

Deploying Microsoft SQL Server on Amazon Web Services

July 2016



© 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Notices

This document is provided for informational purposes only. It represents AWS's current product offerings and practices as of the date of issue of this document, which are subject to change without notice. Customers are responsible for making their own independent assessment of the information in this document and any use of AWS's products or services, each of which is provided "as is" without warranty of any kind, whether express or implied. This document does not create any warranties, representations, contractual commitments, conditions or assurances from AWS, its affiliates, suppliers or licensors. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

Contents

Abstract	5
Microsoft SQL Server Solutions on AWS	5
Amazon RDS for SQL Server	5
SQL Server on Amazon EC2	6
Hybrid Scenarios	6
Choosing Between Microsoft SQL Server Solutions on AWS	7
Amazon RDS for Microsoft SQL Server	8
Starting an Amazon RDS SQL Server DB Instance	10
Security	15
Performance Management	21
High Availability	28
Monitoring and Management	29
Managing Cost	34
Microsoft SQL Server on Amazon EC2	36
Starting a SQL Server Instance on Amazon EC2	36
Security	42
Performance Management	43
High Availability	47
Monitoring and Management	50
Managing Cost	52
Caching	54
Hybrid Scenarios and Data Migration	55
Backups to the Cloud	56
SQL Server Log Shipping—Between On-Premises and Amazon EC2	57

SQL Server AlwaysOn Availability Groups—Between On-Premises and Amazon EC2	58
AWS Database Migration Service	59
Comparison of Microsoft SQL Server Feature Availability on AWS	60
Conclusion	64
Contributors	64
Further Reading	64
Document Revisions	65
Notes	66

Abstract

Amazon Web Services (AWS) is a flexible, cost-effective, easy-to-use cloud computing platform. Relational database management systems are widely deployed within the AWS Cloud. In this whitepaper, we help you understand how to deploy Microsoft SQL Server databases on AWS. You can run SQL Server databases on Amazon Relational Database Service (Amazon RDS) or Amazon Elastic Compute Cloud (Amazon EC2).

The goal of this whitepaper is to explain how you can run SQL Server databases on either Amazon RDS or Amazon EC2, and to give you an understanding of the advantages of each approach. We review in detail how to provision and monitor your SQL Server database, and how to manage scalability, performance, backup and recovery, high availability, and security in both Amazon RDS and Amazon EC2. We also describe how you can set up a disaster recovery solution between an on-premises SQL Server environment and AWS, using native SQL Server features like log shipping, replication, and AlwaysOn Availability Groups. After reading this whitepaper, you should be able to make an educated decision and choose the solution that best fits your needs.

Microsoft SQL Server Solutions on AWS

AWS offers a rich set of features to enable you to run Microsoft SQL Server–based workloads in the cloud. These features offer a variety of controls to effectively manage, scale, and tune SQL Server deployments to match your needs. This whitepaper will discuss these features and controls in greater detail in the following pages.

There are two ways to run SQL Server 2008 R2, 2012, and 2014 in AWS. One is to use Amazon RDS. The other is to run SQL Server on Amazon EC2. The latter option is also available for other versions of SQL Server, subject to Microsoft licensing.

Amazon RDS for SQL Server

Amazon RDS is a service that makes it easier to set up, operate, and scale a relational database in the cloud. Amazon RDS automates installation, disk provisioning and management, patching, minor and major version upgrades,

failed instance replacement, and also backup and recovery of your SQL Server databases. Amazon RDS also offers automated Multi-AZ (Availability Zone) synchronous replication, allowing you to set up a highly available and scalable environment fully managed by AWS.

RDS is a fully managed service. In other words, your database runs on its own database (DB) instance with the compute and storage resources you specify. Because of these advantages, we recommend customers consider Amazon RDS first.

SQL Server on Amazon EC2

Amazon EC2 is a service that provides computing capacity in the cloud. Using Amazon EC2 is similar to running a SQL Server database in-house. You are responsible for administering the database, including backups and recovery, patching the operating system and the database, tuning of the operating system and database parameters, managing security, and configuring high availability or replication. You have full control over the operating system, database installation, and configuration. With Amazon EC2, you can quickly provision and configure DB instances and storage, and you can scale your instances by changing the size of your instances or amount of storage. You can provision your databases in AWS Regions across the world to provide low latency to your end users worldwide. You are responsible for data replication and recovery across your instances in the same or different regions.

Running your own relational database on Amazon EC2 is the ideal scenario if you require a maximum level of control and configurability. You can also use SQL Server services and features that are not available in Amazon RDS.

Hybrid Scenarios

You can also run SQL Server workloads in a hybrid environment. For example, you might have pre-existing commitments on hardware or data center space that makes it impractical to be all in on cloud all at once. Such commitments don't mean you can't take advantage of the scalability, availability, and cost benefits of running a portion of your workload on AWS. Hybrid designs make this possible and can take many forms, from leveraging AWS for long-term SQL backups to running a secondary replica in a SQL Server AlwaysOn Availability Group.

Choosing Between Microsoft SQL Server Solutions on AWS

For SQL Server databases, both Amazon RDS and Amazon EC2 have advantages and certain limitations. Amazon RDS is easier to set up, manage, and maintain. Using RDS can be more cost-effective than running SQL Server in Amazon EC2 and lets you focus on more important tasks, rather than the day-to-day administration of SQL Server and the underlying software stack. Alternatively, running SQL Server in Amazon EC2 gives you more control, flexibility, and choice. Depending on your application and your requirements, you might prefer one over the other.

Start by considering the capabilities and limitations of your proposed solution, as follows:

- Does your workload fit within the features and capabilities offered by Amazon RDS for SQL Server? We will discuss these in greater detail later in this whitepaper.
- Do you need high availability and automated failover capabilities? If you are running a production workload, high availability is a recommended best practice.
- Do you have the resources to manage a cluster on an ongoing basis? These activities include backups, restores, software updates, availability, data durability, optimization, and scaling. Are the same resources better allocated to other business growth activities?

In general, we recommend considering Amazon RDS first. Based on your answers to the considerations preceding, Amazon RDS might be a better choice if the following is true:

- You want to focus on high-level tasks, such as performance tuning and schema optimization, and outsource the following tasks to Amazon: provisioning of the database, management of backup and recovery, management of security patches, upgrades of minor SQL Server versions, and storage management.

- You need a highly available database solution and want to take advantage of the push-button, synchronous Multi-AZ replication offered by Amazon RDS, without having to manually set up and maintain database mirroring, failover clusters, or AlwaysOn Availability Groups.
- You don't want to manage backups and, most importantly, point-in-time recoveries of your database, and prefer that AWS automates and manages these processes.

However, running SQL Server on EC2 might be the better choice if the following is true:

- You need full control over the DB instances, including access to the operating system and software stack.
- You require host access for SQL Server Integration Services (SSIS) packages or bulk insert operations.
- You want your own experienced database administrators managing the databases, including backups, replication, and clustering.
- Your database size and performance needs exceed the current maximums or other limits of Amazon RDS.
- You need to use SQL Server features or options not currently supported by Amazon RDS.

For a detailed side-by-side comparison of SQL Server features available in the AWS environment, see the [Comparison of Microsoft SQL Server Feature Availability on AWS](#) section at the end of this whitepaper.

Amazon RDS for Microsoft SQL Server

Currently, RDS supports the following SQL Server database engines:

- Microsoft SQL Server 2008 R2 SP1 & SP3
- Microsoft SQL Server 2012 RTM SP2 & SP3
- Microsoft SQL Server 2014 RTM

For information on new features in SQL Server 2014 compared to previous versions, see [What's New in SQL Server 2014](#) in the Microsoft TechNet library.¹

Additionally, Amazon RDS for SQL Server supports the following editions of Microsoft SQL Server:

- **Express Edition:** This edition is available at no additional licensing cost for the 2008 R2, 2012, and 2014 versions of SQL Server, and is suitable for small workloads or proof-of-concept deployments. Microsoft limits the amount of memory and size of the individual databases that can be run on the Express edition. This edition is not available in a Multi-AZ deployment.
- **Web Edition:** This edition is available for the 2008 R2, 2012, and 2014 versions and suitable for public and Internet-accessible web workloads. This edition is not available in a Multi-AZ deployment.
- **Standard Edition:** This edition is suitable for most SQL Server workloads, is available for the 2008 R2, 2012 and 2014 versions, and can be deployed in Multi-AZ mode.
- **Enterprise Edition:** This edition is the most feature-rich edition of SQL Server, and can be deployed in Multi-AZ mode. Currently, this edition is not available for SQL Server 2014.

For a detailed feature comparison between the different SQL Server editions, see [Features Supported by the Editions of SQL Server](#) on the Microsoft Developer Network (MSDN) website.²

The capabilities of Amazon RDS for SQL Server are constantly improving, so we recommend checking our website for the latest information on supported SQL Server versions and editions.

In Amazon RDS for SQL Server, the following features and options are supported, depending on the edition of SQL Server:

- Core database engine features
- SQL Server development tools: Visual Studio integration and IntelliSense

- SQL Server management tools: SQL Server Management Studio (SSMS), sqlcmd, SQL Server Profiles (for client side traces), SQL Server Migration Assistant (SSMA), Database Engine Tuning Advisor, and SQL Server Agent
- Safe Common Language Runtime (CLR)
- Service Broker
- Full-text search (except semantic search)
- Secure Sockets Layer (SSL) connection support
- Transparent Data Encryption (TDE)
- Encryption of storage at rest using the AWS Key Management Service (AWS KMS) for all SQL Server license types
- Spatial and location features
- Change tracking
- Database mirroring (used to provide the Multi-AZ capability)
- The ability to use an Amazon RDS SQL DB instance as a data source for reporting, analysis, and integration services

We continuously improve on Amazon RDS for SQL Server capabilities; see the [Microsoft SQL Server on Amazon RDS](#) topic in the *Amazon RDS User Guide* for up-to-date information on supported features and options.³

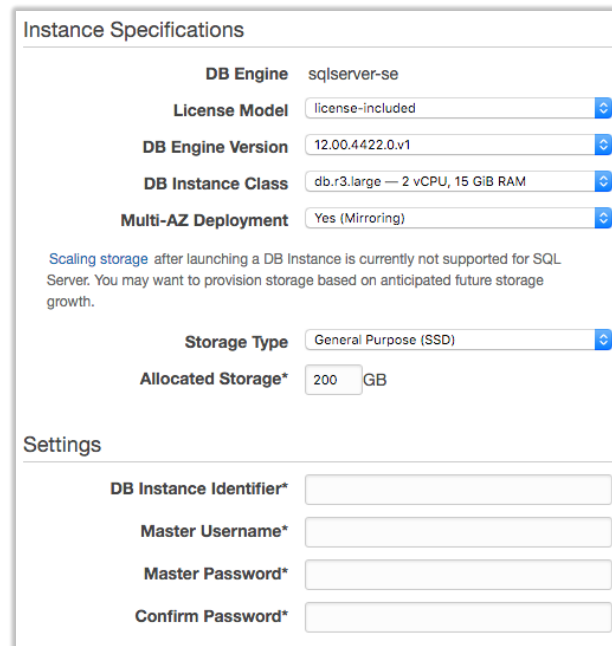
Starting an Amazon RDS SQL Server DB Instance

You can start a SQL Server DB instance on Amazon RDS in several ways. You can do it by using the AWS Management Console, or programmatically using AWS CloudFormation templates, our AWS Command Line Interface (AWS CLI) tools, or AWS SDKs supporting various programming languages. The walkthrough following uses the AWS Management Console for illustration purposes.

Using the AWS Management Console, choose the **RDS** service from the **Services** menu, and choose **Launch a DB Instance** to start the launch wizard.

Choose the **SQL Server** tab on the first screen, choose the edition you wish to deploy, and choose the corresponding **Select** button. If you select SQL Server SE (Standard Edition) or SQL Server EE (Enterprise Edition), a prompt will ask if

you plan to use the RDS instance in a production environment. We recommend always deploying production database workloads with the [Multi-AZ](#) option turned on.



The screenshot displays the 'Instance Specifications' section of the Amazon RDS console. It includes the following configuration options:

- DB Engine:** sqlserver-se
- License Model:** license-included
- DB Engine Version:** 12.00.4422.0.v1
- DB Instance Class:** db.r3.large — 2 vCPU, 15 GiB RAM
- Multi-AZ Deployment:** Yes (Mirroring)
- Storage Type:** General Purpose (SSD)
- Allocated Storage*:** 200 GB

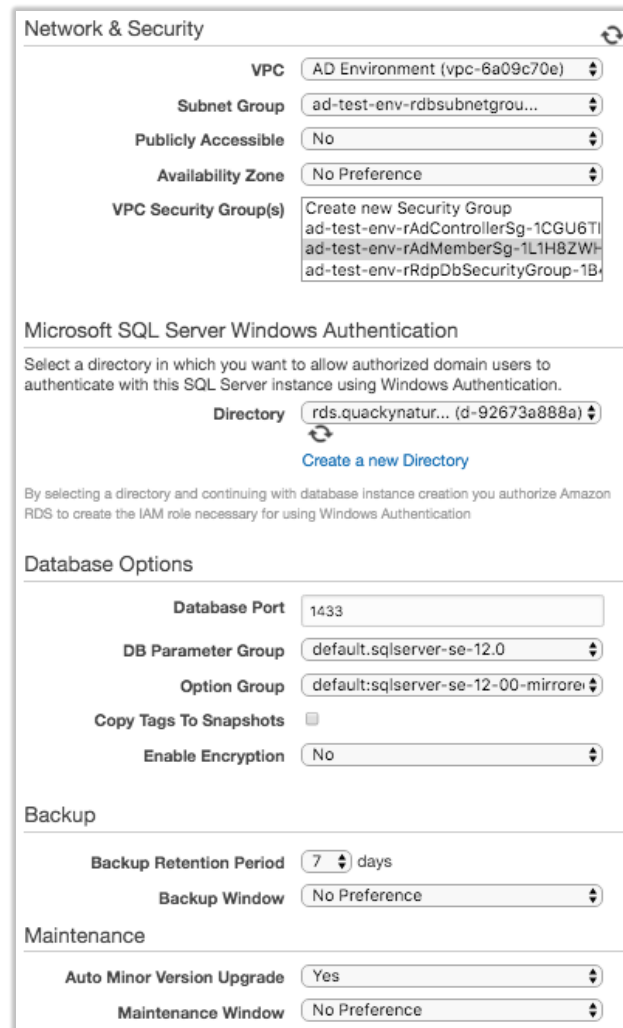
Below the specifications, there is a note: "Scaling storage after launching a DB Instance is currently not supported for SQL Server. You may want to provision storage based on anticipated future storage growth." Under the 'Settings' section, there are four input fields: DB Instance Identifier*, Master Username*, Master Password*, and Confirm Password*.

Figure 1: RDS DB Instance Details

Next, configure the DB instance:

- Choose an option for **License Model**. Customers have only the “license-included” option available for SQL Server Starter Edition and SQL Server Enterprise Edition.
- Choose an option for **DB Engine Version**. This setting corresponds to the versions of SQL Server available (10.50 and later for 2008 R2, 11.00 and later for 2012, or 12.00 and later for 2014). The minor version numbers might change as the engines are updated with new patches.
- Choose an option for **DB Instance Class**. This setting defines the performance characteristics of your DB instance in terms of CPU, RAM, and networking. You can find more details on instance sizing later in this whitepaper.

- Choose a **Multi-AZ Deployment** option. Choosing **Yes**, which is the recommended option for production workloads, will deploy two distinct instances in two separate Availability Zones and mirror the primary DB instance to the secondary. This option is not available for SQL Server Express and SQL Server Web edition, or in regions with fewer than three Availability Zones.
- Set the values for **Storage Type, Allocated Storage, and Provisioned IOPS**, as applicable. These selections determine the amount of database storage, and the performance characteristics of the storage subsystem of the instance. You can set the Provisioned IOPS value only for the Provisioned IOPS storage type. You can find more details on these options later in this whitepaper.
- Set the values for **DB Instance Identifier, Master Username, and Master Password**. Amazon RDS uses SQL Server Authentication for the master user account, and after that account is deployed you can create additional user accounts. You can also use your existing Microsoft Active directory infrastructure to authenticate RDS for SQL Server instances with AWS Directory Service for Microsoft Active Directory (Enterprise Edition).



Network & Security

VPC: AD Environment (vpc-6a09c70e)

Subnet Group: ad-test-env-rdbsubnetgrou...

Publicly Accessible: No

Availability Zone: No Preference

VPC Security Group(s):
 Create new Security Group
 ad-test-env-rAdControllerSg-1CGU6TI
 ad-test-env-rAdMemberSg-1L1H8ZWH
 ad-test-env-rRdpDbSecurityGroup-1B...

Microsoft SQL Server Windows Authentication

Select a directory in which you want to allow authorized domain users to authenticate with this SQL Server instance using Windows Authentication.

Directory: rds.quackynatur... (d-92673a888a)

[Create a new Directory](#)

By selecting a directory and continuing with database instance creation you authorize Amazon RDS to create the IAM role necessary for using Windows Authentication

Database Options

Database Port: 1433

DB Parameter Group: default.sqlserver-se-12.0

Option Group: default.sqlserver-se-12-00-mirrored

Copy Tags To Snapshots:

Enable Encryption: No

Backup

Backup Retention Period: 7 days

Backup Window: No Preference

Maintenance

Auto Minor Version Upgrade: Yes

Maintenance Window: No Preference

Figure 2: RDS DB Instance Settings

Next, configure advanced options for your DB instance:

- Set the **Network & Security** parameters. You can deploy the DB instance in a VPC, or if your account permits, in EC2-Classical. The two networking platforms are described in greater detail in the [Security](#) section following. In a VPC, you also specify the DB subnet group and select a security group to secure access to the instance. These security controls are also discussed in greater detail in the Security section following.
- If needed, join the DB instance to a Microsoft Active Directory domain to use Windows Authentication. You can select the AWS Directory Service

Microsoft AD directory you want to join from the Advanced Configuration Settings while creating your RDS for SQL Server instance. You can find more detailed information on Windows Authentication in the [Data Access Controls](#) section following.

- Set values for the **Database Options**. Advanced DB engine configuration is possible through the use of DB parameter groups and option groups. Additionally, you can also choose to encrypt the instance. RDS provides a default set of configurations that is suitable for most workloads.
- Configure the **Backup and Maintenance** options. You can specify a window of time when Amazon RDS will perform automated backups and maintenance on your instance. You can also opt to have minor DB engine patches and upgrades installed automatically.
- Choose **Launch DB Instance** to have Amazon RDS provision and deploy the DB instance based on the options selected. This process might take several minutes, depending on the size of the instance and the options selected.

Once the instance has been deployed, you can connect to it using SQL Server tools. Amazon RDS provides you with a Domain Name Service (DNS) endpoint for the server, as shown in the example following. *DNS* is a hierarchical distributed naming system that provides translation for numeric IP addresses associated to resources connected to the Internet or private networks.⁴ You use this DNS endpoint as the SQL Server hostname, along with the master user name and password configured for the instance, to connect to the database. Always use the DNS endpoint to connect to the instance, because the underlying IP address might change.

The screenshot shows the AWS Management Console interface for an Amazon RDS DB Instance. The instance is named 'SQL Server SE' and is in an 'available' state. The console displays various configuration details, security and network settings, instance and IOPS information, encryption details, availability and durability settings, and maintenance details.

Configuration Details		Security and Network	
Engine	SQL Server SE 12.00.4422.0.v1	Availability Zone	us-west-2a
License Model	License Included	VPC	AD Environment (vpc-6a09c70e)
Created Time	April 1, 2016 at 1:11:27 PM UTC-7	Subnet Group	ad-test-env-rdbsubnetgroup-1cpv9tfusutu (Complete)
DB Name		Subnets	subnet-6563b501 subnet-09dda50 subnet-dbbd57ad
Username	admin	Security Groups	ad-test-env-rRdpDbSecurityGroup-1B4JR8WEUJLXF (sg-febac099) (active)
Option Group	default:sqlserver-se-12-00-mirrored (in-sync)	Publicly Accessible	No
Parameter Group	default:sqlserver-se-12.0 (in-sync)	Endpoint	rdedemowinauth.clyx9xcjzmjh.us-west-2.rds.amazonaws.com
Copy Tags To Snapshots	No	Port	1433
		Certificate Authority	rds-ca-2015 (Mar 5, 2020)
		Directory	rds.quackynature.com (d-92673a880a)
		Directory Status	joined

Instance and IOPS	
Instance Class	db.r3.large
Storage Type	General Purpose (SSD)
IOPS	disabled
Storage	200 GB

Encryption Details	Availability and Durability	Maintenance Details
Encryption Enabled	No	DB Instance Status
		available
		Multi AZ
		Yes (Mirroring)
		Secondary Zone
		us-west-2c
		Automated Backups
		Enabled (7 Days)
		Latest Restore Time
		April 6, 2016 at 12:26:01 PM UTC-7
		Auto Minor Version Upgrade
		Yes
		Maintenance Window
		fr:09:02-fr:09:32
		Backup Window
		07:18-07:48
		Pending Maintenance
		None

Figure 3: RDS DB Instance Properties

Security

You can use several features and sets of controls to manage the security of your Amazon RDS DB instance. These controls are as follows:

- Network controls, which determine the network configuration underlying your DB instance
- DB instance access controls, which determine administrative and management access to your RDS resources
- Data access controls, which determine access to the data stored in your RDS DB instance databases
- Data-at-rest protection, which affects the security of the data stored in your RDS DB instance
- Data-in-transit protection, which affects the security of data connections to and from your RDS DB instance

Network Controls

At the network layer, controls differ slightly based on whether the instance is deployed in EC2-VPC or EC2-Classic:

- EC2-VPC allows you to define a private, isolated section of the AWS Cloud and launch resources within it. You define the network topology, the IP addressing scheme, and the routing and traffic access control patterns. Newer AWS accounts have access only to this networking platform.
- EC2-Classic allows you to launch resources within a predefined, multitenant network platform. You can control access at the resource or instance level.

Amazon Virtual Private Cloud (Amazon VPC) ClassicLink links existing EC2-Classic resources to a VPC. We recommend that you deploy RDS DB instances in a VPC for the added richness of security controls and use ClassicLink to establish connectivity with existing EC2-Classic resources. For more information, see [ClassicLink](#) in the *Amazon EC2 User Guide*.⁵

In EC2-VPC, *DB subnet groups* are also a security control. They allow you to narrowly control the subnets in which Amazon RDS is allowed to deploy your DB instance. You can control the flow of network traffic between subnets using route tables and network access control lists (NACLs) for stateless filtering. You can designate certain subnets specifically for database workloads, without default routes to the Internet. You can also deny nondatabase traffic at the subnet level to reduce the exposure footprint for these instances.

Security groups are used to filter traffic at the instance level. Security groups act like a stateful firewall, similar in effect to host-based firewalls such as the Microsoft Windows Server Firewall. The rules of a security group define what traffic is allowed to enter the instance (inbound) and what traffic is allowed to exit the instance (outbound). VPC security groups are used for DB instances deployed in a VPC. They can be changed and reassigned without restarting the instances associated with them. DB security groups are used for instances in EC2-Classic. These cannot be reassigned without restarting the instance and cannot filter outbound traffic.

To improve security, we recommend restricting inbound traffic to only database-related traffic (port 1433, unless a custom port number is used) and only traffic

from known sources. Security groups can also accept the ID of a different security group (called the source security group) as the source for traffic. This approach makes it easier to manage sources of traffic to your RDS DB instance in a scalable way. In this case, you don't have to update the security group every time a new server needs to connect to your DB instance; you just have to assign the source security group to it.

Amazon RDS can make DB instances publicly accessible by assigning Internet routable IP addresses to the instances. In most use cases, this approach is not needed or desired, and we recommend setting this option to **No** to limit the potential threat. In cases where direct access to the database over the public Internet is needed, we recommend limiting the sources that can connect to the DB instance to known hosts by using their IP addresses. For this option to be effective, the instance must be launched in a subnet that permits public access and the security groups and NACLs must permit ingress traffic from those sources.

DB instances that are exposed publicly over the Internet and have open security groups, accepting traffic from any source, might be subject to more frequent patching. Such instances can be force-patched when security patches are made available by the vendors involved. This patching can occur even outside the defined DB instance maintenance window, to ensure the safety and integrity of customer resources and our infrastructure. Although there are many ways to secure your databases, we recommend VPC so that your RDS databases are protected with no public routes.

DB Instance Access Controls

Using AWS Identity and Access Management (IAM), you can manage access to your Amazon RDS databases. For example, you can authorize administrative users under your AWS account (or deny them the ability) to create, describe, modify, or delete an Amazon RDS database. You can also enforce multifactor authentication (MFA). For more information on using IAM to manage administrative access to Amazon RDS, see [Authentication and Access Control for Amazon RDS](#) in the *Amazon RDS User Guide*.⁶

Data Access Controls

Amazon RDS for SQL Server supports both SQL Authentication and Windows Authentication, and access control for authenticated users should be configured

using the principle of least privilege. A master account is created automatically when an instance is launched. This master user is put in the `db_owner` group. This login is typically used for administrative purposes only and is granted the roles of `processadmin`, `setupadmin`, and `public`. RDS manages the master user as a login, and creates a user linked to the login in each customer database.

You can create additional users and databases after launch by connecting to the DB instance using the tool of your choice (for example, SQL Server Management Studio). These users should be assigned only the permissions needed for the workload or application that they are supporting to operate correctly. For example, if you as the master user create a user X who then goes ahead and creates a database, user X will be in `db_owner` for this new database and not the master user. Later on, if you reset the master password, the master user will be added to `db_owner` for this new database.

You can also integrate with your existing identity infrastructure based on Microsoft Active Directory and authenticate against Amazon RDS for SQL Server databases using the Windows Authentication method. Using Windows Authentication allows you to keep a single set of credentials for all your users and save time and effort by not having to update these credentials in multiple places.

To use the Windows Authentication method with your Amazon RDS for SQL Server DB instance, sign up for the AWS Directory Service for Microsoft Active Directory (Enterprise Edition). If you don't already have a directory running, you can create a new one. You can then associate directories with both new and existing DB instances.

You can use Active Directory to manage users and groups with access privileges to your SQL Server DB instance, and also join other EC2 instances to that domain. You can also establish a one-way forest trust from an external, existing Active Directory deployment to the directory managed by AWS Directory Service. Doing so will give you the ability to authenticate already existing Active Directory users and groups you have established in your organization with RDS SQL Server instances.

You can also create SQL Server Windows logins on domain-joined DB instances for users and groups in your directory domain, or the trusted domain if

applicable. Logins can be added using a SQL client tool such as SQL Server Management Studio using the following command.

```
CREATE LOGIN [<user or group>] FROM WINDOWS WITH  
DEFAULT_DATABASE = [master], DEFAULT_LANGUAGE =  
[us_english];
```

More information on configuring Windows Authentication with Amazon RDS for SQL Server can be found in the [Using Windows Authentication](#) topic in the *Amazon RDS User Guide*.⁷

Unsupported SQL Server Roles and Permissions in Amazon RDS

The following server-level roles are not currently available in Amazon RDS: bulkadmin, dbcreator, diskadmin, securityadmin, serveradmin and sysadmin

Also, the following server-level permissions are not available on a SQL Server DB instance:

- ADMINISTER BULK OPERATIONS
- ALTER ANY CREDENTIAL
- ALTER ANY EVENT NOTIFICATION
- ALTER ANY SERVER AUDIT
- ALTER RESOURCES
- ALTER SETTINGS (you can use the DB parameter group API actions to modify parameters)
- AUTHENTICATE 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)

Data-at-Rest Protection

Amazon RDS for SQL Server supports the encryption of DB instances with encryption keys managed in AWS KMS. Data that is encrypted at rest includes the underlying storage for a DB instance, its automated backups, and snapshots. You can also encrypt existing DB instances and share encrypted snapshots with other accounts within the same region.

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.

Amazon RDS encrypted instances also help secure 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. To manage the keys used for encrypting and decrypting your Amazon RDS resources, use AWS KMS.

Amazon RDS also supports encryption of data at rest using the TDE feature of SQL Server. This feature is only available in the Enterprise Edition. You can enable TDE by setting up a custom option group with the TDE option enabled (if such a group doesn't already exist), and then associating the DB instance with that group. You can find more details on Amazon RDS support for TDE on the [Options for the Microsoft SQL Server Database Engine](#) topic in the *Amazon RDS User Guide*.⁸

If full data encryption is not feasible, or not desired for your workload, you can selectively encrypt table data using SQL Server column-level encryption, or by encrypting data in the application before it is saved to the DB instance.

Data-in-Transit Protection

Amazon RDS for SQL Server fully supports encrypted connections to the DB instances using SSL. SSL support is available in all AWS Regions for all supported SQL Server editions. Amazon RDS creates an SSL certificate for your SQL Server DB instance when the instance is created. The SSL certificate includes the DB instance endpoint as the Common Name (CN) for the SSL certificate to help guard against spoofing attacks. You can find more details on how to use SSL encryption in [Using SSL with a Microsoft SQL Server DB Instance](#) in the *Amazon RDS User Guide*.⁹

Performance Management

The performance of your SQL Server DB instance is determined primarily by your workload. Depending on your workload, you need to select the right instance type, which affects the compute capacity, amount of memory, and network capacity available to your database. Instance type is also determined by the storage size and type you select when you provision the database.

Instance Sizing

The amount of memory and compute capacity available to your Amazon RDS for SQL Server instance is determined by its DB instance class. Amazon RDS offers a range of DB instance classes to run SQL Server, from 1 vCPU and 1 GB of memory to 32 vCPUs and 244 GB of memory. Not all DB instance classes are available for all SQL Server editions, however. DB instance class availability also varies based on the version and licensing model selected.

At the time of this writing, Amazon RDS for SQL Server supports the following DB instance classes for the various SQL Server editions.

DB Instance Class	vCPU Count	Memory (GB)	Network Performance	SQL Server Express Edition	SQL Server Web Edition	SQL Server Standard Edition	SQL Server Enterprise Edition
db.t2.micro	1	1	Low	Yes			
db.t2.small	1	2	Low	Yes	Yes		Yes
db.t2.medium	2	4	Moderate	Yes	Yes		Yes
db.m4.large	2	6.5	Moderate		Yes	Yes	Yes
db.m4.xlarge	4	13	High		Yes	Yes	Yes
db.m4.2xlarge	8	25.5	High		Yes	Yes	Yes
db.m4.4xlarge	16	53.5	High		Yes	Yes	Yes
db.r3.large	2	15	Moderate		Yes	Yes	Yes
db.r3.xlarge	4	30.5	Moderate		Yes	Yes	Yes
db.r3.2xlarge	8	61	High		Yes	Yes	Yes
db.r3.4xlarge	16	122	High			Yes	Yes
db.r3.8xlarge	32	244	10 Gbps			Yes	Yes
db.m3.medium*	1	3.75	Moderate		Yes	Yes	Yes
db.m3.large*	1	3.75	Moderate		Yes	Yes	Yes
db.m3.xlarge*	4	15	Moderate		Yes	Yes	Yes
db.m3.2xlarge*	8	30	High		Yes	Yes	Yes
db.m2.xlarge*	2	17.1	Moderate		Yes	Yes	Yes
db.m2.2xlarge*	4	34	Moderate		Yes	Yes	Yes
db.m2.4xlarge*	8	68	High		Yes	Yes	Yes
db.t1.micro*	1	0.615	Very Low	Yes			
db.m1.small*	1	1.7	Very Low	Yes	Yes	Yes	Yes
db.m1.medium*	1	3.75	Moderate		Yes	Yes	Yes
db.m1.large*	2	7.5	Moderate		Yes	Yes	Yes
db.m1.xlarge*	4	15	High		Yes	Yes	Yes

* These DB instance classes are previous-generation instance classes and are superseded in terms of both cost-effectiveness and performance by the current generation classes.

Understanding the performance characteristics of your workload is important when identifying the proper DB instance class. If you are unsure how much CPU you need, we recommend that you start with the smallest appropriate DB instance class, then monitor CPU utilization using Amazon CloudWatch. You can modify the DB instance class for an existing Amazon RDS for SQL Server instance, allowing the flexibility to scale up or scale down the instance size depending on the performance characteristics required.

To modify an SQL Server DB instance, sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>. In the navigation pane, choose **Instances**. Select the check box for the DB instance that you want to change, and choose **Modify**.

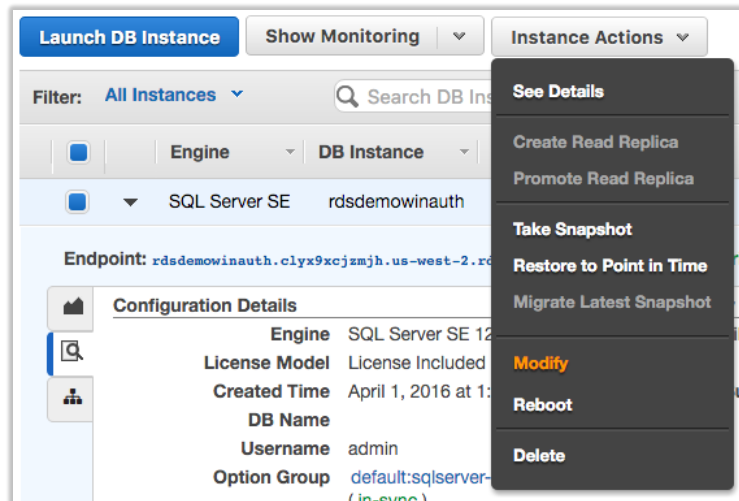


Figure 4: The Modify DB Instance Option

The settings are as follows, and similar to the ones you configure when launching a new DB instance.

Setting	Description
DB Engine Version	In the list provided, choose the version of the SQL Server database engine that you want to use.
DB Instance Class	In the list provided, choose the DB instance class that you want to use.
Multi-AZ Deployment	If you want to create a secondary mirror of your DB instance in another Availability Zone, choose Yes; otherwise, choose No.
Storage Type	You cannot change the storage type for an existing SQL Server DB

Setting	Description
	instance.
Allocated Storage	You cannot change the allocated storage for a SQL Server DB instance.
DB Instance Identifier	You can rename the DB instance by typing a new name. When you change the DB instance identifier, an instance reboot will occur immediately if you set Apply Immediately to true, or will occur during the next maintenance window if you set Apply Immediately to false. This value is stored as a lowercase string.
New Master Password	Type a password that contains from 8 to 128 printable ASCII characters (excluding /, ", a space, and @) for your master user password.
Security Group	Select the security group you want associated with the DB instance.
Certificate Authority	Certificate authority to use for issuing the SSL certificates used for encrypted connections to the database.
Publicly Accessible	Whether the DB instance is accessible from the public internet, and has a public IP address assigned, or not. Review the best practices discussed in the Network Controls section for more details.
Directory	Select a directory in which you want to allow authorized domain users to authenticate with this SQL Server instance using Windows Authentication.
Database Port	Change this setting if you wish to use a different listener port, other than the default 1433 port used by SQL Server.
DB Parameter Group	Select the parameter group you want associated with the DB instance. Changing this setting does not result in an outage. The parameter group name itself is changed immediately, but the actual parameter changes are not applied until you reboot the instance without failover. The DB instance will not be rebooted automatically and the parameter changes will not be applied during the next maintenance window.
Option Group	Select the option group you want associated with the DB instance.
Copy Tags to Snapshots	Whether tags assigned to the DB instance should also be copied over to the snapshots of the DB instance.
Backup Retention Period	Specify the number of days that automatic backups will be retained. To disable automatic backups, set this value to 0. An immediate outage will occur if you change the backup retention period from 0 to a nonzero value or from a nonzero value to 0.
Backup Window	Set the time range during which automated backups of your databases will occur. Specify a start time in Universal Coordinated Time (UTC), and a duration in hours.
Auto Minor Version Upgrade	If you want your DB instance to receive minor engine version upgrades automatically when they become available, choose Yes. Upgrades are installed only during your scheduled maintenance window.
Maintenance Window	Set the time range during which system maintenance, including upgrades, will occur. Specify a start time in UTC and a duration in hours.

By default, changes (including a change to the DB instance class) are applied during the next specified maintenance window. Alternatively, you can use the `apply-immediately` flag to apply the changes immediately.

Disk I/O Management

Amazon RDS for SQL Server simplifies the allocation and management of database storage for instances. You decide the type and amount of storage to use, and also the level of provisioned I/O performance if applicable. Neither the amount of storage or provisioned I/O can be changed on an RDS SQL Server DB instance after the instance has been deployed. This functionality is due to extensibility limitations of striped storage attached to a Windows Server environment.

We recommend that you plan for the growth of storage needs in advance, and provision enough capacity to handle growth from the onset. You can always provision a new DB instance with increased storage, then perform a data migration to the new instance. However, this option might require a window of time during which your database is offline.

Amazon RDS for SQL Server supports three types of storage, each having different characteristics and recommended use cases:

- **Magnetic Storage** (also known as standard storage) uses magnetic disks for storing volume data. It is a cost-effective storage option for applications with light or burst I/O requirements. Performance and burst capabilities can vary greatly depending on the demands placed on shared resources by other customers, which makes this storage type less predictable in terms of performance.
- **General Purpose (SSD)** (also called GP2) is an SSD-backed storage solution with predictable performance and burst capabilities. This option is suitable for workloads that run in larger batches, such as nightly report processing. Credits are replenished while the instance is largely idle, and are then available for bursts of batch jobs.

- Provisioned IOPS storage (or PIOPS storage) is designed to meet the needs of I/O-intensive workloads that are sensitive to storage performance and consistency in random access I/O throughput.

The following table compares the Amazon RDS storage performance characteristics.

Storage Type	Min. Volume Size	Max. Volume Size	Baseline Performance	Burst Capability	Storage Technology	Pricing Criteria
Magnetic Storage	20 GiB	1 TiB*	About 100 IOPS	Yes; several hundred IOPS	Magnetic Disk	Allocated storage and I/O operations
General Purpose	20 GiB (100 GiB recommended)	4 TiB*	3 IOPS/GiB	Yes; up to 3000 IOPS per volume, subject to accrued credits	SSD	Allocated storage
Provisioned IOPS	100 GiB (200 GiB for Standard Edition)	4 TiB*	10 IOPS/GiB up to max. 20,000 IOPS	No; fixed allocation	SSD	Allocated storage and provisioned IOPS

* SQL Server Express Edition limits storage to 10 GB for each database. For example, on RDS you can create up to 30 databases for each instance, effectively limiting you to 300 GB for each instance. Any additional storage on your instance will be unusable.

Although performance characteristics of DB instances change over time as technology and capabilities improve, there are several metrics that can be used to assess performance and help plan deployments. Different workloads and query patterns affect these metrics in different ways, making it difficult to establish a practical baseline reference in a typical environment. We therefore recommend that you test your own workload to determine how these metrics behave in your specific use case.

For Amazon RDS, we provision and measure I/O performance in units of input/output operations per second (IOPS). We count each I/O operation per second that is 256 KiB or smaller as one IOPS.

The average queue depth, a metric available through Amazon CloudWatch, tracks the number of I/O requests in the queue that are waiting to be serviced. These requests have been submitted by the application but haven't been sent to the storage device because the device is busy servicing other I/O requests. Time spent in the queue increases I/O latency, and large queue sizes can indicate an overloaded system from a storage perspective.

Additionally, subject to the DB instance class selected, the current *maximum channel bandwidth* available for storage in Amazon RDS is 4000 Mbps full duplex.¹⁰ This value equates to about 210 MiB/second in each direction. Thus, a perfectly balanced workload, where database queries are evenly split between 50 percent data read operations and 50 percent data write operations, can attain a maximum combined throughput of about 420 MiB/second.

As a result, depending on the storage configuration selected, your overall storage subsystem throughput will be limited either by the maximum IOPS or the maximum channel bandwidth at any time. If your workload is generating a lot of very small sized I/O operations (for example, 8 KiB), you are likely to reach maximum IOPS before the overall bandwidth reaches the channel maximum. However, if I/O operations are large in size (for example, 256 KiB), you might reach the maximum channel bandwidth before maximum IOPS.

As specified in Microsoft documentation, SQL Server stores data in 8 KiB pages, but uses a complex set of techniques to optimize I/O patterns, with the general effect of reducing the number of I/O requests and increasing the I/O request size.¹¹ This approach results in better performance by reading and writing multiple pages at the same time. Amazon RDS accommodates these multipage operations by counting every read or write operation on up to 32 pages as a single I/O operation to the storage system, based on the variable size of IOPS.

SQL Server also attempts to optimize I/O by reading ahead and attempting to keep the queue length nonzero. Therefore, queue depth values that are very low or zero indicate that the storage subsystem is underutilized, and potentially overprovisioned from a I/O capacity perspective.

Using small storage sizes with General Purpose SSD storage can also have a detrimental impact on DB instance performance. If your storage size needs are low, you must ensure that the storage subsystem provides enough I/O performance to match your workload needs. Because IOPS are allocated on a ratio of 3 IOPS for each 1 GB of allocated storage, small storage sizes will also provide small amounts of baseline IOPS. When created, each DB instance comes with an initial allocation of I/O credits. This allocation provides for burst capabilities of up to 3000 IOPS from the start. Once the initial burst credits allocation is exhausted, you must ensure that your ongoing workload needs fit within the baseline I/O performance of the storage size selected.

High Availability

Amazon RDS offers Multi-AZ support for Amazon RDS for SQL Server. This high availability option uses SQL Server Mirroring technology with additional improvements to meet the requirements of enterprise-grade production workloads running on SQL Server. The Multi-AZ deployment option provides enhanced availability and data durability by automatically replicating database updates between two AWS Availability Zones. *Availability Zones* are physically separate locations with independent infrastructure engineered to be insulated from failures in other Availability Zones.

When you create or modify your SQL Server DB instance to run using Multi-AZ, Amazon RDS will automatically provision a primary database in one Availability Zone and maintain a synchronous secondary replica in a different Availability Zone. In the event of planned database maintenance or unplanned service disruption, Amazon RDS will automatically fail over the SQL Server database to the up-to-date secondary so that database operations can resume quickly without any manual intervention.

If an Availability Zone failure or DB instance failure occurs, your availability impact is limited to the time that automatic failover takes to complete, typically 60 seconds. When failing over, Amazon RDS simply flips the canonical name record (CNAME) for your DB instance to point to the secondary, which is in turn promoted to become the new primary. The canonical name record (or endpoint name) is an entry in DNS.

We recommend that you implement retry logic for database connection errors in your application layer by using the canonical name rather than attempt to connect directly to the IP address of the DB instance. We recommend this approach because during a failover the underlying IP address will change to reflect the new primary DB instance.

Amazon RDS automatically performs a failover in the event of any of the following:

- Loss of availability in the primary Availability Zone
- Loss of network connectivity to the primary DB node
- Compute unit failure on the primary DB node
- Storage failure on the primary DB node

Amazon RDS Multi-AZ deployments don't fail over automatically in response to database operations such as long running queries, deadlocks, or database corruption errors. For example, suppose that a customer workload causes high resource usage on a DB instance and that SQL Server times out and triggers failover of individual databases. In this case, RDS recovers the failed databases back to the primary instance.

When operations such as DB instance scaling or system upgrades like OS patching are initiated for Multi-AZ deployments, they are applied first on the secondary instance, prior to the automatic failover of the primary instance, for enhanced availability.

Due to failover optimization of SQL Server, certain workloads can generate greater I/O load on the mirror than on the principal. This functionality can result in higher IOPS on the secondary instance. We therefore recommend that you consider the maximum IOPS needs of both the primary and secondary when provisioning the storage type and IOPS of your RDS DB instance.

Monitoring and Management

Amazon CloudWatch collects many Amazon RDS-specific metrics. You can look at these metrics using the AWS Management Console, the AWS CLI (using the `mon-get-stats` command), or the AWS API. In addition to the system-level

metrics collected for Amazon EC2 instances (such as CPU usage, disk I/O, and network I/O), the Amazon RDS metrics include many database-specific metrics, such as database connections, free storage space, read and write I/O per second, read and write latency, read and write throughput, and available RAM. For a full, up-to-date list, see [Amazon RDS Dimensions and Metrics](#) in the *Amazon CloudWatch Developer Guide*.¹²

In Amazon CloudWatch, you can also configure alarms on these metrics to trigger notifications when the state changes. 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. Notifications are sent to Amazon Simple Notification Service (Amazon SNS) topics or Auto Scaling policies. You can configure these alarms to notify database administrators by email or Amazon Simple Message Service (Amazon SMS) when they get triggered. You can also use notifications as triggers for custom automated response mechanisms or workflows that react to alarm events; however, you need to implement such event handlers separately.

Amazon RDS for SQL Server also supports Enhanced Monitoring. Amazon RDS provides metrics in near 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. Enhanced Monitoring gathers its metrics from an agent on the instance.

Enhanced Monitoring gives you deeper visibility into the health of your Amazon RDS instances in near real time, providing a comprehensive set of 26 new system metrics and aggregated process information, at a detail level of up to 1 second. These monitoring metrics cover a wide range of instance aspects, such as the following:

- General metrics, like uptime, instance, and engine version
- CPU utilization, such as idle, kernel, or user time percentage
- Disk subsystem metrics, including utilization, read and write bytes, and number of I/O operations
- Network metrics, like interface throughput and read and write bytes

- Memory utilization and availability, including physical, kernel, commit charge, system cache, and SQL Server footprint
- System metrics, consisting of number of handles, processes, and threads
- Process list information, grouped by OS processes, RDS processes (management, monitoring, diagnostics agents), and RDS child processes (SQL Server workloads)

Because Enhanced Monitoring delivers metrics to CloudWatch Logs, this feature will incur standard CloudWatch Logs charges. These charges depend on a number of factors:

- The number of DB instances sending metrics to CloudWatch Logs
- The level of detail of metrics sampling—finer detail results in more metrics being delivered to CloudWatch Logs
- The workload running on the DB instance—more compute-intensive workloads have more OS process activity to report

More information and instructions on how to enable the feature can be found in [Viewing DB Instance Metrics](#) in the *Amazon RDS User Guide*.¹³

In addition to CloudWatch metrics, you can use the native SQL Server performance monitoring tools such as dynamic management views, the SQL Server error log, and both client and server-side SQL Server Profiler traces.

Amazon RDS for SQL Server provides two administrative windows of time designed for effective management, described following. The service will assign default time windows to each DB instance, if these aren't customized.

- **Backup window:** The backup window is the period of time during which your DB instance is going to be backed up. Because backups might have a small performance impact on the operation of the instance, we recommend you set the window for a time when this will have minimal impact on your workload.
- **Maintenance window:** The maintenance window is the period of time during which DB instance modifications (such as implementing pending changes to storage or CPU class for the DB instance) and software patching

occur. Your instance might be restarted during this window, if there is a scheduled activity pending and that activity requires a restart, but that is not always the case. We recommend scheduling the maintenance window for a time when your instance has the least traffic, or a potential restart is least disruptive.

Amazon RDS for SQL Server comes with several built-in management features:

- **Automated backup and recovery.** Amazon RDS automatically backs up all of your DB instances. You can set the backup retention period when you create a DB instance. If you don't set the backup retention period, Amazon RDS uses a default retention period of one day. You can modify the backup retention period; valid values are 0 (for no backup retention) to a maximum of 35 days. Automated backups occur daily during the backup window. Amazon RDS uses these periodic data backups in conjunction with your transaction logs to enable you to restore your DB instance to any second during your retention period, up to the `LatestRestorableTime`, typically up to the last 5 minutes.
- **Push-button scaling.** With a few clicks, you can change the DB instance class to increase or decrease the size of your instance's compute capacity, network capacity, and memory. You can choose to make the change immediately, or schedule it for your next maintenance window.
- **Automatic host replacement.** Amazon RDS automatically replaces the compute instance powering your deployment in the event of a hardware failure.
- **Automatic minor version upgrade.** Amazon RDS keeps your database software up to date. You have full control on whether Amazon RDS will deploy such patching automatically, and you can disable this option to prevent that. Regardless of this setting, publicly accessible DB instances with open security groups might be force-patched when security patches are made available by vendors, to ensure the safety and integrity of customer resources and our infrastructure. The patching activity occurs during the weekly, 30-minute maintenance window that you specify when you provision your database (and that you can alter at any time). Such patching occurs infrequently, and your database might become unavailable during part of your maintenance window when a patch is applied. You can minimize the downtime associated with automatic patching if you run in

Multi-AZ mode. In this case, the maintenance is generally performed on the secondary instance. When it is complete, the secondary instance is promoted to primary. The maintenance is then performed on the old primary, which becomes the secondary.

- Preconfigured parameters and options. Amazon RDS provides a default set of DB parameter groups and also option groups for each SQL Server edition and version. These groups contain configuration parameters and options respectively, which allow you to tune the performance and features of your DB instance. By default, Amazon RDS provides an optimal configuration set suitable for most workloads, based on the class of the DB instance that you selected. You can create your own DB parameter and option groups to further tune the performance and features of your DB instance.

You can administer Amazon RDS for SQL Server databases using the same tools you use with on-premises SQL Server instances, such as SQL Server Management Studio.

However, to provide you with a more secure and stable managed database experience, Amazon RDS doesn't provide desktop or administrator access to DB instances, and it restricts access to certain system procedures and tables that require advanced privileges.

Commands to create users, rename users, grant, revoke permissions and set passwords work as they do in Amazon EC2 (or on-premises) databases. The administrative commands that RDS doesn't support are listed in the [Unsupported SQL Server Roles and Permissions in Amazon RDS](#) section earlier in this paper. You can create and manage up to 30 databases on a single DB instance.

Even though direct, file-system level access to the RDS SQL Server DB instance is not available, you can always migrate your data out of RDS instances. Tools like the Microsoft SQL Server Database Publishing Wizard can be used to download the contents of your databases into flat T-SQL files. You can then load these files into any other SQL Server instances, or store them as backups in Amazon Simple Storage Service (Amazon S3) or Amazon Glacier, or on-premises. In addition, you can use the [AWS Database Migration Service](#) to move data to and from RDS.

Managing Cost

Managing the cost of the IT infrastructure is often an important driver for cloud adoption. AWS makes running SQL Server on Amazon a cost-effective proposition by providing a flexible, scalable environment and pricing models that allow you to pay for only the capacity you consume at any given time. Amazon RDS further reduces your costs by reducing the management and administration tasks that you have to perform.

Generally, the cost of operating an Amazon RDS DB instance depends on the following factors:

- The AWS Region the instance is deployed in
- The instance class and storage type selected for the DB instance
- The Multi-AZ mode of the instance
- The pricing model
- How long the instance is running during a given billing period

You can optimize the operating costs of your RDS workloads by controlling the factors listed preceding.

AWS services are available in multiple regions across the world. In regions where our costs of operating our services are lower, we pass those savings on to you. Thus, Amazon RDS hourly prices for the different instance classes vary by the region. If you have the flexibility to deploy your SQL Server workloads in multiple regions, the potential savings from operating in one region as compared to another can be an important factor in choosing the right region.

Amazon RDS also offers different pricing models to match different customer needs:

- On-demand pricing allows you to pay for Amazon RDS DB instances by the hour, with no term commitments. You incur a charge for each hour a given DB instance is running. If your workload doesn't need to run 24/7, or you are deploying temporary databases for staging, testing, or development purposes, on-demand pricing can offer significant advantages.

- Reserved Instances (RI) allow you to lower costs and reserve capacity. Reserved Instances can save you up to 60 percent over On-Demand rates when used in steady state, which tend to be the case for many databases. They can be purchased for 1- or 3-year terms. If your SQL Server database is going to be running more than 25 percent of the time each month, you will most likely financially benefit from using a Reserved Instance.

Overall savings are greater when committing to a 3-year term, compared to running the same workload using on-demand pricing for the same period of time. However, the length of the term needs to be balanced against projections of growth, because the commitment is for a specific instance class. If you expect that your compute and memory needs are going to grow over time for a given DB instance, you might want to opt for a shorter 1-year term and weigh the savings from the Reserved Instance against the overhead of being over-provisioned for some part of that term.

The following pricing options are available for RDS reserved instances:

- With All Upfront Reserved Instances, you pay for the entire Reserved Instance with one upfront payment. This option provides you with the largest discount compared to On-Demand Instance pricing.
- With Partial Upfront Reserved Instances, you make a low upfront payment and are then charged a discounted hourly rate for the instance for the duration of the Reserved Instance term.
- With No Upfront Reserved Instances, you don't make any upfront payments, but are charged a discounted hourly rate for the instance for the duration of the Reserved Instance term. This option still provides you with a significant discount compared to On-Demand Instance pricing, but the discount is usually less than for the other two reserved instance pricing options.

Note that unlike in Amazon EC2, in Amazon RDS you cannot issue a stop command to a DB instance and keep the instance in a stopped state to avoid incurring compute charges. Instead, you can terminate the instance, take a final snapshot prior to termination, and recreate a new Amazon RDS instance from the snapshot when you need it.

Additionally, you can use several other strategies to help optimize costs:

- Use the License Included model to save on SQL Server licenses where the license is included in the Amazon RDS hourly cost. This approach is especially effective for databases that are not running 24/7 and for short projects.
- Terminate DB instances with a last snapshot when they are not needed, then reprovision them from that snapshot when they need to be used again. For example, some development and test databases can be terminated at night and on weekends, and reprovisioned on weekdays in the morning.
- Scale down the size of your DB instance during off-peak times, by using a smaller instance class.

See the [Amazon RDS SQL Server Pricing](#) webpage for up-to-date pricing information for all pricing models and instance classes.¹⁴

Microsoft SQL Server on Amazon EC2

You can also choose to run a Microsoft SQL Server on Amazon EC2, as described following.

Starting a SQL Server Instance on Amazon EC2

To start a SQL Server DB instance on Amazon EC2, you can use the AWS Management Console, or programmatically use CloudFormation templates, AWS CLI tools, or AWS SDKs supporting various programming languages. The walkthrough following uses the AWS Management Console for illustration purposes.

Using the AWS Management Console, choose the **EC2** service from the **Services** menu, and choose **Launch Instance** to start the launch wizard.

AWS lets you deploy a SQL Server instance on Amazon EC2 using Amazon Machine Images (AMIs). An AMI is simply a packaged environment that includes all the necessary bits to set up and boot your instance. Some AMIs have just the operating system (for example Windows Server 2012 R2), and others have the operating system and a version and edition of SQL Server (Windows Server 2012 R2 and SQL Server 2014 Standard Edition, and so on). We recommend that you

use the AMIs available at <http://aws.amazon.com/windows/resources/amis/>. These are available in all AWS Regions.

Some AMIs include an installation of a specific version and edition of SQL Server. When running an Amazon EC2 instance based on one of these AMIs, the SQL Server licensing costs are included in the hourly price to run the Amazon EC2 instance.

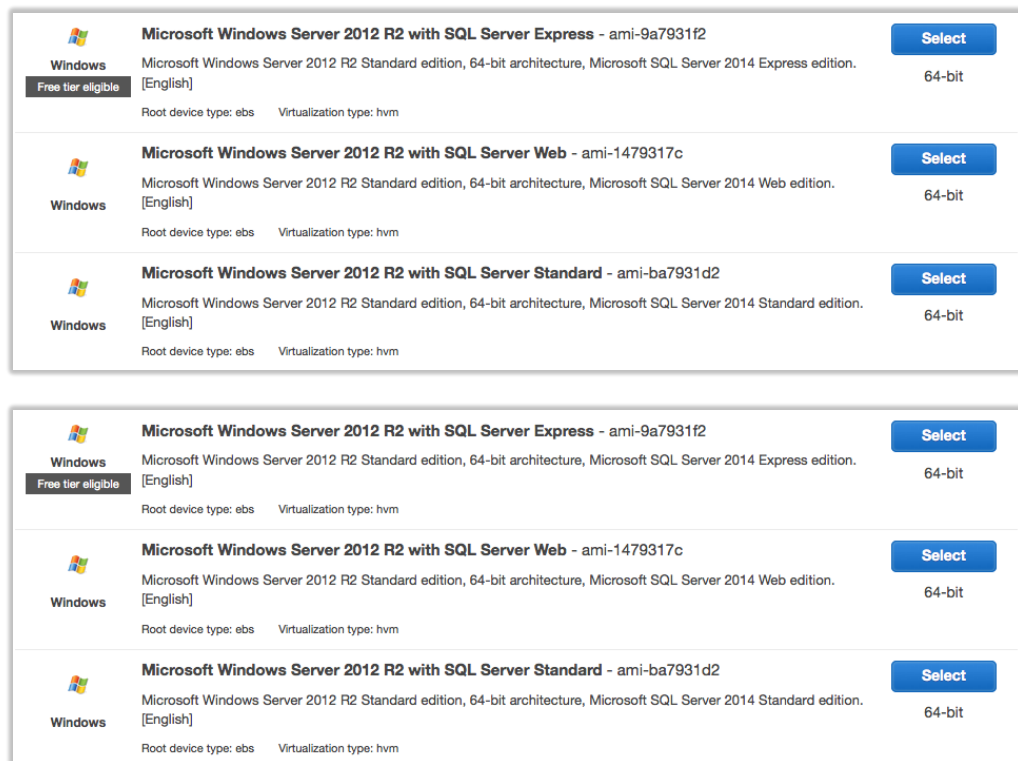


Figure 5: Amazon AMIs with SQL Server

Other AMIs install just the Microsoft Windows operating system. This type of AMI allows you the flexibility to perform a separate, custom installation of SQL Server on the Amazon EC2 instance and bring your own license (BYOL) of Microsoft SQL Server if you have qualifying licenses. For additional information on BYOL qualification criteria, see the AWS [License Mobility](#) webpage.¹⁵

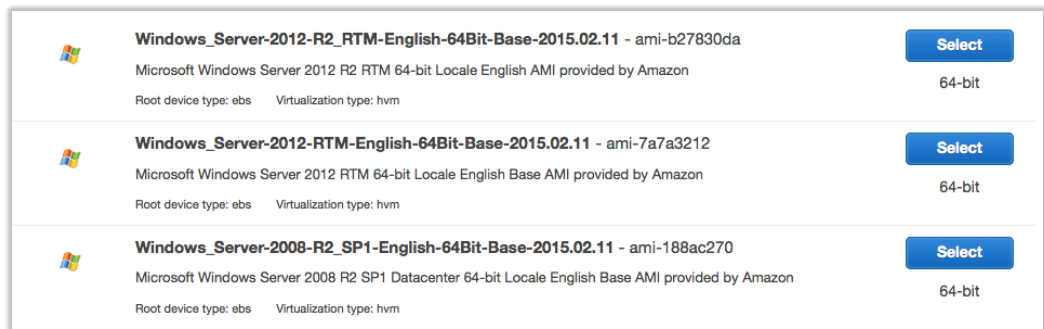


Figure 6: Amazon AMIs with Windows Server

To use an AMI, first select the appropriate AMI, then select the desired instance type. Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. They have varying combinations of compute capacity, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources.

Following is a list of the memory-optimized Amazon EC2 instances. These instance types are optimized for memory-intensive applications and have the lowest cost per GB of RAM among Amazon EC2 instance types. These instance types are recommended for high performance databases, like SQL Server. Consider all five performance characteristics of Amazon EC2 instances when selecting the appropriate EC2 instance.

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
<input checked="" type="checkbox"/>	Memory optimized	r3.large	2	15	1 x 32 (SSD)	-	Moderate
<input type="checkbox"/>	Memory optimized	r3.xlarge	4	30.5	1 x 80 (SSD)	Yes	Moderate
<input type="checkbox"/>	Memory optimized	r3.2xlarge	8	61	1 x 160 (SSD)	Yes	High
<input type="checkbox"/>	Memory optimized	r3.4xlarge	16	122	1 x 320 (SSD)	Yes	High
<input type="checkbox"/>	Memory optimized	r3.8xlarge	32	244	2 x 320 (SSD)	-	10 Gigabit

Figure 7: Instance Type Selector

Next, configure the instance details.

The screenshot displays the 'Step 3: Configure Instance Details' interface in the AWS Management Console. It features several configuration sections:

- Number of instances:** A text input field containing the value '1'.
- Purchasing option:** A radio button labeled 'Request Spot Instances' which is currently unchecked.
- Network:** A dropdown menu showing 'vpc-9199faf4 (10.0.0.0/16) | cloudvikings.net'. To the right is a 'Create new VPC' link.
- Subnet:** A dropdown menu showing 'subnet-6232bd15(10.0.1.0/24) | cloudvikings.'. Below it, it states '248 IP Addresses available'. To the right is a 'Create new subnet' link.
- Auto-assign Public IP:** A dropdown menu set to 'Use subnet setting (Disable)'.
- Placement group:** A dropdown menu set to 'No placement group'.
- Domain join directory:** A dropdown menu showing 'corp.cloudvikings.net (d-906736b115)'. To the right is a 'Create new directory' link.
- IAM role:** A dropdown menu showing 'ec2-admin'. To the right is a 'Create new IAM role' link.
- Shutdown behavior:** A dropdown menu set to 'Stop'.
- Enable termination protection:** A radio button labeled 'Protect against accidental termination' which is currently unchecked.

At the bottom of the form, there are four buttons: 'Cancel', 'Previous', 'Review and Launch' (which is highlighted in blue), and 'Next: Add Storage'.

Figure 8: Configure Instance Details

We recommend running Amazon EC2 instances in a VPC based on the Amazon Virtual Private Cloud (Amazon VPC) service. Amazon VPC lets you provision a logically isolated section of the AWS Cloud where you can launch AWS resources in a virtual network that you define. You have complete control over your virtual networking environment, including selection of your own IP address range, creation of subnets, and configuration of route tables and network gateways. For the **Network** details, choose an already existing VPC or choose **Create new VPC** to create a new VPC. For details on designing a VPC, see the [Amazon VPC Getting Started Guide](#).¹⁶

Configure the remaining instance details options depending on your workload needs. Depending on the type of SQL Server deployment, for example stand-alone, Windows Failover Clustering and AlwaysOn Availability Groups, and so on, you might decide to assign one or multiple static IP addresses to your Amazon EC2 instance. You can do this assignment in the **Network interface** section of **Configure Instance Details**.



Figure 9: Configure Network Interfaces

Add the appropriate storage volumes depending on your workload needs. For more details on select the appropriate volume types, see the [Disk I/O Management](#) section following.

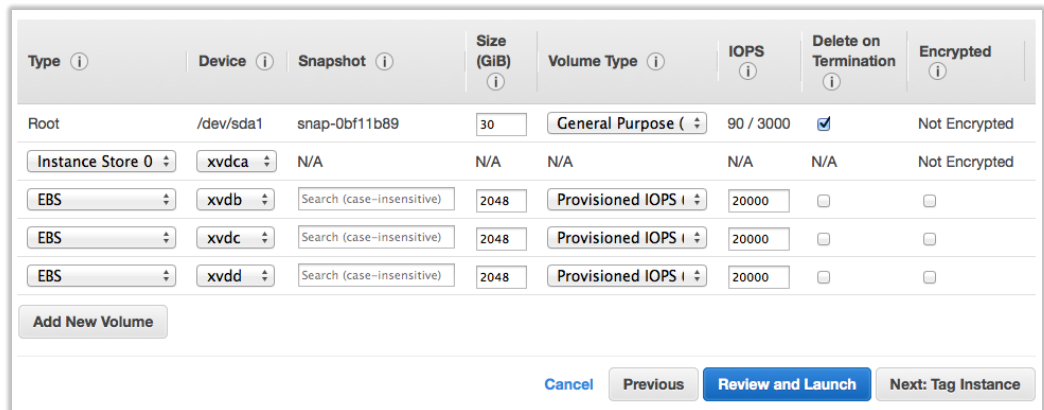


Figure 10: Add Storage

Assign the appropriate tags to the Amazon EC2 instance. We recommend that you assign tags to other Amazon resources, for example Amazon Elastic Block Store (Amazon EBS) volumes, to allow for more control over resource level permissions and cost allocation. For best practices on tagging AWS resources, see [Tagging Your Amazon EC2 Resources](#) in the *Amazon EC2 User Guide*.¹⁷

Step 5: Tag Instance
 A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver. [Learn more](#) about tagging your Amazon EC2 resources.

Key (127 characters maximum)	Value (255 characters maximum)
Name	Windows-SQL2K14-001
Cost Center	132-322-01
Environment	Production
Application	eCommerce
Service	Database

Create Tag (Up to 10 tags maximum)

[Cancel](#) [Previous](#) [Review and Launch](#) [Next: Configure Security Group](#)

Figure 11: Tag Instance

Security groups for Windows instances act as virtual firewalls that control the network traffic for one or more instances. The following screenshot shows the Configure Security Group step.

Step 6: Configure Security Group
 A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group: Create a new security group
 Select an existing security group

Security group name: SQL Server

Description: SQL Server eCommerce Service

Type	Protocol	Port Range	Source
RDP	TCP	3389	Custom IP sg-92490dfa
MS SQL	TCP	1433	Custom IP sg-5258d136

Add Rule

[Cancel](#) [Previous](#) [Review and Launch](#)

Figure 12: Configure Security Group

When you’re finished, review the instance details and choose **Launch**, or make any necessary changes by choosing **Edit** next to the appropriate section, as shown following.

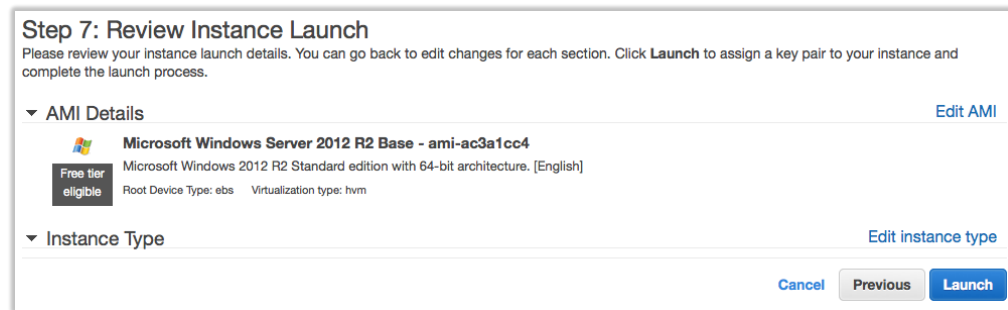


Figure 13: Review Instance Launch

Security

When you run SQL Server on Amazon EC2 instances, you have the responsibility to effectively protect network access to your instances with security groups, adequate operating system settings, and best practices, such as limiting access to open ports and using strong passwords. In addition, you can also configure a host-based firewall or an intrusion detection and prevention system (IDS/IPS) on your instances.

As with Amazon RDS, in EC2 security controls start at the network layer, with the network design itself in EC2-VPC or EC2-Classic, along with subnets, security groups, and network access control lists as applicable. For a more detailed discussion of these features, review the [Amazon RDS Security](#) section preceding.

Using AWS Identity and Access Management (IAM), you can control access to your Amazon EC2 resources and authorize (or deny) users the ability to manage your instances running the SQL Server database and the corresponding EBS volumes. For example, you can restrict the ability to start or stop your Amazon EC2 instances to a subset of your administrators. You can also assign Amazon EC2 roles to your instances, giving them privileges to access other AWS resources that you control. For more information on how to use IAM to manage administrative access to your instances, see [Controlling Access to Amazon EC2 Resources](#) in the *Amazon EC2 User Guide*.¹⁸

In an Amazon EC2 deployment of SQL Server, you are also responsible for patching the OS and application stack of your instances when Microsoft or other third-party vendors release new security or functional patches. This patching

includes work for additional support services and instances, such as Active Directory servers.

You can encrypt the EBS data volumes of your SQL Server instances in Amazon EC2. This option is available to all editions of SQL Server deployed on Amazon EC2 and is not limited to the Enterprise Edition, unlike TDE. When you create an encrypted EBS volume and attach it to a supported instance type, data stored at rest on the volume, disk I/O, and snapshots created from the volume are all encrypted. The encryption occurs on the servers that host Amazon EC2 instances, transparently to your instance, providing encryption of data in transit from EC2 instances to EBS storage as well. Note that encryption of boot volumes is not supported yet. Your data and associated keys are encrypted using the industry-standard AES-256 algorithm.

EBS volume encryption integrates with the AWS KMS. This integration allows you to use your own customer master key (CMK) for volume encryption. Creating and leveraging your own CMK gives you more flexibility, including the ability to create, rotate, disable, and define access controls, and to audit the encryption keys used to protect your data.

Performance Management

The performance of a relational DB instance on AWS depends on many factors, including the Amazon EC2 instance type, the configuration of the database software, the application workload, and the storage configuration. The following sections describe various options that are available to you to tune the performance of the AWS infrastructure on which your SQL Server instance is running.

Instance Sizing

AWS has many different Amazon EC2 instance types available, so you can choose the instance type that best fits your needs. These instance types vary in size, ranging from the smallest instance, the t2.micro with 1 vCPU, 1 GB of memory, and EBS-only storage, to the largest instance, the d2.8xlarge with 36 vCPUs, 244 GB of memory, 48 TB of local storage, and 10 gigabit network performance.

We recommend that you choose Amazon EC2 instances that best fit your workload requirements and have a good balance of CPU, memory, and IO

performance. SQL Server workloads are typically memory-bound, so look at the r3 instances, also referred to as memory-optimized instances. If your workload is more CPU-bound, look at the latest compute-optimized instances of the c4 instance family. If your workload is I/O bound, look at 8xlarge and 10xlarge instances. These instances are available in a number of instances families, like the m4, c4, r3, i2, or d2, which gives you 10 gigabit network performance.

One of the differentiators among all these instance types is that the m4 and c4 instance types are EBS-optimized by default, whereas other instance types, such as the r3 family, can be optionally EBS-optimized. You can find a detailed explanation of EBS-optimized instances in the [Disk I/O Management](#) section following. If your workload is network-bound, again look at instance families that support 10 gigabit network performance, because these instance families also support Enhanced Networking. These include the c4, r3, i2, and d2 instance families.

Enhanced Networking enables you to get significantly higher packet per second (PPS) performance, lower network jitter, and lower latencies by using single root I/O virtualization (SR-IOV). This feature uses a new network virtualization stack that provides higher I/O performance and lower CPU utilization compared to traditional implementations. To take advantage of this feature, you should launch an HVM AMI in Amazon VPC and install the appropriate driver. Driver installation might be necessary for Windows instances; for detailed instructions, see [Enhanced Networking on Windows](#) in the *Amazon EC2 User Guide*.¹⁹

Disk I/O Management

The same storage types available for Amazon RDS are also available when deploying SQL Server on Amazon EC2. Additionally, you also have access to instance storage. Because you have fine-grained control over the storage volumes and strategy to use, you can deploy workloads that require more than 4 TiB in size or 20,000 IOPS in Amazon EC2. Multiple EBS volumes or instance storage disks can even be striped together in a software RAID configuration to aggregate both the storage size and usable IOPS, beyond the capabilities of a single volume.

The two main Amazon EC2 storage options are as follows:

- Instance store volumes: Several Amazon EC2 instance types come with a certain amount of “local” (directly attached) storage, which is ephemeral.

Any data saved on instance storage is no longer available after you stop and restart that instance, or if the underlying hardware fails, which causes an instance restart to happen on a different host server. This characteristic makes instance storage a challenging option for database persistent storage. However, Amazon EC2 instances can have the following benefits:

- Instance store volumes offer good performance for sequential disk access, and don't have a negative impact on your network connectivity. Some customers have found it useful to use these disks to store temporary files to conserve network bandwidth.
- Instance types with large amounts of instance storage offer unmatched I/O performance and are recommended for database workloads, as long as you implement a backup or replication strategy that addresses the ephemeral nature of this storage.
- **EBS volumes:** Similar to Amazon RDS, you can use EBS for persistent block-level storage volumes. Amazon EBS volumes are off-instance storage that persist independently from the life of an instance. Amazon EBS volume data is mirrored across multiple servers in an Availability Zone to prevent the loss of data from the failure of any single component. You can back them up to Amazon S3 by using snapshots. These attributes make EBS volumes suitable for data files, log files, and the flash recovery area. Although the maximum size of an EBS volume is 16 TB, you can address larger database sizes by striping your data across multiple volumes.

Storage Type	Min. Volume Size	Max. Volume Size	Baseline Performance	Burst Capability	Storage Technology
Magnetic Storage	1 GiB	1 TiB	About 100 IOPS.	Yes; several hundred IOPS.	Magnetic Disk
General Purpose	1 GiB	16 TiB	3 IOPS/GB, max. 3000 IOPS for volumes 1 TiB or less, up to 10,000 IOPS for larger volumes.	Yes; up to 3000 IOPS, subject to accrued credits for volumes less than 1 TiB in size.	SSD
Provisioned IOPS	4 GiB	16 TiB	Max. 20,000 IOPS. Ratio between 3 and 30 IOPS for each GiB.	No; fixed allocation. You provision the level you need.	SSD

EBS-optimized instances enable Amazon EC2 instances to fully utilize the provisioned IOPS on an EBS volume. These instances deliver dedicated throughput between Amazon EC2 and Amazon EBS, with options between 500 Mbps and 4,000 Mbps, depending on the instance type. When attached to EBS-optimized instances, Provisioned IOPS volumes are designed to deliver within 10 percent of their provisioned performance 99.9 percent of the time. The combination of EBS-optimized instances and Provisioned IOPS volumes helps to ensure that instances are capable of consistent and high EBS I/O performance. Most databases with high I/O requirements should benefit from this feature. You can also use EBS-optimized instances with standard EBS volumes if you need predictable bandwidth between your instances and EBS. For up-to-date information about the availability of EBS-optimized instances, see the AWS [EC2 Instance Types](#) webpage.²⁰

To scale up random I/O performance, you can increase the number of EBS volumes your data resides on, for example by using eight 100 GB EBS volumes instead of one 800 GB EBS volume. However, remember that using striping generally reduces the operational durability of the logical volume by a degree inversely proportional to the number of EBS volumes in the stripe set. The more volumes you include in a stripe, the larger the pool of data that can get corrupted if a single volume fails, because the data on all other volumes of the stripe gets invalidated also.

EBS volume data is natively replicated, so using RAID 0 (striping) might provide you with sufficient redundancy and availability. You can use RAID 10 (striping and mirroring) instead of RAID 0 to improve the availability of your aggregate storage volume, but in most cases using RAID 10 will decrease your I/O performance compared to RAID 0. The choice between RAID 0 and RAID 10 will depend on your availability requirements and on the number of volumes of your aggregate. At any rate, you should take snapshots of your EBS volumes as often as possible. RAID 5 and RAID 6 are not recommended for Amazon EBS because the parity write operations of these RAID modes consume some of the IOPS available to your volumes.

Data, logs, and temporary files benefit from being stored on independent EBS volumes or volume aggregates because they present different I/O patterns. To take advantage of additional EBS volumes, be sure to evaluate the network load

to help ensure that your instance size is sufficient to provide the network bandwidth required.

I2 instances with instance storage are optimized to deliver tens of thousands of low-latency, random I/O operations per second (IOPS) to applications from direct-attached SSD drives. These instances provide an alternative to EBS volumes for the most I/O demanding workloads. Several instance types are available, with the largest I2 Eight Extra Large providing 244 GB of memory and 8 x 800 GB SSD drives, for a total of 6.4 TB of storage and up to 365,000 read IOPS or 315,000 first write IOPS.

Amazon EC2 offers many options to optimize and tune your I/O subsystem. We encourage you to benchmark your application on several instance types and storage configurations to select the most appropriate configuration. For EBS volumes, we recommend that you monitor the CloudWatch average queue length metric of a given volume, and target an average queue length of 1 for every 500 IOPS for volumes up to 2000 IOPS, and a length between 4 and 8 for volumes with 2000 to 4000 IOPS. Lower metrics indicate overprovisioning, and higher numbers usually indicate your storage system is overloaded.

High Availability

High availability is a design and configuration principle to help protect services or applications from single points of failure. Werner Vogels, CTO of Amazon.com, says, “Everything fails all the time.” By this, he means we need to avoid single points of failure, assume everything fails, and design backwards. The goal is for services and applications to continue to function even if underlying physical hardware fails or is removed or replaced. We will review three native SQL Server features that improve database high availability and ways to deploy these features on AWS.

Log Shipping

Log shipping provides a mechanism to automatically send transaction log backups from a primary database on one DB instance to one or more secondary databases on separate DB instances. Although log shipping is typically considered a disaster recovery feature, it can also provide high availability by allowing secondary DB instances to be promoted as the primary in the event of a failure of the primary DB instance.

Log shipping offers you many benefits to increase the availability of log-shipped databases. Besides the benefits of disaster recovery and high availability already mentioned, log shipping also provides access to secondary databases to use as read-only copies of the database. This feature is available between restore jobs. It can also allow you to configure a lag delay, or a longer delay time, which can allow you to recover accidentally changed data on the primary database before these changes are shipped to the secondary database.

We recommend running the primary and secondary DB instances in separate Availability Zones, and optionally deploying an optional monitor instance to track all the details of log shipping. Backup, copy, restore, and failure events for a log shipping group are available from the monitor instance.

Database Mirroring

Database mirroring is a feature that provides a complete or almost complete mirror of a database, depending on the operating mode, on a separate DB instance. Database mirroring is the technology used by Amazon RDS to provide Multi-AZ support for Amazon RDS for SQL Server. This feature increases the availability and protection of mirrored databases, and provides a mechanism to keep mirrored databases available during upgrades.

In database mirroring, SQL Servers can take one of three roles: the principal server, which hosts the read-write principal version of the database; the mirror server, which hosts the mirror copy of the principal database; and an optional witness server. The witness server is only available in high-safety mode and monitors the state of the database mirror and automates the failover from the primary database to the mirror database.

A mirroring session is established between the principal and mirror servers, which act as partners. They perform complementary roles, as one partner assumes the principal role while the other partner assumes the mirror role. Mirroring performs all inserts, updates, and deletes that are executed against the principal database on the mirror database. Database mirroring can either be a synchronous or asynchronous operation. These operations are performed in the two mirroring operating modes:

- High-safety mode uses synchronous operation. In this mode, the database mirror session synchronizes the inserts, updates, and deletes from the

principal database to the mirror database as quickly as possible using a synchronous operation. As soon as the database is synchronized, the transaction is committed on both partners. This mode has increased transaction latency as each transaction needs to be committed on both the principal and mirror databases. Because of this high latency, we recommend that partners be in the same or different Availability Zones hosted within the same AWS Region when you use this operating mode.

- High-performance mode uses asynchronous operation. Using this mode, the database mirror session synchronizes the inserts, updates, and deletes from the principal database to the mirror database using an asynchronous process. Unlike a synchronous operation, this mode can result in a lag between the time the principal database commits the transactions and the time the mirror database commits the transactions. This mode has minimum transaction latency and is recommended when partners are in different AWS Regions.

SQL Server AlwaysOn Availability Groups

AlwaysOn Availability Groups is an enterprise-level feature that provides high availability and disaster recovery to SQL Server databases. AlwaysOn Availability Groups uses advanced features of Windows Failover Cluster and the Enterprise Edition of SQL Server 2012 and 2014.

These availability groups support the failover of a set of user databases as one distinct unit or group. User databases defined within an availability group consist of primary read-write databases along with multiple sets of related secondary databases. These secondary databases can be made available to the application tier as read-only copies of the primary databases, thus providing a scale-out architecture for read workloads. You can also use the secondary databases for backup operations.

You can implement SQL Server AlwaysOn Availability Groups on Amazon Web Services using services like Windows Server Failover Clustering (WSFC), Amazon EC2, Amazon VPC, Active Directory, and DNS.

Amazon Web Services has published a whitepaper and posted multiple webinars on how to deploy SQL Server AlwaysOn Availability Groups. For details on how to deploy SQL Server AlwaysOn Availability Groups in AWS using CloudFormation, see the AWS whitepaper [Implementing Microsoft Windows](#)

[Server Failover Clustering and SQL Server AlwaysOn Availability Groups in the AWS Cloud.](#)²¹

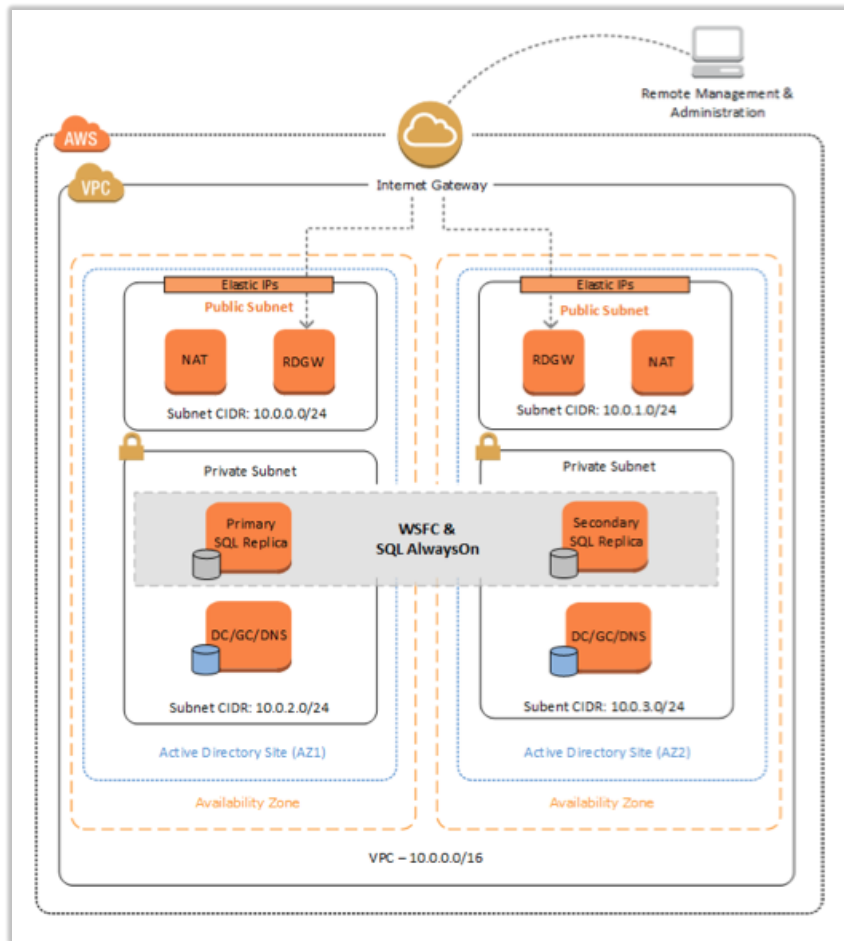


Figure 13: SQL Server AlwaysOn Availability Group

Monitoring and Management

Amazon CloudWatch is an AWS instance-monitoring service that provides detailed CPU, disk, and network utilization metrics for each Amazon EC2 instance and EBS volume. Using these metrics, you can perform detailed reporting and management. This data is available in the AWS Management Console and also the API. Using the API allows for infrastructure automation and orchestration based on load metrics.

Additionally, Amazon CloudWatch supports custom metrics, such as memory utilization or disk utilizations, which are metrics visible only from within the instance. You can publish your own relevant metrics to the service, to consolidate monitoring information. You can also push custom logs to CloudWatch Logs to monitor, store, and access your log files for Amazon EC2 SQL Server instances. You can then retrieve the associated log data from CloudWatch Logs using the Amazon CloudWatch console, the CloudWatch Logs commands in the AWS CLI, or the CloudWatch Logs SDK. This approach allows you to track log events in real time for your SQL Server instances.

As with Amazon RDS, you can configure alarms on Amazon EC2, EBS, and custom metrics to trigger notifications when the state changes. An alarm tracks 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. Notifications are sent to Amazon SNS topics or Auto Scaling policies. You can configure these alarms to notify database administrators by email or Amazon SMS when they get triggered.

In addition, you can use Microsoft and any third-party monitoring tools that have built-in SQL Server monitoring capabilities. Amazon EC2 SQL Server monitoring can be integrated with System Center Operations Manager (SCOM). Open-source monitoring frameworks, such as Nagios, can also be run on Amazon EC2 to monitor your whole AWS environment, including your SQL Server databases.

The management of a SQL Server database on Amazon EC2 is similar to the management of an on-premises database. You can use SQL Server Management Studio, SQL Server Configuration Manager, SQL Server Profiler, and other Microsoft and third-party tools to perform administration or tuning tasks.

AWS also offers the [AWS Management Pack for System Center add-ins](#) to extend the functionality of your existing Microsoft System Center implementation to monitor and control AWS resources from the same interface as your on-premises resources.²² These add-ins are currently available at no additional cost for SCOM versions 2007 and 2012 and System Center Virtual Machine Manager (SCVMM)..

Although you can use Amazon EBS snapshots as a mechanism to back up and restore EBS volumes, the service does not integrate with the Volume Shadow-Copy Service (VSS). You can take a snapshot of an attached volume that is in use.

However, VSS integration is required to ensure that the disk I/O of SQL Server is temporarily paused during the snapshot process. Any data that has not been persisted to disk by SQL Server or the operating system at the time of the EBS snapshot is excluded from the snapshot. Lacking coordination with VSS, there is a risk that the snapshot will not be consistent, and the database files can potentially get corrupted. For this reason, we recommend using third-party backup solutions that are designed for SQL Server workloads.

Managing Cost

AWS's elastic and scalable infrastructure and services make running SQL Server on Amazon a cost-effective proposition, by tracking demand more closely and reducing overprovisioning. As with Amazon RDS, the costs of running SQL Server on Amazon EC2 depend on several factors. Because you have more control over your infrastructure and resources when deploying SQL Server on Amazon EC2, there are a few additional dimensions to optimize cost on, compared to Amazon RDS:

- The AWS Region the instance is deployed in
- Instance type and EBS optimization
- The type of instance tenancy selected
- The high availability solution selected
- The storage type and size selected for the EC2 instance
- The Multi-AZ mode of the instance
- The pricing model
- How long it is running during a given billing period

As with Amazon RDS, Amazon EC2 hourly instance costs vary by the region. If you have flexibility about where you can deploy your workloads geographically, we recommend deploying your workload in the region with the cheapest EC2 costs for your particular use case.

Different instance types have different hourly charges. Generally, current generation instance types have lower hourly charges compared to previous generation instance types, along with better performance due to newer hardware

architectures. We recommend that you test your workloads on new instance types as these become available, and plan to migrate your workloads to new instance types, if the cost vs. performance ratio makes sense for your use case.

Many EC2 instance types are available with the EBS Optimized option. This option is available for an additional hourly surcharge, and provides additional, dedicated networking capacity for EBS I/O. This dedicated capacity ensures a predictable amount of networking capacity to sustain predictable EBS I/O. Some current generation instance types, such as the C4, M4 and D2 instance types, are EBS-optimized by default and don't have an additional surcharge for the optimization.

Dedicated Instances are Amazon EC2 instances that run in a VPC on hardware that's dedicated to a single customer. Your Dedicated Instances are physically isolated at the host hardware level from your instances that aren't Dedicated Instances and from instances that belong to other AWS accounts. We recommend deploying EC2 SQL Server instances in dedicated tenancy if you have certain regulatory needs. Dedicated tenancy has a per-region surcharge for each hour a customer runs at least one instance in dedicated tenancy. The hourly cost for instance types operating in dedicated tenancy is different for standard tenancy. Up-to-date pricing information is available on the [Amazon EC2 Dedicated Instances](#) pricing page.²³

You also have the option to provision EC2 Dedicated Hosts. These are physical servers with Ec2 instance capacity fully dedicated to your use. Dedicated Hosts can help you address compliance requirements and reduce costs by allowing you to use your existing server-bound software licenses. More information is available on the [Amazon EC2 Dedicated Hosts](#) webpage.²⁴.

Amazon EC2 Reserved Instances allow you to lower costs and reserve capacity. Reserved Instances can save you up to 70 percent over On-Demand rates when used in steady state. They can be purchased for one or three year terms. If your SQL Server database is going to be running more than 60 percent of the time, you will most likely financially benefit from using a Reserved Instance. Unlike with On-Demand pricing, the capacity reservation is made for the entire duration of the term, whether a specific instance is using the reserved capacity or not.

The following pricing options are available for EC2 reserved instances:

- With All Upfront Reserved Instances, you pay for the entire Reserved Instance with one upfront payment. This option provides you with the largest discount compared to On-Demand Instance pricing.
- With Partial Upfront Reserved Instances, you make a low upfront payment and are then charged a discounted hourly rate for the instance for the duration of the Reserved Instance term.
- With No Upfront Reserved Instances, you don't make any upfront payments, but will be charged a discounted hourly rate for the instance for the duration of the Reserved Instance term. This option still provides you with a significant discount compared to On-Demand Instance pricing, but the discount is usually less than for the other two reserved instance pricing options.

Additionally, the following options can be combined to reduce your cost of operating SQL Server on EC2:

- Use the Windows Server with SQL Server AMIs where licensing is included. The cost of the SQL Server license is included in the hourly cost of the instance. You are only paying for the SQL Server license when the instance is running. This approach is especially effective for databases that are not running 24/7 and for short projects.
- Shut down DB instances when they are not needed. For example, some development and test databases can be shut down at night and on weekends and restarted on weekdays in the morning.
- Scale down the size of your databases during off-peak times.

Caching

Whether using SQL Server on Amazon EC2 or Amazon RDS, SQL Server users confronted with heavy workloads should look into reducing this load by caching data so that the web and application servers don't have to repeatedly access the database for common or repeat data sets. Deploying a caching layer between the business logic layer and the database is a common architectural design pattern to reduce the amount of read traffic and connections to the database itself.

The effectiveness of the cache depends largely on the following aspects:

- Generally, the more read-heavy the query patterns of the application are on the database, the more effective caching can be.
- Commonly, the more repetitive query patterns are, with queries returning infrequently changing datasets, the more you can benefit from caching.

Leveraging caching usually requires changes to applications. The logic of checking, populating, and updating a cache is normally implemented in the application, data, and database abstraction layer or Object Relational Mapper (ORM).

Several tools can address your caching needs. You have the option to use a managed service similar to Amazon RDS, but for caching engines. You can also choose from different caching engines that have slightly different feature sets:

- **Amazon ElastiCache:** In a similar fashion to Amazon RDS preceding, ElastiCache allows you to provision fully managed caching clusters supporting both Memcached and Redis (details following). ElastiCache simplifies and offloads the management, monitoring, and operation of a Memcached or Redis environment, enabling you to focus on the differentiating parts of your applications.
- **Memcached:** An open-source, high-performance, distributed in-memory object caching system, Memcached is an in-memory object store for small chunks of arbitrary data (strings, objects), such as results of database calls. Memcached is widely adopted and mostly used to speed up dynamic web applications by alleviating database load.
- **Redis:** An open-source, high-performance, in-memory key-value NoSQL data engine, Redis stores structured key-value data and provides rich query capabilities over your data. The contents of the data store can also be persisted to disk. Redis is widely adopted to speed up a variety of analytics workloads by storing and querying more complex or aggregate datasets in-memory relieving some of the load off backend SQL databases.

Hybrid Scenarios and Data Migration

Some AWS customers already have SQL Server running in their on-premises or co-located data center but want to use the AWS Cloud to enhance their architecture to provide a more highly available solution or one that offers disaster

recovery. Other customers are looking to migrate workloads to AWS without incurring significant downtime. These efforts often can stretch over a significant amount of time. AWS offers several services and tools to assist customers in these use cases, and SQL Server has several replication technologies that offer high availability and disaster recovery solutions. These features differ depending on the SQL Server version and edition.

Backups to the Cloud

AWS storage solutions allow you to pay for only what you need. AWS doesn't require capacity planning, purchasing capacity in advance, or any large upfront payments. You get the benefits of AWS storage solutions without the upfront investment and hassle of setting up and maintaining an on-premises system.

Amazon Simple Storage Service (Amazon S3)

Using Amazon S3, you can take advantage of the flexibility and pricing of cloud storage. S3 gives you the ability to back up SQL Server databases to a highly secure, available, durable, reliable storage solution. Many third-party backup solutions are designed to securely store SQL Server backups in Amazon S3. You can also design and develop a SQL Server backup solution yourself by using AWS tools like the AWS CLI, AWS Tools for Windows PowerShell, or a wide variety of SDKs for .NET or Java, and also the AWS Toolkit for Visual Studio.

AWS Storage Gateway

AWS Storage Gateway is a service connecting an on-premises software appliance with cloud-based storage to provide seamless and secure integration between an organization's on-premises IT environment and AWS's storage infrastructure. The service allows you to securely store data in the AWS Cloud for scalable and cost-effective storage. AWS Storage Gateway supports industry-standard storage protocols that work with your existing applications. It provides low-latency performance by maintaining frequently accessed data on-premises while securely storing all of your data encrypted in Amazon S3. AWS Storage Gateway enables your existing on-premises-to-cloud backup applications to store primary backups on Amazon S3's scalable, reliable, secure, and cost-effective storage service.

SQL Server Log Shipping—Between On-Premises and Amazon EC2

Some AWS customers have already deployed SQL Server using a Windows Server Failover Cluster design in an on-premises or co-located facility. This approach provides high availability in the event of component failure within a data center but doesn't protect against a significant outage impacting multiple components or the entire data center.

Other AWS customers have been using SQL Server synchronous mirroring to provide an high availability solution in their on-premises data center. Again, this provides high availability in the event of component failure within the data center but doesn't protect against a significant outage impacting multiple components or the entire data center.

You can extend your existing on-premises high-availability solution and provide a disaster recover solution with AWS by using the native SQL Server feature of log shipping. SQL Server transaction logs can ship from on-premises or co-located data centers to a SQL Server instance running on an Amazon EC2 instance within a VPC. This data can be securely transmitted over a dedicated network connection using AWS Direct Connect or over a secure VPN tunnel. Once shipped to the Amazon EC2 instance, these transaction log backups are applied to secondary DB instances. You can configure one or multiple databases as secondary databases. An optional third Amazon EC2 instance can be configured to act as a monitor, an instance that monitors the status of backup and restore operations and raises events if these operations fail.

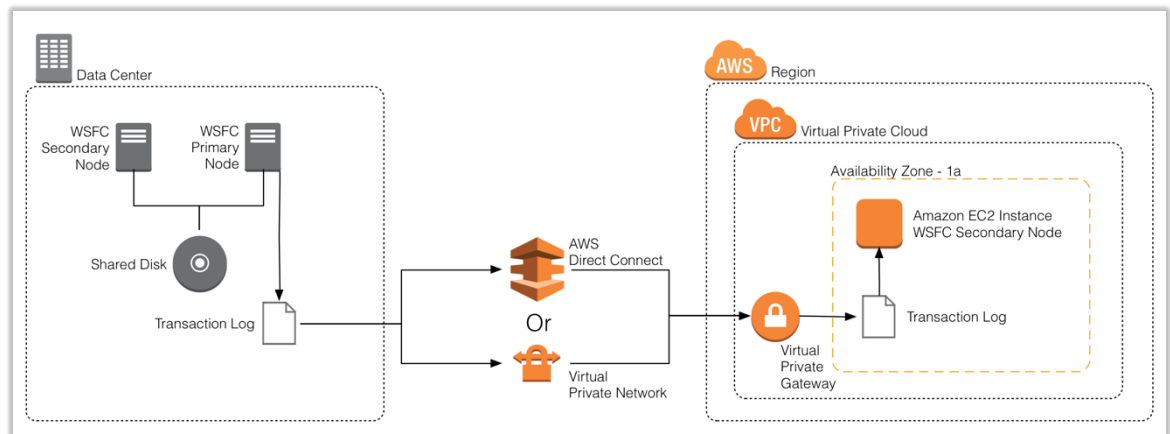


Figure 14: Hybrid SQL Server Log Shipping

SQL Server AlwaysOn Availability Groups—Between On-Premises and Amazon EC2

SQL Server AlwaysOn Availability Groups is an advanced, enterprise-level feature to provide high availability and disaster recovery solutions. This feature is available when deploying the Enterprise Edition of SQL Server 2012 or 2014 within the AWS Cloud on Amazon EC2 or on physical or virtual machines deployed in on-premises or co-located data centers. If you have existing on-premises deployments of SQL Server AlwaysOn Availability Groups, you might want to use the AWS Cloud to provide an even higher level of availability and disaster recovery. To do so, you can extend your data center into a VPC by using a dedicated network connection like AWS Direct Connect or setting secure VPN tunnels between these two environments.

Following are a few things you might consider doing when planning a hybrid implementation of SQL Server AlwaysOn Availability Groups:

- Establish secure, reliable, and consistent network connection between on-premises and AWS (using AWS Direct Connect or VPN).
- Create a VPC based on the Amazon VPC service.
- Use Amazon VPC route tables and security groups to enable the appropriate communicate between the new environments.

- Extend Active Directory domains into the VPC by deploying domain controllers as Amazon EC2 instances or using the AWS Directory Service AD Connector service.
- Use synchronous mode between SQL Server instances within the same environment (for example, all instances on-premises or all instances in AWS).
- Use asynchronous mode between SQL Server instances in different environments (for example, instances in AWS and on-premises).

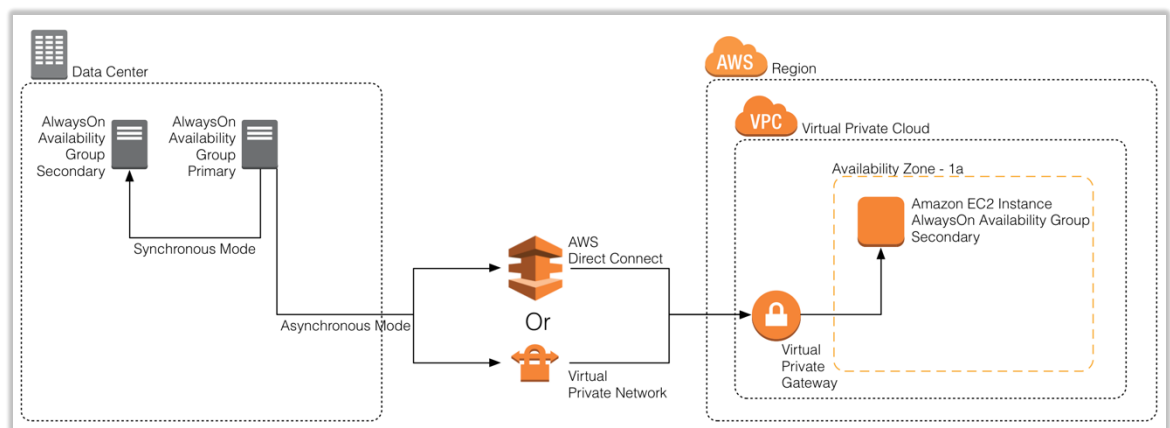


Figure 15: Hybrid Windows Server Failover Cluster

AWS Database Migration Service

[AWS Database Migration Service](#) (AWS DMS) helps you migrate databases to AWS easily and securely.²⁵ When you use the AWS Database Migration Service, the source database remains fully operational during the migration, minimizing downtime to applications that rely on the database.

You can begin a database migration with just a few clicks in the AWS Management Console. Once the migration has started, AWS manages many of the complexities of the migration process like data type transformation, compression, and parallel transfer (for faster data transfer) while ensuring that data changes to the source database that occur during the migration process are automatically replicated to the target. The service is intended to support migrations to and from AWS hosted databases, where both the source and destination engine are the same, and also heterogeneous data sources.

Comparison of Microsoft SQL Server Feature Availability on AWS

The following table shows a side-by-side comparison of available features of SQL Server in the AWS environment.

	Amazon RDS	Amazon EC2
SQL Server Editions	Supported Versions	Supported Versions
Express	2008 R2, 2012, 2014	2005, 2008, 2008 R2, 2012, 2014, 2016
Web	2008 R2, 2012, 2014	2008 R2, 2012, 2014, 2016
Standard	2008 R2, 2012, 2014	2005, 2008, 2008 R2, 2012, 2014, 2016
Enterprise	2008 R2, 2012	2005, 2008, 2008 R2, 2012, 2014, 2016
SQL Server Editions	Installation Method	Installation Method
Express	N/A	AMI, Manual install
Web	N/A	AMI, Manual install
Standard	N/A	AMI, Manual install
Enterprise	N/A	AMI, Manual install
Manageability Benefits	Supported	Supported
Managed Automated Backups	Yes	No (need to configure and manage maintenance plans, or use third-party solutions)
Multi-AZ with Automated Failover	Yes	Enterprise Edition only (with manual configuration of AlwaysOn Availability Groups)
Built-in Instance and Database Monitoring and Metrics	Yes	No (push your own metrics to CloudWatch or use third-party solution)
Automatic Software Patching	Yes	No
Preconfigured Parameters	Yes	No (default SQL Server installation only)

	Amazon RDS	Amazon EC2
DB Event Notifications	Yes	No (manually track and manage DB events)
SQL Server Feature	Supported	Supported
SQL Authentication	Yes	Yes
Windows Authentication	Yes	Yes
TDE (encryption at rest)	Yes (Enterprise Edition only)	Yes (Enterprise Edition only)
Encrypted Storage using AWS KMS	Yes (all editions)	Yes (all editions)
SSL (encryption in transit)	Yes	Yes
Database Replication	No	Yes
Log Shipping	No	Yes
Database Mirroring	Yes (Multi-AZ)	Yes
AlwaysOn Availability Groups	No	Yes
Max Number of DBs per Instance	30	None
Rename existing databases	Yes (Single-AZ only)	Yes (not available for databases in Availability Groups or enabled for mirroring)
Max Size of DB Instance	4 TiB	None
Min Size of DB Instance	20 GB (Web Exp) 200 GB (Std Ent)	None
Increase Storage Size	No	Yes
BACKUP Command	No	Yes
RESTORE Command	No	Yes
SQL Server Analysis Services	Data source only*	Yes
SQL Server Integration Services	Data source only*	Yes
SQL Server Reporting Services	Data source only*	Yes
Data Quality Services	No	Yes
Master Data Services	No	Yes
Custom Set Time Zones	No (UTC Only)	Yes

	Amazon RDS	Amazon EC2
SQL Server Mgmt Studio	Yes	Yes
sqlcmd	Yes	Yes
SQL Server Profiler	Yes (client side traces)	Yes
SQL Server Migration Assistance	Yes	Yes
DB Engine Tuning Advisor	Yes	Yes
SQL Server Agent	Yes	Yes
Safe CLR	Yes	Yes
Full-text search	Yes (except semantic search)	Yes
Spatial and location features	Yes	Yes
Change Tracking	Yes	Yes
Columnstore Indexes	2012 & newer (Ent)	2012 & newer (Std Ent)
Flexible Server Roles	2012 & newer	2012 & newer
Partially Contained Databases	2012 & newer	2012 & newer
Sequences	2012 & newer	2012 & newer
THROW statement	2012 & newer	2012 & newer
UTF-16 Support	2012 & newer	2012 & newer
New Query Optimizer	2014	2014
Delayed Transaction Durability (lazy commit)	2014	2014
Maintenance Plans	No**	Yes
Database Mail	No***	Yes
Linked Servers	No	Yes
MSDTC	No	Yes
Service Broker	Yes (except Endpoints)	Yes
Performance Data Collector	No	Yes
WCF Data Services	No	Yes
FILESTREAM	No	Yes
Policy-Based Management	No	Yes
SQL Server Audit	No	Yes

	Amazon RDS	Amazon EC2
BULK INSERT	No	Yes
OPENROWSET	No	Yes
Data Quality Services	No	Yes
Buffer Pool Extensions	No	Yes
File Tables	No	Yes

* Amazon RDS SQL Server DB instances can be used as data sources for SSRS, SSAS, and SSIS, but cannot run those services on the DB instance.

** Amazon RDS provides a separate set of features to facilitate backup and recovery of databases.

*** We encourage our customers use the Amazon Simple Email Service (Amazon SES) to send outbound emails originating from AWS resources and ensure a high degree of deliverability.

For detailed list of features supported by the editions of SQL Server, see [High Availability](#) on MSDN.²⁶

Conclusion

AWS provides two deployment platforms to deploy your SQL Server databases: Amazon RDS and Amazon EC2. Each platform provides unique benefits that might be beneficial to your specific use case, but you have the flexibility to use one or both depending on your needs. Understanding how to manage performance, high availability, security, and monitoring in these environments is outlined in this whitepaper.

Contributors

The following individuals and organizations contributed to this document:

- Russell Day, Solutions Architect, Amazon Web Services
- Darryl Osborne, Solutions Architect, Amazon Web Services
- Vlad Vlasceanu, Solutions Architect, Amazon Web Services

Further Reading

- Microsoft on AWS: <http://aws.amazon.com/microsoft/>
- Implementing Active Directory Domain Services in the AWS Cloud: <http://aws.amazon.com/microsoft/whitepapers/ad-reference-architecture/>
- Deploy Remote Desktop Gateway on the AWS Cloud: <http://aws.amazon.com/microsoft/whitepapers/rdgateway-reference-architecture/>
- Implementing Microsoft DirectAccess and NAT in the AWS Cloud: <http://aws.amazon.com/microsoft/whitepapers/ms-direct-access/>
- Secure Microsoft Applications on AWS: <http://aws.amazon.com/microsoft/whitepapers/secure-microsoft-applications-on-aws/>
- Implementing Microsoft Windows Server Failover Clustering and SQL Server AlwaysOn Availability Groups in the AWS Cloud:

<http://aws.amazon.com/microsoft/whitepapers/microsoft-wsfc-sql-alwayson/>

- AWS Directory Service: <https://aws.amazon.com/directoryservice/>

Document Revisions

Date	Revision
May 2015	Initial publication
July 2016	Updated with information on new features and changes: release of Amazon RDS SQL Server Windows Authentication; availability of SQL Server 2014 in Amazon RDS; new RDS Reserved DB Instance pricing model, availability of the AWS Database Migration Service; other minor corrections and content updates.

Notes

¹ What's New in SQL Server:

[https://technet.microsoft.com/en-us/library/bb500435\(v=sql.120\).aspx](https://technet.microsoft.com/en-us/library/bb500435(v=sql.120).aspx)

² Features Supported by the Editions of SQL Server:

[https://msdn.microsoft.com/en-us/library/cc645993\(v=sql.120\).aspx](https://msdn.microsoft.com/en-us/library/cc645993(v=sql.120).aspx)

³ Microsoft SQL Server on Amazon RDS:

http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_SQLServer.html#SQLServer.Concepts.General.FeatureSupport

⁴ Domain Name System (DNS):

http://en.wikipedia.org/wiki/Domain_Name_System

⁵ VPC ClassicLink:

<http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/vpc-classiclink.html>

⁶ Authentication and Access Control for Amazon RDS:

<http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/UsingWithRDS.IAM.html>

⁷ Using Windows Authentication with an Amazon RDS DB Instance Running Microsoft SQL Server:

http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_SQLServerWinAuth.html

⁸ Options for the Microsoft SQL Server Database Engine:

<http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Appendix.SQLServer.Options.html>

⁹ Using SSL with a Microsoft SQL Server DB Instance:

http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_SQLServer.html#SQLServer.Concepts.General.SSL

¹⁰ Facts About Amazon RDS Storage:

http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Storage.html#CHAP_Storage.FactsAbout

¹¹ Microsoft SQL Server I/O Patterns:

<http://blogs.msdn.com/b/psssql/archive/2010/03/24/how-it-works-bob-dorr-sql-server-i-o-presentation.aspx>

- 12 Amazon RDS Dimensions and Metrics:
<http://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/rds-metricscollected.html>
- 13 Enhanced Monitoring:
http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_Monitoring.html
- 14 Amazon RDS SQL Server Pricing:
<http://aws.amazon.com/rds/sqlserver/pricing/>
- 15 License Mobility:
<https://aws.amazon.com/windows/resources/licensemobility/>
- 16 VPC Getting Started Guide:
<http://docs.aws.amazon.com/AmazonVPC/latest/GettingStartedGuide/ExerciseOverview.html>
- 17 Tagging Your Amazon EC2 Resources:
http://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/Using_Tags.html
- 18 Controlling Access to Amazon EC2 Resources:
<http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/UsingIAM.html>
- 19 Enhanced Networking for Windows:
<http://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/enhanced-networking.html>
- 20 EC2 Instance Types:
<http://aws.amazon.com/ec2/instance-types/>
- 21 Implementing Microsoft Windows Server Failover Clustering and SQL Server AlwaysOn Availability Groups in the AWS Cloud:
<http://aws.amazon.com/windows/resources/whitepapers/alwayson/>
- 22 AWS Add-ins for Microsoft System Center:
<http://aws.amazon.com/windows/system-center/>
- 23 Amazon EC2 Dedicated Instances:
<http://aws.amazon.com/ec2/purchasing-options/dedicated-instances/>
- 24 Amazon EC2 Dedicated Hosts:
<http://aws.amazon.com/ec2/dedicated-hosts/>

²⁵ AWS Database Migration Service:

<https://aws.amazon.com/dms/>

²⁶ High Availability:

http://msdn.microsoft.com/en-us/library/cc645993%28v=sql.120%29.aspx#High_availability