminate an autovacuum session that is blocking a command from running, or executing slower than a manually issued vacuum command.

Supported PostgreSQL Features and Extensions

Amazon RDS supports many of the most common PostgreSQL extensions and features.

Topics

PostgreSQL Extensions and Modules Supported on Amazon RDS

PostgreSQL supports many PostgreSQL extensions and modules. Extensions and modules expand on the functionality provided by the PostgreSQL engine. The following sections show the extensions and modules supported by Amazon RDS for the major PostgreSQL versions.

Topics

You can find a list of extensions supported by Amazon RDS in the default DB parameter group for that PostgreSQL version. You can also see the current extensions list using `psql` by showing the `rds.extensions` parameter as in the following example.

```
SHOW rds.extensions;
```

Note
Parameters added in a minor version release might display inaccurately when using the `rds.extensions` parameter in `psql`.

PostgreSQL Version 9.6.x Extensions and Modules Supported on Amazon RDS

The following tables show PostgreSQL extensions and modules for PostgreSQL version 9.6.x that are currently supported by PostgreSQL on Amazon RDS. "N/A" indicates that the extension or module is not available for that PostgreSQL version. For more information on PostgreSQL extensions, see Packaging Related Objects into an Extension.

Extension	9.6.1	9.6.2	9.6.3	9.6.5
address_standardizer	2.3.0	2.3.2	2.3.2	2.3.2
address_standardizer_data_us	2.3.0	2.3.2	2.3.2	2.3.2
bloom	1.0	1.0	1.0	1.0
btree_gin	1.0	1.0	1.0	1.0
btree_gist	1.2	1.2	1.2	1.2
chkpass	1.0	1.0	1.0	1.0
citext	1.3	1.3	1.3	1.3
cube	1.2	1.2	1.2	1.2
dblink	1.2	1.2	1.2	1.2
dict_int	1.0	1.0	1.0	1.0
dict_xsyn	1.0	1.0	1.0	1.0
earthdistance	1.1	1.1	1.1	1.1

Extension	9.6.1	9.6.2	9.6.3	9.6.5
fuzzystrmatch	1.1	1.1	1.1	1.1
hstore	1.4	1.4	1.4	1.4
hstore_plperl	1.0	1.0	1.0	1.0
intagg	1.1	1.1	1.1	1.1
intarray	1.2	1.2	1.2	1.2
ip4r	2.0	2.0	2.0	2.0
isn	1.1	1.1	1.1	1.1
log_fdw—see Using the `log_fdw` Extension (p. 1163).	N/A	1.0	1.0	1.0
ltree	1.1	1.1	1.1	1.1
pgaudit	N/A	N/A	1.1	1.1
pg_buffercache	1.2	1.2	1.2	1.2
pg_freespacemap	N/A	1.1	1.1	1.1
pg_hint_plan	N/A	1.1.3	1.1.3	1.1.3
pg_prewarm	1.1	1.1	1.1	1.1
pg_repack	N/A	N/A	1.4.0	1.4.1
pg_stat_statements	1.4	1.4	1.4	1.4
pg_trgm	1.3	1.3	1.3	1.3
pg_visibility	1.1	1.1	1.1	1.1
pgcrypto	1.3	1.3	1.3	1.3
pgrowlocks	1.2	1.2	1.2	1.2
pgrouting	N/A	N/A	N/A	2.3.2
pgstattuple	1.4	1.4	1.4	1.4
plcoffee	1.5.3	1.5.3	1.5.3	1.5.3
plls	1.5.3	1.5.3	1.5.3	1.5.3
plperl	1.0	1.0	1.0	1.0
plpgsql	1.0	1.0	1.0	1.0
pltcl	1.0	1.0	1.0	1.0
plv8	1.5.3	1.5.3	1.5.3	1.5.3
PostGIS	2.3.0	2.3.2	2.3.2	2.3.2

Extension	9.6.1	9.6.2	9.6.3	9.6.5
postgis_tiger_geocoder	2.3.0	2.3.2	2.3.2	2.3.2
postgis_topology	2.3.0	2.3.2	2.3.2	2.3.2
postgres_fdw	1.0	1.0	1.0	1.0
postgresql-hll	N/A	N/A	N/A	2.10.2
sslinfo	1.2	1.2	1.2	1.2
tablefunc	1.0	1.0	1.0	1.0
test_parser	1.0	1.0	1.0	1.0
tsearch2	1.0	1.0	1.0	1.0
tsm_system_rows	1.0	1.0	1.0	1.0
tsm_system_time	1.0	1.0	1.0	1.0
unaccent	1.1	1.1	1.1	1.1
uuid-ossp	1.1	1.1	1.1	1.1

The following modules are supported as shown for versions of PostgreSQL 9.6.

Module	9.6.1	9.6.2	9.6.3	9.6.5
auto_explain	N/A	N/A	Supported	Supported
decoder_raw	N/A	N/A	N/A	Supported
test_decoder	Supported	Supported	Supported	Supported
wal2json	N/A	N/a	Supported	Supported

PostgreSQL Version 9.5.x Extensions Supported on Amazon RDS

The following tables show PostgreSQL extensions and modules for PostgreSQL version 9.5.x that are currently supported by PostgreSQL on Amazon RDS. "N/A" indicates that the extension or module is not available for that PostgreSQL version. For more information on PostgreSQL extensions, see Packaging Related Objects into an Extension.

Extension	9.5.2	9.5.4	9.5.6	9.5.7	9.5.9
address_standardizer	2.2.2	2.2.2	2.2.5	2.2.5	2.2.5
address_standardizer_data_us	2.2.2	2.2.2	2.2.5	2.2.5	2.2.5
bloom	N/A	N/A	N/A	N/A	N/A
btree_gin	1.0	1.0	1.0	1.0	1.0
btree_gist	1.1	1.1	1.1	1.1	1.1
chkpass	1.0	1.0	1.0	1.0	1.0

Extension	9.5.2	9.5.4	9.5.6	9.5.7	9.5.9
citext	1.1	1.1	1.1	1.1	1.1
cube	1.0	1.0	1.0	1.0	1.0
dblink	1.1	1.1	1.1	1.1	1.1
dict_int	1.0	1.0	1.0	1.0	1.0
dict_xsyn	1.0	1.0	1.0	1.0	1.0
earthdistance	1.0	1.0	1.0	1.0	1.0
fuzzystrmatch	1.0	1.0	1.0	1.0	1.0
hstore	1.3	1.3	1.3	1.3	1.3
hstore_plperl	1.0	1.0	1.0	1.0	1.0
intagg	1.0	1.0	1.0	1.0	1.0
intarray	1.0	1.0	1.0	1.0	1.0
ip4r	2.0	2.0	2.0	2.0	2.0
isn	1.0	1.0	1.0	1.0	1.0
log_fdw—see Using the `log_fdw` Extension (p. 1163).	N/A	N/A	N/A	N/A	N/A
ltree	1.0	1.0	1.0	1.0	1.0
pgaudit	N/A	N/A	N/A	1.0.5	1.0.5
pg_buffercache	1.1	1.1	1.1	1.1	1.1
pg_freespacemap	N/A	N/A	1.0	1.0	1.0
pg_hint_plan	N/A	N/A	1.1.3	1.1.3	1.1.3
pg_prewarm	1.0	1.0	1.0	1.0	1.0
pg_stat_statements	1.3	1.3	1.3	1.3	1.3
pg_trgm	1.1	1.1	1.1	1.1	1.1
pg_visibility	N/A	N/A	N/A	N/A	N/A
pgcrypto	1.2	1.2	1.2	1.2	1.2
pgrowlocks	1.1	1.1	1.1	1.1	1.1
pgstattuple	1.3	1.3	1.3	1.3	1.3
plcoffee	1.4.4	1.4.4	1.4.4	1.4.4	1.4.4
plls	1.4.4	1.4.4	1.4.4	1.4.4	1.4.4
plperl	1.0	1.0	1.0	1.0	1.0

Extension	9.5.2	9.5.4	9.5.6	9.5.7	9.5.9
plpgsql	1.0	1.0	1.0	1.0	1.0
pltcl	1.0	1.0	1.0	1.0	1.0
plv8	1.4.4	1.4.4	1.4.4	1.4.4	1.4.4
PostGIS	2.2.2	2.2.2	2.2.5	2.2.5	2.2.5
postgis_tiger_geocoder	2.2.2	2.2.2	2.2.5	2.2.5	2.2.5
postgis_topology	2.2.2	2.2.2	2.2.5	2.2.5	2.2.5
postgres_fdw	1.0	1.0	1.0	1.0	1.0
sslinfo	1.0	1.0	1.0	1.0	1.0
tablefunc	1.0	1.0	1.0	1.0	1.0
test_parser	1.0	1.0	1.0	1.0	1.0
tsearch2	1.0	1.0	1.0	1.0	1.0
tsm_system_rows	N/A	N/A	1.0	1.0	1.0
tsm_system_time	N/A	N/A	1.0	1.0	1.0
unaccent	1.0	1.0	1.0	1.0	1.0
uuid-ossp	1.0	1.0	1.0	1.0	1.0

The following modules are supported as shown for versions of PostgreSQL 9.5.

Module	9.5.2	9.5.4	9.5.6	9.5.7	9.5.9
auto_explain	N/A	N/A	N/A	Supported	Supported
test_decoder	N/A	N/A	Supported	Supported	Supported
wal2json	N/A	N/a	N/A	Supported	Supported

PostgreSQL Version 9.4.x Extensions and Modules Supported on Amazon RDS

The following tables show the PostgreSQL extensions and modules for PostgreSQL version 9.4.x that are currently supported by PostgreSQL on Amazon RDS. "N/A" indicates that the extension or module is not available for that PostgreSQL version. For more information on PostgreSQL extensions, see Packaging Related Objects into an Extension.

Extension	9.4.7	9.4.9	9.4.11
address_standardizer	N/A	N/A	N/A
address_standardizer_data_us	N/A	N/A	N/A
bloom	N/A	N/A	N/A
btree_gin	1.0	1.0	1.0

Extension	9.4.7	9.4.9	9.4.11
btree_gist	1.0	1.0	1.0
chkpass	1.0	1.0	1.0
citext	1.0	1.0	1.0
cube	1.0	1.0	1.0
dblink	1.1	1.1	1.1
dict_int	1.0	1.0	1.0
dict_xsyn	1.0	1.0	1.0
earthdistance	1.0	1.0	1.0
fuzzystrmatch	1.0	1.0	1.0
hstore	1.3	1.3	1.3
hstore_plperl	N/A	N/A	N/A
intagg	1.0	1.0	1.0
intarray	1.0	1.0	1.0
ip4r	2.0	2.0	2.0
isn	1.0	1.0	1.0
log_fdw—see Using the `log_fdw` Extension (p. 1163).	N/A	N/A	N/A
ltree	1.0	1.0	1.0
pg_buffercache	1.0	1.0	1.0
pg_freespacemap	N/A	N/A	N/A
pg_hint_plan	N/A	N/A	N/A
pg_prewarm	1.0	1.0	1.0
pg_stat_statements	1.2	1.2	1.2
pg_trgm	1.1	1.1	1.1
pg_visibility	N/A	N/A	N/A
pgcrypto	1.1	1.1	1.1
pgrowlocks	1.1	1.1	1.1
pgstattuple	1.2	1.2	1.2
plcoffee	1.4.4	1.4.4	1.4.4
plls	1.4.4	1.4.4	1.4.4
plperl	1.0	1.0	1.0

Extension	9.4.7	9.4.9	9.4.11
plpgsql	1.0	1.0	1.0
pltcl	1.0	1.0	1.0
plv8	1.4.4	1.4.4	1.4.4
PostGIS	2.1.8	2.1.8	2.1.8
postgis_tiger_geocoder	2.1.8	2.1.8	2.1.8
postgis_topology	2.1.8	2.1.8	2.1.8
postgres_fdw	1.0	1.0	1.0
sslinfo	1.0	1.0	1.0
tablefunc	1.0	1.0	1.0
test_parser	1.0	1.0	1.0
tsearch2	1.0	1.0	1.0
tsm_system_rows	N/A	N/A	N/A
tsm_system_time	N/A	N/A	N/A
unaccent	1.0	1.0	1.0
uuid-ossp	1.0	1.0	1.0

The following modules are supported as shown for versions of PostgreSQL 9.4.

Module	9.4.7	9.4.9	9.4.11	9.4.12	9.4.14
test_decoder	N/A	N/A	N/A	Supported	Supported

PostgreSQL Version 9.3.x Extensions Supported on Amazon RDS

The following table shows PostgreSQL extensions for PostgreSQL version 9.3.x that are currently supported by PostgreSQL on Amazon RDS. "N/A" indicates that the extension is not available for that PostgreSQL version. For more information on PostgreSQL extensions, see Packaging Related Objects into an Extension.

Extension	9.3.12	9.3.14	9.3.16
address_standardizer	N/A	N/A	N/A
address_standardizer_data_us	N/A	N/A	N/A
bloom	N/A	N/A	N/A
btree_gin	1.0	1.0	1.0
btree_gist	1.0	1.0	1.0
chkpass	1.0	1.0	1.0

Extension	9.3.12	9.3.14	9.3.16
citext	1.0	1.0	1.0
cube	1.0	1.0	1.0
dblink	1.1	1.1	1.1
dict_int	1.0	1.0	1.0
dict_xsyn	1.0	1.0	1.0
earthdistance	1.0	1.0	1.0
fuzzystrmatch	1.0	1.0	1.0
hstore	1.2	1.2	1.2
hstore_plperl	N/A	N/A	N/A
intagg	1.0	1.0	1.0
intarray	1.0	1.0	1.0
ip4r	N/A	N/A	N/A
isn	1.0	1.0	1.0
log_fdw—see Using the `log_fdw` Extension (p. 1163).	N/A	N/A	N/A
ltree	1.0	1.0	1.0
pg_buffercache	N/A	N/A	N/A
pg_freespacemap	N/A	N/A	N/A
pg_hint_plan	N/A	N/A	N/A
pg_prewarm	N/A	N/A	N/A
pg_stat_statements	1.1	1.1	1.1
pg_trgm	1.1	1.1	1.1
pg_visibility	N/A	N/A	N/A
pgcrypto	1.0	1.0	1.0
pgrowlocks	1.1	1.1	1.1
pgstattuple	N/A	N/A	N/A
plcoffee	1.4.4	1.4.4	1.4.4
plls	1.4.4	1.4.4	1.4.4
plperl	1.0	1.0	1.0
plpgsql	1.0	1.0	1.0
pltcl	1.0	1.0	1.0

Extension	9.3.12	9.3.14	9.3.16
plv8	1.4.4	1.4.4	1.4.4
PostGIS	2.1.8	2.1.8	2.1.8
postgis_tiger_geocoder	2.1.8	2.1.8	2.1.8
postgis_topology	2.1.8	2.1.8	2.1.8
postgres_fdw	1.0	1.0	1.0
sslinfo	1.0	1.0	1.0
tablefunc	1.0	1.0	1.0
test_parser	1.0	1.0	1.0
tsearch2	1.0	1.0	1.0
tsm_system_rows	N/A	N/A	N/A
tsm_system_time	N/A	N/A	N/A
unaccent	1.0	1.0	1.0
uuid-ossp	1.0	1.0	1.0

PostgreSQL Extension Support for PostGIS on Amazon RDS

The following table shows the PostGIS component versions that ship with the Amazon RDS PostgreSQL versions.

Version	PostGIS	GEOS	GDAL	PROJ
9.3.12	2.1.8 r13780	3.5.0-CAPI-1.9.0 r4084	GDAL 1.11.4, released 2016/01/25	Rel. 4.9.2, 08 September 2015
9.3.14	2.1.8 r13780	3.5.0-CAPI-1.9.0 r4084	GDAL 1.11.5, released 2016/07/01	Rel. 4.9.2, 08 September 2015
9.3.16	2.1.8 r13780	3.5.0-CAPI-1.9.0 r4084	GDAL 1.11.5, released 2016/07/01	Rel. 4.9.2, 08 September 2015
9.3.17	2.1.8 r13780	3.5.0-CAPI-1.9.0 r4084	GDAL 1.11.5, released 2016/07/01	Rel. 4.9.2, 08 September 2015
9.4.7	2.1.8 r13780	3.5.0-CAPI-1.9.0 r4084	GDAL 1.11.4, released 2016/01/25	Rel. 4.9.2, 08 September 2015
9.4.9	2.1.8 r13780	3.5.0-CAPI-1.9.0 r4084	GDAL 1.11.5, released 2016/07/01	Rel. 4.9.2, 08 September 2015

Version	PostGIS	GEOS	GDAL	PROJ
9.4.11	2.1.8 r13780	3.5.0-CAPI-1.9.0 r4084	GDAL 1.11.5, released 2016/07/01	Rel. 4.9.2, 08 September 2015
9.4.12	2.1.8 r13780	3.5.0-CAPI-1.9.0 r4084	GDAL 1.11.5, released 2016/07/01	Rel. 4.9.2, 08 September 2015
9.5.2	2.2.2 r14797	3.5.0-CAPI-1.9.0 r4084	GDAL 2.0.2, released 2016/01/26	Rel. 4.9.2, 08 September 2015
9.5.4	2.2.2 r14797	3.5.0-CAPI-1.9.0 r4084	GDAL 2.0.3, released 2016/07/01	Rel. 4.9.2, 08 September 2015
9.5.6	2.2.5 r15298	3.5.1-CAPI-1.9.1 r4246	GDAL 2.0.3, released 2016/07/01	Rel. 4.9.3, 15 August 2016
9.5.7	2.2.5 r15298	3.5.1-CAPI-1.9.1 r4246	GDAL 2.0.3, released 2016/07/01	Rel. 4.9.3, 15 August 2016
9.6.1	2.3.0 r15146	3.5.0-CAPI-1.9.0 r4084	GDAL 2.1.1, released 2016/07/07	Rel. 4.9.2, 08 September 2015
9.6.2	2.3.2 r15302	3.5.1-CAPI-1.9.1 r4246	GDAL 2.1.3, released 2017/20/01	Rel. 4.9.3, 15 August 2016
9.6.3	2.3.2 r15302	3.5.1-CAPI-1.9.1 r4246	GDAL 2.1.3, released 2017/20/01	Rel. 4.9.3, 15 August 2016

Before you can use the PostGIS extension, you must create it by running the following command.

```
CREATE EXTENSION POSTGIS;
```

Using the `log_fdw` Extension

The `log_fdw` extension is new for Amazon RDS for PostgreSQL version 9.6.2 and later. Using this extension, you can access your database engine log using a SQL interface. In addition to viewing the *stderr* log files that are generated by default on RDS, you can view CSV logs (set the `log_destination` parameter to `csvlog`) and build foreign tables with the data neatly split into several columns.

This extension introduces two new functions that make it easy to create foreign tables for database logs:

- `list_postgres_log_files()` – Lists the files in the database log directory and the file size in bytes.
- `create_foreign_table_for_log_file(table_name text, server_name text, log_file_name text)` – Builds a foreign table for the specified file in the current database.

All functions created by `log_fdw` are owned by `rds_superuser`. Members of the `rds_superuser` role can grant access to these functions to other database users.

The following example shows how to use the `log_fdw` extension.

To use the `log_fdw` extension

1. Get the `log_fdw` extension.

```
postgres=> CREATE EXTENSION log_fdw;
CREATE EXTENSION
```

2. Create the log server as a foreign data wrapper.

```
postgres=> CREATE SERVER log_server FOREIGN DATA WRAPPER log_fdw;
CREATE SERVER
```

3. Select all from a list of log files.

```
postgres=> SELECT * from list_postgres_log_files() order by 1;
```

   A sample response is as follows.

```
            file_name              | file_size_bytes
-----------------------------------+-----------------
 postgresql.log.2016-08-09-22.csv |            1111
 postgresql.log.2016-08-09-23.csv |            1172
 postgresql.log.2016-08-10-00.csv |            1744
 postgresql.log.2016-08-10-01.csv |            1102
(4 rows)
```

4. Create a table with a single 'log_entry' column for non-CSV files.

```
postgres=> SELECT create_foreign_table_for_log_file('my_postgres_error_log',
     'log_server', 'postgresql.log.2016-08-09-22.csv');
```

   A sample response is as follows.

```
----------------------------------

(1 row)
```

5. Select a sample of the log file. The following code retrieves the log time and error message description.

```
postgres=> SELECT log_time, message from my_postgres_error_log order by 1;
```

   A sample response is as follows.

```
```

```
              log_time              |                                  message

---------------------------------
+------------------------------------------------------------------------------
Tue Aug 09 15:45:18.172 2016 PDT | ending log output to stderr
Tue Aug 09 15:45:18.175 2016 PDT | database system was interrupted; last known up at
 2016-08-09 22:43:34 UTC
Tue Aug 09 15:45:18.223 2016 PDT | checkpoint record is at 0/90002E0
Tue Aug 09 15:45:18.223 2016 PDT | redo record is at 0/90002A8; shutdown FALSE
Tue Aug 09 15:45:18.223 2016 PDT | next transaction ID: 0/1879; next OID: 24578
Tue Aug 09 15:45:18.223 2016 PDT | next MultiXactId: 1; next MultiXactOffset: 0
Tue Aug 09 15:45:18.223 2016 PDT | oldest unfrozen transaction ID: 1822, in database 1
(7 rows)
```

Supported PostgreSQL Features

Amazon RDS supports many of the most common PostgreSQL features. These include:

Topics

Logical Replication for PostgreSQL on Amazon RDS

Beginning with PostgreSQL version 9.4, PostgreSQL supports the streaming of WAL changes using logical replication slots. Amazon RDS supports logical replication for a PostgreSQL DB instance version 9.4.9 and higher and 9.5.4 and higher. Using logical replication, you can set up logical replication slots on your instance and stream database changes through these slots to a client like pg_recvlogical. Logical slots are created at the database level and support replication connections to a single database.

PostgreSQL logical replication on Amazon RDS is enabled by a new parameter, a new replication connection type, and a new security role. The client for the replication can be any client that is capable of establishing a replication connection to a database on a PostgreSQL DB instance.

The most common clients for PostgreSQL logical replication are AWS Database Migration Service or a custom-managed host on an AWS EC2 instance. The logical replication slot knows nothing about the receiver of the stream; there is no requirement that the target be a replica database. If you set up a logical replication slot and don't read from the slot, data can be written to your DB instance's storage and you can quickly fill up the storage on your instance.

For more information on using logical replication with PostgreSQL, see the PostgreSQL documentation.

To enable logical replication for an Amazon RDS for PostgreSQL DB instance, you must do the following:

- The AWS user account initiating the logical replication for the PostgreSQL database on Amazon RDS must have the rds_superuser role and the rds_replication role. The rds_replication role grants permissions to manage logical slots and to stream data using logical slots.
- Set the `rds.logical_replication` parameter to 1. It is a static parameter that requires a reboot to take effect. As part of applying this parameter, we set the `wal_level`, `max_wal_senders`,

`max_replication_slots`, and `max_connections` parameters. These parameter changes can increase WAL generation, so you should only set the `rds.logical_replication` parameter when you are using logical slots.

- Create a logical replication slot as explained following. This process requires a decoding plugin to be specified; currently we support the 'test_decoding' output plugin that ships with PostgreSQL.

Working with Logical Replication Slots

You can use SQL commands to work with logical slots. For example, the following command creates a logical slot named test_slot using the default PostgreSQL output plugin test_decoding.

```
SELECT * FROM pg_create_logical_replication_slot('test_slot', 'test_decoding');
```

The output should be similar to the following.

```
slot_name    | xlog_position
-----------------+---------------
regression_slot | 0/16B1970
(1 row)
```

To list logical slots, use the following command.

```
SELECT * FROM pg_replication_slots;
```

To drop a logical slot, use the following command.

```
SELECT pg_drop_replication_slot('test_slot');
```

The output should be similar to the following.

```
pg_drop_replication_slot
-----------------------

(1 row)
```

For more examples on working with logical replication slots, see Logical Decoding Examples in the PostgreSQL documentation.

Once you create the logical replication slot, you can start streaming. The following example shows how logical decoding is controlled over the streaming replication protocol, using the program pg_recvlogical included in the PostgreSQL distribution. This requires that client authentication is set up to allow replication connections.

```
pg_recvlogical -d postgres --slot test_slot -U master
    --host sg-postgresql1.c6c8mresaghgv0.us-west-2.rds.amazonaws.com
```

```
    -f -  --start
```

Event Triggers for PostgreSQL on Amazon RDS

PostgreSQL versions 9.4.9 and later and version 9.5.4 and later support event triggers, and Amazon RDS supports event triggers for these versions. The master user account can be used to create, modify, rename, and delete event triggers. Event triggers are at the DB instance level, so they can apply to all databases on an instance.

For example, the following code creates an event trigger that prints the current user at the end of every DDL command.

```
CREATE OR REPLACE FUNCTION raise_notice_func()
                RETURNS event_trigger
                LANGUAGE plpgsql AS
$$
BEGIN
                RAISE NOTICE 'In trigger function: %', current_user;
END;
$$;

CREATE EVENT TRIGGER event_trigger_1
                ON ddl_command_end
EXECUTE PROCEDURE raise_notice_func();
```

For more information about PostgreSQL event triggers, see Event Triggers in the PostgreSQL documentation.

There are several limitations to using PostgreSQL event triggers on Amazon RDS. These include:

- You cannot create event triggers on read replicas. You can, however, create event triggers on a read replica master. The event triggers are then copied to the read replica. The event triggers on the read replica don't fire on the read replica when changes are pushed from the master. However, if the read replica is promoted, the existing event triggers fire when database operations occur.
- To perform a major version upgrade to a PostgreSQL DB instance that uses event triggers, you must delete the event triggers before you upgrade the instance.

Huge Pages for Amazon RDS for PostgreSQL

Amazon RDS for PostgreSQL supports multiple page sizes for PostgreSQL versions 9.4.11 and later, 9.5.6 and later, and 9.6.2 and later. This support includes 4 K and 2 MB page sizes.

Huge pages reduce overhead when using large contiguous chunks of memory. You allocate huge pages for your application by using calls to *mmap* or *SYSV* shared memory. You enable huge pages on an Amazon RDS for PostgreSQL database by using the `huge_pages` parameter. Set this parameter to "on" to use huge pages; the default value is "off."

When you set the `huge_pages` parameter to "on," Amazon RDS uses huge pages based on the available shared memory. If the DB instance is unable to use huge pages due to shared memory constraints, Amazon RDS prevents the instance from starting and sets the status of the DB instance to an incompatible parameters state. In this case, you can set the `huge_pages` parameter to "off" to allow Amazon RDS to start the DB instance.

The `shared_buffers` parameter is key to setting the shared memory pool that is required for using huge pages. The default value for the `shared_buffers` parameter is set to a percentage of the total 8K

pages available for that instance's memory. When you use huge pages, those pages are allocated in the huge pages collocated together. Amazon RDS puts a DB instance into an incompatible parameters state if the shared memory parameters are set to require more than 90 percent of the DB instance memory. For more information about setting shared memory for PostgreSQL, see the PostgreSQL documentation.

Note
Huge pages are not supported for the db.m1, db.m2, and db.m3 DB instance classes.

Tablespaces for PostgreSQL on Amazon RDS

Tablespaces are supported in PostgreSQL on Amazon RDS for compatibility; since all storage is on a single logical volume, tablespaces cannot be used for IO splitting or isolation. We have benchmarks and practical experience that shows that a single logical volume is the best setup for most use cases.

Autovacuum for PostgreSQL on Amazon RDS

The PostgreSQL auto-vacuum is an optional, but highly recommended, parameter that by default is turned on for new PostgreSQL DB instances. Do not turn this parameter off. For more information on using auto-vacuum with Amazon RDS PostgreSQL, see Working with PostgreSQL Autovacuum on Amazon RDS (p. 1209).

RAM Disk for the stats_temp_directory

The Amazon RDS for PostgreSQL parameter, `rds.pg_stat_ramdisk_size`, can be used to specify the system memory allocated to a RAM disk for storing the PostgreSQL `stats_temp_directory`. The RAM disk parameter is available for all PostgreSQL versions on Amazon RDS.

Under certain workloads, setting this parameter can improve performance and decrease IO requirements. For more information about the `stats_temp_directory`, see the PostgreSQL documentation.

To enable a RAM disk for your `stats_temp_directory`, set the `rds.pg_stat_ramdisk_size` parameter to a non-zero value in the parameter group used by your DB instance. The parameter value is in MB. You must reboot the DB instance before the change takes effect.

For example, the following AWS CLI command sets the RAM disk parameter to 256 MB.

```
postgres=>aws rds modify-db-parameter-group \
    --db-parameter-group-name pg-95-ramdisk-testing \
    --parameters "ParameterName=rds.pg_stat_ramdisk_size, ParameterValue=256,
 ApplyMethod=pending-reboot"
```

After you reboot, run the following command to see the status of the `stats_temp_directory`:

```
postgres=>show stats_temp_directory;
```

The command should return the following:

```
stats_temp_directory
---------------------------
/rdsdbramdisk/pg_stat_tmp
(1 row)
```

ALTER ENUM for PostgreSQL

Amazon RDS PostgreSQL versions 9.6.2 and 9.5.6 and later support the ability to alter enumerations. This feature is not available in other versions on Amazon RDS.

The following code shows an example of altering an enum value.

```
postgres=> CREATE TYPE rainbow AS ENUM ('red', 'orange', 'yellow', 'green', 'blue',
 'purple');
CREATE TYPE
postgres=> CREATE TABLE t1 (colors rainbow);
CREATE TABLE
postgres=> INSERT INTO t1 VALUES ('red'), ( 'orange');
INSERT 0 2
postgres=> SELECT * from t1;
colors
--------
red
orange
(2 rows)
postgres=> ALTER TYPE rainbow RENAME VALUE 'red' TO 'crimson';
ALTER TYPE
postgres=> SELECT * from t1;
colors
---------
crimson
orange
(2 rows)
```

Limits for PostgreSQL DB Instances

You can have up to 40 PostgreSQL DB instances. The following is a list of limitations for PostgreSQL on Amazon RDS:

- The maximum storage size for PostgreSQL DB instances is the following:
  - General Purpose (SSD) storage: 16 TB
  - Provisioned IOPS storage: 16 TB
  - Magnetic storage: 3 TB
- The minimum storage size for PostgreSQL DB instances is the following:
  - General Purpose (SSD) storage: 5 GB
  - Provisioned IOPS storage: 100 GB
  - Magnetic storage: 5 GB
- Amazon RDS reserves up to 3 connections for system maintenance. If you specify a value for the user connections parameter, you need to add 3 to the number of connections that you expect to use.

Upgrading a PostgreSQL DB Instance

There are two types of upgrades you can manage for your PostgreSQL DB instance:

- OS Updates – Occasionally, Amazon RDS might need to update the underlying operating system of your DB instance to apply security fixes or OS changes. You can decide when Amazon RDS applies OS updates by using the RDS console, AWS Command Line Interface (AWS CLI), or RDS API.

  For more information about OS updates, see Updating the Operating System for a DB Instance or DB Cluster (p. 108).

- Database Engine Upgrades – When Amazon RDS supports a new version of a database engine, you can upgrade your DB instances to the new version. There are two kinds of upgrades: major version upgrades and minor version upgrades. Amazon RDS supports both major and minor version upgrades for PostgreSQL DB instances.

  For more information about PostgreSQL DB engine upgrades, see Upgrading the PostgreSQL DB Engine (p. 1191).

Using SSL with a PostgreSQL DB Instance

Amazon RDS supports Secure Socket Layer (SSL) encryption for PostgreSQL DB instances. Using SSL, you can encrypt a PostgreSQL connection between your applications and your PostgreSQL DB instances. You can also force all connections to your PostgreSQL DB instance to use SSL.

Topics

SSL support is available in all AWS regions for PostgreSQL. Amazon RDS creates an SSL certificate for your PostgreSQL DB instance when the instance is created. If you enable SSL certificate verification, then the SSL certificate includes the DB instance endpoint as the Common Name (CN) for the SSL certificate to guard against spoofing attacks.

To connect to a PostgreSQL DB instance over SSL

1. Download the certificate stored at https://s3.amazonaws.com/rds-downloads/rds-combined-ca-bundle.pem.
2. Import the certificate into your operating system.
3. Connect to your PostgreSQL DB instance over SSL by appending `sslmode=verify-full` to your connection string. When you use `sslmode=verify-full`, the SSL connection verifies the DB instance endpoint against the endpoint in the SSL certificate.

   Use the `sslrootcert` parameter to reference the certificate, for example, `sslrootcert=rds-ssl-ca-cert.pem`.

The following is an example of using the `psql` program to connect to a PostgreSQL DB instance :

```
$ psql -h testpg.cdhmuqifdpib.us-east-1.rds.amazonaws.com -p 5432 \
    "dbname=testpg user=testuser sslrootcert=rds-ca-2015-root.pem sslmode=verify-full"
```

Requiring an SSL Connection to a PostgreSQL DB Instance

You can require that connections to your PostgreSQL DB instance use SSL by using the `rds.force_ssl` parameter. By default, the `rds.force_ssl` parameter is set to 0 (off). You can set the `rds.force_ssl` parameter to 1 (on) to require SSL for connections to your DB instance. Updating the `rds.force_ssl` parameter also sets the PostgreSQL `ssl` parameter to 1 (on) and modifies your DB instance's `pg_hba.conf` file to support the new SSL configuration.

You can set the `rds.force_ssl` parameter value by updating the parameter group for your DB instance. If the parameter group for your DB instance isn't the default one, and the `ssl` parameter is already set to 1 when you set `rds.force_ssl` to 1, you don't need to reboot your DB instance. Otherwise, you must reboot your DB instance for the change to take effect. For more information on parameter groups, see Working with DB Parameter Groups (p. 170).

When the `rds.force_ssl` parameter is set to 1 for a DB instance, you see output similar to the following when you connect, indicating that SSL is now required:

```
$ psql postgres -h SOMEHOST.amazonaws.com -p 8192 -U someuser
psql (9.3.12, server 9.4.4)
WARNING: psql major version 9.3, server major version 9.4.
Some psql features might not work.
SSL connection (cipher: DHE-RSA-AES256-SHA, bits: 256)
Type "help" for help.

postgres=>
```

Determining the SSL Connection Status

The encrypted status of your connection is shown in the logon banner when you connect to the DB instance:

```
Password for user master:
psql (9.3.12)
SSL connection (cipher: DHE-RSA-AES256-SHA, bits: 256)
Type "help" for help.

postgres=>
```

You can also load the `sslinfo` extension and then call the `ssl_is_used()` function to determine if SSL is being used. The function returns `t` if the connection is using SSL, otherwise it returns `f`.

```
postgres=> create extension sslinfo;
CREATE EXTENSION

postgres=> select ssl_is_used();
 ssl_is_used
---------
t
(1 row)
```

You can use the `select ssl_cipher()` command to determine the SSL cipher:

```
postgres=> select ssl_cipher();
ssl_cipher
--------------------
DHE-RSA-AES256-SHA
(1 row)
```

If you enable `set rds.force_ssl` and restart your instance, non-SSL connections are refused with the following message:

```
$ export PGSSLMODE=disable
$ psql postgres -h SOMEHOST.amazonaws.com -p 8192 -U someuser
psql: FATAL: no pg_hba.conf entry for host "host.ip", user "someuser", database "postgres",
 SSL off
$
```

Creating a DB Instance Running the PostgreSQL Database Engine

The basic building block of Amazon RDS is the DB instance. This is the environment in which you will run your PostgreSQL databases.

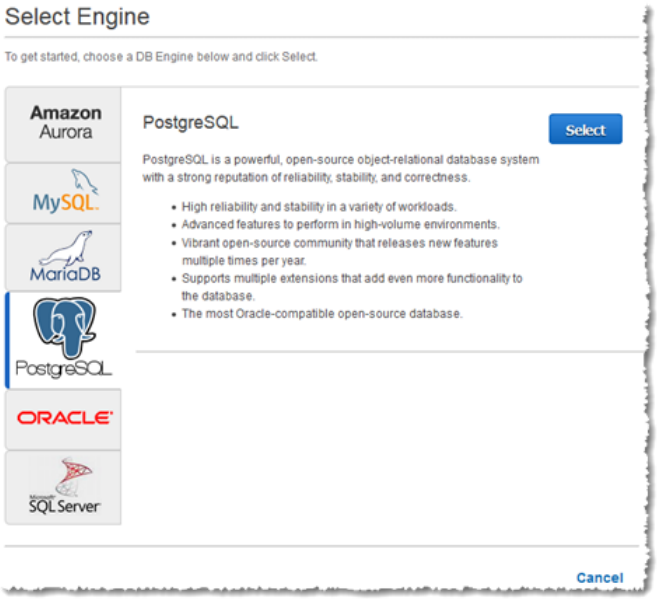
Important
You must complete the tasks in the Setting Up for Amazon RDS (p. 5) section before you can create or connect to a DB instance.

AWS Management Console

To launch a PostgreSQL DB instance

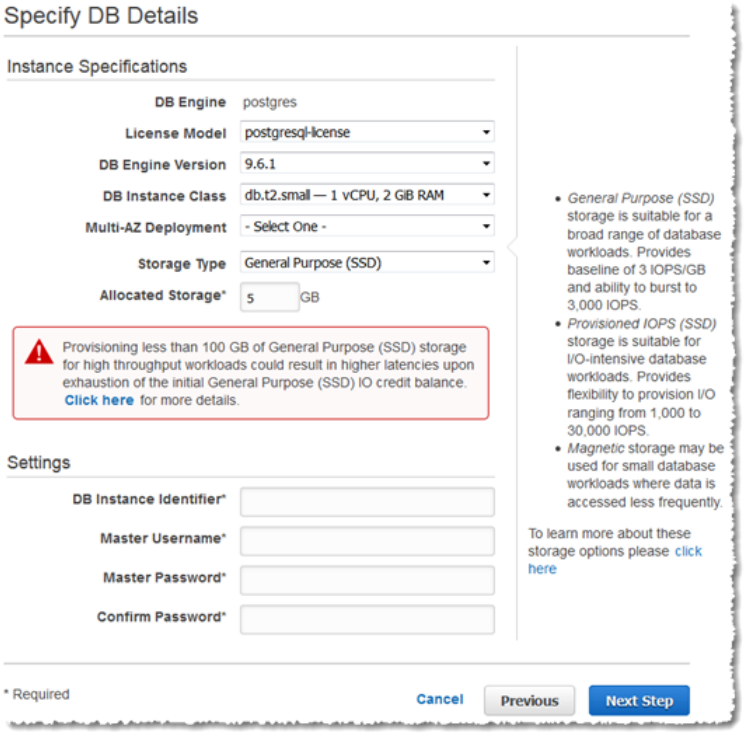
1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the top right corner of the AWS Management Console, select the AWS Region where you want to create the DB instance.
3. In the navigation pane, click **DB Instances**.
4. Click **Launch DB Instance** to start the **Launch DB Instance Wizard**.

   The wizard opens on the **Select Engine** page.



5. On the **Select Engine** page, click the PostgreSQL icon and then click the **Select** button for the PostgreSQL DB engine.
6. Next, the **Production?** page asks if you are planning to use the DB instance you are creating for production. If you are, select **Yes**. By selecting **Yes**, the failover option **Multi-AZ** and the **Provisioned IOPS** storage option are preselected in the following step. Click **Next** when you are finished.
7. On the **Specify DB Details** page, specify your DB instance information. Click **Next** when you are finished.

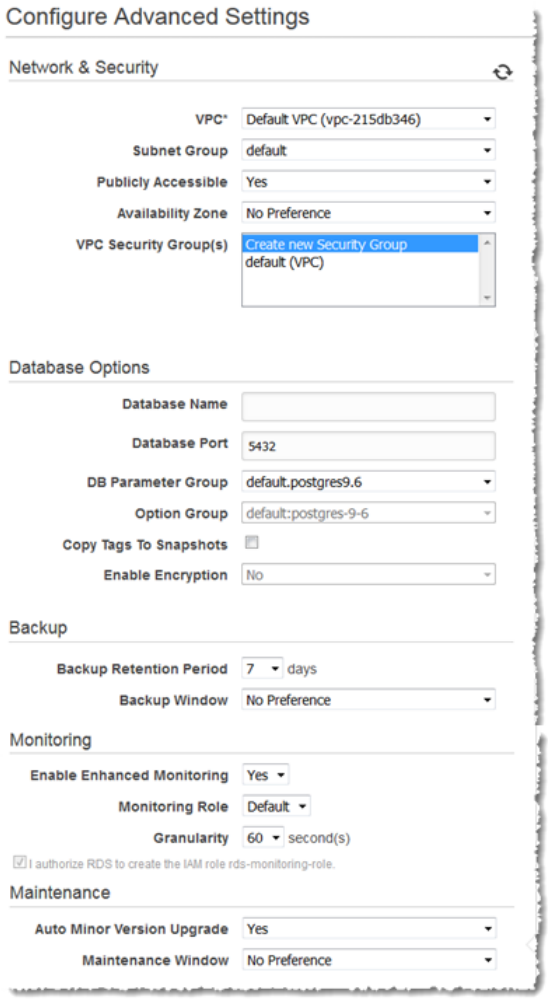
For this parameter...	...Do this:
License Model	PostgreSQL has only one license model. Select the default, **`postgresql-license`**, to use the general license agreement for PostgreSQL.
DB Engine Version	Select the version of PostgreSQL that you want to work with.
DB Instance Class	Select a DB instance class that defines the processing and memory requirements for the DB instance. For more information about all the DB instance class options, see DB Instance Class (p. 92).
Multi-AZ Deployment	Determine if you want to create a standby replica of your DB instance in another Availability Zone for failover support. For more information about multiple Availability Zones, see Regions and Availability Zones (p. 97).
Allocated Storage	Type a value to allocate storage for your database (in gigabytes). In some cases, allocating a higher amount of storage for your DB instance than the size of your database can improve I/O performance. The minimum and maximum storage allowed depends on the storage type. For more information, see Storage for Amazon RDS (p. 410).
Storage Type	Select the storage type you want to use. For more information about storage, see Storage for Amazon RDS (p. 410).
DB Instance Identifier	Type a name for the DB instance that is unique for your account in the AWS Region you selected. You might choose to add some intelligence to the name such as including the AWS Region and DB engine you selected, for example **`postgresql-instance1`**.
Master Username	Type a name using alphanumeric characters that you will use as the master user name to log on to your DB instance. For information on the default privileges granted to the master user name, see Amazon RDS PostgreSQL Planning Information (p. 1147)
Master Password and **Confirm Password**	Type a password that contains from 8 to 128 printable ASCII characters (excluding /,", and @) for your master user password. Retype the password in the **Confirm Password** text box.



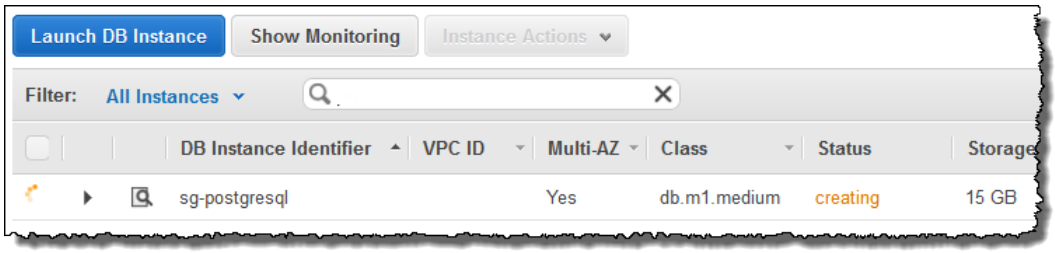
8. On the **Configure Advanced Settings** page, provide additional information that Amazon RDS needs to launch the PostgreSQL DB instance. The table shows settings for an example DB instance. Specify your DB instance information, then click **Launch DB Instance**.

For this parameter...	...Do this:
VPC	This setting depends on the platform you are on. If you are a new customer to AWS, select the default VPC shown. If you are creating a DB instance on the previous E2-Classic platform that does not use a VPC, select **`Not in VPC`**. For more information about VPC, see Amazon Virtual Private Cloud (VPCs) and Amazon RDS (p. 390).
DB Subnet Group	This setting depends on the platform you are on. If you are a new customer to AWS, select **`default`**, which is the default DB subnet group that was created for your account. If you are creating a DB instance on the previous E2-Classic platform and you want your DB instance in a specific VPC, select the DB subnet group you created for that VPC. For more information about VPC, see Amazon Virtual Private Cloud (VPCs) and Amazon RDS (p. 390).
Publicly Accessible	Select **`Yes`** to give the DB instance a public IP address, meaning that it is accessible outside the VPC (the DB instance also needs to be in a public subnet in the VPC); otherwise, select **`No`**, so the DB instance will only be accessible from inside the VPC. For more information about hiding DB instances from public access, see Hiding a DB Instance in a VPC from the Internet (p. 401).

For this parameter...	...Do this:
Availability Zone	Use the default value of **`No Preference`** unless you want to specify an Availability Zone.
VPC Security Group	If you are a new customer to AWS, select the default VPC. If you created a VPC security group, select the VPC security group you previously created.
Database Name	If you want to specify a database name for the default database, type a name for your database of up to 63 alpha-numeric characters. If you do not provide a name, the default "postgres" database is created. To create additional databases, connect to the DB instance and use the SQL command CREATE DATABASE. For more information about connecting to the DB instance, see Connecting to a DB Instance Running the PostgreSQL Database Engine (p. 1179).
Database Port	Specify a port you want to use to access the database. PostgreSQL installations default to port 5432 .
Parameter Group	Select a parameter group. Each PostgreSQL version has a default parameter group you can use, or you can create your own parameter group. For more information about parameter groups, see Working with DB Parameter Groups (p. 170).
Option Group	Option groups are currently not used with PostgreSQL DB instances. For more information about option groups, see Working with Option Groups (p. 153).
Copy Tags To Snapshots	Choose this option to have any DB instance tags copied to a DB snapshot when you create a snapshot. For more information, see Tagging Amazon RDS Resources (p. 129).
Enable Encryption	Select `Yes` to enable encryption at rest for this DB instance. For more information, see Encrypting Amazon RDS Resources (p. 355).
Backup Retention Period	Set the number of days you want automatic backups of your database to be retained. For non-trivial instances set this value to **`1`** or greater.
Backup Window	Unless you have a specific time that you want to have your database backup, use the default of **`No Preference`**.
Auto Minor Version Upgrade	Select **`Yes`** to enable your DB instance to receive minor DB engine version upgrades automatically when they become available.
Maintenance Window	Select the 30 minute window in which pending modifications to your DB instance are applied. If you the time period doesn't matter, select **`No Preference`**.



9. On the final page of the wizard, click **Close**.
10. On the Amazon RDS console, the new DB instance appears in the list of DB instances. The DB instance will have a status of **creating** until the DB instance is created and ready for use. When the state changes to **available**, you can connect to the DB instance. Depending on the DB instance class and store allocated, it could take several minutes for the new instance to be available.



CLI

To create a PostgreSQL DB instance, use the AWS CLI `create-db-instance` command with the following parameters:

- `--db-instance-identifier`
- `--allocated-storage`
- `--db-instance-class`
- `--engine`
- `--master-username`
- `--master-user-password`

Example

For Linux, OS X, or Unix:

```
aws rds create-db-instance
    --db-instance-identifier pgdbinstance \
    --allocated-storage 20 \
    --db-instance-class db.t2.small \
    --engine postgres \
    --master-username masterawsuser \
    --master-user-password masteruserpassword
```

For Windows:

```
aws rds create-db-instance
    --db-instance-identifier pgdbinstance ^
    --allocated-storage 20 ^
    --db-instance-class db.t2.small ^
    --engine postgres ^
    --master-username masterawsuser ^
    --master-user-password masteruserpassword
```

This command should produce output similar to the following:

```
DBINSTANCE  pgdbinstance  db.t2.small  postgres  20  sa  creating  3  ****  n  9.3
SECGROUP  default  active
PARAMGRP  default.PostgreSQL9.3  in-sync
```

API

To create a PostgreSQL DB instance, use the Amazon RDS API`CreateDBInstance` command with the following parameters:

- `Engine = postgres`
- `DBInstanceIdentifier = pgdbinstance`
- `DBInstanceClass = db.t2.small`
- `AllocatedStorage = 20`
- `BackupRetentionPeriod = 3`
- `MasterUsername = masterawsuser`
- `MasterUserPassword = masteruserpassword`

Example

```
https://rds.amazonaws.com/
    ?Action=CreateDBInstance
    &AllocatedStorage=20
    &BackupRetentionPeriod=3
    &DBInstanceClass=db.t2.small
    &DBInstanceIdentifier=pgdbinstance
    &DBName=mydatabase
    &DBSecurityGroups.member.1=mysecuritygroup
    &DBSubnetGroup=mydbsubnetgroup
    &Engine=postgres
    &MasterUserPassword=<masteruserpassword>
    &MasterUsername=<masterawsuser>
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2013-09-09
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20140212/us-west-2/rds/aws4_request
    &X-Amz-Date=20140212T190137Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=60d520ca0576c191b9eac8dbfe5617ebb6a6a9f3994d96437a102c0c2c80f88d
```

Related Topics

- Amazon RDS DB Instances (p. 90)
- DB Instance Class (p. 92)
- Deleting a DB Instance (p. 126)

Connecting to a DB Instance Running the PostgreSQL Database Engine

Once Amazon RDS provisions your DB instance, you can use any standard SQL client application to connect to the instance. It is important to note that the security group you assigned to the DB instance when you created it must allow access to the DB instance. If you have difficulty connecting to the DB instance, the problem is most often with the access rules you set up in the security group you assigned to the DB instance.

You can use the AWS Management Console, the AWS CLI describe-db-instances command, or the Amazon RDS API DescribeDBInstances action to list the details of an Amazon RDS DB instance, including its endpoint. If an endpoint value is `myinstance.123456789012.us-east-1.rds.amazonaws.com:5432`, then you would specify the following values in a PostgreSQL connection string:

- For host or host name, specify

```
myinstance.123456789012.us-east-1.rds.amazonaws.com
```

- For port, specify

```
5432
```

Two common causes of connection failures to a new DB instance are:

- The DB instance was created using a security group that does not authorize connections from the device or Amazon EC2 instance where the PostgreSQL application or utility is running. If the DB instance was created in a VPC, it must have a VPC security group that authorizes the connections. If the DB instance was created outside of a VPC, it must have a DB security group that authorizes the connections.
- The DB instance was created using the default port of 5432, and your company has firewall rules blocking connections to that port from devices in your company network. To fix this failure, modify the instance to use a different port.

This section shows two ways to connect to a PostgreSQL DB instance. The first example uses *pgAdmin*, a popular Open Source administration and development tool for PostgreSQL. You can download and use *pgAdmin* without having a local instance of PostgreSQL on your client computer. The second example uses *psql*, a command line utility that is part of a PostgreSQL installation. To use *psql*, you must have a PostgreSQL installed on your client computer or have installed the *psql* client on your machine.

In this example, you connect to a PostgreSQL DB instance using pgAdmin.

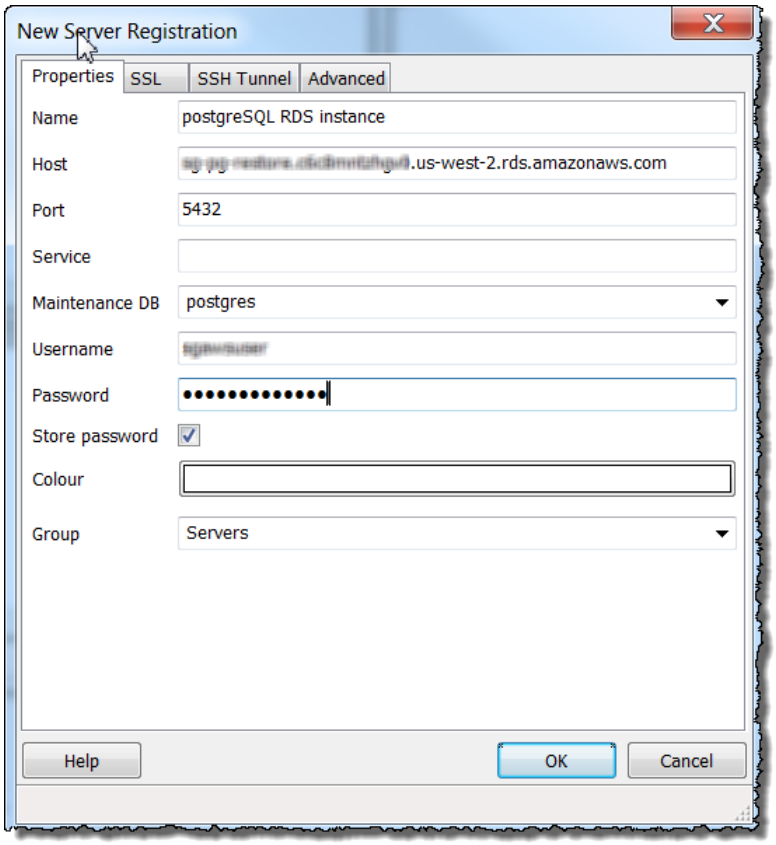
Using pgAdmin to Connect to a PostgreSQL DB Instance

To connect to a PostgreSQL DB instance using pgAdmin

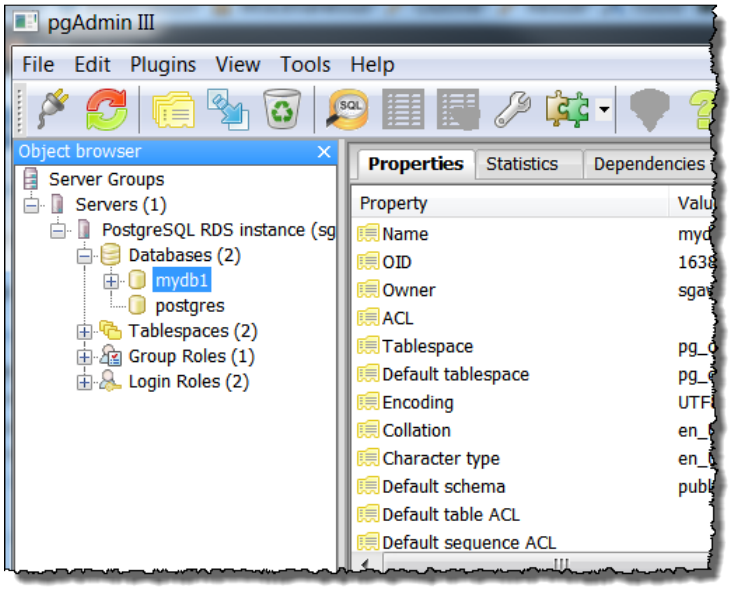
1. Launch the *pgAdmin* application on your client computer. You can install *pgAdmin* from http://www.pgadmin.org/.
2. Select **Add Server** from the **File** menu.
3. In the **New Server Registration** dialog box, enter the DB instance endpoint (for example, mypostgresql.c6c8dntfzzhgv0.us-west-2.rds.amazonaws.com) in the **Host** text box. Do not include

the colon or port number as shown on the Amazon RDS console (mypostgresql.c6c8dntfzzhgv0.us-west-2.rds.amazonaws.com:*5432*).

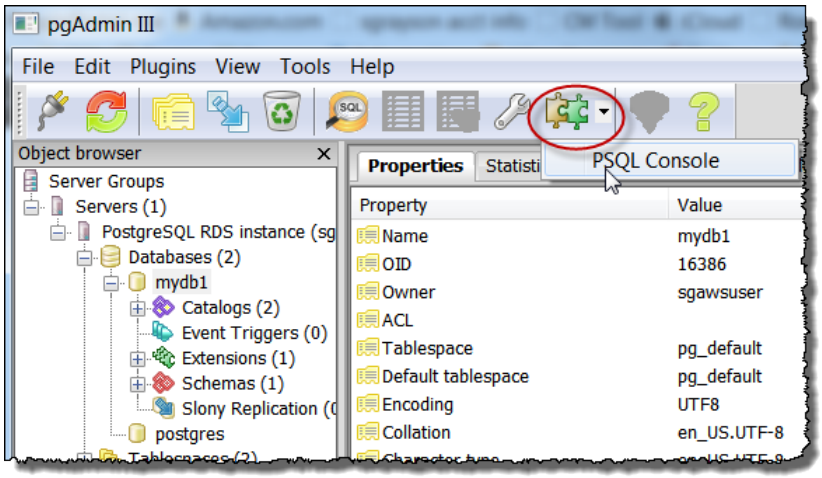
Enter the port you assigned to the DB instance into the **Port** text box. Enter the user name and user password you entered when you created the DB instance into the **Username** and **Password** text boxes, respectively.



4. Click **OK**.
5. In the **Object browser**, expand the **Server Groups**. Select the Server (the DB instance) you created, and then select the database name.



6. Click the plugin icon and click **PSQL Console**. The *psql* command window opens for the default database you created.



7. Use the command window to enter SQL or *psql* commands. Type `\q` to close the window.

Using *psql* to Connect to a PostgreSQL DB Instance

If your client computer has PostgreSQL installed, you can use a local instance of *psql* to connect to a PostgreSQL DB instance. To connect to your PostgreSQL DB instance using *psql*, you need to provide host information and access credentials.

The following format is used to connect to a PostgreSQL DB instance on Amazon RDS. Note that you will be prompted for a password; use the --no-password option for batch jobs or scripts.

For Linux, OS X, or Unix:

```
psql \
   --host=<DB instance endpoint> \
   --port=<port> \
   --username <master user name> \
```

```
   --password \
   --dbname=<database name>
```

For Windows:

```
psql ^
   --host=<DB instance endpoint> ^
   --port=<port> ^
   --username <master user name> ^
   --password ^
   --dbname=<database name>
```

For example, the following command connects to a database called `mypgdb` on a PostgreSQL DB instance called `mypostgresql` using fictitious credentials:

```
psql --host=mypostgresql.c6c8mwvfdgv0.us-west-2.rds.amazonaws.com --port=5432 --
username=awsuser --password --dbname=mypgdb
```

Troubleshooting Connection Issues

By far the most common problem that occurs when attempting to connect to a database on a DB instance is the access rules in the security group assigned to the DB instance. If you used the default DB security group when you created the DB instance, chances are good that the security group did not have the rules that will allow you to access the instance. For more information about Amazon RDS security groups, see Amazon RDS Security Groups (p. 375)

The most common error is *could not connect to server: Connection timed out*. If you receive this error, check that the host name is the DB instance endpoint and that the port number is correct. Check that the DB security group assigned to the DB instance has the necessary rules to allow access through any firewall your connection may be going through.

Related Topics

- Amazon RDS DB Instances (p. 90)
- Creating a DB Instance Running the PostgreSQL Database Engine (p. 1172)
- Amazon RDS Security Groups (p. 375)
- Deleting a DB Instance (p. 126)

Modifying a DB Instance Running the PostgreSQL Database Engine

You can change the settings of a DB instance to accomplish tasks such as adding additional storage or changing the DB instance class. This topic guides you through modifying an Amazon RDS PostgreSQL DB instance, and describes the settings for PostgreSQL instances. For information about additional tasks, such as renaming, rebooting, deleting, tagging, or upgrading an Amazon RDS DB instance, see Amazon RDS DB Instance Lifecycle (p. 111). We recommend that you test any changes on a test instance before modifying a production instance so you better understand the impact of a change. This is especially important when upgrading database versions.

You can have the changes apply immediately or have them applied during the DB instance's next maintenance window. Applying changes immediately can cause an outage in some cases; for more information on the impact of the **Apply Immediately** option when modifying a DB instance, see Modifying an Amazon RDS DB Instance and Using the Apply Immediately Parameter (p. 114).

AWS Management Console

To modify a PostgreSQL DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **DB Instances**, and then select the DB instance that you want to modify.
3. Choose **Instance Actions**, and then choose **Modify**. The **Modify DB Instance** page appears.
4. Change any of the settings that you want. For information about each setting, see Settings for PostgreSQL DB Instances (p. 1185).
5. To apply the changes immediately, select **Apply Immediately**. Selecting this option can cause an outage in some cases. For more information, see The Impact of Apply Immediately (p. 114).
6. When all the changes are as you want them, choose **Continue**.
7. On the confirmation page, review your changes. If they are correct, choose **Modify DB Instance** to save your changes.

   Alternatively, choose **Back** to edit your changes, or choose **Cancel** to cancel your changes.

CLI

To modify a PostgreSQL DB instance, use the AWS CLI command `modify-db-instance`.

Example

The following code modifies `pgdbinstance` by setting the backup retention period to 1 week (7 days) and disabling automatic minor version upgrades. These changes are applied during the next maintenance window.

Parameters

- `--db-instance-identifier`—the name of the db instance
- `--backup-retention-period`—the number of days to retain automatic backups.
- `--no-auto-minor-version-upgrade`—disallow automatic minor version upgrades. To allow automatic minor version upgrades, use `--auto-minor-version-upgrade`.

- `--no-apply-immediately`—apply changes during the next maintenance window. To apply changes immediately, use `--apply-immediately`.

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier pgdbinstance \
    --backup-retention-period 7 \
    --no-auto-minor-version-upgrade \
    --no-apply-immediately
```

For Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier pgdbinstance ^
    --backup-retention-period 7 ^
    --no-auto-minor-version-upgrade ^
    --no-apply-immediately
```

API

To modify a PostgreSQL DB instance, use the `ModifyDBInstance` action.

Example

The following code modifies `pgdbinstance` by setting the backup retention period to 1 week (7 days) and disabling automatic minor version upgrades. These changes are applied during the next maintenance window.

Parameters

- `DBInstanceIdentifier`—the name of the db instance
- `BackupRetentionPeriod`—the number of days to retain automatic backups.
- `AutoMinorVersionUpgrade=false`—disallow automatic minor version upgrades. To allow automatic minor version upgrades, set the value to `true`.
- `ApplyImmediately=false`—apply changes during the next maintenance window. To apply changes immediately, set the value to `true`.

```
https://rds.us-east-1.amazonaws.com/
   ?Action=ModifyDBInstance
   &ApplyImmediately=false
   &AutoMinorVersionUpgrade=false
   &BackupRetentionPeriod=7
   &DBInstanceIdentifier=mydbinstance
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &Version=2013-09-09
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20131016/us-east-1/rds/aws4_request
   &X-Amz-Date=20131016T233051Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
   &X-Amz-Signature=087a8eb41cb1ab0fc9ec1575f23e73757ffc6a1e42d7d2b30b9cc0be988cff97
```

Settings for PostgreSQL DB Instances

The following table contains details about which settings you can modify, which settings you can't modify, when the changes can be applied, and whether the changes cause downtime for the DB instance.

Setting	Setting Description	When the Change Occurs	Downtime Notes
Allocated Storage	The storage, in gigabytes, that you want to allocate for your DB instance. You can only increase the allocated storage, you can't reduce the allocated storage. You can't modify allocated storage if the DB instance status is `storage-optimization` or if the allocated storage for the DB instance has been modified in the last six hours. The maximum storage allowed depends on the storage type. For more information, see Storage for Amazon RDS (p. 410).	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	No downtime. Performance may be degraded during the change.
Auto Minor Version Upgrade	If you want your DB instance to receive minor engine version upgrades automatically when they become available, click **Yes**. Upgrades are installed only during your scheduled maintenance window.	–	–
Backup Retention Period	The number of days that automatic backups are retained. To disable automatic backups, set the backup retention period to 0. For more information, see Working With Backups (p. 201).	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false and you change the setting from a non-zero value to another non-zero value, the change is applied asynchronously, as soon as possible. Otherwise, the change occurs during the next maintenance window.	An outage occurs if you change from 0 to a non-zero value, or from a non-zero value to 0.
Backup Window	The time range during which automated backups of your databases occur. The backup window is a start time in Universal Coordinated Time (UTC), and a duration in hours.	The change is applied asynchronously, as soon as possible.	–

Setting	Setting Description	When the Change Occurs	Downtime Notes
	For more information, see Working With Backups (p. 201).		
Certificate Authority	The certificate that you want to use.	–	–
Copy Tags to Snapshots	If you have any DB instance tags, this option copies them when you create a DB snapshot. For more information, see Tagging Amazon RDS Resources (p. 129).	–	–
Database Port	The port that you want to use to access the database. The port value must not match any of the port values specified for options in the option group for the DB instance.	The change occurs immediately. This setting ignores the **Apply Immediately** setting.	The DB instance is rebooted immediately.
DB Engine Version	The version of the PostgreSQL database engine that you want to use. Before you upgrade your production DB instances, we recommend that you test the upgrade process on a test instance to verify its duration and to validate your applications.	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	An outage occurs during this change.
DB Instance Class	The DB instance class that you want to use. For more information, see DB Instance Class (p. 92)	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	An outage occurs during this change.
DB Instance Identifier	The DB instance identifier. This value is stored as a lowercase string. For more information about the effects of renaming a DB instance, see Renaming a DB Instance (p. 116).	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	An outage occurs during this change. The DB instance is rebooted.

Setting	Setting Description	When the Change Occurs	Downtime Notes
DB Parameter Group	The parameter group that you want associated with the DB instance. For more information, see Working with DB Parameter Groups (p. 170).	The parameter group change occurs immediately. However, parameter changes only occur when you reboot the DB instance manually without failover. For more information, see Rebooting a DB Instance (p. 119).	An outage doesn't occur during this change. However, parameter changes only occur when you reboot the DB instance manually without failover.
Enable Enhanced Monitoring	**Yes** to enable gathering metrics in real time for the operating system that your DB instance runs on. For more information, see Enhanced Monitoring (p. 258).	–	–
License Model	Select the PostgreSQL License.	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	An outage occurs during this change.
Maintenance Window	The time range during which system maintenance occurs. System maintenance includes upgrades, if applicable. The maintenance window is a start time in Universal Coordinated Time (UTC), and a duration in hours. If you set the window to the current time, there must be at least 30 minutes between the current time and end of the window to ensure any pending changes are applied. For more information, see The Amazon RDS Maintenance Window (p. 103).	The change occurs immediately. This setting ignores the **Apply Immediately** setting.	If there are one or more pending actions that cause an outage, and the maintenance window is changed to include the current time, then those pending actions are applied immediately, and an outage occurs.
Multi-AZ Deployment	**Yes** to deploy your DB instance in multiple Availability Zones; otherwise, **No**. For more information, see Regions and Availability Zones (p. 97).	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	–

Setting	Setting Description	When the Change Occurs	Downtime Notes
New Master Password	The password for your master user. The password must contain from 8 to 30 alphanumeric characters.	The change is applied asynchronously, as soon as possible. This setting ignores the **Apply Immediately** setting.	–
Option Group	No options are available for PostgreSQL DB instances. For more information, see Working with Option Groups (p. 153).	–	–
Publicly Accessible	**Yes** to give the DB instance a public IP address, meaning that it is accessible outside the VPC. To be publicly accessible, the DB instance also has to be in a public subnet in the VPC. **No** to make the DB instance accessible only from inside the VPC. For more information, see Hiding a DB Instance in a VPC from the Internet (p. 401).	The change occurs immediately. This setting ignores the **Apply Immediately** setting.	–
Security Group	The security group you want associated with the DB instance. For more information, see Working with DB Security Groups (EC2-Classic Platform) (p. 380).	The change is applied asynchronously, as soon as possible. This setting ignores the **Apply Immediately** setting.	–

Setting	Setting Description	When the Change Occurs	Downtime Notes
Storage Type	The storage type that you want to use. For more information, see Amazon RDS Storage Types (p. 410).	If **Apply Immediately** is set to true, the change occurs immediately. If **Apply Immediately** is set to false, the change occurs during the next maintenance window.	The following changes all result in a brief outage while the process starts. After that, you can use your database normally while the change takes place. • From **General Purpose (SSD)** to **Magnetic**. • From **General Purpose (SSD)** to **Provisioned IOPS (SSD)**, if the DB instance is single-AZ or if you are using a custom parameter group and the DB instance is a read replica. There is no outage for a multi-AZ DB instance or for the source DB instance of a read replica. • From **Magnetic** to **General Purpose (SSD)**. • From **Magnetic** to **Provisioned IOPS (SSD)**. • From **Provisioned IOPS (SSD)** to **Magnetic**. • From **Provisioned IOPS (SSD)** to **General Purpose (SSD)**, if the DB instance is single-AZ or if you are using a custom parameter group and the DB instance is a read replica. There is no outage for a multi-AZ

Setting	Setting Description	When the Change Occurs	Downtime Notes
			DB instance or for the source DB instance of a read replica.
Subnet Group	The subnet group for the DB instance. You can use this setting to move your DB instance to a different VPC. If your DB instance is not in a VPC, you can use this setting to move your DB instance into a VPC. For more information, see Moving a DB Instance Not in a VPC into a VPC (p. 405).	–	–

Related Topics

- the section called “Rebooting a DB Instance” (p. 119) (p. 119)
- the section called “Connecting to a DB Instance Running the PostgreSQL Database Engine” (p. 1179) (p. 1179)
- the section called “Upgrading the PostgreSQL DB Engine” (p. 1191)

Upgrading the PostgreSQL DB Engine

When Amazon Relational Database Service (Amazon RDS) supports a new version of a database engine, you can upgrade your DB instances to the new version. There are two kinds of upgrades: major version upgrades and minor version upgrades.

Amazon RDS supports major and minor version upgrades for PostgreSQL DB instances.

Major version upgrades can contain database changes that are not backward-compatible with existing applications. As a result, Amazon Relational Database Service (Amazon RDS) doesn't apply major version upgrades automatically; you must manually modify your DB instance. You can initiate a major version upgrade manually by modifying your instance. However, there are recommended steps to follow when performing a major version upgrade. For details, see Major Version Upgrades (p. 1191).

You can initiate a minor version upgrade manually by modifying your instance, or select the **Auto Minor Version Upgrade** option when creating or modifying a DB instance to have your instance automatically upgraded once the new version is tested and approved by Amazon RDS.

AWS RDS does not automatically upgrade PostgreSQL extensions. To upgrade an extension, you must use the ALTER EXTENSION UPDATE command. For example, to upgrade PostGIS when you upgrade the PostgreSQL DB engine from 9.4.x to 9.5.x, you would run the following command:

```
ALTER EXTENSION POSTGIS UPDATE TO '2.2.2'
```

> **Note**
> If you are running the PostGIS extension in your Amazon RDS PostgreSQL instance, make sure and follow the PostGIS upgrade instructions before you upgrade PostgreSQL.

Overview of Upgrading

If your backup retention period is greater than 0, Amazon RDS takes two DB snapshots during both the major and minor upgrade process. The first DB snapshot is of the DB instance before any upgrade changes have been made. If the upgrade doesn't work for your databases, you can restore this snapshot to create a DB instance running the old version. The second DB snapshot is taken after the upgrade completes.

> **Note**
> Amazon RDS only takes DB snapshots if you have set the backup retention period for your DB instance to a number greater than 0. To change your backup retention period, see Modifying a DB Instance Running the Oracle Database Engine (p. 967).

After an upgrade is complete, you can't revert to the previous version of the database engine. If you want to return to the previous version, restore the DB snapshot that was taken before the upgrade to create a new DB instance.

If your DB instance is in a Multi-AZ deployment, both the primary and standby replicas are upgraded. The primary and standby DB instances are upgraded at the same time, and you experience an outage until the upgrade is complete.

Major Version Upgrades

Major version upgrades can contain database changes that are not backward-compatible with previous versions of the database. This functionality can cause your existing applications to stop working correctly. As a result, Amazon RDS doesn't apply major version upgrades automatically; you must modify your DB instance manually to perform a major version upgrade. You should thoroughly test any upgrade to verify that your applications work correctly before applying the upgrade to your production DB

instances. A best practice we recommend is to perform the major version upgrade on a restored instance that you create from a DB snapshot.

Amazon RDS supports an in-place upgrade from the following:

- A PostgreSQL 9.3.x DB instance to a PostgreSQL 9.4.x DB instance
- A PostgreSQL 9.4.x DB instance to a PostgreSQL 9.5.x DB instance
- A PostgreSQL 9.5.x DB instance to a PostgreSQL 9.6.x DB instance

Amazon RDS uses the `pg_upgrade` utility found at http://www.postgresql.org/docs/9.4/static/pgupgrade.html to safely upgrade your instance.

Because some PostgreSQL minor versions updates for 9.3 were released after major version 9.4 was released, you cannot upgrade from version 9.3.9 to 9.4.1, and you cannot upgrade from version 9.3.10 to 9.4.1 or 9.4.4.

Read Replicas cannot undergo a major version upgrade. The source instance can undergo a major version upgrade, but all Read Replicas remain as readable nodes on the previous engine version. After a source instance is upgraded, its Read Replicas can no longer replicate changes performed on the source instance. We recommend that you either promote your Read Replicas, or delete and recreate them after the source instance has upgraded to a different major version.

Major Version Upgrade Process

We recommend the following process when upgrading an Amazon RDS PostgreSQL DB instance:

1. **Have a version-compatible parameter group ready** – If you are using a custom parameter group, you must specify either a default parameter group for the new DB engine version or create your own custom parameter group for the new DB engine version. Associating the new parameter group with the DB instance requires a customer-initiated database reboot after the upgrade completes. The instance's parameter group status will show `pending-reboot` if the instance needs to be rebooted to apply the parameter group changes. An instance's parameter group status can be viewed in the AWS console or by using a "describe" call such as `describe-db-instances`.
2. **Check for unsupported usage:**

    1. **Prepared transactions** – Commit or roll back all open prepared transactions before attempting an upgrade.

        You can use the following query to verify that there are no open prepared transactions on your instance:

        ```
        SELECT count(*) FROM pg_catalog.pg_prepared_xacts;
        ```

    2. **The `line` data type** – If you are upgrading an RDS PostgreSQL 9.3 instance, you must remove all uses of the `line` data type before attempting an upgrade, because the `line` data type was not fully implemented in PostgreSQL until version 9.4.

        You can use the following query on each database to be upgraded to verify that there are no uses of the `line` data type in each database:

        ```
        SELECT count(*) FROM pg_catalog.pg_class c, pg_catalog.pg_namespace n,
         pg_catalog.pg_attribute a
          WHERE c.oid = a.attrelid
              AND NOT a.attisdropped
              AND a.atttypid = 'pg_catalog.line'::pg_catalog.regtype
              AND c.relnamespace = n.oid
        ```

```
    AND n.nspname !~ '^pg_temp_'
    AND n.nspname !~ '^pg_toast_temp_'
    AND n.nspname NOT IN ('pg_catalog', 'information_schema');
```

Note
To list all databases on an instance, use the following query:

```
SELECT d.datname FROM pg_catalog.pg_database d WHERE d.datallowconn = true;
```

3. **Reg* data types** – Remove all uses of the *reg** data types before attempting an upgrade, because these data types contain information that cannot be persisted with `pg_upgrade`. Uses of *reg** data types cannot be upgraded, except for `regtype` and `regclass`. Remove all usages before attempting an upgrade.

   You can use the following query to verify that there are no uses of unsupported *reg** data types in each database:

```
SELECT count(*) FROM pg_catalog.pg_class c, pg_catalog.pg_namespace n,
 pg_catalog.pg_attribute a
  WHERE c.oid = a.attrelid
      AND NOT a.attisdropped
      AND a.atttypid IN ('pg_catalog.regproc'::pg_catalog.regtype,
                         'pg_catalog.regprocedure'::pg_catalog.regtype,
                         'pg_catalog.regoper'::pg_catalog.regtype,
                         'pg_catalog.regoperator'::pg_catalog.regtype,
                         'pg_catalog.regconfig'::pg_catalog.regtype,
                         'pg_catalog.regdictionary'::pg_catalog.regtype)
      AND c.relnamespace = n.oid
      AND n.nspname NOT IN ('pg_catalog', 'information_schema');
```

3. Perform a `VACUUM` operation before upgrading your instance. The `pg_upgrade` utility vacuums each database when you upgrade to a different major version. If you haven't performed a `VACUUM` operation, the upgrade process can take much longer, causing increased downtime for your RDS instance.
4. Perform a dry run of your major version upgrade. We highly recommend testing major version upgrade on a duplicate of your production database before attempting it on your production database. To create a duplicate test instance, you can either restore your database from a recent snapshot or point-in-time restore your database to its latest restorable time. After you have completed the major version upgrade, consider testing your application on the upgraded database with a similar workload in order to verify that everything works as expected. After the upgrade is verified, you can delete this test instance.
5. We recommend that you perform a backup before performing the major version upgrade so that you have a known restore point for your database. Note that we create a DB snapshot of your DB instance before and after upgrading.
6. Upgrade your production instance. If the dry-run major version upgrade was successful, you should now be able to upgrade your production database with confidence.

You can use Amazon RDS to view two logs that the `pg_upgrade` utility produces: `pg_upgrade_internal.log` and `pg_upgrade_server.log`. Amazon RDS appends a timestamp to the file name for these logs. You can view these logs as you can any other log.

You cannot perform a point-in-time restore of your instance to a point in time during the upgrade process. During the upgrade process, RDS takes an automatic backup of the instance after the upgrade has been performed. You can perform a point-in-time restore to times before the upgrade began and after the automatic backup of your instance has completed.

The `public` and `template1` databases and the `public` schema in every database on the instance are renamed during the major version upgrade. These objects will appear in the logs with their original name and a random string appended. The string is appended so that custom settings such as the `locale` and `owner` are preserved during the major version upgrade. Once the upgrade completes, the objects are renamed back to their original names.

Minor Version Upgrades for PostgreSQL

Minor version upgrades occur automatically if a minor upgrade has been tested and approved by Amazon RDS and you selected the **Auto Minor Version Upgrade** option. In all other cases, you must modify the DB instance manually to perform a minor version upgrade. If you select the **Auto Minor Version Upgrade** option when creating or modifying a DB instance, you can have your instance automatically upgraded after the new version is tested and approved by Amazon RDS.

If your PostgreSQL DB instance is using read replication, you must upgrade all of the Read Replicas before upgrading the source instance. If the DB instance is in a Multi-AZ deployment, both the primary and standby replicas are upgraded, and the instance might not be available until the upgrade is complete.

AWS Management Console

To apply a DB engine major version upgrade to a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Instances**.
3. Choose the check box for the DB instance that you want to upgrade.
4. Choose **Instance Actions**, and then choose **Modify**.
5. For **DB Engine Version**, choose the new version.
6. To upgrade immediately, select **Apply Immediately**. To delay the upgrade to the next maintenance window, clear **Apply Immediately**.
7. Choose **Continue**.
8. Review the modification summary information. To proceed with the upgrade, choose **Modify DB Instance**. To cancel the upgrade, choose **Cancel** or **Back**.

CLI

To upgrade the engine version of a DB instance, use the AWS CLI modify-db-instance command. Specify the following parameters:

- `--db-instance-identifier` – the name of the db instance.
- `--engine-version` – the version number of the database engine to upgrade to.
- `--allow-major-version-upgrade` – to to upgrade major version.
- `--no-apply-immediately` – apply changes during the next maintenance window. To apply changes immediately, use `--apply-immediately`.

Example

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier <mydbinstance> \
```

```
    --engine-version <new_version> \
    --allow-major-version-upgrade \
    --apply-immediately
```

For Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier <mydbinstance> ^
    --engine-version <new_version> ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

API

To upgrade the engine version of a DB instance, use the ModifyDBInstance action. Specify the following parameters:

- `DBInstanceIdentifier` – the name of the db instance, for example *`mydbinstance`*.
- `EngineVersion` – the version number of the database engine to upgrade to.
- `AllowMajorVersionUpgrade` – set to `true` to upgrade major version.
- `ApplyImmediately` – whether to apply changes immediately or during the next maintenance window. To apply changes immediately, set the value to *`true`*. To apply changes during the next maintenance window, set the value to *`false`*.

Example

```
https://rds.us-east-1.amazonaws.com/
   ?Action=ModifyDBInstance
   &ApplyImmediately=false
   &DBInstanceIdentifier=mydbinstance
   &EngineVersion=new_version
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &Version=2013-09-09
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20131016/us-east-1/rds/aws4_request
   &X-Amz-Date=20131016T233051Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
   &X-Amz-Signature=087a8eb41cb1ab5f99e81575f23e73757ffc6a1e42d7d2b30b9cc0be988cff97
```

Related Topics

- Amazon RDS Maintenance (p. 102)
- Updating the Operating System for a DB Instance or DB Cluster (p. 108)

Importing Data into PostgreSQL on Amazon RDS

If you have an existing PostgreSQL deployment that you want to move to Amazon RDS, the complexity of your task depends on the size of your database and the types of database objects that you are transferring. For example, consider a database that contains datasets on the order of gigabytes, along with stored procedures and triggers. Such a database is going to be more complicated than a simple database with only a few megabytes of test data and no triggers or stored procedures.

We recommend that you use native PostgreSQL database migration tools under the following conditions:

- You have a homogeneous migration, where you are migrating from a database with the same database engine as the target database.
- You are migrating an entire database.
- The native tools allow you to migrate your system with minimal downtime.

In most other cases, performing a database migration using AWS Database Migration Service (AWS DMS) is the best approach. AWS DMS can migrate databases without downtime and, for many database engines, continue ongoing replication until you are ready to switch over to the target database. You can migrate to either the same database engine or a different database engine using AWS DMS. If you are migrating to a different database engine than your source database, you can use the AWS Schema Conversion Tool to migrate schema objects that are not migrated by AWS DMS. For more information about AWS DMS, see What is AWS Database Migration Service.

Modify your DB parameter group to include the following settings *for your import only*. You should test the parameter settings to find the most efficient settings for your DB instance size. You also need to revert back to production values for these parameters after your import completes.

Modify your DB instance settings to the following:

- Disable DB instance backups (set backup_retention to 0).
- Disable Multi-AZ.

Modify your DB parameter group to include the following settings. You should only use these settings when importing data. You should test the parameter settings to find the most efficient settings for your DB instance size. You also need to revert back to production values for these parameters after your import completes.

Parameter	Recommended Value When Importing	Description
`maintenance_work_mem`	524288, 1048576, 2097152 or 4194304 (in KB). These settings are comparable to 512 MB, 1 GB, 2 GB, and 4 GB.	The value for this setting depends on the size of your host. This parameter is used during CREATE INDEX statements and each parallel command can use this much memory. Calculate the best value so that you don't set this value so high that you run out of memory.
`checkpoint_segments`	256	The value for this setting consumes more disk space, but gives you less contention on your WAL logs. For PostgreSQL versions 9.5.x and 9.6.x, this value would be `max_wal_size`.
`checkpoint_timeout`	1800	The value for this setting allows for less frequent WAL rotation.

Parameter	Recommended Value When Importing	Description
synchronous_commit	Off	Disable this setting to speed up writes. Turning this parameter off can increase the risk of data loss in the event of a server crash (do not turn off FSYNC)
wal_buffers	8192	This is value is in 8 KB units. This again helps your WAL generation speed
autovacuum	Off	Disable the PostgreSQL auto vacuum parameter while you are loading data so that it doesn't use resources

Use the `pg_dump -Fc` (compressed) or `pg_restore -j` (parallel) commands with these settings.

Note
The PostgreSQL command `pg_dumpall` requires super_user permissions that are not granted when you create a DB instance, so it cannot be used for importing data.

Importing a PostgreSQL Database from an Amazon EC2 Instance

If you have data in a PostgreSQL server on an Amazon EC2 instance and want to move it to a PostgreSQL DB instance, you can use the following process. The following list shows the steps to take. Each step is discussed in more detail in the following sections.

1. Create a file using pg_dump that contains the data to be loaded
2. Create the target DB instance
3. Use *psql* to create the database on the DB instance and load the data
4. Create a DB snapshot of the DB instance

Step 1: Create a file using `pg_dump` that contains the data to be loaded

`pg_dump` uses the COPY command to create a schema and data dump of a PostgreSQL database. The dump script generated by `pg_dump` loads data into a database with the same name and recreates the tables, indexes, and foreign keys. You can use the `pg_restore` command and the `-d` parameter to restore the data to a database with a different name.

Before you create the data dump, you should query the tables to be dumped to get a row count so you can confirm the count on the target DB instance.

The following command creates a dump file called mydb2dump.sql for a database called mydb2.

```
prompt>pg_dump dbname=mydb2 -f mydb2dump.sql
```

Step 2: Create the target DB instance

Create the target PostgreSQL DB instance using either the Amazon RDS console, AWS CLI, or API. Create the instance with the backup retention setting set to 0 and disable Multi-AZ. Doing so allows faster data import. You must create a database on the instance before you can dump the data. The database can

have the same name as the database that is contained the dumped data. Alternatively, you can create a database with a different name. In this case, you use the pg_restore command and the -d parameter to restore the data into the newly named database.

For example, the following commands can be used to dump, restore, and rename a database.

```
pg_dump -Fc -v -h [endpoint of instance] -U [master username] [database] > [database].dump
createdb [new database name]
pg_restore -v -h [endpoint of instance] -U [master username] -d [new database
 name] [database].dump
```

Step 3: Use *psql* to create the database on the DB instance and load the data

You can use the same connection you used to execute the pg_dump command to connect to the target DB instance and recreate the database. Using *psql*, you can use the master user name and master password to create the database on the DB instance

The following example uses *psql* and a dump file named mydb2dump.sql to create a database called mydb2 on a PostgreSQL DB instance called mypginstance:

For Linux, OS X, or Unix:

```
psql \
   -f mydb2dump.sql \
   --host mypginstance.c6c8mntzhgv0.us-west-2.rds.amazonaws.com \
   --port 8199 \
   --username myawsuser \
   --password password \
   --dbname mydb2
```

For Windows:

```
psql ^
   -f mydb2dump.sql ^
   --host mypginstance.c6c8mntzhgv0.us-west-2.rds.amazonaws.com ^
   --port 8199 ^
   --username myawsuser ^
   --password password ^
   --dbname mydb2
```

Step 4: Create a DB snapshot of the DB instance

Once you have verified that the data was loaded into your DB instance, we recommend that you create a DB snapshot of the target PostgreSQL DB instance. DB snapshots are complete backups of your DB instance that can be used to restore your DB instance to a known state. A DB snapshot taken immediately after the load protects you from having to load the data again in case of a mishap and can also be used to seed new database instances. For information about creating a DB snapshot, see Creating a DB Snapshot (p. 207).

Using the \copy Command to Import Data to a Table on a PostgreSQL DB Instance

You can run the \copy command from the *psql* prompt to import data into a table on a PostgreSQL DB instance. The table must already exist on the DB instance. For more information on the \copy command, see the PostgreSQL documentation.

Note
The \copy command does not provide confirmation of actions, such as a count of rows inserted. PostgreSQL does provide error messages if the copy command fails due to an error.

Create a .csv file from the data in the source table, log on to the target database on the PostgreSQL instance using *psql*, and then run the following command. This example uses *source-table* as the source table name, *source-table.csv* as the .csv file, and *target-db* as the target database:

```
target-db=> \copy source-table from 'source-table.csv' with DELIMITER ',';
```

You can also run the following command from your client computer command prompt. This example uses *source-table* as the source table name, *source-table.csv* as the .csv file, and *target-db* as the target database:

For Linux, OS X, or Unix:

```
$psql target-db \
    -U <admin user> \
    -p <port> \
    -h <DB instance name> \
    -c "\copy source-table from 'source-table.csv' with DELIMITER ','"
```

For Windows:

```
$psql target-db ^
    -U <admin user> ^
    -p <port> ^
    -h <DB instance name> ^
    -c "\copy source-table from 'source-table.csv' with DELIMITER ','"
```

Common DBA Tasks for PostgreSQL

This section describes the Amazon RDS-specific implementations of some common DBA tasks for DB instances running the PostgreSQL database engine. In order to deliver a managed service experience, Amazon RDS does not provide shell access to DB instances, and it restricts access to certain system procedures and tables that require advanced privileges.

For information about working with PostgreSQL log files on Amazon RDS, see PostgreSQL Database Log Files (p. 322)

Topics

Creating Roles

When you create a DB instance, the master user system account that you create is assigned to the `rds_superuser` role. The `rds_superuser` role is a pre-defined Amazon RDS role similar to the PostgreSQL superuser role (customarily named postgres in local instances), but with some restrictions. As with the PostgreSQL superuser role, the `rds_superuser` role has the most privileges on your DB instance and you should not assign this role to users unless they need the most access to the DB instance.

The following example shows how to create a user and then grant the user the `rds_superuser` role. User-defined roles, such as `rds_superuser`, have to be granted.

```
create role testuser with password 'testuser' login;
CREATE ROLE
grant rds_superuser to testuser;
GRANT ROLE
```

Managing PostgreSQL Database Access

By default, when PostgreSQL database objects are created, they receive "public" access privileges. You can revoke all privileges to a database and then explicitly add privileges back as you need them.

As the master user, you can remove all privileges from a database using the following command format:

```
revoke all on database <database name> from public;
REVOKE
```

You can then add privileges back to a user. For example, the following command grants connect access to a user named *mytestuser* to a database named *test*.

```
grant connect on database test to mytestuser;
```

```
GRANT
```

Note that on a local instance, you could specify database privileges in the pg_hba.conf file, but when using PostgreSQL with Amazon RDS it is better to restrict privileges at the Postgres level. Changes to the pg_hba.conf file require a server restart so you cannot edit the pg_hba.conf in Amazon RDS, but privilege changes at the Postgres level occur immediately.

Working with PostgreSQL Parameters

PostgreSQL parameters that you would set for a local PostgreSQL instance in the *postgresql.conf* file are maintained in the DB parameter group for your DB instance. If you create a DB instance using the default parameter group, the parameter settings are in the parameter group called *default.postgres9.6*.

When you create a DB instance, the parameters in the associated DB parameter group are loaded. You can modify parameter values by changing values in the parameter group. You can also change parameter values, if you have the security privileges to do so, by using the ALTER DATABASE, ALTER ROLE, and the SET commands. Note that you cannot use the command line `postgres` command nor the `env PGOPTIONS` command because you will have no access to the host.

Keeping track of PostgreSQL parameter settings can occasionally be difficult. Use the following command to list current parameter settings and the default value:

```
select name, setting, boot_val, reset_val, unit
from pg_settings
order by name;
```

For an explanation of the output values, see the `pg_settings` topic in the PostgreSQL documentation.

If you set the memory settings too large for `max_connections`, `shared_buffers`, or `effective_cache_size`, you will prevent the PostgreSQL instance from starting up. Note that some parameters use units that you might not be familiar with; for example, `shared_buffers` sets the number of 8 KB shared memory buffers used by the server.

The following error is written to the *postgres.log* file when the instance is attempting to start up, but incorrect parameter settings are preventing it from starting.

```
2013-09-18 21:13:15 UTC::@:[8097]:FATAL:  could not map anonymous shared
memory: Cannot allocate memory
2013-09-18 21:13:15 UTC::@:[8097]:HINT:  This error usually means that
PostgreSQL's request for a shared memory segment exceeded available memory or
swap space. To reduce the request size (currently 3514134274048 bytes), reduce
PostgreSQL's shared memory usage, perhaps by reducing shared_buffers or
max_connections.
```

There are two types of PostgreSQL parameters, static and dynamic. Static parameters require that the DB instance be rebooted before they are applied. Dynamic parameters can be applied immediately. The following table shows parameters you can modify for a PostgreSQL DB instance and the parameter's type:

Parameter Name	Apply_Type	Description
`application_name`	Dynamic	Sets the application name to be reported in statistics and logs.
`array_nulls`	Dynamic	Enables input of NULL elements in arrays.
`authentication_timeout`	Dynamic	Sets the maximum allowed time to complete client authentication.

Parameter Name	Apply_Type	Description
autovacuum	Dynamic	Starts the autovacuum subprocess.
autovacuum_analyze_scale_factor	Dynamic	Number of tuple inserts, updates, or deletes prior to analyze as a fraction of reltuples.
autovacuum_analyze_threshold	Dynamic	Minimum number of tuple inserts, updates, or deletes prior to analyze.
autovacuum_naptime	Dynamic	Time to sleep between autovacuum runs.
autovacuum_vacuum_cost_delay	Dynamic	Vacuum cost delay, in milliseconds, for autovacuum.
autovacuum_vacuum_cost_limit	Dynamic	Vacuum cost amount available before napping, for autovacuum.
autovacuum_vacuum_scale_factor	Dynamic	Number of tuple updates or deletes prior to vacuum as a fraction of reltuples.
autovacuum_vacuum_threshold	Dynamic	Minimum number of tuple updates or deletes prior to vacuum.
backslash_quote	Dynamic	Sets whether a backslash (\) is allowed in string literals.
bgwriter_delay	Dynamic	Background writer sleep time between rounds.
bgwriter_lru_maxpages	Dynamic	Background writer maximum number of LRU pages to flush per round.
bgwriter_lru_multiplier	Dynamic	Multiple of the average buffer usage to free per round.
bytea_output	Dynamic	Sets the output format for bytea.
check_function_bodies	Dynamic	Checks function bodies during CREATE FUNCTION.
checkpoint_completion_target	Dynamic	Time spent flushing dirty buffers during checkpoint, as fraction of checkpoint interval.
checkpoint_segments	Dynamic	Sets the maximum distance in log segments between automatic WAL checkpoints.
checkpoint_timeout	Dynamic	Sets the maximum time between automatic WAL checkpoints.
checkpoint_warning	Dynamic	Enables warnings if checkpoint segments are filled more frequently than this.
client_encoding	Dynamic	Sets the client's character set encoding.
client_min_messages	Dynamic	Sets the message levels that are sent to the client.
commit_delay	Dynamic	Sets the delay in microseconds between transaction commit and flushing WAL to disk.
commit_siblings	Dynamic	Sets the minimum concurrent open transactions before performing commit_delay.

Parameter Name	Apply_Type	Description
constraint_exclusion	Dynamic	Enables the planner to use constraints to optimize queries.
cpu_index_tuple_cost	Dynamic	Sets the planner's estimate of the cost of processing each index entry during an index scan.
cpu_operator_cost	Dynamic	Sets the planner's estimate of the cost of processing each operator or function call.
cpu_tuple_cost	Dynamic	Sets the planner's estimate of the cost of processing each tuple (row).
cursor_tuple_fraction	Dynamic	Sets the planner's estimate of the fraction of a cursor's rows that will be retrieved.
datestyle	Dynamic	Sets the display format for date and time values.
deadlock_timeout	Dynamic	Sets the time to wait on a lock before checking for deadlock.
debug_pretty_print	Dynamic	Indents parse and plan tree displays.
debug_print_parse	Dynamic	Logs each query's parse tree.
debug_print_plan	Dynamic	Logs each query's execution plan.
debug_print_rewritten	Dynamic	Logs each query's rewritten parse tree.
default_statistics_target	Dynamic	Sets the default statistics target.
default_tablespace	Dynamic	Sets the default tablespace to create tables and indexes in.
default_transaction_deferrable	Dynamic	Sets the default deferrable status of new transactions.
default_transaction_isolation	Dynamic	Sets the transaction isolation level of each new transaction.
default_transaction_read_only	Dynamic	Sets the default read-only status of new transactions.
default_with_oids	Dynamic	Creates new tables with OIDs by default.
effective_cache_size	Dynamic	Sets the planner's assumption about the size of the disk cache.
effective_io_concurrency	Dynamic	Number of simultaneous requests that can be handled efficiently by the disk subsystem.
enable_bitmapscan	Dynamic	Enables the planner's use of bitmap-scan plans.
enable_hashagg	Dynamic	Enables the planner's use of hashed aggregation plans.
enable_hashjoin	Dynamic	Enables the planner's use of hash join plans.
enable_indexscan	Dynamic	Enables the planner's use of index-scan plans.
enable_material	Dynamic	Enables the planner's use of materialization.

Parameter Name	Apply_Type	Description
enable_mergejoin	Dynamic	Enables the planner's use of merge join plans.
enable_nestloop	Dynamic	Enables the planner's use of nested-loop join plans.
enable_seqscan	Dynamic	Enables the planner's use of sequential-scan plans.
enable_sort	Dynamic	Enables the planner's use of explicit sort steps.
enable_tidscan	Dynamic	Enables the planner's use of TID scan plans.
escape_string_warning	Dynamic	Warns about backslash (\) escapes in ordinary string literals.
extra_float_digits	Dynamic	Sets the number of digits displayed for floating-point values.
from_collapse_limit	Dynamic	Sets the FROM-list size beyond which subqueries are not collapsed.
fsync	Dynamic	Forces synchronization of updates to disk.
full_page_writes	Dynamic	Writes full pages to WAL when first modified after a checkpoint.
geqo	Dynamic	Enables genetic query optimization.
geqo_effort	Dynamic	GEQO: effort is used to set the default for other GEQO parameters.
geqo_generations	Dynamic	GEQO: number of iterations of the algorithm.
geqo_pool_size	Dynamic	GEQO: number of individuals in the population.
geqo_seed	Dynamic	GEQO: seed for random path selection.
geqo_selection_bias	Dynamic	GEQO: selective pressure within the population.
geqo_threshold	Dynamic	Sets the threshold of FROM items beyond which GEQO is used.
gin_fuzzy_search_limit	Dynamic	Sets the maximum allowed result for exact search by GIN.
hot_standby_feedback	Dynamic	Determines whether a hot standby sends feedback messages to the primary or upstream standby.
intervalstyle	Dynamic	Sets the display format for interval values.
join_collapse_limit	Dynamic	Sets the FROM-list size beyond which JOIN constructs are not flattened.
lc_messages	Dynamic	Sets the language in which messages are displayed.
lc_monetary	Dynamic	Sets the locale for formatting monetary amounts.
lc_numeric	Dynamic	Sets the locale for formatting numbers.
lc_time	Dynamic	Sets the locale for formatting date and time values.

Parameter Name	Apply_Type	Description
log_autovacuum_min_duration	Dynamic	Sets the minimum execution time above which autovacuum actions will be logged.
log_checkpoints	Dynamic	Logs each checkpoint.
log_connections	Dynamic	Logs each successful connection.
log_disconnections	Dynamic	Logs end of a session, including duration.
log_duration	Dynamic	Logs the duration of each completed SQL statement.
log_error_verbosity	Dynamic	Sets the verbosity of logged messages.
log_executor_stats	Dynamic	Writes executor performance statistics to the server log.
log_filename	Dynamic	Sets the file name pattern for log files.
log_hostname	Dynamic	Logs the host name in the connection logs.
log_lock_waits	Dynamic	Logs long lock waits.
log_min_duration_statement	Dynamic	Sets the minimum execution time above which statements will be logged.
log_min_error_statement	Dynamic	Causes all statements generating an error at or above this level to be logged.
log_min_messages	Dynamic	Sets the message levels that are logged.
log_parser_stats	Dynamic	Writes parser performance statistics to the server log.
log_planner_stats	Dynamic	Writes planner performance statistics to the server log.
log_rotation_age	Dynamic	Automatic log file rotation will occur after N minutes.
log_rotation_size	Dynamic	Automatic log file rotation will occur after N kilobytes.
log_statement	Dynamic	Sets the type of statements logged.
log_statement_stats	Dynamic	Writes cumulative performance statistics to the server log.
log_temp_files	Dynamic	Logs the use of temporary files larger than this number of kilobytes.
maintenance_work_mem	Dynamic	Sets the maximum memory to be used for maintenance operations.
max_stack_depth	Dynamic	Sets the maximum stack depth, in kilobytes.
max_standby_archive_delay	Dynamic	Sets the maximum delay before canceling queries when a hot standby server is processing archived WAL data.
max_standby_streaming_delay	Dynamic	Sets the maximum delay before canceling queries when a hot standby server is processing streamed WAL data.

Parameter Name	Apply_Type	Description
quote_all_identifiers	Dynamic	Adds quotes (") to all identifiers when generating SQL fragments.
random_page_cost	Dynamic	Sets the planner's estimate of the cost of a non-sequentially fetched disk page.
rds.log_retention_period	Dynamic	Amazon RDS will delete PostgreSQL logs that are older than N minutes.
search_path	Dynamic	Sets the schema search order for names that are not schema-qualified.
seq_page_cost	Dynamic	Sets the planner's estimate of the cost of a sequentially fetched disk page.
session_replication_role	Dynamic	Sets the sessions behavior for triggers and rewrite rules.
sql_inheritance	Dynamic	Causes subtables to be included by default in various commands.
ssl_renegotiation_limit	Dynamic	Sets the amount of traffic to send and receive before renegotiating the encryption keys.
standard_conforming_strings	Dynamic	Causes ... strings to treat backslashes literally.
statement_timeout	Dynamic	Sets the maximum allowed duration of any statement.
synchronize_seqscans	Dynamic	Enables synchronized sequential scans.
synchronous_commit	Dynamic	Sets the current transactions synchronization level.
tcp_keepalives_count	Dynamic	Maximum number of TCP keepalive retransmits.
tcp_keepalives_idle	Dynamic	Time between issuing TCP keepalives.
tcp_keepalives_interval	Dynamic	Time between TCP keepalive retransmits.
temp_buffers	Dynamic	Sets the maximum number of temporary buffers used by each session.
temp_tablespaces	Dynamic	Sets the tablespaces to use for temporary tables and sort files.
timezone	Dynamic	Sets the time zone for displaying and interpreting time stamps.
track_activities	Dynamic	Collects information about executing commands.
track_counts	Dynamic	Collects statistics on database activity.
track_functions	Dynamic	Collects function-level statistics on database activity.
track_io_timing	Dynamic	Collects timing statistics on database I/O activity.
transaction_deferrable	Dynamic	Indicates whether to defer a read-only serializable transaction until it can be executed with no possible serialization failures.

Parameter Name	Apply_Type	Description
transaction_isolation	Dynamic	Sets the current transactions isolation level.
transaction_read_only	Dynamic	Sets the current transactions read-only status.
transform_null_equals	Dynamic	Treats expr=NULL as expr IS NULL.
update_process_title	Dynamic	Updates the process title to show the active SQL command.
vacuum_cost_delay	Dynamic	Vacuum cost delay in milliseconds.
vacuum_cost_limit	Dynamic	Vacuum cost amount available before napping.
vacuum_cost_page_dirty	Dynamic	Vacuum cost for a page dirtied by vacuum.
vacuum_cost_page_hit	Dynamic	Vacuum cost for a page found in the buffer cache.
vacuum_cost_page_miss	Dynamic	Vacuum cost for a page not found in the buffer cache.
vacuum_defer_cleanup_age	Dynamic	Number of transactions by which vacuum and hot cleanup should be deferred, if any.
vacuum_freeze_min_age	Dynamic	Minimum age at which vacuum should freeze a table row.
vacuum_freeze_table_age	Dynamic	Age at which vacuum should scan a whole table to freeze tuples.
wal_writer_delay	Dynamic	WAL writer sleep time between WAL flushes.
work_mem	Dynamic	Sets the maximum memory to be used for query workspaces.
xmlbinary	Dynamic	Sets how binary values are to be encoded in XML.
xmloption	Dynamic	Sets whether XML data in implicit parsing and serialization operations is to be considered as documents or content fragments.
autovacuum_freeze_max_age	Static	Age at which to autovacuum a table to prevent transaction ID wraparound.
autovacuum_max_workers	Static	Sets the maximum number of simultaneously running autovacuum worker processes.
max_connections	Static	Sets the maximum number of concurrent connections.
max_files_per_process	Static	Sets the maximum number of simultaneously open files for each server process.
max_locks_per_transaction	Static	Sets the maximum number of locks per transaction.
max_pred_locks_per_transaction	Static	Sets the maximum number of predicate locks per transaction.
max_prepared_transactions	Static	Sets the maximum number of simultaneously prepared transactions.

Parameter Name	Apply_Type	Description
shared_buffers	Static	Sets the number of shared memory buffers used by the server.
ssl	Static	Enables SSL connections.
track_activity_query_size	Static	Sets the size reserved for pg_stat_activity.current_query, in bytes.
wal_buffers	Static	Sets the number of disk-page buffers in shared memory for WAL.

Amazon RDS uses the default PostgreSQL units for all parameters. The following table shows the PostgreSQL unit value for each parameter.

Parameter Name	Unit
effective_cache_size	8 KB
segment_size	8 KB
shared_buffers	8 KB
temp_buffers	8 KB
wal_buffers	8 KB
wal_segment_size	8 KB
log_rotation_size	KB
log_temp_files	KB
maintenance_work_mem	KB
max_stack_depth	KB
ssl_renegotiation_limit	KB
temp_file_limit	KB
work_mem	KB
log_rotation_age	min
autovacuum_vacuum_cost_delay	ms
bgwriter_delay	ms
deadlock_timeout	ms
lock_timeout	ms
log_autovacuum_min_duration	ms
log_min_duration_statement	ms
max_standby_archive_delay	ms
max_standby_streaming_delay	ms

Parameter Name	Unit
statement_timeout	ms
vacuum_cost_delay	ms
wal_receiver_timeout	ms
wal_sender_timeout	ms
wal_writer_delay	ms
archive_timeout	s
authentication_timeout	s
autovacuum_naptime	s
checkpoint_timeout	s
checkpoint_warning	s
post_auth_delay	s
pre_auth_delay	s
tcp_keepalives_idle	s
tcp_keepalives_interval	s
wal_receiver_status_interval	s

Working with PostgreSQL Autovacuum on Amazon RDS

The autovacuum feature for PostgreSQL databases is a feature that we strongly recommend you use to maintain the health of your PostgreSQL DB instance. Because autovacuum checks for tables that have had a large number of inserted, updated or deleted tuples, it can be used to prevent transaction ID wraparound. Autovacuum automates the execution of the VACUUM and the ANALYZE command; using autovacuum is required by PostgreSQL, not imposed by Amazon RDS, and its use is critical to good performance. The feature is enabled by default for all new Amazon RDS PostgreSQL DB instances, and the related configuration parameters are appropriately set by default. Since our defaults are somewhat generic, you can benefit from tuning parameters to your specific workload. This section can help you perform the needed autovacuum tuning.

For information on creating a process that warns you about transaction ID wraparound, see the AWS Database Blog entry Implement an Early Warning System for Transaction ID Wraparound in Amazon RDS for PostgreSQL.

Topics

- Other Parameters That Affect Autovacuum (p. 1215)
- Autovacuum Logging (p. 1216)

Maintenance Work Memory

One of the most important parameters influencing autovacuum performance is the `maintenance_work_mem` parameter. This parameter determines how much memory you allocate for autovacuum to use to scan a database table and to hold all the row IDs that are going to be vacuumed. If you set the value of the `maintenance_work_mem` parameter too low, the vacuum process might have to scan the table multiple times to complete its work, possibly impacting performance.

When doing calculations to determine the `maintenance_work_mem` parameter value, keep in mind two things:

- The default unit is (KB) for this parameter
- The `maintenance_work_mem` parameter works in conjunction with the `autovacuum_max_workers` parameter. If you have many small tables, allocate more `autovacuum_max_workers` and less `maintenance_work_mem`. If you have large tables (say 100GB+), allocate more memory and fewer workers. You need to have enough memory allocated to succeed on your biggest table. Each `autovacuum_max_workers` can use the memory you allocate, so you should make sure the combination of workers and memory equal the total memory you want to allocate.

In general terms, for large hosts, set the `maintenance_work_mem` parameter to a value between one and two gigabytes. For extremely large hosts, set the parameter to a value between two and four gigabytes. The value you set for this parameter should depend on the workload. Amazon RDS has updated its default for this parameter to be `GREATEST({DBInstanceClassMemory/63963136*1024},65536)`.

Determining if the Tables in Your Database Need Vacuuming

A PostgreSQL database can have two billion "in-flight" unvacuumed transactions before PostgreSQL takes dramatic action to avoid data loss. If the number of unvacuumed transactions reaches (2^31 - 10,000,000), the log will start warning that vacuuming is needed. If the number of unvacuumed transactions reaches (2^31 - 1,000,000), PostgreSQL sets the database to read only and requires an offline, single-user, standalone vacuum. This requires multiple hours or days (depending on size) of downtime. A very detailed explanation of TransactionID wraparound is found in the Postgres documentation.

The following query can be used to show the number of unvacuumed transactions in a database. The datfrozenxid column of a database's pg_database row is a lower bound on the normal XIDs appearing in that database; it is the minimum of the per-table relfrozenxid values within the database.

```
select datname, age(datfrozenxid) from pg_database order
by age(datfrozenxid) desc limit 20;
```

For example, the results of running the above query could be the following:

```
datname    | age
mydb       | 1771757888
template0  | 1721757888
template1  | 1721757888
rdsadmin   | 1694008527
postgres   | 1693881061
(5 rows)
```

When the age of a database hits two billion, TransactionID (XID) wraparound occurs and the database will go into read only. This query can be used to produce a metric and run a few times a day. By default, autovacuum is set to keep the age of transactions to no more than 200,000,000 (`autovacuum_freeze_max_age`).

A sample monitoring strategy might look like this:

- Autovacuum_freeze_max_age is set to 200 million.
- If a table hits 500 million unvacuumed transactions, a low-severity alarm is triggered. This isn't an unreasonable value, but it could indicate that autovacuum isn't keeping up.
- If a table ages to one billion, this should be treated as an actionable alarm. In general, you want to keep ages closer to autovacuum_freeze_max_age for performance reasons. Investigation using the following steps is recommended.
- If a table hits 1.5 billion unvacuumed transactions, a high-severity alarm is triggered. Depending on how quickly your database uses XID's, this alarm would indicate that the system is running out of time to run autovacuum and immediate resolution should be considered.

If a table is constantly breaching these thresholds, you need further modify your autovacuum parameters. By default, VACUUM (which has cost-based delays disabled) is more aggressive than default autovacuum, but, also more intrusive to the system as a whole.

We have the following recommendations:

- Be aware and enable a monitoring mechanism so that you are aware of the age of your oldest transactions.
- For busier tables, perform a manual vacuum freeze regularly during a maintenance window in addition to relying on autovacuum. For information on performing a manual vacuum freeze, see Performing a Manual Vacuum Freeze (p. 1213)

Determining Which Tables Are Currently Eligible for Autovacuum

Often, it is one or two tables in need of vacuuming. Tables whose relfrozenxid value is more than autovacuum_freeze_max_age transactions old are always targeted by autovacuum. Otherwise, if the number of tuples made obsolete since the last VACUUM exceeds the "vacuum threshold", the table is vacuumed.

The autovacuum threshold is defined as:

```
Vacuum threshold = vacuum base threshold + vacuum scale factor * number of tuples
```

While you are connected to your database, run the following query to see a list of tables that autovacuum sees as eligible for vacuuming:

```
WITH vbt AS (SELECT setting AS autovacuum_vacuum_threshold FROM
pg_settings WHERE name = 'autovacuum_vacuum_threshold')
    , vsf AS (SELECT setting AS autovacuum_vacuum_scale_factor FROM
pg_settings WHERE name = 'autovacuum_vacuum_scale_factor')
    , fma AS (SELECT setting AS autovacuum_freeze_max_age FROM
pg_settings WHERE name = 'autovacuum_freeze_max_age')
    , sto AS (select opt_oid, split_part(setting, '=', 1) as param,
split_part(setting, '=', 2) as value from (select oid opt_oid,
unnest(reloptions) setting from pg_class) opt)
SELECT
    '"'||ns.nspname||'"."'||c.relname||'"' as relation
```

```
    , pg_size_pretty(pg_table_size(c.oid)) as table_size
    , age(relfrozenxid) as xid_age
    , coalesce(cfma.value::float, autovacuum_freeze_max_age::float)
autovacuum_freeze_max_age
    , (coalesce(cvbt.value::float, autovacuum_vacuum_threshold::float)
+ coalesce(cvsf.value::float,autovacuum_vacuum_scale_factor::float) *
pg_table_size(c.oid)) as autovacuum_vacuum_tuples
    , n_dead_tup as dead_tuples
FROM pg_class c join pg_namespace ns on ns.oid = c.relnamespace
join pg_stat_all_tables stat on stat.relid = c.oid
join vbt on (1=1) join vsf on (1=1) join fma on (1=1)
left join sto cvbt on cvbt.param = 'autovacuum_vacuum_threshold' and
c.oid = cvbt.opt_oid
left join sto cvsf on cvsf.param = 'autovacuum_vacuum_scale_factor' and
c.oid = cvsf.opt_oid
left join sto cfma on cfma.param = 'autovacuum_freeze_max_age' and
c.oid = cfma.opt_oid
WHERE c.relkind = 'r' and nspname <> 'pg_catalog'
and (
    age(relfrozenxid) >= coalesce(cfma.value::float,
autovacuum_freeze_max_age::float)
    or
    coalesce(cvbt.value::float, autovacuum_vacuum_threshold::float) +
coalesce(cvsf.value::float,autovacuum_vacuum_scale_factor::float) *
pg_table_size(c.oid) <= n_dead_tup
   -- or 1 = 1
)
ORDER BY age(relfrozenxid) DESC LIMIT 50;
```

Determining if Autovacuum is Currently Running and For How Long

If you need to manually vacuum a table, you need to determine if autovacuum is currently running. If it is, you may need to adjust parameters to make it run more efficiently, or terminate autovacuum so you can manually run VACUUM.

Use the following query to determine if autovacuum is running, and how long it has been running. This requires RDS Postgres 9.3.12+, 9.4.7+ and 9.5.2+ to have full visibility into rdsadmin processes currently running.

```
SELECT datname, usename, pid, waiting, current_timestamp - xact_start
AS xact_runtime, query
FROM pg_stat_activity WHERE upper(query) like '%VACUUM%' ORDER BY
xact_start;
```

After running the query, you will see output similar to the following:

```
datname | usename  |  pid  | waiting |      xact_runtime       |
query  --
 mydb    | rdsadmin | 16473 | f       | 33 days 16:32:11.600656 |
 autovacuum: VACUUM ANALYZE public.mytable1 (to prevent wraparound)
 mydb    | rdsadmin | 22553 | f       | 14 days 09:15:34.073141 |
 autovacuum: VACUUM ANALYZE public.mytable2 (to prevent wraparound)
 mydb    | rdsadmin | 41909 | f       | 3 days 02:43:54.203349  |
 autovacuum: VACUUM ANALYZE public.mytable3
 mydb    | rdsadmin |   618 | f       | 00:00:00                |
 SELECT datname, usename, pid, waiting, current_timestamp - xact_start
 AS xact_runtime, query+
         |          |       |         |                         |
         FROM pg_stat_activity
         +
```

```
         |          |       |         |                         |
         WHERE query like '%VACUUM%'
         +
         |          |       |         |                         |
         ORDER BY xact_start;
(4 rows)
```

Several issues may cause long running (multiple days) autovacuum session , but the most common issue is that yourmaintenance_work_mem parameter value is set too low for the size of the table or rate of updates.

We recommend that you use the following formula to set the maintenance_work_mem parameter value:

```
GREATEST({DBInstanceClassMemory/63963136*1024},65536)
```

Short running autovacuum sessions can also indicate problems:

- It can indicate that there aren't enough autovacuum_max_workers for your workload. You will need to indicate the number of workers.
- It can indicate that there is an index corruption (autovacuum will crash and restart on the same relation but make no progress). You will need to run a manual vacuum freeze verbose ___table___ to see the exact cause.

Performing a Manual Vacuum Freeze

You might want to perform a manual vacuum on a table that has a vacuum process already running. This is useful if you have identified a table with an "XID age" approaching 2 billion (or above any threshold you are monitoring).

The following steps are a guideline, and there are several variations to the process. For example, during testing, you find that the maintenance_work_mem parameter value was set too small and that you need to take immediate action on a table but don't want to bounce the instance at the moment. Using the queries listed above, you determine which table is the problem and notice a long running autovacuum session. You know you need to change the maintenance_work_mem parameter setting, but you also need to take immediate action and vacuum the table in question. The following procedure shows what you would do in this situation:

To manually perform a vacuum freeze

1. Open two sessions to the database containing the table you want to vacuum. For the second session, use "screen" or another utility that maintains the session if your connection is dropped.
2. In session one, get the PID of the autovacuum session running on the table. This action requires that you are running RDS Postgres 9.3.12 or later, 9.4.7 or later, or 9.5.2 or later to have full visibility into the running rdsadmin processes.

   Run the following query to get the PID of the autovacuum session:

   ```
   SELECT datname, usename, pid, waiting, current_timestamp - xact_start
   AS xact_runtime, query
   FROM pg_stat_activity WHERE upper(query) like '%VACUUM%' ORDER BY
   xact_start;
   ```

3. In session two, calculate the amount of memory you will need for this operation. In this example, we determine that we can afford to use up to 2GB of memory for this operation, so we set maintenance_work_mem for the current session to 2 GB.

```
set maintenance_work_mem='2 GB';
SET
```

4. In session two, issue a vacuum freeze verbose for the table. The verbose setting is useful because, while there is no progress report for this in Postgres currently, you can see activity.

```
\timing on
Timing is on.
vacuum freeze verbose pgbench_branches;
INFO:  vacuuming "public.pgbench_branches"
INFO:  index "pgbench_branches_pkey" now contains 50 row versions in 2 pages
DETAIL:  0 index row versions were removed.
0 index pages have been deleted, 0 are currently reusable.
CPU 0.00s/0.00u sec elapsed 0.00 sec.
INFO:  index "pgbench_branches_test_index" now contains 50 row versions in 2 pages
DETAIL:  0 index row versions were removed.
0 index pages have been deleted, 0 are currently reusable.
CPU 0.00s/0.00u sec elapsed 0.00 sec.
INFO:  "pgbench_branches": found 0 removable, 50 nonremovable row versions
     in 43 out of 43 pages
DETAIL:  0 dead row versions cannot be removed yet.
There were 9347 unused item pointers.
0 pages are entirely empty.
CPU 0.00s/0.00u sec elapsed 0.00 sec.
VACUUM
Time: 2.765 ms
```

5. In session one, if autovacuum was blocking, you will see in pg_stat_activity that waiting is "T" for your vacuum session. In this case, you need to terminate the autovacuum process.

```
select pg_terminate_backend('the_pid');
```

6. At this point, your session begins. It's important to note that autovacuum will restart immediately as this table is probably the highest on its list of work. You will need to initiate your command in session 2 and then terminate the autovacuum process in session one.

Reindexing a Table When Autovacuum is Running

If an index has become corrupt, autovacuum will continue to process the table and fail. If you attempt a manual vacuum in this situation, you will receive an error message similar to the following:

```
mydb=# vacuum freeze pgbench_branches;
ERROR: index "pgbench_branches_test_index" contains unexpected
   zero page at block 30521
HINT: Please REINDEX it.
```

When the index is corrupted and autovacuum is attempting to run against the table, you will contend with an already running autovacuum session. When you issue a "REINDEX " command, you will be taking out an exclusive lock on the table and write operations will be blocked as well as reads that use that specific index.

To reindex a table when autovacuum is running against the table

1. Open two sessions to the database containing the table you want to vacuum. For the second session, use "screen" or another utility that maintains the session if your connection is dropped.
2. In session one, get the PID of the autovacuum session running on the table. This action requires that you are running RDS Postgres 9.3.12 or later, 9.4.7 or later, or 9.5.2 or later to have full visibility into the running rdsadmin processes.

Run the following query to get the PID of the autovacuum session:

```
SELECT datname, usename, pid, waiting, current_timestamp - xact_start
AS xact_runtime, query
FROM pg_stat_activity WHERE upper(query) like '%VACUUM%' ORDER BY
xact_start;
```

3. In session two, issue the reindex command

```
\timing on
Timing is on.
reindex index pgbench_branches_test_index;
REINDEX
Time: 9.966 ms
```

4. In session one, if autovacuum was blocking, you will see in *pg_stat_activity* that waiting is "T" for your vacuum session. In this case, you will need to terminate the autovacuum process.

```
select pg_terminate_backend('the_pid');
```

5. At this point, your session begins. It's important to note that autovacuum will restart immediately as this table is probably the highest on its list of work. You will need to initiate your command in session 2 and then terminate the autovacuum process in session one.

Other Parameters That Affect Autovacuum

This query will show the values of some of the parameters that directly impact autovacuum and its behavior. The autovacuum parameters are described fully in the Postgres documentation.

```
select name, setting, unit, short_desc
from pg_settings
where name in (
'autovacuum_max_workers',
'autovacuum_analyze_scale_factor',
'autovacuum_naptime',
'autovacuum_analyze_threshold',
'autovacuum_analyze_scale_factor',
'autovacuum_vacuum_threshold',
'autovacuum_vacuum_scale_factor',
'autovacuum_vacuum_threshold',
'autovacuum_vacuum_cost_delay',
'autovacuum_vacuum_cost_limit',
'vacuum_cost_limit',
'autovacuum_freeze_max_age',
'maintenance_work_mem',
'vacuum_freeze_min_age');
```

While these all affect autovacuum, some of the most important ones are:

- Maintenance_Work_mem
- Autovacuum_freeze_max_age
- Autovacuum_max_workers
- Autovacuum_vacuum_cost_delay
- Autovacuum_vacuum_cost_limit

Table-Level Parameters

Autovacuum related storage parameters can be set at a table level which may be preferred to altering the behavior of the entire database. For large tables, it may be required to set aggressive settings and you may not want to make autovacuum behave that way for all tables.

This query will show which tables currently have table level options in place:

```
select relname, reloptions
from pg_class
where reloptions is not null;
```

An example where this might be useful is on tables that are much larger than the rest of your tables. If you have one 300GB table and 30 other tables less than 1GB, it would be reasonable to set some specific parameters for your large table so you don't alter the entire behavior of your system.

```
alter table mytable set (autovacuum_vacuum_cost_delay=0);
```

This will disable the cost-based autovacuum delay for this table at the expense of more resource usage on your system. Normally, autovacuum will pause for autovacuum_vacuum_cost_delay each time autovacuum_cost_limit is reached. More details can be read in the Postgres documentation regarding cost-based vacuuming.

Autovacuum Logging

By default, the *postgresql.log* doesn't contain information about the autovacuum process. If you are using PostgreSQL 9.4.5 or later, you can see output in the PostgreSQL error log from the autovacuum worker operations by setting the `rds.force_autovacuum_logging_level` parameter. Allowed values are `disabled, debug5, debug4, debug3, debug2, debug1, info, notice, warning, error, log, fatal,` and `panic`. The default value is `disabled` because the other allowable values can add significant amount of information to your logs.

We recommend that you set the value of the `rds.force_autovacuum_logging_level` parameter to `log` and that you set the `log_autovacuum_min_duration` parameter to a value from 1000 or 5000. If you set this value to 5000, Amazon RDS writes activity to the log that takes more than five seconds and shows "vacuum skipped" messages when application locking is causing autovacuum to intentionally skip tables. If you are troubleshooting a problem and need more detail, you can use a different logging level value, such as `debug1` or `debug3`. Use these debug parameters for a short period of time because these settings produce extremely verbose content written to the error log file. For more information about these debug settings, see the PostgreSQL documentation.

NOTE: PostgreSQL version 9.4.7 and later includes improved visibility of autovacuum sessions by allowing the `rds_superuser` account to view autovacuum sessions in `pg_stat_activity`. For example, you can identify and terminate an autovacuum session that is blocking a command from running, or executing slower than a manually issued vacuum command.

Audit Logging for a PostgreSQL DB Instance

There are several parameters you can set to log activity that occurs on your PostgreSQL DB instance. These ways include:

- The `log_statement` parameter can be used to log user activity in your PostgreSQL database. For more information, see PostgreSQL Database Log Files (p. 322).
- The `rds.force_admin_logging_level` parameter logs actions by the RDS internal user (rdsadmin) in the databases on the DB instance, and writes the output to the Postgres error log. Allowed values

are disabled, debug5, debug4, debug3, debug2, debug1, info, notice, warning, error, log, fatal, and panic. The default value is disabled.

- The `rds.force_autovacuum_logging_level` parameter logs autovacuum worker operations in all databases on the DB instance, and writes the output to the Postgres error log. Allowed values are disabled, debug5, debug4, debug3, debug2, debug1, info, notice, warning, error, log, fatal, and panic. The default value is disabled. The Amazon RDS recommended setting for rds.force_autovacuum_logging_level: is LOG. Set log_autovacuum_min_duration to a value from 1000 or 5000. Setting this value to 5000 will write activity to the log that takes more than 5 seconds and will show "vacuum skipped" messages. For more information on this parameter, see Best Practices for Working with PostgreSQL (p. 87).

Working with the `pgaudit` Extension

The `pgaudit` extension provides detailed session and object audit logging for Amazon RDS for PostgreSQL version 9.6.3 and later and version 9.5.7 version and later. You can enable session auditing or object auditing using this extension.

With session auditing, you can log audit events from various sources and includes the fully-qualified command text when available. For example, you can use session auditing to log all READ statements that connect to a database by setting pgaudit.log to 'READ'.

With object auditing, you can refine the audit logging to work with specific commands. For example, you can specify that you want audit logging for READ operations on a specific number of tables.

To use object based logging with the `pgaudit` extension

1. Create a specific database role called `rds_pgaudit`. Use the following command to create the role:

```
CREATE ROLE rds_pgaudit;
CREATE ROLE
```

2. Modify the parameter group that is associated with your DB instance to use the shared preload libraries that contain `pgaudit` and set the parameter `pgaudit.role`. The `pgaudit.role` must be set to the role `rds_pgaudit`.

   The following command modifies a custom parameter group:

```
aws rds modify-db-parameter-group
   --db-parameter-group-name rds-parameter-group-96
   --parameters
 "ParameterName=pgaudit.role,ParameterValue=rds_pgaudit,ApplyMethod=pending-reboot"
   --parameters
 "ParameterName=shared_preload_libraries,ParameterValue=pgaudit,ApplyMethod=pending-
reboot"
   --region us-west-2
```

3. Reboot the instance so that the DB instance will pick up the changes to the parameter group. The following command reboots a DB instance:

```
aws rds reboot-db-instance --db-instance-identifier rds-test-instance --region us-
west-2
```

4. Run the following command to confirm that `pgaudit` has been initialized.

```
show shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pgaudit
(1 row)
```

5. Run the following command to create the `pgaudit` extension:

```
CREATE EXTENSION pgaudit;
CREATE EXTENSION
```

6. Run the following command to confirm `pgaudit.role` is set to *rds_pgaudit*:

```
show pgaudit.role;
pgaudit.role
------------------
rds_pgaudit
```

To test the audit logging, run several commands that you have chosen to audit. For example, you might run the following commands:

```
CREATE TABLE t1 (id int);
CREATE TABLE
GRANT SELECT ON t1 TO rds_pgaudit;
GRANT
select * from t1;
id
----
(0 rows)
```

The database logs will contain an entry similar to the following:

```
...
2017-06-12 19:09:49 UTC:…:rds_test@postgres:[11701]:LOG: AUDIT:
OBJECT,1,1,READ,SELECT,TABLE,public.t1,select * from t1;
...
```

For information on viewing the logs, see Amazon RDS Database Log Files (p. 303).

Working with the pg_repack Extension

You can use the `pg_repack` extension to remove bloat from tables and indexes. This extension is supported on Amazon RDS for PostgreSQL versions 9.6.3 and later. For more information on the `pg_repack` extension, see the Github project documentation.

To use the `pg_repack` extension

1. Install the `pg_repack` extension on your Amazon RDS for PostgreSQL DB instance by running the following command.

```
CREATE EXTENSION pg_repack;
```

2. Use the pg_repack client utility to connect to a database. Use a database role that has *rds_superuser* priviledges to connect to the database. In the following connection example, the *rds_test* role has *rds_superuser* privileges, and the database endpoint used is *rds-test-instance.cw7jjfgdr4on8.us-west-2.rds.amazonaws.com*.

```
pg_repack -h rds-test-instance.cw7jjfgdr4on8.us-west-2.rds.amazonaws.com -U rds_test -k
 postgres
```

   Connect using the -k option. The -a option is not supported.
3. The response from the pg_repack client provides information on the tables on the DB instance that are repacked.

```
INFO: repacking table "pgbench_tellers"
INFO: repacking table "pgbench_accounts"
INFO: repacking table "pgbench_branches"
```

Working with PostGIS

PostGIS is an extension to PostgreSQL for storing and managing spatial information. If you are not familiar with PostGIS you can get a good general overview at PostGIS Introduction.

You need to perform a bit of setup before you can use the PostGIS extension. The following list shows what you need to do. Each step is described in greater detail later in this section.

- Connect to the DB instance using the master username used to create the DB instance
- Load the PostGIS extensions
- Transfer ownership of the extensions to the`rds_superuser` role
- Transfer ownership of the objects to the `rds_superuser` role
- Test the extensions

Step 1: Connect to the DB Instance Using the Master Username Used to Create the DB Instance

First, you connect to the DB instance using the master user name that was used to create the DB instance. That name is automatically assigned the `rds_superuser` role. You need the `rds_superuser` role that is needed to do the remaining steps.

The following example uses SELECT to show you the current user; in this case, the current user should be the master username you chose when creating the DB instance:

```
select current_user;
 current_user
 -------------
  myawsuser
 (1 row)
```

Step 2: Load the PostGIS Extensions

Use the CREATE EXTENSION statements to load the PostGIS extensions. Note that you must also load the `fuzzystrmatch` extension. You can then use the `\dn` *psql* command to list the owners of the PostGIS schemas.

```
create extension postgis;
CREATE EXTENSION
create extension fuzzystrmatch;
CREATE EXTENSION
create extension postgis_tiger_geocoder;
CREATE EXTENSION
create extension postgis_topology;
CREATE EXTENSION
\dn
     List of schemas
     Name     |   Owner
--------------+-----------
 public       | myawsuser
 tiger        | rdsadmin
 tiger_data   | rdsadmin
 topology     | rdsadmin
(4 rows)
```

Step 3: Transfer Ownership of the Extensions to the rds_superuser Role

Use the ALTER SCHEMA statements to transfer ownership of the schemas to the `rds_superuser` role.

```
alter schema tiger owner to rds_superuser;
ALTER SCHEMA
alter schema tiger_data owner to rds_superuser;
ALTER SCHEMA
alter schema topology owner to rds_superuser;
ALTER SCHEMA
\dn
       List of schemas
     Name     |     Owner
--------------+---------------
 public       | myawsuser
 tiger        | rds_superuser
tiger_data    | rds_superuser
 topology     | rds_superuser
(4 rows)
```

Step 4: Transfer Ownership of the Objects to the rds_superuser Role

Use the following function to transfer ownership of the PostGIS objects to the `rds_superuser` role. Run the following statement from the psql prompt to create the function:

```
CREATE FUNCTION exec(text) returns text language plpgsql volatile AS $f$ BEGIN EXECUTE $1;
 RETURN $1; END; $f$;
```

Next, run this query to run the exec function that in turn executes the statements and alters the permissions:

```
SELECT exec('ALTER TABLE ' || quote_ident(s.nspname) || '.' || quote_ident(s.relname) || '
 OWNER TO rds_superuser;')
  FROM (
    SELECT nspname, relname
    FROM pg_class c JOIN pg_namespace n ON (c.relnamespace = n.oid)
    WHERE nspname in ('tiger','topology') AND
    relkind IN ('r','S','v') ORDER BY relkind = 'S')
s;
```

Step 5: Test the Extensions

Add `tiger` to your search path using the following command:

```
SET search_path=public,tiger;
```

Test `tiger` by using the following SELECT statement:

```
select na.address, na.streetname, na.streettypeabbrev, na.zip
from normalize_address('1 Devonshire Place, Boston, MA 02109') as na;
 address | streetname | streettypeabbrev |  zip
---------+------------+------------------+-------
       1 | Devonshire | Pl               | 02109
(1 row)
```

Test `topology` by using the following SELECT statement:

```
select topology.createtopology('my_new_topo',26986,0.5);
 createtopology
----------------
              1
(1 row)
```

Using pgBadger for Log Analysis with PostgreSQL

You can use a log analyzer such as pgbadger to analyze PostgreSQL logs. Although the *pgbadger* documentation states that the %l pattern (log line for session/process) should be a part of the prefix, if you provide the current rds log_line_prefix as a parameter to *pgbadger* it should still produce a report.

For example, the following command would correctly format an Amazon RDS PostgreSQL log file dated 2014-02-04 using *pgbadger*:

```
./pgbadger -p '%t:%r:%u@%d:[%p]:' postgresql.log.2014-02-04-00
```

Viewing the Contents of pg_config

In PostgreSQL version 9.6.1, you can see the compile-time configuration parameters of the currently installed version of PostgreSQL using the new view pg_config. You can use the view by calling the pg_config function as shown in the following sample:

```
select * from pg_config();
       name        |             setting

-------------------
+-------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------
 BINDIR             | /rdsdbbin/postgres-9.6.1.R1/bin
 DOCDIR             | /rdsdbbin/postgres-9.6.1.R1/share/doc
 HTMLDIR            | /rdsdbbin/postgres-9.6.1.R1/share/doc
 INCLUDEDIR         | /rdsdbbin/postgres-9.6.1.R1/include
 PKGINCLUDEDIR      | /rdsdbbin/postgres-9.6.1.R1/include
 INCLUDEDIR-SERVER | /rdsdbbin/postgres-9.6.1.R1/include/server
 LIBDIR             | /rdsdbbin/postgres-9.6.1.R1/lib
 PKGLIBDIR          | /rdsdbbin/postgres-9.6.1.R1/lib
 LOCALEDIR          | /rdsdbbin/postgres-9.6.1.R1/share/locale
 MANDIR             | /rdsdbbin/postgres-9.6.1.R1/share/man
 SHAREDIR           | /rdsdbbin/postgres-9.6.1.R1/share
 SYSCONFDIR         | /rdsdbbin/postgres-9.6.1.R1/etc
 PGXS               | /rdsdbbin/postgres-9.6.1.R1/lib/pgxs/src/makefiles/pgxs.mk
 CONFIGURE          | '--prefix=/rdsdbbin/postgres-9.6.1.R1' '--with-openssl' '--with-perl'
 '--with-tcl' '--with-ossp-uuid' '--with-libxml' '--with-libraries=/rdsdbbin
/postgres-9.6.1.R1/lib' '--with-includes=/rdsdbbin/postgres-9.6.1.R1/include' '--enable-
debug'
 CC                 | gcc
 CPPFLAGS           | -D_GNU_SOURCE -I/usr/include/libxml2 -I/rdsdbbin/postgres-9.6.1.R1/
include
 CFLAGS             | -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-
statement
 -Wendif-labels -Wmissing-format-attribute -Wformat-security -fno-strict-
aliasing -fwrapv -fexcess-precision=standard -g -O2
 CFLAGS_SL          | -fpic
 LDFLAGS            | -L../../src/common -L/rdsdbbin/postgres-9.6.1.R1/lib -Wl,--as-needed -
Wl,
 -rpath,'/rdsdbbin/postgres-9.6.1.R1/lib',--enable-new-dtags
 LDFLAGS_EX         |
 LDFLAGS_SL         |
 LIBS               | -lpgcommon -lpgport -lxml2 -lssl -lcrypto -lz -lreadline -lrt -lcrypt
 -ldl -lm
 VERSION            | PostgreSQL 9.6.1
(23 rows)
```

If you attempt to access the view directly, the request fails.

```
select * from pg_config;
ERROR:  permission denied for relation pg_config
```

Limits for Amazon RDS

This topic describes the resource limits and naming constraints for Amazon RDS.

Topics

Limits in Amazon RDS

Each AWS account has limits, per region, on the number of Amazon RDS resources that can be created. Once a limit for a resource has been reached, additional calls to create that resource will fail with an exception.

The following table lists the resources and their limits per region.

Resource	Default Limit
Clusters	40
Cluster parameter groups	50
Cross-region snapshots copy requests	5
DB Instances	40
Event subscriptions	20
Manual snapshots	100
Manual cluster snapshots	100
Option groups	20
Parameter groups	50
Read replicas per master	5
Reserved instances (purchased per month)	40
Rules per security group	20
Security groups	25
Security groups (VPC)	5
Subnet groups	50
Subnets per subnet group	20
Tags per resource	50
Total storage for all DB instances	100 TB

Naming Constraints in Amazon RDS

The following table describes naming constraints in Amazon RDS.

DB instance identifier	• Must contain 1 to 63 alphanumeric characters or hyphens. • First character must be a letter. • Cannot end with a hyphen or contain two consecutive hyphens. • Must be unique for all DB instances per AWS account, per region.
Database name	Database name constraints differ for each database engine. **MySQL, Amazon Aurora, and MariaDB** • Must contain 1 to 64 alphanumeric characters. • Cannot be a word reserved by the database engine. **PostgreSQL** • Must contain 1 to 63 alphanumeric characters. • Must begin with a letter or an underscore. Subsequent characters can be letters, underscores, or digits (0-9). • Cannot be a word reserved by the database engine. **Oracle** • Cannot be longer than 8 characters. **SQL Server** • Not applicable. For SQL Server, you create your databases after you create your DB instance. Database names follow the usual SQL Server naming rules.
Master user name	Master user name constraints differ for each database engine. **MySQL and Amazon Aurora** • Must contain 1 to 16 alphanumeric characters. • First character must be a letter. • Cannot be a word reserved by the database engine. **Oracle** • Must contain 1 to 30 alphanumeric characters. • First character must be a letter. • Cannot be a word reserved by the database engine. **SQL Server**

	• Must contain 1 to 64 alphanumeric characters. • First character must be a letter. • Cannot be a word reserved by the database engine. **PostgreSQL** • Must contain 1 to 63 alphanumeric characters. • First character must be a letter. • Cannot be a word reserved by the database engine. **MariaDB** • Must contain 1 to 16 alphanumeric characters. • Cannot be a word reserved by the database engine.
Master password	The password for the master database user can be any printable ASCII character except "/", """, or "@". Master password constraints differ for each database engine. **MySQL, Amazon Aurora, and MariaDB** • Must contain 8 to 41 characters. **Oracle** • Must contain 8 to 30 characters. **SQL Server** • Must contain 8 to 128 characters. **PostgreSQL** • Must contain 8 to 128 characters .
DB parameter group name	• Must contain from 1 to 255 alphanumeric characters. • First character must be a letter. • Cannot end with a hyphen or contain two consecutive hyphens.

File Size Limits in Amazon RDS

Aurora File Size Limits in Amazon RDS

With Amazon Aurora, the table size limit is only constrained by the size of the Aurora cluster volume, which has a maximum of 64 terabytes (TB). As a result, the maximum table size for a table in an Aurora database is 64 TB.

MySQL File Size Limits in Amazon RDS

For Amazon RDS MySQL DB instances, the maximum provisioned storage limit constrains the size of a table to a maximum size of 16 TB when using InnoDB file-per-table tablespaces. This limit also constrains the system tablespace to a maximum size of 16 TB. InnoDB file-per-table tablespaces (with tables each in their own tablespace) is set by default for Amazon RDS MySQL DB instances. For more information, see Storage for Amazon RDS (p. 410).

> **Note**
> Some existing DB instances have a lower limit. For example, MySQL DB instances created prior to April 2014 have a file and table size limit of 2 TB. This 2 TB file size limit also applies to DB instances or Read Replicas created from DB snapshots taken prior to April 2014, regardless of when the DB instance was created.

There are advantages and disadvantages to using InnoDB file-per-table tablespaces, depending on your application. To determine the best approach for your application, go to InnoDB File-Per-Table Mode in the MySQL documentation.

We don't recommend allowing tables to grow to the maximum file size. In general, a better practice is to partition data into smaller tables, which can improve performance and recovery times.

One option that you can use for breaking a large table up into smaller tables is partitioning. Partitioning distributes portions of your large table into separate files based on rules that you specify. For example, if you store transactions by date, you can create partitioning rules that distribute older transactions into separate files using partitioning. Then periodically, you can archive the historical transaction data that doesn't need to be readily available to your application. For more information, see Partitioning in the MySQL documentation.

To determine the file size of a table

Use the following SQL command to determine if any of your tables are too large and are candidates for partitioning. To update table statistics, issue an `ANALYZE TABLE` command on each table. For more information, see ANALYZE TABLE in the MySQL documentation.

```
SELECT TABLE_SCHEMA, TABLE_NAME,
    round(((DATA_LENGTH + INDEX_LENGTH) / 1024 / 1024), 2) As "Approximate size (MB)",
 DATA_FREE
    FROM information_schema.TABLES
    WHERE TABLE_SCHEMA NOT IN ('mysql', 'information_schema', 'performance_schema');
```

To enable InnoDB file-per-table tablespaces

- To enable InnoDB file-per-table tablespaces, set the *innodb_file_per_table* parameter to `1` in the parameter group for the DB instance.

To disable InnoDB file-per-table tablespaces

- To disable InnoDB file-per-table tablespaces, set the *innodb_file_per_table* parameter to `0` in the parameter group for the DB instance.

For information on updating a parameter group, see Working with DB Parameter Groups (p. 170).

When you have enabled or disabled InnoDB file-per-table tablespaces, you can issue an `ALTER TABLE` command to move a table from the global tablespace to its own tablespace, or from its own tablespace to the global tablespace as shown in the following example:

```
ALTER TABLE table_name ENGINE=InnoDB, ALGORITHM=COPY;
```

MariaDB File Size Limits in Amazon RDS

For Amazon RDS MariaDB DB instances, the maximum provisioned storage limit constrains the size of a table to a maximum size of 16 TB when using InnoDB file-per-table tablespaces. This limit also constrains the system tablespace to a maximum size of 16 TB. InnoDB file-per-table tablespaces (with tables each in their own tablespace) is set by default for Amazon RDS MariaDB DB instances. For more information, see Storage for Amazon RDS (p. 410).

There are advantages and disadvantages to using InnoDB file-per-table tablespaces, depending on your application. To determine the best approach for your application, go to InnoDB File-Per-Table Mode in the MySQL documentation.

We don't recommend allowing tables to grow to the maximum file size. In general, a better practice is to partition data into smaller tables, which can improve performance and recovery times.

One option that you can use for breaking a large table up into smaller tables is partitioning. Partitioning distributes portions of your large table into separate files based on rules that you specify. For example, if you store transactions by date, you can create partitioning rules that distribute older transactions into separate files using partitioning. Then periodically, you can archive the historical transaction data that doesn't need to be readily available to your application. For more information, go to https://dev.mysql.com/doc/refman/5.6/en/partitioning.html in the MySQL documentation.

To determine the file size of a table

Use the following SQL command to determine if any of your tables are too large and are candidates for partitioning. To update table statistics, issue an `ANALYZE TABLE` command on each table. For more information, see ANALYZE TABLE in the MySQL documentation.

```
SELECT TABLE_SCHEMA, TABLE_NAME,
    round(((DATA_LENGTH + INDEX_LENGTH) / 1024 / 1024), 2) As "Approximate size (MB)",
 DATA_FREE
    FROM information_schema.TABLES
    WHERE TABLE_SCHEMA NOT IN ('mysql', 'information_schema', 'performance_schema');
```

To enable InnoDB file-per-table tablespaces

- To enable InnoDB file-per-table tablespaces, set the *innodb_file_per_table* parameter to `1` in the parameter group for the DB instance.

To disable InnoDB file-per-table tablespaces

- To disable InnoDB file-per-table tablespaces, set the *innodb_file_per_table* parameter to `0` in the parameter group for the DB instance.

For information on updating a parameter group, see Working with DB Parameter Groups (p. 170).

When you have enabled or disabled InnoDB file-per-table tablespaces, you can issue an `ALTER TABLE` command to move a table from the global tablespace to its own tablespace, or from its own tablespace to the global tablespace as shown in the following example:

```
ALTER TABLE table_name ENGINE=InnoDB, ALGORITHM=COPY;
```

Troubleshooting

Use the following sections to help troubleshoot problems you have with Amazon RDS.

Topics

Cannot Connect to Amazon RDS DB Instance

When you cannot connect to a DB instance, the following are common causes:

- The access rules enforced by your local firewall and the ingress IP addresses that you authorized to access your DB instance in the instance's security group are not in sync. The problem is most likely the ingress rules in your security group. By default, DB instances do not allow access; access is granted through a security group. To grant access, you must create your own security group with specific ingress and egress rules for your situation. For more information about setting up a security group, see Provide Access to the DB Instance in the VPC by Creating a Security Group (p. 8).
- The port you specified when you created the DB instance cannot be used to send or receive communications due to your local firewall restrictions. In this case, check with your network administrator to determine if your network allows the specified port to be used for inbound and outbound communication.
- Your DB instance is still being created and is not yet available. Depending on the size of your DB instance, it can take up to 20 minutes before an instance is available.

Testing a Connection to an Amazon RDS DB Instance

You can test your connection to a DB instance using common Linux or Windows tools.

From a Linux or Unix terminal, you can test the connection by typing the following (replace *<DB-instance-endpoint>* with the endpoint and *<port>* with the port of your DB instance):

```
$nc -zv <DB-instance-endpoint> <port>
```

For example, the following shows a sample command and the return value:

```
  $nc -zv postgresql1.c6c8mn7tsdgv0.us-west-2.rds.amazonaws.com 8299

  Connection to postgresql1.c6c8mn7tsdgv0.us-west-2.rds.amazonaws.com 8299 port [tcp/vvr-
data] succeeded!
```

Windows users can use Telnet to test the connection to a DB instance. Note that Telnet actions are not supported other than for testing the connection. If a connection is successful, the action returns no message. If a connection is not successful, you receive an error message such as the following:

```
  C:\>telnet sg-postgresql1.c6c8mntzhgv0.us-west-2.rds.amazonaws.com 819

  Connecting To sg-postgresql1.c6c8mntzhgv0.us-west-2.rds.amazonaws.com...Could not open
  connection to the host, on port 819: Connect failed
```

If Telnet actions return success, your security group is properly configured.

Troubleshooting Connection Authentication

If you can connect to your DB instance but you get authentication errors, you might want to reset the master user password for the DB instance. You can do this by modifying the RDS instance; for more information, see one of the following topics:

- Modifying a DB Instance Running the MySQL Database Engine (p. 843)
- Modifying a DB Instance Running the Oracle Database Engine (p. 967)
- Modifying a DB Instance Running the Microsoft SQL Server Database Engine (p. 756)
- Modifying a DB Instance Running the PostgreSQL Database Engine (p. 1183)

Amazon RDS Security Issues

To avoid security issues, never use your master AWS user name and password for a user account. Best practice is to use your master AWS account to create IAM users and assign those to DB user accounts. You can also use your master account to create other user accounts, if necessary. For more information on creating IAM users, see Create an IAM User (p. 5).

Error Message "Failed to retrieve account attributes, certain console functions may be impaired."

There are several reasons you would get this error; it could be because your account is missing permissions, or your account has not been properly set up. If your account is new, you may not have waited for the account to be ready. If this is an existing account, you could lack permissions in your access policies to perform certain actions such as creating a DB instance. To fix the issue, your IAM administrator needs to provide the necessary roles to your account. For more information, see the IAM documentation.

Resetting the DB Instance Owner Role Password

You can reset the assigned permissions for your DB instance by resetting the master password. For example, if you lock yourself out of the `db_owner` role on your SQL Server database, you can reset the `db_owner` role password by modifying the DB instance master password. By changing the DB instance password, you can regain access to the DB instance, access databases using the modified password for

the `db_owner`, and restore privileges for the `db_owner` role that may have been accidentally revoked. You can change the DB instance password by using the Amazon RDS console, the AWS CLI command modify-db-instance, or by using the ModifyDBInstance action. For more information about modifying a SQL Server DB instance, see Modifying a DB Instance Running the Microsoft SQL Server Database Engine (p. 756).

Amazon RDS DB Instance Outage or Reboot

A DB instance outage can occur when a DB instance is rebooted, when the DB instance is put into a state that prevents access to it, and when the database is restarted. A reboot can occur when you manually reboot your DB instance or when you change a DB instance setting that requires a reboot before it can take effect.

When you modify a setting for a DB instance, you can determine when the change is applied by using the **Apply Immediately** setting. To see a table that shows DB instance actions and the effect that setting the **Apply Immediately** value has, see Modifying an Amazon RDS DB Instance and Using the Apply Immediately Parameter (p. 114).

A DB instance reboot only occurs when you change a setting that requires a reboot, or when you manually cause a reboot. A reboot can occur immediately if you change a setting and request that the change take effect immediately or it can occur during the DB instance's maintenance window.

A DB instance reboot occurs immediately when one of the following occurs:

- You change the backup retention period for a DB instance from 0 to a nonzero value or from a nonzero value to 0 and set **Apply Immediately** to *true*.
- You change the DB instance class, and **Apply Immediately** is set to *true*.
- You change the storage type from **Magnetic (Standard)** to **General Purpose (SSD**) or **Provisioned IOPS (SSD)**, or from **Provisioned IOPS (SSD)** or **General Purpose (SSD)** to **Magnetic (Standard)**. from standard to PIOPS.

A DB instance reboot occurs during the maintenance window when one of the following occurs:

- You change the backup retention period for a DB instance from 0 to a nonzero value or from a nonzero value to 0, and **Apply Immediately** is set to *false*.
- You change the DB instance class, and **Apply Immediately** is set to *false*.

When you change a static parameter in a DB parameter group, the change will not take effect until the DB instance associated with the parameter group is rebooted. The change requires a manual reboot; the DB instance will not automatically be rebooted during the maintenance window.

Amazon RDS DB Parameter Changes Not Taking Effect

If you change a parameter in a DB parameter group but you don't see the changes take effect, you most likely need to reboot the DB instance associated with the DB parameter group. When you change a dynamic parameter, the change takes effect immediately; when you change a static parameter, the change won't take effect until you reboot the DB instance associated with the parameter group.

You can reboot a DB instance using the RDS console or explicitly calling the `RebootDbInstance` API action (without failover, if the DB instance is in a Multi-AZ deployment). The requirement to reboot

the associated DB instance after a static parameter change helps mitigate the risk of a parameter misconfiguration affecting an API call, such as calling `ModifyDBInstance` to change DB instance class or scale storage. For more information, see Modifying Parameters in a DB Parameter Group (p. 172).

Amazon RDS DB Instance Running Out of Storage

If your DB instance runs out of storage space, it might no longer be available. We highly recommend that you constantly monitor the `FreeStorageSpace` metric published in CloudWatch to ensure that your DB instance has enough free storage space.

If your database instance runs out of storage, its status will change to *storage-full*. For example, a call to the `DescribeDBInstances` action for a DB instance that has used up its storage will output the following:

```
aws rds describe-db-instances --db-instance-identifier mydbinstance

DBINSTANCE  mydbinstance  2009-12-22T23:06:11.915Z  db.m3.large  mysql5.6  50  sa
storage-full  mydbinstance.clla4j4jgyph.us-east-1.rds.amazonaws.com  3306
us-east-1b  3
 SECGROUP  default  active
 PARAMGRP  default.mysql5.6  in-sync
```

To recover from this scenario, add more storage space to your instance using the `ModifyDBInstance` action or the following AWS CLI command:

For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --allocated-storage 60 \
    --apply-immediately
```

For Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --allocated-storage 60 ^
    --apply-immediately
```

```
DBINSTANCE  mydbinstance  2009-12-22T23:06:11.915Z  db.m3.large  mysql5.6  50  sa
storage-full  mydbinstance.clla4j4jgyph.us-east-1.rds.amazonaws.com  3306
us-east-1b  3  60
 SECGROUP  default  active
 PARAMGRP  default.mysql5.6  in-sync
```

Now, when you describe your DB instance, you will see that your DB instance will have *modifying* status, which indicates the storage is being scaled.

```
aws rds describe-db-instances --db-instance-identifier mydbinstance
```

```
DBINSTANCE  mydbinstance  2009-12-22T23:06:11.915Z  db.m3.large  mysql5.6  50  sa
modifying  mydbinstance.clla4j4jgyph.us-east-1.rds.amazonaws.com
3306  us-east-1b  3  60
 SECGROUP  default  active
```

```
 PARAMGRP  default.mysql5.6  in-sync
```

Once storage scaling is complete, your DB instance status will change to *available*.

```
aws rds describe-db-instances --db-instance-identifier mydbinstance
```

```
DBINSTANCE  mydbinstance  2009-12-22T23:06:11.915Z  db.m3.large  mysql5.6  60  sa
available  mydbinstance.clla4j4jgyph.us-east-1.rds.amazonaws.com  3306
us-east-1b  3
 SECGROUP  default  active
 PARAMGRP  default.mysql5.6  in-sync
```

Note that you can receive notifications when your storage space is exhausted using the `DescribeEvents` action. For example, in this scenario, if you do a `DescribeEvents` call after these operations you will see the following output:

```
aws rds describe-events --source-type db-instance --source-identifier mydbinstance
```

```
2009-12-22T23:44:14.374Z  mydbinstance  Allocated storage has been exhausted db-instance
2009-12-23T00:14:02.737Z  mydbinstance  Applying modification to allocated storage db-
instance
2009-12-23T00:31:54.764Z  mydbinstance  Finished applying modification to allocated storage
```

Amazon RDS Insufficient DB Instance Capacity

If you get an `InsufficientDBInstanceCapacity` error when you try to modify a DB instance class, it might be because the DB instance is on the EC2-Classic platform and is therefore not in a VPC. Some DB instance classes require a VPC. For example, if you are on the EC2-Classic platform and try to increase capacity by switching to a DB instance class that requires a VPC, this error results. For information about Amazon Elastic Compute Cloud instance types that are only available in a VPC, see Instance Types Available Only in a VPC in the *Amazon Elastic Compute Cloud User Guide*.

To correct the problem, you can move the DB instance into a VPC. For more information, see Moving a DB Instance Not in a VPC into a VPC (p. 405).

For information about modifying a DB instance, see Modifying an Amazon RDS DB Instance and Using the Apply Immediately Parameter (p. 114). For information about troubleshooting instance capacity issues for Amazon EC2, see Troubleshooting Instance Capacity in the *Amazon Elastic Compute Cloud User Guide*.

Amazon RDS MySQL and MariaDB Issues

Index Merge Optimization Returns Wrong Results

This issue applies only to MySQL DB instances.

Queries that use index merge optimization might return wrong results due to a bug in the MySQL query optimizer that was introduced in MySQL 5.5.37. When you issue a query against a table with multiple indexes the optimizer scans ranges of rows based on the multiple indexes, but does not merge the results together correctly. For more information on the query optimizer bug, go to http://

bugs.mysql.com/bug.php?id=72745 and http://bugs.mysql.com/bug.php?id=68194 in the MySQL bug database.

For example, consider a query on a table with two indexes where the search arguments reference the indexed columns.

```
SELECT * FROM table1
  WHERE indexed_col1 = 'value1' AND indexed_col2 = 'value2';
```

In this case, the search engine searches both indexes. However, due to the bug, the merged results are incorrect.

To resolve this issue, you can do one of the following:

- Set the `optimizer_switch` parameter to `index_merge=off` in the DB parameter group for your MySQL DB instance. For information on setting DB parameter group parameters, see Working with DB Parameter Groups (p. 170).
- Upgrade your MySQL DB instance to MySQL version 5.6 or 5.7. For more information, see Upgrading a MySQL DB Snapshot (p. 857).
- If you cannot upgrade your instance or change the `optimizer_switch` parameter, you can work around the bug by explicitly identifying an index for the query, for example:

```
SELECT * FROM table1
  USE INDEX covering_index
  WHERE indexed_col1 = 'value1' AND indexed_col2 = 'value2';
```

For more information, go to Index Merge Optimization.

Diagnosing and Resolving Lag Between Read Replicas

After you create a MySQL or MariaDB Read Replica and the Read Replica is available, Amazon RDS first replicates the changes made to the source DB instance from the time the create Read Replica operation was initiated. During this phase, the replication lag time for the Read Replica will be greater than 0. You can monitor this lag time in Amazon CloudWatch by viewing the Amazon RDS `ReplicaLag` metric.

The `ReplicaLag` metric reports the value of the `Seconds_Behind_Master` field of the MySQL or MariaDB `SHOW SLAVE STATUS` command. For more information, see SHOW SLAVE STATUS. When the `ReplicaLag` metric reaches 0, the replica has caught up to the source DB instance. If the `ReplicaLag` metric returns -1, replication might not be active. To troubleshoot a replication error, see Diagnosing and Resolving a MySQL or MariaDB Read Replication Failure (p. 1234). A `ReplicaLag` value of -1 can also mean that the `Seconds_Behind_Master` value cannot be determined or is `NULL`.

The `ReplicaLag` metric returns -1 during a network outage or when a patch is applied during the maintenance window. In this case, wait for network connectivity to be restored or for the maintenance window to end before you check the `ReplicaLag` metric again.

Because the MySQL and MariaDB read replication technology is asynchronous, you can expect occasional increases for the `BinLogDiskUsage` metric on the source DB instance and for the `ReplicaLag` metric on the Read Replica. For example, a high volume of write operations to the source DB instance can occur in parallel, while write operations to the Read Replica are serialized using a single I/O thread, can lead to a lag between the source instance and Read Replica. For more information about Read Replicas and MySQL, go to Replication Implementation Details in the MySQL documentation. For more information about Read Replicas and MariaDB, go to Replication Overview in the MariaDB documentation.

You can reduce the lag between updates to a source DB instance and the subsequent updates to the Read Replica by doing the following:

- Set the DB instance class of the Read Replica to have a storage size comparable to that of the source DB instance.
- Ensure that parameter settings in the DB parameter groups used by the source DB instance and the Read Replica are compatible. For more information and an example, see the discussion of the `max_allowed_packet` parameter in the next section.
- Disable the query cache. For tables that are modified often, using the query cache can increase replica lag because the cache is locked and refreshed often. If this is the case, you might see less replica lag if you disable the query cache. You can disable the query cache by setting the `query_cache_type parameter` to 0 in the DB parameter group for the DB instance. For more information on the query cache, see Query Cache Configuration.
- Warm the InnoDB for MySQL or XtraDB for MariaDB buffer pool on the Read Replica. If you have a small set of tables that are being updated often, and you are using the InnoDB or XtraDB table schema, then dump those tables on the Read Replica. Doing this causes the database engine to scan through the rows of those tables from the disk and then cache them in the buffer pool, which can reduce replica lag. The following shows an example.

  For Linux, OS X, or Unix:

```
PROMPT> mysqldump \
    -h <endpoint> \
    --port=<port> \
    -u=<username> \
    -p <password> \
    database_name table1 table2 > /dev/null
```

  For Windows:

```
PROMPT> mysqldump ^
    -h <endpoint> ^
    --port=<port> ^
    -u=<username> ^
    -p <password> ^
    database_name table1 table2 > /dev/null
```

Diagnosing and Resolving a MySQL or MariaDB Read Replication Failure

Amazon RDS monitors the replication status of your Read Replicas and updates the **Replication State** field of the Read Replica instance to **Error** if replication stops for any reason. You can review the details of the associated error thrown by the MySQL or MariaDB engines by viewing the **Replication Error** field. Events that indicate the status of the Read Replica are also generated, including RDS-EVENT-0045 (p. 283), RDS-EVENT-0046 (p. 283), and RDS-EVENT-0047 (p. 283). For more information about events and subscribing to events, see Using Amazon RDS Event Notification (p. 279). If a MySQL error message is returned, review the error in the MySQL error message documentation. If a MariaDB error message is returned, review the error in the MariaDB error message documentation.

Common situations that can cause replication errors include the following:

- The value for the `max_allowed_packet` parameter for a Read Replica is less than the `max_allowed_packet` parameter for the source DB instance.

  The `max_allowed_packet` parameter is a custom parameter that you can set in a DB parameter group that is used to specify the maximum size of data manipulation language (DML) that can be executed on the database. If the `max_allowed_packet` parameter value for the source DB instance is smaller than the `max_allowed_packet` parameter value for the Read Replica, the replication

process can throw an error and stop replication. The most common error is `packet bigger than 'max_allowed_packet' bytes`. You can fix the error by having the source and Read Replica use DB parameter groups with the same `max_allowed_packet` parameter values.

- Writing to tables on a Read Replica. If you are creating indexes on a Read Replica, you need to have the `read_only` parameter set to *0* to create the indexes. If you are writing to tables on the Read Replica, it can break replication.
- Using a non-transactional storage engine such as MyISAM. Read replicas require a transactional storage engine. Replication is only supported for the InnoDB for MySQL and XtraDB for MariaDB storage engines.

  You can convert a MyISAM table to InnoDB with the following command:

  ```
  alter table <schema>.<table_name> engine=innodb;
  ```
- Using unsafe non-deterministic queries such as `SYSDATE()`. For more information, see Determination of Safe and Unsafe Statements in Binary Logging.

The following steps can help resolve your replication error:

- If you encounter a logical error and you can safely skip the error, follow the steps described in Skipping the Current Replication Error (p. 905). Your MySQL or MariaDB DB instance must be running a version that includes the `mysql_rds_skip_repl_error` procedure. For more information, see mysql.rds_skip_repl_error (p. 918).
- If you encounter a binlog position issue, you can change the slave replay position with the `mysql_rds_next_master_log` command. Your MySQL or MariaDB DB instance must be running a version that supports the `mysql_rds_next_master_log` command in order to change the slave replay position. For version information, see mysql.rds_next_master_log (p. 919).
- If you encounter a temporary performance issue due to high DML load, you can set the `innodb_flush_log_at_trx_commit` parameter to 2 in the DB parameter group on the Read Replica. Doing this can help the Read Replica catch up, though it temporarily reduces atomicity, consistency, isolation, and durability (ACID).
- You can delete the Read Replica and create an instance using the same DB instance identifier so that the endpoint remains the same as that of your old Read Replica.

If a replication error is fixed, the **Replication State** changes to **replicating**. For more information, see Troubleshooting a MySQL or MariaDB Read Replica Problem (p. 150).

Creating Triggers with Binary Logging Enabled Requires SUPER Privilege

When trying to create triggers in an RDS MySQL or MariaDB DB instance, you might receive the following error:

```
"You do not have the SUPER privilege and binary logging is enabled"
```

To use triggers when binary logging is enabled requires the SUPER privilege, which is restricted for RDS MySQL and MariaDB DB instances. You can create triggers when binary logging is enabled without the SUPER privilege by setting the `log_bin_trust_function_creators` parameter to true. To set the `log_bin_trust_function_creators` to true, create a new DB parameter group or modify an existing DB parameter group.

To create a new DB parameter group that allows you to create triggers in your RDS MySQL or MariaDB DB instance with binary logging enabled, use the following CLI commands. To modify an existing parameter group, start with step 2.

To create a new parameter group to allow triggers with binary logging enabled using the CLI

1. Create a new parameter group.

   For Linux, OS X, or Unix:

```
aws rds create-db-parameter-group \
    --db-parameter-group-name allow-triggers \
    --db-parameter-group-family mysql15.5 \
    --description "parameter group allowing triggers"
```

   For Windows:

```
aws rds create-db-parameter-group ^
    --db-parameter-group-name allow-triggers ^
    --db-parameter-group-family mysql15.5 ^
    --description "parameter group allowing triggers"
```

2. Modify the DB parameter group to allow triggers.

   For Linux, OS X, or Unix:

```
aws rds modify-db-parameter-group \
    --db-parameter-group-name allow-triggers \
    --parameters "name=log_bin_trust_function_creators,value=true, method=pending-
reboot"
```

   For Windows:

```
aws rds modify-db-parameter-group ^
    --db-parameter-group-name allow-triggers ^
    --parameters "name=log_bin_trust_function_creators,value=true, method=pending-
reboot"
```

3. Modify your DB instance to use the new DB parameter group.

   For Linux, OS X, or Unix:

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --db-parameter-group-name allow-triggers \
    --apply-immediately
```

   For Windows:

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --db-parameter-group-name allow-triggers ^
    --apply-immediately
```

4. In order for the changes to take effect, manually reboot the DB instance.

```
aws rds reboot-db-instance mydbinstance
```

Diagnosing and Resolving Point-In-Time Restore Failures

Restoring a DB Instance That Includes Temporary Tables

When attempting a Point-In-Time Restore (PITR) of your MySQL or MariaDB DB instance, you might encounter the following error:

```
Database instance could not be restored because there has been incompatible database
 activity for restore
functionality. Common examples of incompatible activity include using temporary tables, in-
memory tables,
or using MyISAM tables. In this case, use of Temporary table was detected.
```

PITR relies on both backup snapshots and binlogs from MySQL or MariaDB to restore your DB instance to a particular time. Temporary table information can be unreliable in binlogs and can cause a PITR failure. If you use temporary tables in your MySQL or MariaDB DB instance, you can minimize the possibility of a PITR failure by performing more frequent backups. A PITR failure is most probable in the time between a temporary table's creation and the next backup snapshot.

Restoring a DB Instance That Includes In-Memory Tables

You might encounter a problem when restoring a database that has in-memory tables. In-memory tables are purged during a restart. As a result, your in-memory tables might be empty after a reboot. We recommend that when you use in-memory tables, you architect your solution to handle empty tables in the event of a restart. If you are using in-memory tables with replicated DB instances, you might need to recreate the Read Replicas after a restart if a Read Replica reboots and is unable to restore data from an empty in-memory table.

For more information about backups and PITR, see Working With Backups (p. 201) and Restoring a DB Instance to a Specified Time (p. 237).

Slave Down or Disabled Error

When you call the `mysql.rds_skip_repl_error` command, you might receive the following error message: `Slave is down or disabled.`

This error message appears because replication has stopped and could not be restarted.

If you need to skip a large number of errors, the replication lag can increase beyond the default retention period for binary log files. In this case, you might encounter a fatal error due to binary log files being purged before they have been replayed on the replica. This purge causes replication to stop, and you can no longer call the `mysql.rds_skip_repl_error` command to skip replication errors.

You can mitigate this issue by increasing the number of hours that binary log files are retained on your replication master. After you have increased the binlog retention time, you can restart replication and call the `mysql.rds_skip_repl_error` command as needed.

To set the binlog retention time, use the mysql.rds_set_configuration (p. 923) procedure and specify a configuration parameter of 'binlog retention hours' along with the number of hours to retain binlog files on the DB cluster, up to 720 (30 days). The following example sets the retention period for binlog files to 48 hours:

```
CALL mysql.rds_set_configuration('binlog retention hours', 48);
```

Read Replica Create Fails or Replication Breaks With Fatal Error 1236

After changing default parameter values for a MySQL or MariaDB DB instance, you might encounter one of the following problems:

- You are unable to create a Read Replica for the DB instance.
- Replication fails with `fatal error 1236`.

Some default parameter values for MySQL or MariaDB DB instances help to ensure the database is ACID compliant and Read Replicas are crash-safe by making sure that each commit is fully synchronized by writing the transaction to the binary log before it is committed. Changing these parameters from their default values to improve performance can cause replication to fail when a transaction has not been written to the binary log.

To resolve this issue, set the following parameter values:

- `sync-binlog = 1`
- `innodb_support_xa = 1`
- `innodb_flush_log_at_trx_commit = 1`

Amazon Aurora Issues

No Space Left on Device Error

You might encounter the following error message from Amazon Aurora:

```
ERROR 3 (HY000): Error writing file '/rdsdbdata/tmp/XXXXXXXX' (Errcode: 28 - No space left
 on device)
```

Each DB instance in an Amazon Aurora DB cluster uses local SSD storage to store temporary tables for a session. This local storage for temporary tables does not autogrow like the Aurora cluster volume. Instead, the amount of local storage is limited. The limit is based on the DB instance class for DB instances in your DB cluster. To find the amount of local SSD storage for R3 DB instance types, go to Memory Optimized R3 instances.

If your workload cannot be modified to reduce the amount temporary storage required, then you can scale your DB instances up to use a DB instance class that has more local SSD storage.

Amazon RDS Oracle GoldenGate Issues

Retaining Logs for Sufficient Time

The source database must retain archived redo logs. The duration for log retention is specified in hours. The duration should exceed any potential downtime of the source instance or any potential period of communication or networking issues for the source instance, so that Oracle GoldenGate can recover logs from the source instance as needed. The absolute minimum value required is one (1) hour of logs retained. If you don't have log retention enabled, or if the retention value is too small, you will receive the following message:

```
2014-03-06 06:17:27  ERROR   OGG-00446  error 2 (No such file or directory)
opening redo log /rdsdbdata/db/GGTEST3_A/onlinelog/o1_mf_2_9k4bp1n6_.log
for sequence 1306Not able to establish initial position for begin time 2014-03-06
 06:16:55.
```

Cannot Connect to Amazon RDS SQL Server DB Instance

When you have problems connecting to a DB instance using SQL Server Management Studio, the following are some common causes:

- The access rules enforced by your local firewall and the IP addresses you authorized to access your DB instance in the instance's security group are not in sync. If you use your DB instance's endpoint and port with Microsoft SQL Server Management Studio and cannot connect, the problem is most likely the egress or ingress rules on your firewall. To grant access, you must create your own security group with specific ingress and egress rules for your situation. For more information about security groups, see Amazon RDS Security Groups (p. 375).
- The port you specified when you created the DB instance cannot be used to send or receive communications due to your local firewall restrictions. In this case, check with your network administrator to determine if your network allows the specified port to be used for inbound and outbound communication.
- Your DB instance is still being created and is not yet available. Depending on the size of your DB instance, it can take up to 20 minutes before an instance is available.

If you can send and receive communications through the port you specified, check for the following SQL Server errors:

- **Could not open a connection to SQL Server - Microsoft SQL Server, Error: 53** – You must include the port number when you specify the server name when using Microsoft SQL Server Management Studio. For example, the server name for a DB instance (including the port number) might be: **`sqlsvr-pdz.c6c8mdfntzgv0.region.rds.amazonaws.com,1433`**.
- **No connection could be made because the target machine actively refused it - Microsoft SQL Server, Error: 10061** – In this case, you reached the DB instance but the connection was refused. This error is often caused by an incorrect user name or password.

Cannot Connect to Amazon RDS PostgreSQL DB Instance

The most common problem when attempting to connect to a PostgreSQL DB instance is that the security group assigned to the DB instance has incorrect access rules. By default, DB instances do not allow access; access is granted through a security group. To grant access, you must create your own security group with specific ingress and egress rules for your situation. For more information about creating a security group for your DB instance, see Provide Access to the DB Instance in the VPC by Creating a Security Group (p. 8).

The most common error is `could not connect to server: Connection timed out`. If you receive this error, check that the host name is the DB instance endpoint and that the port number is correct. Check that the security group assigned to the DB instance has the necessary rules to allow access through your local firewall.

Amazon RDS Application Programming Interface (API)

In addition to the AWS Management Console, and the AWS Command Line Interface (AWS CLI), the Amazon Relational Database Service also provides an application programming interface (API). You can use the API to automate many of the tasks for managing your DB instances and other objects on Amazon RDS.

- For the alphabetical list of API actions, see API Actions.
- For the alphabetical list of data types, see Data Types.
- For a list of common query parameters, see Common Parameters.
- For descriptions of the error codes, see Common Errors.

For more information about the AWS CLI, see AWS Command Line Interface Reference for Amazon RDS.

For information on using the Amazon RDS API, see the following topics:

Topics

Using the Query API

The following sections discuss the parameters and request authentication used with the Query API.

Query Parameters

HTTP Query-based requests are HTTP requests that use the HTTP verb GET or POST and a Query parameter named `Action`.

Each Query request must include some common parameters to handle authentication and selection of an action.

Some operations take lists of parameters. These lists are specified using the `param.`*`n`* notation. Values of *`n`* are integers starting from 1.

For information about Amazon RDS regions and endpoints, go to Amazon Relational Database Service (RDS) in the Regions and Endpoints section of the *Amazon Web Services General Reference*.

Query Request Authentication

You can only send Query requests over HTTPS, and you must include a signature in every Query request. You must use either a signature version 2 or signature version 4. This section describes how to create a signature version 2. For information about creating a signature version 4, see Signature Version 4 Signing Process.

The following are the basic steps used to authenticate requests to AWS. This process assumes you are registered with AWS and have an access key ID and secret access key.

Tip
You can find your access key ID and secret access key in the Security Credentials section of the AWS Your Account page.

To authenticate requests to AWS

1. The sender constructs a request to AWS.
2. The sender calculates the request signature, a Keyed-Hashing for Message Authentication Code (HMAC) with a SHA-1 hash function, as defined in the next section of this topic.
3. The sender of the request sends the request data, the signature, and access key ID (the key identifier of the secret access key used) to AWS.
4. AWS uses the access key ID to look up the secret access key.
5. AWS generates a signature from the request data and the secret access key using the same algorithm used to calculate the signature in the request.
6. If the signatures match, the request is considered to be authentic. If the comparison fails, the request is discarded, and AWS returns an error response.

Note
If a request contains a `Timestamp` parameter, the signature calculated for the request expires 15 minutes after its value. If a request contains an `Expires` parameter, the signature expires at the time specified by the `Expires` parameter.

To calculate the request signature

1. Create the canonicalized query string that you need later in this procedure:
   a. Sort the UTF-8 query string components by parameter name with natural byte ordering. The parameters can come from the GET URI or from the POST body (when Content-Type is application/x-www-form-urlencoded).
   b. URL encode the parameter name and values according to the following rules:
      i. Do not URL encode any of the unreserved characters that RFC 3986 defines. These unreserved characters are A–Z, a–z, 0–9, hyphen (-), underscore (_), period (.), and tilde (~).
      ii. Percent encode all other characters with %XY, where X and Y are hex characters 0–9 and uppercase A–F.
      iii. Percent encode extended UTF-8 characters in the form %XY%ZA....
      iv. Percent encode the space character as %20 (and not +, as common encoding schemes do).
   c. Separate the encoded parameter names from their encoded values with the equals sign (=) (ASCII character 61), even if the parameter value is empty.
   d. Separate the name-value pairs with an ampersand (&) (ASCII code 38).
2. Create the string to sign according to the following pseudo-grammar (the "\n" represents an ASCII newline).

```
StringToSign = HTTPVerb + "\n" +
ValueOfHostHeaderInLowercase + "\n" +
HTTPRequestURI + "\n" +
CanonicalizedQueryString <from the preceding step>
```

   The HTTPRequestURI component is the HTTP absolute path component of the URI up to, but not including, the query string. If the HTTPRequestURI is empty, use a forward slash (/).
3. Calculate an RFC 2104-compliant HMAC with the string you just created, your secret access key as the key, and SHA256 or SHA1 as the hash algorithm.

For more information, go to RFC 2104.

4. Convert the resulting value to base 64.
5. Include the value as the value of the `Signature` parameter in the request.

For example, the following is an example request (line breaks added for clarity).

```
https://rds.amazonaws.com/
    ?Action=DescribeDBInstances
    &DBInstanceIdentifier=myinstance
    &Version=2010-01-01
    &Timestamp=2010-05-10T17%3A09%3A03.726Z
    &SignatureVersion=2
    &SignatureMethod=HmacSHA256
    &AWSAccessKeyId=<Your AWS Access Key ID>
```

For the preceding Query string, you calculate the HMAC signature over the following string.

```
GET\n
rds.amazonaws.com\n
AWSAccessKeyId=<Your AWS Access Key ID>
&Action=DescribeDBInstances
&DBInstanceIdentifier=myinstance
&Timestamp=2010-05-10T17%3A09%3A03.726Z
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Version=2009-10-16
```

The result is the following signed request.

```
https://rds.amazonaws.com/
?Action=DescribeDBInstances
 &DBInstanceIdentifier=myinstance
 &Version=2010-01-01
 &Timestamp=2010-05-10T17%3A09%3A03.726Z
 &Signature=<URLEncode(Base64Encode(Signature))>
 &SignatureVersion=2
 &SignatureMethod=HmacSHA256
 &AWSAccessKeyId=<Your AWS Access Key ID>
```

Troubleshooting Applications on Amazon RDS

Topics

Amazon Relational Database Service; provides specific and descriptive errors to help you troubleshoot problems while interacting with the Amazon RDS API.

Retrieving Errors

Typically, you want your application to check whether a request generated an error before you spend any time processing results. The easiest way to find out if an error occurred is to look for an `Error` node in the response from the Amazon RDS API.

XPath syntax provides a simple way to search for the presence of an `Error` node, as well as an easy way to retrieve the error code and message. The following code snippet uses Perl and the XML::XPath module to determine if an error occurred during a request. If an error occurred, the code prints the first error code and message in the response.

```
use XML::XPath;
    my $xp = XML::XPath->new(xml =>$response);
    if ( $xp->find("//Error") )
    {print "There was an error processing your request:\n", " Error code: ",
    $xp->findvalue("//Error[1]/Code"), "\n", " ",
    $xp->findvalue("//Error[1]/Message"), "\n\n"; }
```

Troubleshooting Tips

We recommend the following processes to diagnose and resolve problems with the Amazon Relational Database Service API.

- Verify that Amazon Relational Database Service is operating normally in the region you are targeting by visiting http://status.aws.amazon.com.
- Check the structure of your request

  Each Amazon Relational Database Service operation has a reference page in the *Amazon RDS API Reference*. Double-check that you are using parameters correctly. In order to give you ideas regarding what might be wrong, look at the sample requests or user scenarios to see if those examples are doing similar operations.
- Check the forum

  Amazon RDS has a development community forum where you can search for solutions to problems others have experienced along the way. To view the forum, go to

  https://forums.aws.amazon.com/

RDS REST API Reference

Standard API syntax cannot be used in certain scenarios. The `DownloadCompleteDBLogFile` action is a REST API action that you can use to retrieve log files. Because a database log file can be arbitrarily large, the `DownloadCompleteDBLogFile` REST API is provided to enable streaming of the log file contents.

DownloadCompleteDBLogFile

Description

Downloads the contents of the specified database log file.

Request Parameters

DBInstanceIdentifier

The customer-assigned name of the DB instance that contains the log file you want to download.

LogFileName

The name of the log file to be downloaded.

Syntax

```
GET /v13/downloadCompleteLogFile/DBInstanceIdentifier/LogFileName HTTP/1.1
Content-type: application/json
host: rds.region.amazonaws.com
```

Response Elements

The DownloadCompleteDBLogFile REST API returns the contents of the requested log file as a stream.

Errors

DBInstanceNotFound

DBInstanceIdentifier does not refer to an existing DB instance.

HTTP Status Code: 404

Examples

The following example downloads the log file named *log/ERROR.6* for the DB instance named *sample-sql* in the *us-west-2* region.

```
GET /v13/downloadCompleteLogFile/sample-sql/log/ERROR.6 HTTP/1.1
host: rds.us-west-2.amazonaws.com
X-Amz-Security-Token: AQoDYXdzEIH//////////
wEa0AIXLhngC5zp9CyB1R6abwKrXHVR5efnAVN3XvR7IwqKYalFSn6UyJuEFTft9nObglx4QJ+GXV9cpACkETq=
X-Amz-Date: 20140903T233749Z
X-Amz-Algorithm: AWS4-HMAC-SHA256
X-Amz-Credential: AKIADQKE4SARGYLE/20140903/us-west-2/rds/aws4_request
X-Amz-SignedHeaders: host
X-Amz-Content-SHA256: e3b0c44298fc1c229afbf4c8996fb92427ae41e4649b934de495991b7852b855
X-Amz-Expires: 86400
X-Amz-Signature: 353a4f14b3f250142d9afc34f9f9948154d46ce7d4ec091d0cdabbcf8b40c558
```

The rds-download-db-logfile Command

You can also download complete log files using the rds-download-db-logfile command. Because the AWS CLI does not currently support the rds-download-db-logfile command, you must use the deprecated RDS CLI to run the rds-download-db-logfile command. You can get the last version of the RDS CLI in a ZIP file at http://s3.amazonaws.com/rds-downloads/RDSCli.zip.

Use the following syntax when using the RDS CLI with the rds-download-db-logfile command:

```
rds-download-db-logfile db-instance-identifier
--log-file-name value
[General Options]
```

For example, the following command downloads a log named *log/ERROR.4* for the *myexampledb* RDS DB instance and stored the log in a file called errorlog.txt.

```
PROMPT> rds-download-db-logfile myexampledb --region us-west-2 --log-file-name log/ERROR.4
 > errorlog.txt
```

Related Topics

- Downloading a Database Log File (p. 304)
- download-db-logfile-portion
- DownloadDBLogFilePortion
- RDS Query API Documentation
- Signature Version 4 Signing Process

Resources for Amazon RDS

The following table lists related resources that you'll find useful as you work with Amazon RDS.

Resource	Description
Amazon RDS FAQs	The FAQ answers the top questions that developers ask about Amazon RDS.
API Reference	The API reference contains a comprehensive description of all Amazon RDS APIs and data types.
AWS CLI Reference	The Command Line Tools Reference contains a comprehensive description of all the command line tools and their options.
AWS Management Console	Access and manage Amazon Web Services through a simple and intuitive web-based user interface. The AWS Management Console allows you to perform most of the functions of Amazon RDS without programming.
AWS Support	Based on your current or planned use-cases, AWS Support provides a unique combination of tools and expertise to help you do amazing things with AWS.
Conditions of Use	Detailed information about the copyright and trademark usage at Amazon.com and other topics.
Contact Us	A central contact point for inquiries concerning AWS billing, account, events, abuse etc.
Discussion Forums	A community-based forum for developers to discuss technical questions related to Amazon RDS.
Product Information	The primary web page for information about Amazon RDS.
Release Notes	Release notes identify new features, corrections, and known issues.
Tools for Amazon Web Services	The place to find developer tools, SDKs, IDE Toolkits, and Command Line Tools for developing and managing your AWS applications.

Document History

The following table describes the important changes to the documentation since the last release of the Amazon Relational Database Service User Guide.

- **Latest documentation update:** December 6, 2017
- **Current API version:** 2014-10-31

Change	Description	Date Changed
Added Aurora PostgreSQL HIPAA compliance	Aurora PostgreSQL now supports building HIPAA compliant applications, see Working with Amazon Aurora PostgreSQL (p. 640).	December 6, 2017
Additional AWS Regions available for Amazon Aurora with PostgreSQL compatibility	Amazon Aurora with PostgreSQL compatibility is now available in four new AWS Regions. For more information, see Availability for Amazon Aurora PostgreSQL (p. 641).	November 22, 2017
Modify storage for Amazon RDS DB instances running Microsoft SQL Server	You can now modify the storage of your Amazon RDS DB instances running SQL Server. For more information, see Modifying a DB Instance Running the Microsoft SQL Server Database Engine (p. 756).	November 21, 2017
Amazon RDS supports 16 TiB storage for Linux-based engines	You can now create MySQL, MariaDB, PostgreSQL, and Oracle RDS DB instances with up to 16 TiB of storage. For more information, see Storage for Amazon RDS (p. 410).	November 21, 2017
Amazon RDS supports fast scale up of storage	You can now add storage to MySQL, MariaDB, PostgreSQL, and Oracle RDS DB instances in a few minutes. For more information, see Adding Storage and Changing Storage Type for MariaDB, MySQL, Oracle, and PostgreSQL (p. 415).	November 21, 2017
Amazon RDS supports MariaDB versions 10.1.26 and 10.0.32	You can now create Amazon RDS DB instances running MariaDB versions 10.1.26 and 10.0.32. For more information, see MariaDB on Amazon RDS Versions (p. 667).	November 20, 2017
Amazon RDS for Microsoft SQL Server now supports new DB instance classes	You can now create Amazon RDS DB instances running SQL Server that use the db.r4 and db.m4.16xlarge DB instance classes. For more information, see DB Instance Class Support for Microsoft SQL Server (p. 723).	November 20, 2017
Amazon RDS for MySQL and MariaDB now supports new DB instance classes	You can now create Amazon RDS DB instances running MySQL and MariaDB that use the db.r4, db.m4.16xlarge, db.t2.xlarge, and db.t2.2xlarge DB instance classes. For more information, see DB Instance Class (p. 92).	November 20, 2017
SQL Server 2017	You can now create Amazon RDS DB instances running Microsoft SQL Server 2017. You can also create DB	November 17, 2017

Change	Description	Date Changed
	instances running SQL Server 2016 SP1 CU5. For more information, see Microsoft SQL Server on Amazon RDS (p. 720).	
Restore MySQL backups from Amazon S3	You can now create a backup of your on-premises database, store it on Amazon S3, and then restore the backup file onto a new Amazon RDS DB instance running MySQL. For more information, see Importing Data into an Amazon RDS MySQL DB Instance (p. 860).	November 17, 2017
Auto Scaling with Aurora Replicas	Amazon Aurora MySQL now supports Aurora Auto Scaling. Aurora Auto Scaling dynamically adjusts the number of Aurora Replicas based on increases or decreases in connectivity or workload. For more information, see Using Amazon Aurora Auto Scaling with Aurora Replicas (p. 577).	November 17, 2017
Oracle default edition support	Amazon RDS for Oracle DB instances now supports setting the default edition for the DB instance. For more information, see Setting the Default Edition for a DB Instance (p. 1058).	November 3, 2017
Oracle DB instance file validation	Amazon RDS for Oracle DB instances now supports validating DB instance files with the Oracle Recovery Manager (RMAN) logical validation utility. For more information, see Validating DB Instance Files (p. 1058).	November 3, 2017
Oracle July 2017 PSU	Amazon RDS for Oracle has released database engine versions 12.1.0.2.v9 and 11.2.0.4.v13 to support the July 2017 Oracle Database Patch Set Update (PSU). For more information, see Appendix: Oracle Database Engine Release Notes (p. 1120).	November 3, 2017
Management Agent for OEM 13c	Amazon RDS Oracle DB instances now support the Management Agent for Oracle Enterprise Manager (OEM) Cloud Control 13c. For more information, see Oracle Management Agent for Enterprise Manager Cloud Control (p. 1010).	November 1, 2017
PostgreSQL 9.6.5, 9.5.9, 9.4.14, and 9.3.19	You can now create Amazon RDS DB instances running PostgreSQL versions 9.6.5., 9.5.9, 9.4.14, and 9.3.19. For more information, see Supported PostgreSQL Database Versions (p. 1147).	November 1, 2017
Asynchronous key prefetch for Aurora with MySQL compatibility	Asynchronous key prefetch (AKP) improves the performance of noncached index joins, by prefetching keys in memory ahead of when they are needed. For more information, see Working with Asynchronous Key Prefetch in Amazon Aurora (p. 594).	October 26, 2017
Storage reconfiguration for Microsoft SQL Server snapshots	You can now reconfigure the storage when you restore a snapshot to an Amazon RDS DB instance running Microsoft SQL Server. For more information, see Restoring from a DB Snapshot (p. 209).	October 26, 2017

Change	Description	Date Changed
MySQL 5.7.19, 5.6.37, and 5.5.57	You can now create Amazon RDS DB instances running MySQL versions 5.7.19, 5.6.37, and 5.5.57. For more information, see MySQL on Amazon RDS Versions (p. 822).	October 25, 2017
General availability of Amazon Aurora with PostgreSQL compatibility	Amazon Aurora with PostgreSQL compatibility makes it simple and cost-effective to set up, operate, and scale your new and existing PostgreSQL deployments, thus freeing you to focus on your business and applications. For more information, see Working with Amazon Aurora PostgreSQL (p. 640).	October 24, 2017
Amazon RDS for Oracle DB instances support new DB instance classes	Amazon RDS Oracle DB instances now support Memory Optimized Next Generation (db.r4) instance classes. Amazon RDS Oracle DB instances also now support the following new current generation instance classes: db.m4.16xlarge, db.t2.xlarge, and db.t2.2xlarge. For more information, see DB Instance Class (p. 92) and DB Instance Class Support for Oracle (p. 934).	October 23, 2017
New feature	Your new and existing Reserved Instances can now cover multiple sizes in the same DB instance class. Size-flexible reserved instances are available for DB instances with the same AWS Region, database engine, and instance family, and across AZ configuration. Size-flexible reserved instances are available for the following database engines: Amazon Aurora, MariaDB, MySQL, Oracle (Bring Your Own License), PostgreSQL. For more information, see Size-Flexible Reserved Instances (p. 189).	October 11, 2017
New feature	You can now use the Oracle SQLT option to tune a SQL statement for optimal performance. For more information, see Oracle SQLT (p. 1025).	September 22, 2017
New feature	If you have existing manual DB snapshots of your Amazon RDS Oracle DB instances, you can now upgrade them to a later version of the Oracle database engine. For more information, see Upgrading an Oracle DB Snapshot (p. 980).	September 20, 2017
New feature	You can now use Oracle Spatial to store, retrieve, update, and query spatial data in your Amazon RDS DB instances running Oracle. For more information, see Oracle Spatial (p. 1019).	September 15, 2017
New feature	You can now use Oracle Locator to support internet and wireless service-based applications and partner-based GIS solutions with your Amazon RDS DB instances running Oracle. For more information, see Oracle Locator (p. 1014).	September 15, 2017
New feature	You can now use Oracle Multimedia to store, manage, and retrieve images, audio, video, and other heterogeneous media data in your Amazon RDS DB instances running Oracle. For more information, see Oracle Multimedia (p. 1017).	September 15, 2017

Change	Description	Date Changed
New feature	You can now export audit logs from your Amazon Aurora MySQL DB clusters to Amazon CloudWatch Logs. For more information, see Exporting Audit Log Data From Amazon Aurora to Amazon CloudWatch Logs (p. 576).	September 14, 2017
New feature	Amazon RDS now supports multiple versions of Oracle Application Express (APEX) for your DB instances running Oracle. For more information, see Oracle Application Express (p. 994).	September 13, 2017
New feature	You can now use Amazon Aurora to migrate an unencrypted or encrypted DB snapshot or Amazon RDS MySQL DB instance to an encrypted Aurora MySQL DB cluster. For more information, see Migrating an RDS MySQL Snapshot to Aurora (p. 502) and Migrating Data from a MySQL DB Instance to an Amazon Aurora MySQL DB Cluster by Using an Aurora Read Replica (p. 507).	September 5, 2017
New feature	You can use Amazon RDS for Microsoft SQL Server databases to build HIPAA-compliant applications. For more information, see Compliance Program Support for Microsoft SQL Server DB Instances (p. 725).	August 31, 2017
New feature	You can now use Amazon RDS for MariaDB databases to build HIPAA-compliant applications. For more information, see MariaDB on Amazon RDS (p. 666).	August 31, 2017
New feature	You can now create Amazon RDS DB instances running Microsoft SQL Server with allocated storage up to 16 TB, and Provisioned IOPS to storage ranges of 1:1–50:1. For more information, see Storage for Amazon RDS (p. 410).	August 22, 2017
New feature	You can now use Multi-AZ deployments for DB instances running Microsoft SQL Server in the EU (Frankfurt) region. For more information, see Multi-AZ Deployments for Microsoft SQL Server with Database Mirroring (p. 787).	August 3, 2017
New feature	You can now create Amazon RDS DB instances running MariaDB versions 10.1.23 and 10.0.31. For more information, see MariaDB on Amazon RDS Versions (p. 667).	July 17, 2017
New feature	Amazon RDS now supports Microsoft SQL Server Enterprise Edition with the License Included model in all AWS Regions. For more information, see License Included (p. 735).	July 13, 2017

Change	Description	Date Changed
New feature	Amazon RDS for Oracle now supports Linux kernel huge pages for increased database scalability. The use of huge pages results in smaller page tables and less CPU time spent on memory management, increasing the performance of large database instances. You can use huge pages with your Amazon RDS DB instances running all editions of Oracle versions 12.1.0.2 and 11.2.0.4. For more information, see Using Huge Pages with an Oracle DB Instance (p. 945).	July 7, 2017
New feature	Updated to support encryption at rest (EAR) for db.t2.small and db.t2.medium DB instance classes for all non-Aurora DB engines. For more information, see Availability of Amazon RDS Encrypted Instances (p. 356).	June 27, 2017
New feature	Updated to support Amazon Aurora in the EU (Frankfurt) region. For more information, see Amazon Aurora on Amazon RDS (p. 428).	June 16, 2017
New feature	You can now specify an option group when you copy an DB snapshot across AWS regions. For more information, see Option Group Considerations (p. 214).	June 12, 2017
New feature	You can now copy DB snapshots created from specialized DB instances across AWS regions. You can copy snapshots from DB instances that use Oracle TDE, Microsoft SQL Server TDE, and Microsoft SQL Server Multi-AZ with Mirroring. For more information, see Copying a DB Snapshot (p. 215).	June 12, 2017
New feature	Amazon Aurora now allows you to quickly and cost-effectively copy all of your databases in an Amazon Aurora DB cluster. For more information, see Cloning Databases in an Aurora DB Cluster (p. 479).	June 12, 2017
New feature	Amazon RDS now supports Microsoft SQL Server 2016 SP1 CU2. For more information, see Microsoft SQL Server on Amazon RDS (p. 720).	June 7, 2017
New feature	Amazon RDS for Oracle has released database engine versions 12.1.0.2.v8 and 11.2.0.4.v12 to support the April 2017 Oracle Database Patch Set Update (PSU). For more information, see Appendix: Oracle Database Engine Release Notes (p. 1120).	May 23, 2017
New Feature	Amazon RDS now supports PostgreSQL versions 9.6.2, 9.5.6, 9.4.11, and 9.3.16. For more information, see Supported PostgreSQL Database Versions (p. 1147)	May 3, 2017
Preview	Public preview of Amazon Aurora with PostgreSQL Compatibility. For more information, see Working with Amazon Aurora PostgreSQL (p. 640).	April 19, 2017

Change	Description	Date Changed
New feature	Amazon Aurora now allows you to execute an ALTER TABLE *tbl_name* ADD COLUMN *col_name column_definition* operation nearly instantaneously. The operation completes without requiring the table to be copied and without materially impacting other DML statements. For more information, see Altering Tables in Amazon Aurora Using Fast DDL (p. 522).	April 5, 2017
New feature	We have added a new monitoring command, SHOW VOLUME STATUS, to display the number of nodes and disks in a volume. For more information, see Displaying Volume Status for an Aurora DB Cluster (p. 523).	April 5, 2017
New feature	Amazon RDS for Oracle now includes the January 2017 Oracle Database Patch Set Update (PSU). This adds support for database engine versions 12.1.0.2.v7 and 11.2.0.4.v11. For more information, see Appendix: Oracle Database Engine Release Notes (p. 1120).	March 21, 2017
New feature	You can now use your own custom logic in your custom password verification functions for Oracle on Amazon RDS. For more information, see Creating Custom Functions to Verify Passwords (p. 1049).	March 21, 2017
New feature	You can now access your online and archived redo log files on your Oracle DB instances on Amazon RDS. For more information, see Accessing Transaction Logs (p. 1071).	March 21, 2017
New feature	You can now copy both encrypted and unencrypted DB cluster snapshots between regions in the same account. For more information, see Copying a DB Cluster Snapshot (p. 221).	March 7, 2017
New feature	You can now copy both encrypted and unencrypted DB cluster snapshots between accounts in the same region. For more information, see Copying a DB Cluster Snapshot Across Accounts (p. 226).	March 7, 2017
New feature	You can now share encrypted DB cluster snapshots between accounts in the same region. For more information, see Sharing a DB Snapshot or DB Cluster Snapshot (p. 230).	March 7, 2017
New feature	You can now replicate encrypted Amazon Aurora MySQL DB clusters to create cross-region Aurora Replicas. For more information, see Replicating Amazon Aurora MySQL DB Clusters Across AWS Regions (p. 528).	March 7, 2017
New feature	You can now require that all connections to your DB instance running Microsoft SQL Server use Secure Sockets Layer (SSL). For more information, see Using SSL with a Microsoft SQL Server DB Instance (p. 791).	February 27, 2017
New feature	You can now set your local time zone to one of 15 additional time zones. For more information, see Supported Time Zones (p. 731).	February 27, 2017

Change	Description	Date Changed
New feature	You can now use the Amazon RDS procedure `msdb.dbo.rds_shrink_tempdbfile` to shrink the tempdb database on your DB instances running Microsoft SQL Server. For more information, see Shrinking the tempdb Database (p. 801).	February 17, 2017
New feature	You can now compress your backup file when you export your Enterprise and Standard Edition Microsoft SQL Server database from an Amazon RDS DB instance to Amazon S3. For more information, see Compressing Backup Files (p. 775).	February 17, 2017
New feature	Amazon RDS now supports custom DNS servers to resolve DNS names used in outbound network access on your DB instances running Oracle. For more information, see Setting Up a Custom DNS Server (p. 1053).	January 26, 2017
New feature	Amazon RDS now supports creating an encrypted Read Replica in another region. For more information, see Replicating a Read Replica Across AWS Regions (p. 142) and CreateDBInstanceReadReplica.	January 23, 2017
New feature	Amazon RDS now supports upgrading a MySQL DB snapshot from MySQL 5.1 to MySQL 5.5. For more information, see Upgrading a MySQL DB Snapshot (p. 857) and ModifyDBSnapshot.	January 20, 2017
New feature	Amazon RDS now supports copying an encrypted DB snapshot to another region for the MariaDB, MySQL, Oracle, PostgreSQL, and Microsoft SQL Server database engines. For more information, see Copying a DB Snapshot (p. 215) and CopyDBSnapshot.	December 20, 2016
New feature	Amazon RDS now supports migrating an Amazon RDS MySQL 5.6 DB snapshot to a new DB instance running MariaDB 10.1. For more information, see Migrating Data from a MySQL DB Snapshot to a MariaDB DB Instance (p. 702).	December 20, 2016
New feature	Amazon Aurora MySQL now supports spatial indexing. Spatial indexing improves query performance on large datasets for queries that use spatial data. For more information, see Amazon Aurora MySQL and Spatial Data (p. 485).	December 14, 2016
New feature	Amazon RDS for Oracle now includes the October 2016 Oracle Database Patch Set Update (PSU). This adds support for Oracle database engine versions 12.1.0.2.v6 and 11.2.0.4.v10. For more information, see Appendix: Oracle Database Engine Release Notes (p. 1120).	December 12, 2016
New feature	Amazon RDS now supports outbound network access on your DB instances running Oracle. You can use utl_http, utl_tcp, and utl_smtp to connect from your DB instance to the network. For more information, see Using utl_http, utl_tcp, and utl_smtp with an Oracle DB Instance (p. 947).	December 5, 2016

Change	Description	Date Changed
New feature	Amazon RDS has retired support for MySQL version 5.1. However, you can restore existing MySQL 5.1 snapshots to a MySQL 5.5 instance. For more information, see Supported Storage Engines for MySQL on Amazon RDS (p. 824).	November 15, 2016
New feature	Amazon RDS now supports PostgreSQL version 9.6.1. For more information, see PostgreSQL Version 9.6.1 on Amazon RDS (p. 1149).	November 11, 2016
New feature	Amazon RDS now supports Microsoft SQL Server 2016 RTM CU2. For more information, see Microsoft SQL Server on Amazon RDS (p. 720).	November 4, 2016
New feature	Amazon RDS now supports major version upgrades for DB instances running Oracle. You can now upgrade your Oracle DB instances from 11g to 12c. For more information, see Upgrading the Oracle DB Engine (p. 975).	November 2, 2016
New feature	You can now create DB instances running Microsoft SQL Server 2014 Enterprise Edition. Amazon RDS now supports SQL Server 2014 SP2 for all editions and all regions. For more information, see Microsoft SQL Server on Amazon RDS (p. 720).	October 25, 2016
New feature	Amazon Aurora MySQL now integrates with other AWS services: You can load text or XML data into a table from an Amazon S3 bucket, or invoke an AWS Lambda function from database code. For more information, see Integrating Amazon Aurora MySQL with Other AWS Services (p. 550).	October 18, 2016
New feature	You can now access the tempdb database on your Amazon RDS DB instances running Microsoft SQL Server. You can access the tempdb database by using Transact-SQL through Microsoft SQL Server Management Studio (SSMS), or any other standard SQL client application. For more information, see Accessing the tempdb Database on Microsoft SQL Server DB Instances on Amazon RDS (p. 801).	September 29, 2016
New feature	You can now use the UTL_MAIL package with your Amazon RDS DB instances running Oracle. For more information, see Oracle UTL_MAIL (p. 1038).	September 20, 2016
New feature	Amazon RDS for Oracle now includes the July 2016 Oracle Database Patch Set Update (PSU). This adds support for Oracle database engine versions 12.1.0.2.v5, 12.1.0.1.v6, and 11.2.0.4.v9. For more information, see Appendix: Oracle Database Engine Release Notes (p. 1120).	September 20, 2016

Change	Description	Date Changed
New features	You can now set the time zone of your new Microsoft SQL Server DB instances to a local time zone, to match the time zone of your applications. For more information, see Local Time Zone for Microsoft SQL Server DB Instances (p. 731).	September 19, 2016
New features	Added support for new PostgreSQL versions 9.5.4, 9.4.9, and 9.3.14. Also added support for PostgreSQL logical replication, PostgreSQL event triggers, and RAM disk for the PostgreSQL stats_temp_directory. For more information, see Supported PostgreSQL Database Versions (p. 1147), Logical Replication for PostgreSQL on Amazon RDS (p. 1165), Event Triggers for PostgreSQL on Amazon RDS (p. 1167), and RAM Disk for the stats_temp_directory (p. 1168).	September 14, 2016
New feature	You can now use the Oracle Label Security option to control access to individual table rows in your Amazon RDS DB instances running Oracle 12c. With Oracle Label Security, you can enforce regulatory compliance with a policy-based administration model, and ensure that an access to sensitive data is restricted to only users with the appropriate clearance level. For more information, see Oracle Label Security (p. 1000).	September 8, 2016
New feature	You can now connect to an Amazon Aurora DB cluster using the reader endpoint, which load-balances connections across the Aurora Replicas that are available in the DB cluster. As clients request new connections to the reader endpoint, Aurora distributes the connection requests among the Aurora Replicas in the DB cluster. This functionality can help balance your read workload across multiple Aurora Replicas in your DB cluster. For more information, see Aurora Endpoints (p. 431).	September 8, 2016
New feature	You can now support the Oracle Enterprise Manager Cloud Control on your Amazon RDS DB instances running Oracle. You can enable the Management Agent on your DB instances, and share data with your Oracle Management Service (OMS). For more information, see Oracle Management Agent for Enterprise Manager Cloud Control (p. 1010).	September 1, 2016
New feature	This release adds support to get an ARN for a resource. For more information, see Getting an Existing ARN (p. 186).	August 23, 2016
New feature	You can now assign up to 50 tags for each Amazon RDS resource, for managing your resources and tracking costs. For more information, see Tagging Amazon RDS Resources (p. 129).	August 19, 2016

Change	Description	Date Changed
New feature	Amazon RDS now supports the License Included model for Oracle Standard Edition Two. For more information, see Creating a DB Instance Running the Oracle Database Engine (p. 949). You can now change the license model of your Amazon RDS DB instances running Microsoft SQL Server and Oracle. For more information, see Licensing Microsoft SQL Server on Amazon RDS (p. 735) and Oracle Licensing (p. 933).	August 5, 2016
New feature	You can now use the AWS Management Console to easily move your DB instance to a different VPC, or to a different subnet group in the same VPC. For more information, see Updating the VPC for a DB Instance (p. 404). If your DB instance is not in a VPC, you can now use the AWS Management Console to easily move your DB instance into a VPC. For more information, see Moving a DB Instance Not in a VPC into a VPC (p. 405).	August 4, 2016
New feature	Amazon RDS now supports native backup and restore for Microsoft SQL Server databases using full backup files (.bak files). You can now easily migrate SQL Server databases to Amazon RDS, and import and export databases in a single, easily-portable file, using Amazon S3 for storage, and AWS KMS for encryption. For more information, see Importing and Exporting SQL Server Databases (p. 769).	July 27, 2016
New feature	You can now copy the source files from a MySQL database to an Amazon Simple Storage Service (Amazon S3) bucket, and then restore an Amazon Aurora DB cluster from those files. This option can be considerably faster than migrating data using `mysqldump`. For more information, see Migrating Data from MySQL by Using an Amazon S3 Bucket (p. 488).	July 20, 2016
New feature	You can now restore an unencrypted Amazon Aurora DB cluster snapshot to create an encrypted Amazon Aurora DB cluster by including an AWS Key Management Service (AWS KMS) encryption key during the restore operation. For more information, see Encrypting Amazon RDS Resources (p. 355).	June 30, 2016
New feature	Amazon RDS for Oracle now includes the April 2016 Oracle Database Patch Set Update (PSU). This PSU adds support for Oracle database engine versions 12.1.0.2.v4, 12.1.0.1.v5, and 11.2.0.4.v8. For more information, see Appendix: Oracle Database Engine Release Notes (p. 1120).	June 17, 2016

Change	Description	Date Changed
New feature	You can use the Oracle Repository Creation Utility (RCU) to create a repository on Amazon RDS for Oracle. For more information, see Using the Oracle Repository Creation Utility on Amazon RDS for Oracle (p. 1112).	June 17, 2016
New feature	Adds support for PostgreSQL cross-region Read Replicas. For more information, see Replicating a Read Replica Across AWS Regions (p. 142).	June 16, 2016
New feature	You can now use the AWS Management Console to easily add Multi-AZ with Mirroring to a Microsoft SQL Server DB instance. For more information, see Adding Multi-AZ with Mirroring to a Microsoft SQL Server DB Instance (p. 787).	June 9, 2016
New feature	You can now use Multi-AZ Deployments Using SQL Server Mirroring in the following additional regions: Asia Pacific (Sydney), Asia Pacific (Tokyo), and South America (Sao Paulo). For more information, see Multi-AZ Deployments for Microsoft SQL Server with Database Mirroring (p. 787).	June 9, 2016
New feature	Updated to support Amazon Aurora cross-region DB clusters that are Read Replicas. For more information, see Replicating Amazon Aurora MySQL DB Clusters Across AWS Regions (p. 528).	June 1, 2016
New feature	Updated to support MariaDB version 10.1. For more information, see MariaDB on Amazon RDS (p. 666).	June 1, 2016
New feature	Enhanced Monitoring is now available for Oracle DB instances. For more information, see Enhanced Monitoring (p. 258) and Modifying a DB Instance Running the Oracle Database Engine (p. 967).	May 27, 2016
New feature	Updated to support manual snapshot sharing for Amazon Aurora DB cluster snapshots. For more information, see Sharing a DB Snapshot or DB Cluster Snapshot (p. 230).	May 18, 2016
New feature	You can now use the MariaDB Audit Plugin to log database activity on MariaDB and MySQL database instances. For more information, see Appendix: Options for MariaDB Database Engine (p. 709) and Options for MySQL DB Instances (p. 897).	April 27, 2016
New feature	In-place, major version upgrades are now available for upgrading from MySQL version 5.6 to version 5.7. For more information, see Upgrading the MySQL DB Engine (p. 851).	April 26, 2016
New feature	Enhanced Monitoring is now available for Microsoft SQL Server DB instances. For more information, see Enhanced Monitoring (p. 258).	April 22, 2016
New feature	Added support for PostgreSQL versions 9.5.2, 9.4.7, and 9.3.12. For more information, see Supported PostgreSQL Database Versions (p. 1147).	April 8, 2016

Change	Description	Date Changed
New feature	Updated to provide an Amazon Aurora **Clusters** view in the Amazon RDS console. For more information, see Viewing an Amazon Aurora DB Cluster (p. 461).	April 1, 2016
New feature	Updated to support Oracle database versions 11.2.0.4.v7, 12.1.0.1.v4, and 12.1.0.2.v3 with the January 2016 Oracle Patch Set Updates (PSU). For more information, see Appendix: Oracle Database Engine Release Notes (p. 1120).	April 1, 2016
New feature	Updated to support Amazon Aurora and SQL Server Multi-AZ with mirroring in the Asia Pacific (Seoul) region. For more information, see Amazon Aurora on Amazon RDS (p. 428) and Multi-AZ Deployments for Microsoft SQL Server with Database Mirroring (p. 787).	March 31, 2016
New feature	PostgreSQL DB instances have the ability to require connections to use SSL. For more information, see Using SSL with a PostgreSQL DB Instance (p. 1170).	March 25, 2016
New feature	Enhanced Monitoring is now available for PostgreSQL DB instances. For more information, see Enhanced Monitoring (p. 258).	March 25, 2016
New feature	Microsoft SQL Server DB instances can now use Windows Authentication for user authentication. For more information, see Using Windows Authentication with a Microsoft SQL Server DB Instance (p. 812).	March 23, 2016
New feature	Enhanced Monitoring is now available in the Asia Pacific (Seoul) region. For more information, see Enhanced Monitoring (p. 258).	March 16, 2016
New feature	You can now customize the order in which Aurora Replicas are promoted to primary instance during a failover. For more information, see Fault Tolerance for an Aurora DB Cluster (p. 468).	March 14, 2016
New feature	Updated to support encryption when migrating to an Aurora DB cluster. For more information, see Migrating Data to an Amazon Aurora DB Cluster (p. 466).	March 2, 2016
New feature	Updated to support local time zone for Aurora DB clusters. For more information, see Local Time Zone for Amazon Aurora DB Clusters (p. 434).	March 1, 2016
New feature	Updated to add support for MySQL version 5.7 for current generation Amazon RDS DB instance classes.	February 22, 2016
New feature	Updated to support Amazon Aurora in the Asia Pacific (Sydney) region. For more information, see Amazon Aurora on Amazon RDS (p. 428).	February 11, 2016
New feature	Updated to support *db.r3* and *db.t2* DB instance classes in the AWS GovCloud (US) region.	February 11, 2016

Change	Description	Date Changed
New feature	Updated to support encrypting copies of DB snapshots and sharing encrypted DB snapshots. For more information, see Copying a DB Snapshot or DB Cluster Snapshot (p. 213) and Sharing a DB Snapshot or DB Cluster Snapshot (p. 230).	February 11, 2016
New feature	Updated to support SSL for Oracle DB Instances. For more information, see SSL Support for Oracle DB Instances (p. 936).	February 9, 2016
New feature	Updated to support local time zone for MySQL and MariaDB DB instances. For more information, see Local Time Zone for MySQL DB Instances (p. 828) and Local Time Zone for MariaDB DB Instances (p. 676).	December 21, 2015
New feature	Updated to support Enhanced Monitoring of OS metrics for MySQL and MariaDB instances and Aurora DB clusters. For more information, see Viewing DB Instance Metrics (p. 254).	December 18, 2015
New feature	Updated to support Oracle Standard Edition Two with Bring-Your- Own-License licensing. Also added support for Oracle versions 11.2.0.4.v5, 12.1.0.1.v3, and 12.1.0.2.v2. For more information, see Appendix: Oracle Database Engine Release Notes (p. 1120).	December 14, 2015
New feature	Updated to support db.t2, db.r3, and db.m4 DB instance classes for MySQL version 5.5. For more information, see DB Instance Class (p. 92).	December 4, 2015
New feature	Updated to support modifying the database port for an existing DB instance.	December 3, 2015
New feature	Updated to support three new extensions for PostgreSQL versions 9.3.10 and 9.4.5 DB instances. For more information, see Supported PostgreSQL Database Versions (p. 1147).	December 1, 2015
New feature	Updated to support PostgreSQL versions 9.3.10 and 9.4.5 DB instances. For more information, see Supported PostgreSQL Database Versions (p. 1147).	November 27, 2015
New feature	Updated to support major version upgrades of the database engine for PostgreSQL instances. For more information, see Upgrading the PostgreSQL DB Engine (p. 1191).	November 19, 2015
New feature	Updated to support modifying the public accessibility of an existing DB instance. Updated to support db.m4 standard DB instance classes.	November 11, 2015
New feature	Updated to support manual DB snapshot sharing. For more information, see Sharing a DB Snapshot or DB Cluster Snapshot (p. 230).	October 28, 2015
New feature	Updated to support Microsoft SQL Server 2014 for the Web, Express, and Standard editions.	October 26, 2015

Change	Description	Date Changed
New feature	Updated to support the MySQL-based MariaDB database engine. For more information, see MariaDB on Amazon RDS (p. 666).	October 7, 2015
New feature	Updated to support Amazon Aurora in the Asia Pacific (Tokyo) region. For more information, see Amazon Aurora on Amazon RDS (p. 428).	October 7, 2015
New feature	Updated to support db.t2 burst-capable DB instance classes for all DB engines and the addition of the db.t2.large DB instance class. For more information, see DB Instance Class (p. 92).	September 25, 2015
New feature	Updated to support Oracle DB instances on R3 and T2 DB instance classes. For more information, see DB Instance Class (p. 92).	August 5, 2015
New feature	Updated to support PostgreSQL versions 9.4.4 and 9.3.9. For more information, see Supported PostgreSQL Database Versions (p. 1147).	July 30, 2015
New feature	Microsoft SQL Server Enterprise Edition is now available with the License Included service model. For more information, see License Included (p. 735).	July 29, 2015
New feature	Amazon Aurora has officially released. The Amazon Aurora DB engine supports multiple DB instances in a DB cluster. For detailed information, see Amazon Aurora on Amazon RDS (p. 428).	July 27, 2015
New feature	Updated to support copying tags to DB snapshots.	July 20, 2015
New feature	Updated to support Oracle 12c database version "12.1.0.2", including the In-Memory option, Oracle 11g April PSU patches, and improved integration with AWS CloudHSM.	July 20, 2015
New feature	Updated to support increases in storage size for all DB engines and an increase in Provisioned IOPS for SQL Server.	June 18, 2015
New feature	Updated options for reserved DB instances.	June 15, 2015
New feature	Updated to support Oracle version 12c.	April 2, 2015
New feature	Updated to support PostgreSQL versions 9.3.6 and 9.4.1.	March 18, 2015
New feature	Updated to support using Amazon CloudHSM with Oracle DB instances using TDE.	January 8, 2015
New feature	Updated to support encrypting data at rest and new API version 2014-10-31.	January 6, 2015
New feature	Updated to support Oracle version 11.2.0.4.v3 that includes the PSU released in October 2014.	November 20, 2014

Change	Description	Date Changed
New feature	Updated to include the new Amazon DB engine: Aurora. The Amazon Aurora DB engine supports multiple DB instances in a DB cluster. Amazon Aurora is currently in preview release and is subject to change. For detailed information, see Amazon Aurora on Amazon RDS (p. 428).	November 12, 2014
New feature	Updated to support PostgreSQL Read Replicas.	November 10, 2014
New features	Updated to support Oracle 11.2.0.4v2.	October 16, 2014
New API and features	Updated to support the GP2 storage type and new API version 2014-09-01. Updated to support the ability to copy an existing option or parameter group to create a new option or parameter group.	October 7, 2014
New feature	Updated to support InnoDB Cache Warming for DB instances running MySQL version 5.6.19 and later.	September 3, 2014
New feature	Updated to support SSL certificate verification when connecting to MySQL version 5.6, SQL Server, and PostgreSQL database engines.	August 5, 2014
New feature	Updated to support the db.t2 burst-capable DB instance classes.	August 4, 2014
New feature	Updated to support the db.r3 memory-optimized DB instance classes for use with the MySQL (version 5.6), SQL Server, and PostgreSQL database engines.	May 28, 2014
New feature	Updated to support SQL Server Multi-AZ deployments using SQL Server Mirroring.	May 19, 2014
New feature	Updated to support upgrades from MySQL version 5.5 to version 5.6.	April 23, 2014
New feature	Updated to support Oracle 11.2.0.4.	April 23, 2014
New feature	Updated to support Oracle GoldenGate.	April 3, 2014
New feature	Updated to support the M3 DB instance classes.	February 20, 2014
New feature	Updated to support the Oracle Timezone option.	January 13, 2014
New feature	Updated to support replication between Amazon RDS MySQL DB instances in different regions.	November 26, 2013
New feature	Updated to support the PostgreSQL DB engine.	November 14, 2013
New feature	Updated to support SQL Server transparent data encryption (TDE).	November 7, 2013
New API and new feature	Updated to support cross region DB snapshot copies; new API version, 2013-09-09.	October 31, 2013
New features	Updated to support Oracle Statspack.	September 26, 2013

Change	Description	Date Changed
New features	Updated to support using replication to import or export data between instances of MySQL running in Amazon RDS and instances of MySQL running on-premises or on Amazon EC2.	September 5, 2013
New features	Updated to support the db.cr1.8xlarge DB instance class for MySQL 5.6.	September 4, 2013
New feature	Updated to support replication of Read Replicas.	August 28, 2013
New feature	Updated to support parallel Read Replica creation.	July 22, 2013
New feature	Updated to support fine-grained permissions and tagging for all Amazon RDS resources.	July 8, 2013
New feature	Updated to support MySQL 5.6 for new instances, including support for the MySQL 5.6 memcached interface and binary log access.	July 1, 2013
New feature	Updated to support major version upgrades from MySQL 5.1 to MySQL 5.5.	June 20, 2013
New feature	Updated DB parameter groups to allow expressions for parameter values.	June 20, 2013
New API and new feature	Updated to support Read Replica status; new API version, 2013-05-15.	May 23, 2013
New features	Updated to support Oracle Advanced Security features for native network encryption and Oracle Transparent Data Encryption.	April 18, 2013
New features	Updated to support major version upgrades for SQL Server and additional functionality for Provisioned IOPS.	March 13, 2013
New feature	Updated to support VPC By Default for RDS.	March 11, 2013
New API and feature	Updated to support log access; new API version 2013-02-12	March 4, 2013
New feature	Updated to support RDS event notification subscriptions.	February 4, 2013
New API and feature	Updated to support DB instance renaming and the migration of DB security group members in a VPC to a VPC security group.	January 14, 2013
New feature	Updated for AWS GovCloud (US) support.	December 17, 2012
New feature	Updated to support m1.medium and m1.xlarge DB Instance classes.	November 6, 2012
New feature	Updated to support Read Replica promotion.	October 11, 2012
New feature	Updated to support SSL in Microsoft SQL Server DB Instances.	October 10, 2012
New feature	Updated to support Oracle micro DB Instances.	September 27, 2012
New feature	Updated to support SQL Server 2012.	September 26, 2012

Change	Description	Date Changed
New API and feature	Updated to support provisioned IOPS. API version 2012-09-17.	September 25, 2012
New features	Updated for SQL Server support for DB Instances in VPC and Oracle support for Data Pump.	September 13, 2012
New feature	Updated for support for SQL Server Agent.	August 22, 2012
New feature	Updated for support for tagging of DB Instances.	August 21, 2012
New features	Updated for support for Oracle APEX and XML DB, Oracle time zones, and Oracle DB Instances in a VPC.	August 16, 2012
New features	Updated for support for SQL Server Database Engine Tuning Advisor and Oracle DB Instances in VPC.	July 18, 2012
New feature	Updated for support for option groups and first option, Oracle Enterprise Manager Database Control.	May 29, 2012
New feature	Updated for support for Read Replicas in Amazon Virtual Private Cloud.	May 17, 2012
New feature	Updated for Microsoft SQL Server support.	May 8, 2012
New features	Updated for support for forced failover, Multi-AZ deployment of Oracle DB Instances, and nondefault character sets for Oracle DB Instances.	May 2, 2012
New feature	Updated for Amazon Virtual Private Cloud (VPC) Support.	February 13, 2012
Updated content	Updated for new Reserved Instance types.	December 19, 2011
New feature	Updated for Oracle engine support.	May 23, 2011
Updated content	Console updates.	May 13, 2011
Updated content	Edited content for shortened backup and maintenance windows.	February 28, 2011
New feature	Added support for MySQL 5.5.	January 31, 2011
New feature	Added support for Read Replicas.	October 4, 2010
New feature	Added support for AWS Identity and Access Management (IAM).	September 2, 2010
New feature	Added DB Engine Version Management.	August 16, 2010
New feature	Added Reserved DB Instances.	August 16, 2010
New Feature	Amazon RDS now supports SSL connections to your DB Instances.	June 28, 2010
New Guide	This is the first release of the Amazon Relational Database Service User Guide.	June 7, 2010