RDBMS in the Cloud:

Deploying SQL Server on AWS

Darryl Osborne

Vlad Vlasceanu

June 2015

Notices

Contents

Abstract 5

SQL Server Solutions on AWS 6

Amazon RDS for SQL Server 6

SQL Server on Amazon EC2 6

Hybrid Scenarios 7

Choosing between SQL Server Solutions on AWS 7

Amazon RDS for SQL Server 9

Starting a Microsoft SQL Server RDS Instance 11

Security 15

Performance Management 19

High Availability 26

Monitoring and Management 27

Managing Cost 30

SQL Server on Amazon EC2 33

Starting a Microsoft SQL Server Instance 33

Security 38

Performance Management 40

High Availability 44

Monitoring and Management 48

Managing Cost 49

Caching 52

Hybrid Scenarios 53

Backups to the Cloud 53

SQL Server Log Shipping – Between On-Premises and Amazon EC2 57

SQL Server AlwaysOn Availability Groups – Between On-Premises and Amazon EC2 58
Amazon RDS Migration Tool 59
Conclusion 60
Further Reading 61
Appendix 62
I. SQL Server Feature Availability on AWS 62
II. SQL Server Roles and Permissions Not Supported in Amazon RDS 66
Notes 67

Abstract

Amazon Web Services (AWS) is a flexible, cost-effective, easy-to-use cloud computing platform. Relational database management systems, or RDBMS, are widely deployed within the Amazon cloud. In this whitepaper, we help you understand how to deploy 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-premise SQL Server environment and AWS, using native SQL Server features like log shipping, replication, and AlwaysOn Availability Groups. After reading this whitepaper you will be able to make an educated decision and choose the solution that best fits your needs.

SQL Server Solutions on AWS

Amazon Web Services (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 and 2012 in AWS. One is to use the Amazon Relational Database Service (Amazon RDS, or RDS). The other is to run SQL Server on the Amazon Elastic Compute Cloud (Amazon EC2, or EC2). The latter option is also available for other versions of SQL Server, such as 2014, 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 version upgrades, failed instance replacement, as well as 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.

If you want Amazon to handle the day-to-day management of your SQL Server databases, Amazon RDS is the preferred way. This enables you to focus on higher-level tasks, such as schema optimization, query tuning, and application development, and eliminate the undifferentiating work that goes into maintenance and operation of the databases. However, certain SQL Server services and features are not supported in Amazon RDS. We will cover feature availability in greater detail throughout this document. For more information about Amazon RDS, see http://aws.amazon.com/rds/sqlserver/.

SQL Server on Amazon EC2

Amazon EC2 is a service that provides resizable computing capacity in the cloud. Using Amazon EC2 is similar to running a SQL Server database in-house. You have full control over the operating system, database installation, and configuration. 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. With Amazon EC2, you can quickly provision and configure database 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.

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. For more information about Amazon EC2, see http://aws.amazon.com/documentation/ec2/.

Hybrid Scenarios

Customers can also choose to run their SQL Server workload in a hybrid environment. Some customers have outstanding commitments on hardware or datacenter space that makes it impractical to be all in on cloud all at once. This doesn't mean they can't take advantage of the scalability, availability, cost benefits of running a portion of their 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 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. It 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 understanding 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 throughout this document.
- Do you need high availability and automated failover capabilities? If you are running a production workload, high availability is a recommended best practice.

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

- You would rather 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 setup and maintain database mirroring, failover clusters, or AlwaysOn Availability Groups.
- You do not 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:

- You need full control over the database instances, including access to the operating system and software stack.
- 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.

- You want to setup a disaster recovery solution with SQL Server running in AWS as the source.
- You need to use a SQL Server version not supported by Amazon RDS (e.g. 2014 at the time of this writing)

For a detailed side-by-side comparison of SQL Server features available in the AWS environment, please refer to **Appendix I. SQL Server Feature Availability on AWS** at the end of this document.

Amazon RDS for SQL Server

At the time of this writing RDS supports the following SQL Server database engines:

- Microsoft SQL Server 2008 R2 SP1 CU3 (DB Engine Version 10.50.2789.0.v1)
- Microsoft SQL Server 2012 RTM (DB Engine Version 11.00.2100.60.v1)

For information on new features in SQL Server 2012 compared to previous versions, please see: http://technet.microsoft.com/en-us/library/bb500435(v=sql.110).aspx

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 both the 2008 R2 and 2012 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 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 both the 2008 R2 and 2012 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 both 2008 R2 and 2012 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.

For a detailed feature comparison between the different SQL Server editions, please see: http://msdn.microsoft.com/en-us/library/cc645993%28v=sql.105%29.ASPX for SQL Server 2008 R2 and http://msdn.microsoft.com/en-us/library/cc645993(v=sql.110).ASPX for SQL Server 2012.

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, subject to the various editions:

- 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 (client side traces), SQL Server Migration Assistant (SSMA), Database Engine Tuning Advisor, SQL Server Agent
- Safe CLR
- Full-text search (except semantic search)
- SSL connection support
- Transparent Data Encryption (available in Enterprise Edition only)
- Encryption of storage at rest using the Amazon Key Management Service
- Spatial and location features
- Change Tracking
- Database Mirroring (Standard and Enterprise Editions only, 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

SQL Server 2012 supports the following additional key database engine features:

- Columnstore indexes (available in Enterprise Edition only)
- Online index creation, dropping and rebuilding XML indexes (available in Enterprise Edition only)
- varchar(max), nvarchar(max), and varbinary(max) data types (available in Enterprise Edition only)
- Flexible server roles
- Partially Contained Databases
- Sequences
- THROW statement
- New and enhanced spatial types
- UTF-16 support
- ALTER ANY SERVER ROLE server-level permission

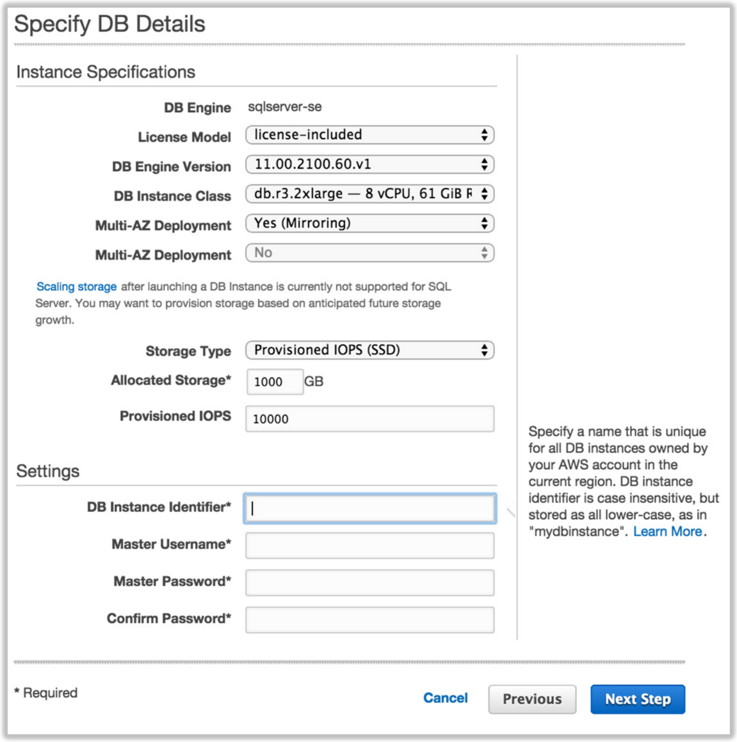
We continuously improve on Amazon RDS for SQL Server capabilities, please review the following page for up-to-date information on supported features and options:
http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_SQLServer.html#SQLServer.Concepts.General.FeatureSupport.

Starting a Microsoft SQL Server RDS Instance

Starting a SQL Server database instance on Amazon RDS is easy. You can do it by using the AWS Management Console, or programmatically using CloudFormation templates, our CLI tools or AWS SDKs supporting various programming languages. The walkthrough below uses the AWS Management Console for illustration purposes.

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

Choose the **SQL Server** tab on the first screen, pick the edition you wish to deploy and click the corresponding **Select** button. If you select SQL Server SE (Standard Edition), you are asked 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.


Specify DB Details

Instance Specifications

DB Engine	sqlserver-se
License Model	license-included
DB Engine Version	11.00.2100.60.v1
DB Instance Class	db.r3.2xlarge — 8 vCPU, 61 GiB R
Multi-AZ Deployment	Yes (Mirroring)
Multi-AZ Deployment	No

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.

Storage Type	Provisioned IOPS (SSD)
Allocated Storage*	1000 GB
Provisioned IOPS	10000

Settings

DB Instance Identifier*	
Master Username*	
Master Password*	
Confirm Password*	

Specify a name that is unique for all DB instances owned by your AWS account in the current region. DB instance identifier is case insensitive, but stored as all lower-case, as in "mydbinstance". Learn More.

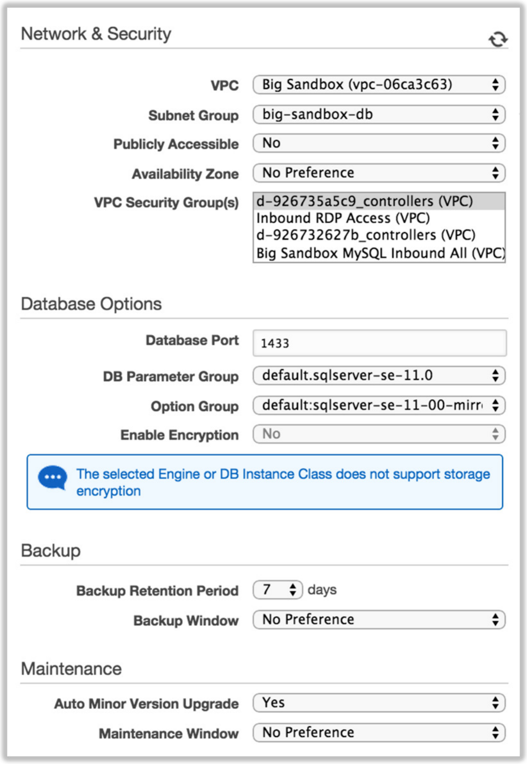
* Required

Figure 1: RDS DB instance details

Next, configure the DB instance:

- Select the **License Model**. Typically, customers have only the "license-included" option available.
- Select the **DB Engine Version**. This corresponds to the two versions of SQL Server available (10.50[...] for 2008 R2, 11.00[...] for 2012). The minor version numbers may change as the engines are updated with new patches.
- Select the **DB Instance Class**. This defines the performance characteristics of your database instance in terms of CPU, RAM and networking. More details on instance sizing can be found further in this document.

- Select the **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 database instance to the secondary. This option is not available for SQL Server Express and SQL Server Web edition, or in regions with fewer than 3 Availability Zones.
- Set the values for the **Storage Type, Allocated Storage,** and **Provisioned IOPS** fields, as applicable. These selections determine the amount of database storage, and the performance characteristics of the storage subsystem of the instance. The Provisioned IOPS value can only be set for the Provisioned IOPS storage type. The options will be discussed in greater detail further in this document.
- Set the **DB Instance Identifier, Master Username,** and **Master Password**. Amazon RDS uses SQL Server Authentication, and once deployed you can create additional user accounts.



Network & Security

VPC	Big Sandbox (vpc-06ca3c63)
Subnet Group	big-sandbox-db
Publicly Accessible	No
Availability Zone	No Preference
VPC Security Group(s)	d-926735a5c9_controllers (VPC) Inbound RDP Access (VPC) d-926732627b_controllers (VPC) Big Sandbox MySQL Inbound All (VPC)

Database Options

Database Port	1433
DB Parameter Group	default.sqlserver-se-11.0
Option Group	default:sqlserver-se-11-00-mirr
Enable Encryption	No

The selected Engine or DB Instance Class does not support storage encryption

Backup

Backup Retention Period	7 days
Backup Window	No Preference

Maintenance

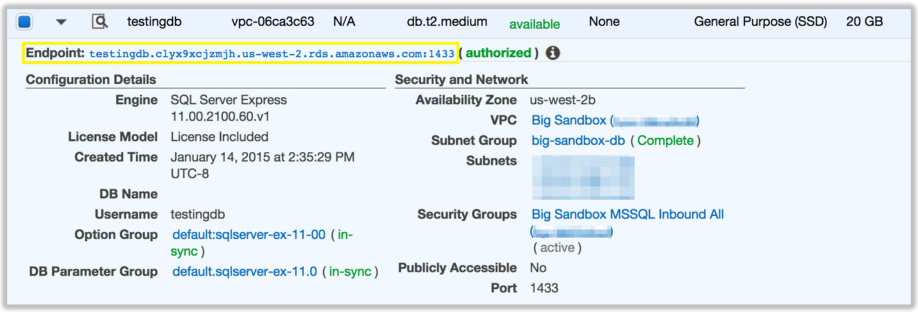
Auto Minor Version Upgrade	Yes
Maintenance Window	No Preference

Figure 2: RDS DB instance settings

Next, configure advanced options for your database instance:

- Set the **Network & Security** parameters. You can deploy the database instance in a VPC, or if your account permits, in EC2-Classic. The two networking platforms are described in greater detailed in the Security section below. In a VPC, you will also have to specify the DB Subnet Group. Select a Security Group to secure access to the instance. These security controls are discussed in greater detail in the Security section of this document.
- Set 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 are 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.
- Click **Launch DB Instance** to have Amazon RDS provision and deploy the database instance based on the options selected. This process may 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 ecosystem tools. Amazon RDS provides you with a **DNS Endpoint** for the server, as shown in the example below. Use this DNS Endpoint as the SQL Server hostname, along with the Master Username and Password configured for the instance in order to connect to the database. Always use the DNS Endpoint to connect to the instance, as the underlying IP address may change.



testingdb vpc-06ca3c63 N/A db.t2.medium available None General Purpose (SSD) 20 GB

Endpoint: `testingdb.clyx9xcjzmjh.us-west-2.rds.amazonaws.com:1433` (authorized)

Configuration Details

Engine	SQL Server Express 11.00.2100.60.v1
License Model	License Included
Created Time	January 14, 2015 at 2:35:29 PM UTC-8
DB Name	
Username	testingdb
Option Group	default:sqlserver-ex-11-00 (in-sync)
DB Parameter Group	default.sqlserver-ex-11.0 (in-sync)

Security and Network

Availability Zone	us-west-2b
VPC	Big Sandbox ([redacted])
Subnet Group	big-sandbox-db (Complete)
Subnets	[redacted]
Security Groups	Big Sandbox MSSQL Inbound All ([redacted]) (active)
Publicly Accessible	No
Port	1433

Figure 3: RDS DB instance properties

Security

There are several features and sets of controls available to manage the security of your Amazon RDS database 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 only have access to this networking platform.
- **EC2-Classic** allows you to launch resources within a pre-defined, multi-tenant network platform. You can control access at the resource, or instance level.

With the recent availability of Amazon VPC ClassicLink[1], a mechanism to link EC2-Classic resources to a VPC, we recommend you deploy RDS database instances in a VPC for the added richness of security controls.

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 non-database 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 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 database instances deployed in a VPC. They can be changed and re-assigned without restarting the instances associated with them. DB Security Groups are used for instances in EC2-Classic. They cannot be re-assigned without restarting the instance, and cannot filter outbound traffic.

To improve the security posture, we recommend restricting inbound traffic to only database-related traffic (port 1433, unless a custom port number is used) and only from known sources. Security groups also accept the ID of a different security group (source security group) as the source for traffic. This makes it easier to manage sources of traffic to your RDS database instance in a scalable way. 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 it the source security group.

Amazon RDS can make database instances publicly accessible by assigning Internet routable IP addresses to the instances. In most use cases this is not needed or desired, and we recommend setting this option to 'No' to limit the threat surface. 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 their IP address. 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, may be subject to more frequent patching. Such instances can be force-patched when security patches are made available by the vendors. This patching can occur even outside the defined DB instance maintenance window, to address the safety and integrity of customer resources and our infrastructure.

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 (or deny) administrative users under your AWS account to create, describe, modify, or delete an Amazon RDS database. You can also enforce Multi-Factor Authentication (MFA). For more information on using IAM to manage administrative access to Amazon RDS, see http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/UsingWithRDS.IAM.html.

Data Access Controls

Currently, Amazon RDS for SQL Server supports SQL authentication only, and access control for SQL authenticated users should be configured using the Principle of Least Privileges. A master account is created automatically when the instance is launched. This login should be used for administrative purposes only and is granted the following roles: `processadmin`, `setupadmin` and `public`. RDS will manage the master user as a login, and create a user linked to the login in each customer database. This master user will also be put in the `db_owner` group.

You can create additional users and databases after launch by connecting to the database instance using the tool of your choice (e.g. SQL Server Management Studio). These users should be assigned only the permissions needed for the workload or application they are supporting to operate correctly.

Data-At-Rest Protection

There are several options for protecting data-at-rest in a DB instance:

- Encrypted Amazon RDS DB instances using Amazon KMS;
- SQL Server Transparent Data Encryption (TDE);
- SQL Server column-level;
- Encrypting data in the application before it is saved to the database instance.

Recently, Amazon RDS for SQL Server added support for encrypting DB instances with encryption keys managed in Amazon KMS. Data that is encrypted at rest includes the underlying storage for a DB instance, its automated backups, read replicas, and snapshots.

Amazon RDS encrypted instances use the industry standard AES-256 encryption algorithm to encrypt your data on the server that hosts your Amazon RDS instance. Once your data is encrypted, Amazon RDS handles authentication of access and decryption of your data transparently with a minimal impact on performance. You don't need to modify your database client applications to use encryption.

Amazon RDS encrypted instances also 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 the AWS Key Management Service (AWS KMS)[2].

Amazon RDS also supports encryption of data at rest using the Transparent Data Encryption (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 one does not already exist), and then associating the database instance with that group. More details on Amazon RDS support for TDE can be found here: http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Appendix.SQLServer.Options.html

Additionally, you can always use SQL Server's column level encryption or encrypt your data in the application before sending it to your SQL Server database. In the latter scenario, the application is responsible for data encryption and decryption activities, and the DB instance does not have any means to access the encrypted data. Both options give you more control to only encrypt certain fields that contain sensitive data. However, application level encryption may limit your ability to use SQL queries to join, sort, group or specify conditions on fields containing encrypted data. You can also use AWS KMS to manage your application's encryption keys.

Data-In-Transit Protection

Amazon RDS for SQL Server fully supports encrypted connections to the database 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 guard against spoofing attacks. More details on how to use SSL encryption are available in the Amazon RDS for SQL Server documentation at: http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_SQLServer.html#SQLServer.Concepts.General.SSL

Performance Management

The performance of your SQL Server DB instance is largely determined by the instance type you select, which affects the compute capacity, amount of memory and network capacity available to your database. The storage size and type you select when you provision the database will also affect performance.

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 1GB of memory to 32 vCPUs and 244 GB of memory. Not all DB instance classes are available for all SQL Server editions, however.

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

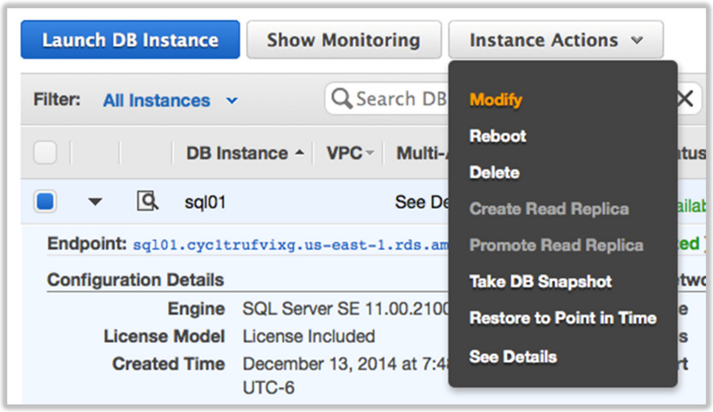
DB Instance Class	vCPU Count	Memory (GB)	Network Performance	SQL Server Express Edition	SQL Server Web Edition	SQL Server Standard Edition	SQL Server Enterprise Edition
db.t2.micro	1	1	Low	Yes			
db.t2.small	1	2	Low	Yes	Yes		Yes
db.t2.medium	2	4	Moderate		Yes		Yes
db.m3.medium	1	3.75	Moderate		Yes	Yes	Yes
db.m3.large	1	3.75	Moderate		Yes	Yes	Yes

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.m3.xlarge	4	15	Moderate		Yes	Yes	Yes
db.m3.2xlarge	8	30	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
db.r3.8xlarge	32	244	10 Gbps				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	.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 the typical SQL workload you plan to run is important when identifying the proper DB instance class. If you are unsure how much CPU you need, we recommend 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, click Instances. Select the check box for the DB instance that you want to change, and click **Modify**.



sql01 See De

Endpoint: sql01.cycltrufvixg.us-east-1.rds.am

Configuration Details

Engine SQL Server SE 11.00.2100

License Model License Included

Created Time December 13, 2014 at 7:4 UTC-6

Figure 4: Modify DB instance

The settings are as follows:

Setting	Description
DB Engine Version	In the list provided, click the version of the SQL Server database engine that you want to use.
DB Instance Class	In the list provided, click the DB instance class that you want to use. For information about instance classes, see DB Instance Class.
Multi-AZ Deployment	If you want to create a secondary mirror of your DB instance in another Availability Zone, click Yes; otherwise, click No. For more information on Multi-AZ deployments using SQL Server Mirroring, see Planning your Multi-AZ Deployments Using SQL Server Mirroring.
Allocated Storage	You cannot change the allocated storage for a SQL Server DB instance.
Storage Type	You cannot change the storage type for an existing 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. For more information about security groups, see Working with DB Security Groups.
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

Setting	Description
	will NOT be rebooted automatically and the parameter changes will NOT be applied during the next maintenance window. For more information about parameter groups, see Working with DB Parameter Groups.
Option Group	Select the option group you want associated with the DB instance. For more information about option groups, see Working with Option Groups.
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 non-zero value or from a non-zero 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, click 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 as well as 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 database instance once the instance has been deployed. This is due to extensibility limitations of striped storage attached to a Windows Server environment.

We recommend you plan for the growth of storage needs in advance, and provision enough capacity to handle growth from the onset. You can always change the size of your DB instance by restoring from a snapshot, or provisioning a new DB instance from a snapshot with increased storage, but these options require a window of time during which your database is offline.

Amazon RDS for SQL Server supports 3 types of storage, each having different characteristics and recommended use cases:

- **Magnetic Storage** (also known as standard storage) leverages 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, as well as burst capabilities. It can deliver single-digit millisecond latencies, with a base performance of 3 IOPS/GB and the ability to burst to 3,000 IOPS based on a predictable credits system[3]. 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. Amazon RDS delivers within 10 percent of the provisioned IOPS performance 99.9 percent of the time over a given year. Allocations follow a fixed 10:1 ratio of IOPS to GB of storage, with PIOS storage allocated in increments of 100 GB and 1000 IOPS.

AWS recently introduced larger storage options with capacities of up to 4TB and 20,000 IOPS.

Amazon RDS storage performance characteristics at a glance:

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	4 TiB*	3 IOPS/GiB	Yes; up to 3000 IOPS per	SSD	Allocated storage

Storage Type	Min. Volume Size	Max. Volume Size	Baseline Performance	Burst Capability	Storage Technology	Pricing Criteria
	(100 GiB recommended)			volume, subject to accrued credits		
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 10GB/database, limiting overall useable database storage to 300GB per DB instance (max. 30 databases/instance allowed). Any additional storage may be unusable.

While 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 compare it with on-premise systems. 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 you test your own workload to determine how these metrics behave in your specific use case.

When sizing your storage subsystem, keep the following metrics and limits under consideration:

- Subject to the DB instance class selected, the current **maximum channel bandwidth** available for storage in RDS is 1000 Mbps full duplex[4]. This equates to about 105 MiB/s in each direction, thus a perfectly balanced workload, where database queries are evenly split between 50% data reads and 50% data writes, can attain a maximum combined throughput of about 210 MiB/s.
- In 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 via CloudWatch, represents the number of I/O requests in the queue, waiting to be serviced. These requests have been submitted by the application but have not 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 are indicative of an overloaded system from a storage perspective.

Depending on the storage configuration selected, your overall storage subsystem throughput will be limited either by the maximum IOPS, or the maximum channel bandwidth. If your workload is generating a lot of very small sized I/O operations (e.g. 8 KiB) you are likely to reach maximum IOPS before the overall bandwidth reaches the channel maximum. On the other hand if I/O operations are large in size (e.g. 256 KiB), you are likely to reach the maximum channel bandwidth before maximizing IOPS.

Microsoft SQL Server stores data in 8 KiB pages but it 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[5]. This results in better performance by reading and writing multiple pages at the same time. Amazon RDS accommodates these multi-page operations by counting every read or write operation on up to 32 pages as a single I/O operation to the storage system.

SQL Server will also attempt to optimize I/O by reading ahead, and attempting to keep the queue length non-zero. Therefore, low or zero queue numbers indicate that the storage subsystem is underutilized, and potentially overprovisioned with SQL Server not needing to use the available I/O capacity.

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 provides for burst capabilities of up to 3000 IOPS from the get-go. 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. You only accumulate new burst I/O credits while you consume less than the baseline IOPS provided.

For example: a 20GB General Purpose SSD volume comes with just 60 baseline IOPS. While you can burst to 3000 IOPS initially, your workload would need to fit within the 60 IOPS budget in the long term.

Additionally, several other factors may reduce I/O performance transiently:

- **First Touch Penalty**: The first time each storage block is accessed it will take longer to service the request as the storage block is initialized. This penalty is only incurred once per block and will not be incurred again for the lifetime of the instance.
- **DB Snapshot Creation**: This activity also consumes capacity, and may reduce performance.
- **Multi-AZ** peer creation: When switching from a Single-AZ to a Multi-AZ DB instance, data is copied over to the secondary node, causing a decrease in performance while the initial data synchronization is performed between the nodes.

High Availability

Amazon RDS offers Multi-AZ support for Amazon RDS for SQL Server. This high availability (HA) option leverages 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 database 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 automatic failover takes to complete: typically one to two minutes. When failing over, Amazon RDS simply flips the canonical name record (`CNAME`) for your DB Instance to point at the secondary, which is in turn promoted to become the new primary. The canonical name record (or endpoint name) is an entry in the Domain Name System (DNS)[6], a hierarchical distributed naming system that provides translation for numeric IP addresses associated to resources connected to the Internet or private networks.

We recommend you implement database connection error retry logic in your application layer, and rather than attempt to connect directly to the IP address of the DB instance, use the canonical name instead. This is 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 do not failover automatically in response to database operations such as long running queries, deadlocks or database corruption errors.

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

Monitoring and Management

Amazon CloudWatch collects many Amazon RDS-specific metrics. You can look at these metrics using the AWS Management Console, the command line (using the `mon-get-stats` command) or the API. In addition to the system-level metrics collected traditionally 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 the CloudWatch documentation at: http://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/rds-metricscollected.html.

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 (SNS) topics or Auto Scaling policies. You can configure these alarms to notify database administrators via email or even 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 would need to implement such event handlers separately.

For effective management, you should configure database administrative windows of time to meet your workload and business needs. Amazon RDS will assign default administrative windows to each database instance, if these windows aren't customized.

- **Backup Window**: The backup window is the period of time during which your DB instance is backed up. Since backups may 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) or software patching occur. Most maintenance occurs during the maintenance window, and your instance may be restarted during this window, if the work necessitates a restart. Because of the performance impact and possibility of restarts, we recommend scheduling the maintenance window for the time when your instance has the least traffic.

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

- **Automated backup and recovery**. Amazon RDS automatically back 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 five minutes.
- **Push-button scaling**. With a few simple clicks you can change the database instance class in order to increase or decrease the size of your instance in terms of compute capacity, network capacity, and memory. You can choose to make the change immediately, or schedule it for your 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 may be force-patched when security patches are made available by the vendors, to address 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 may become unavailable during part of your maintenance window when a patch is applied. You can also 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.
- **Pre-configured parameters and options**. Amazon RDS provides a default set of DB Parameter Groups, as well as Option Groups for each SQL

Server edition and version. These groups contain configuration parameters and options respectively, that allow you to tune the performance and features of your DB instance. By default Amazon RDS provides an optimal set of configuration suitable for most workloads, based on the class of the DB instance that you selected. You can create your own DB Parameter Groups 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 would use with on-premise SQL Server instances, such as SQL Server Management Studio.

However, in order to provide you with a secure and stable managed database experience, Amazon RDS does not provide desktop or Administrator access to DB instances, and it restricts access to certain system procedures and tables that require advanced privileges. You also must use SQL Server Authentication, as Windows authentication is currently not supported.

Commands to create users, rename users, grant, revoke permissions and set password, work as they do in Amazon EC2 (or on-premise) databases. Unsupported administrative commands are listed in Appendix II. SQL Server Roles and Permissions Not Supported in Amazon RDS. 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. These files can then be loaded into any other SQL Server instances, or stored as backups in S3, Glacier or on-premise.

For a complete list of server roles and permissions that are currently not available in Amazon RDS for SQL Server, please refer to Appendix II.

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 **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 it is running during a given billing period

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

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. 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 database 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% over On-Demand rates when used in steady state, which tend to be the case for many databases. They can be purchased for one or three year terms. If your SQL Server database

is going to be running more than 25% 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, since 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 may 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.

Currently, there are 3 types of Reserved Instances available:

- **Heavy Utilization RIs** are recommended for workloads that are running 24/7 over a given month. They also provide the best discount compared to on-demand pricing. You will be charged for a full month of operation whether you had a DB instance running during that period of time or not.
- **Medium Utilization RIs** are recommended for workloads that are running about 60% of the time within a given month. You will still see savings as compared to on-demand pricing, but you will not be charged for the time a DB instance is not running.
- **Light Utilization RIs** are similar to Medium Utilization RIs, and are recommended for workloads that are running about 30-40% of the time over a given month.

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 is especially effective for databases that are not running 24/7 and for short projects.

- Terminate database instances with a last snapshot when they are not needed, then re-provision them from that snapshot when they need to be used again. For example, some development and test databases could be terminated at night and on weekends, and re-provisioned on weekdays in the morning.
- Scale down the size of your DB instance during off-peak times.

Please refer to the Amazon RDS SQL Server pricing web page for up to date pricing information for all pricing models and instance classes: http://aws.amazon.com/rds/sqlserver/pricing/.

SQL Server on Amazon EC2

Starting a Microsoft SQL Server Instance

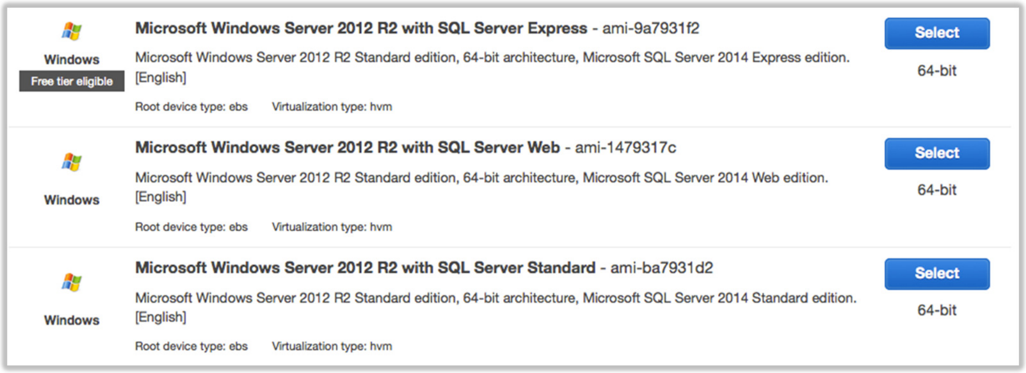
Starting a SQL Server database instance on Amazon EC2 is easy. You can do it by using the AWS Management Console, or programmatically using CloudFormation templates, CLI tools or AWS SDKs supporting various programming languages. The walkthrough below uses the AWS Management Console for illustration purposes.

Using the AWS Management Console, select the **EC2** service from the **Services** menu, and click **Launch Instance** to invoke the launch wizard.

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



Windows
Free tier eligible

Microsoft Windows Server 2012 R2 with SQL Server Express - ami-9a7931f2
Microsoft Windows Server 2012 R2 Standard edition, 64-bit architecture, Microsoft SQL Server 2014 Express edition. [English]
Root device type: ebs Virtualization type: hvm
Select
64-bit

Windows

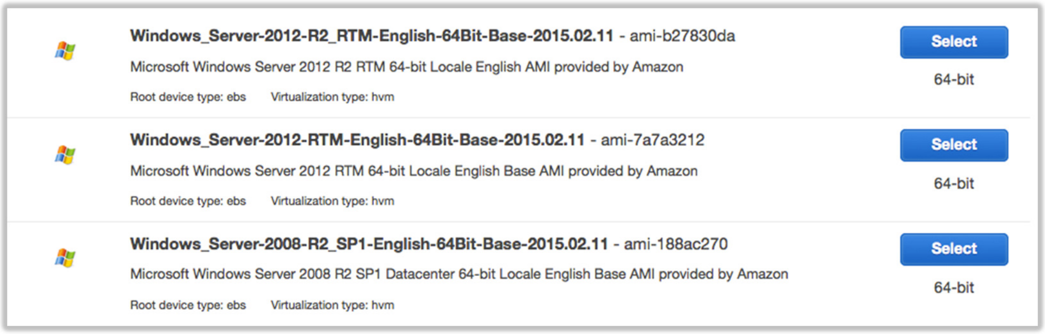
Microsoft Windows Server 2012 R2 with SQL Server Web - ami-1479317c
Microsoft Windows Server 2012 R2 Standard edition, 64-bit architecture, Microsoft SQL Server 2014 Web edition. [English]
Root device type: ebs Virtualization type: hvm
Select
64-bit

Windows

Microsoft Windows Server 2012 R2 with SQL Server Standard - ami-ba7931d2
Microsoft Windows Server 2012 R2 Standard edition, 64-bit architecture, Microsoft SQL Server 2014 Standard edition. [English]
Root device type: ebs Virtualization type: hvm
Select
64-bit

Figure 5: Amazon AMIs with SQL Server

Other Amazon Machine Images (AMIs) install just the Windows operating system. This allows you the flexibility to perform a separate, custom installation of SQL Server on the Amazon EC2 instance.



Windows_Server-2012-R2_RTM-English-64Bit-Base-2015.02.11 - ami-b27830da
Microsoft Windows Server 2012 R2 RTM 64-bit Locale English AMI provided by Amazon
Root device type: ebs Virtualization type: hvm
Select
64-bit

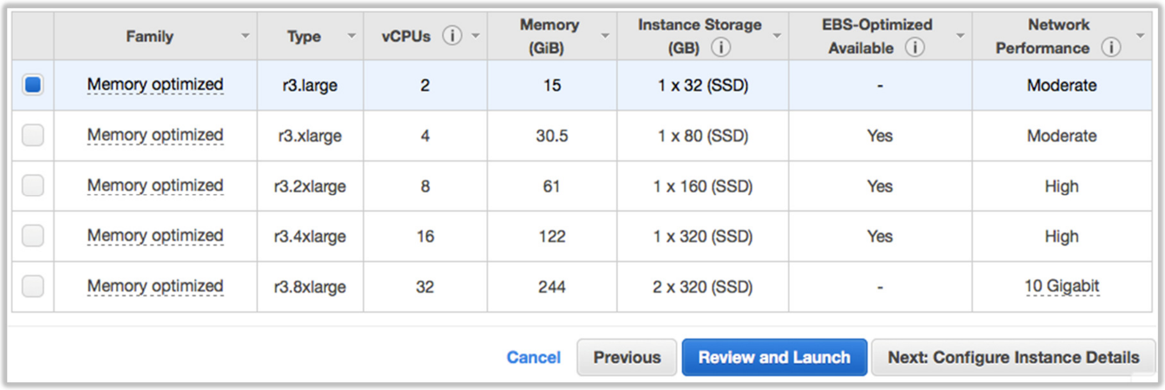
Windows_Server-2012-RTM-English-64Bit-Base-2015.02.11 - ami-7a7a3212
Microsoft Windows Server 2012 RTM 64-bit Locale English Base AMI provided by Amazon
Root device type: ebs Virtualization type: hvm
Select
64-bit

Windows_Server-2008-R2_SP1-English-64Bit-Base-2015.02.11 - ami-188ac270
Microsoft Windows Server 2008 R2 SP1 Datacenter 64-bit Locale English Base AMI provided by Amazon
Root device type: ebs Virtualization type: hvm
Select
64-bit

Figure 6: Amazon AMIs with Windows Server

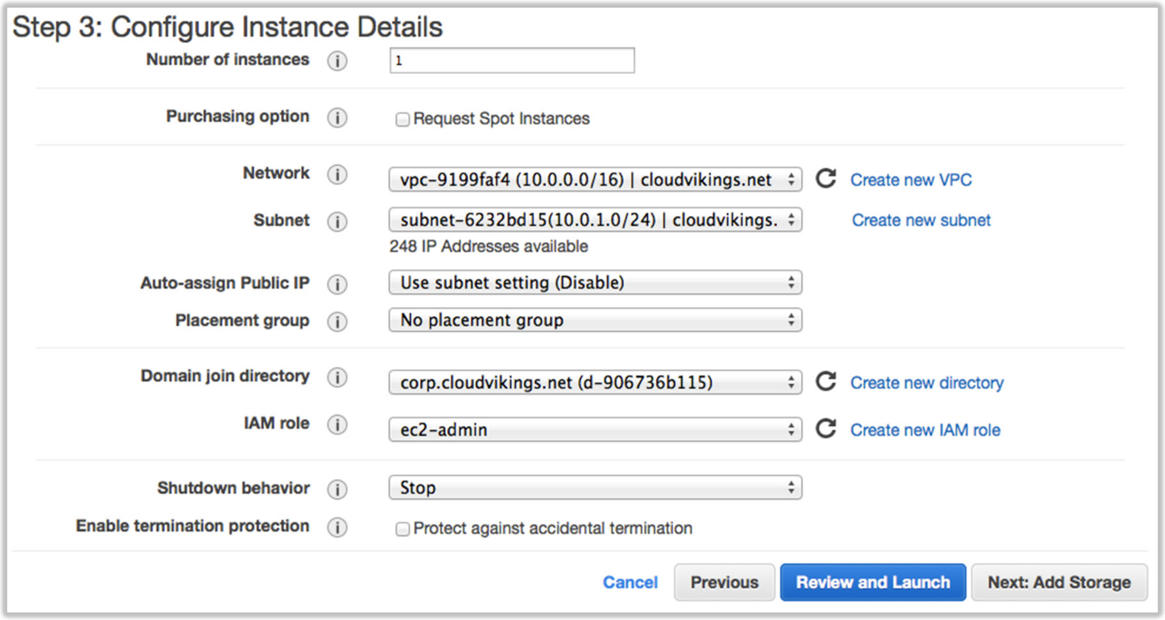
First select the appropriate AMI, then select the desired instance type. Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. They have varying combinations of compute capacity, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources. Below 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
■	Memory optimized	r3.large	2	15	1 x 32 (SSD)	-	Moderate
☐	Memory optimized	r3.xlarge	4	30.5	1 x 80 (SSD)	Yes	Moderate
☐	Memory optimized	r3.2xlarge	8	61	1 x 160 (SSD)	Yes	High
☐	Memory optimized	r3.4xlarge	16	122	1 x 320 (SSD)	Yes	High
☐	Memory optimized	r3.8xlarge	32	244	2 x 320 (SSD)	-	10 Gigabit

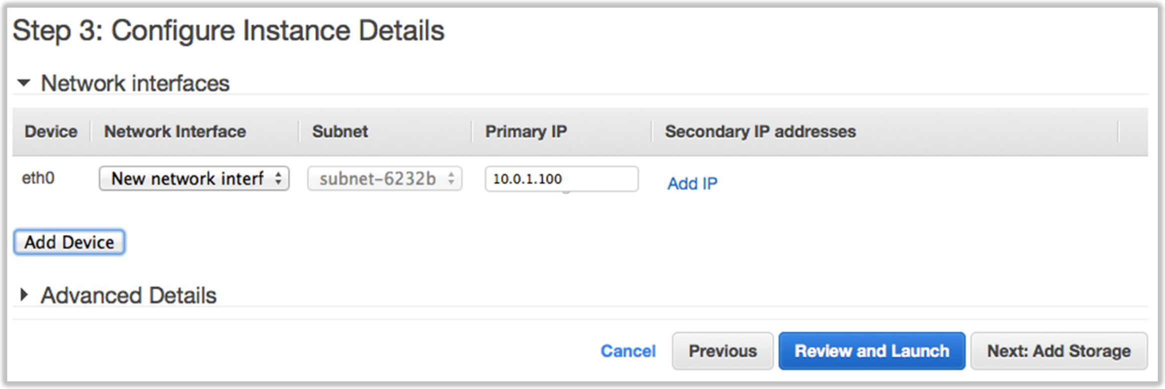
Figure 7: Instance Type Selector



Step 3: Configure Instance Details

Number of instances	1
Purchasing option	☐ Request Spot Instances
Network	vpc-9199faf4 (10.0.0.0/16) | cloudvikings.net
Subnet	subnet-6232bd15(10.0.1.0/24) | cloudvikings. 248 IP Addresses available
Auto-assign Public IP	Use subnet setting (Disable)
Placement group	No placement group
Domain join directory	corp.cloudvikings.net (d-906736b115)
IAM role	ec2-admin
Shutdown behavior	Stop
Enable termination protection	☐ Protect against accidental termination

Figure 8: Configure Instance Details

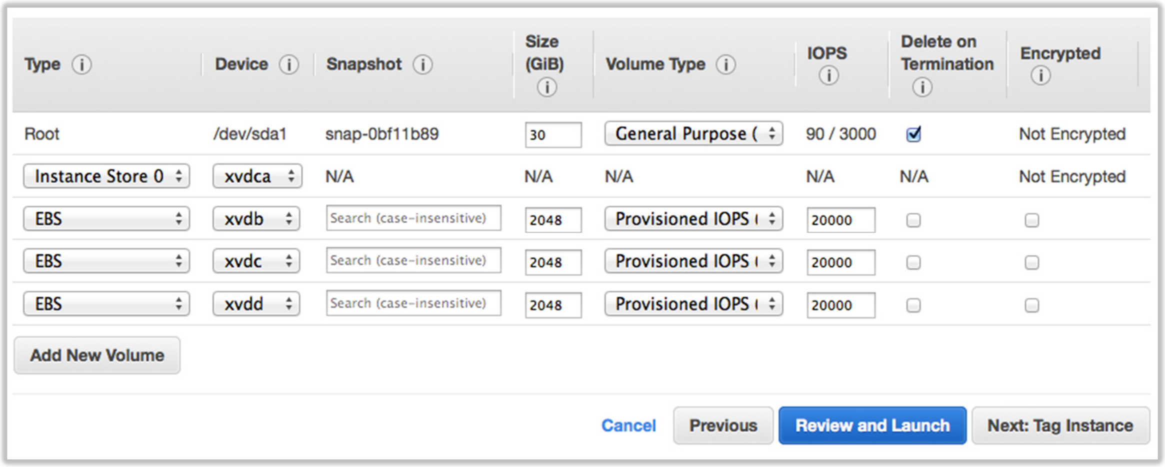


Device	Network Interface	Subnet	Primary IP	Secondary IP addresses
eth0	New network interf	subnet-6232b	10.0.1.100	Add IP

Figure 9: Configure Instance Details

Configure the instance details. We recommend running Amazon EC2 instances in an Amazon Virtual Private Cloud (VPC). Amazon Virtual Private Cloud (Amazon VPC) lets you provision a logically isolated section of the Amazon Web Services (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, select an already existing VPC or select **Create new VPC** to create a new VPC. For details on designing a VPC, review the **VPC Getting Started Guide**. Configure the remaining instance details options depending on your workload needs.

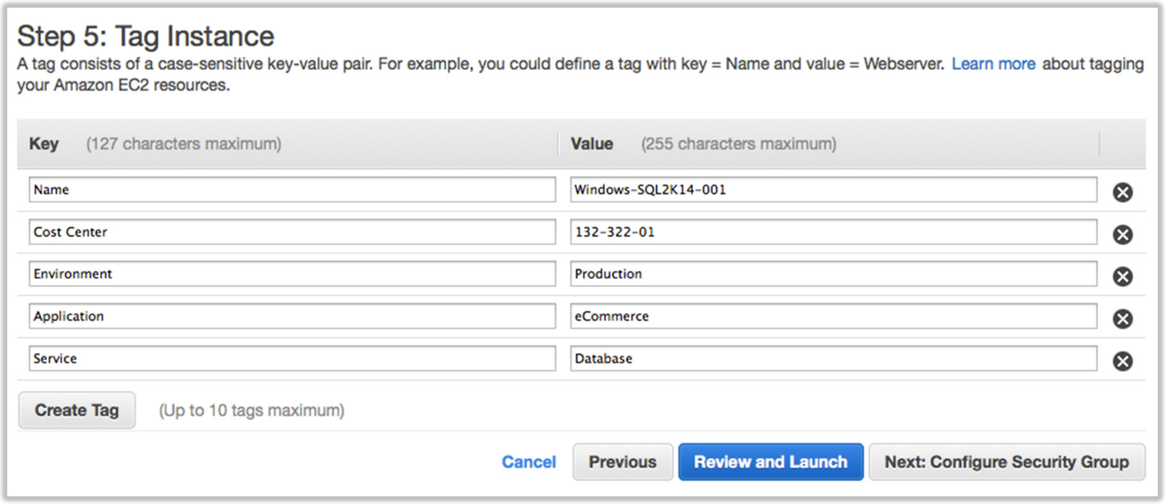
Depending on the type of SQL Server deployment, e.g. stand-along, Windows Failover Clustering & AlwaysOn Availability Groups, etc., you may decide to assign one or multiple static IP addresses to your Amazon EC2 instance. This can be done in the **Network interface** section of **Configure Instance Details**.



Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Delete on Termination	Encrypted
Root	/dev/sda1	snap-0bf11b89	30	General Purpose (	90 / 3000	✓	Not Encrypted
Instance Store 0	xvdca	N/A	N/A	N/A	N/A	N/A	Not Encrypted
EBS	xvdb	Search (case-insensitive)	2048	Provisioned IOPS (	20000	☐	☐
EBS	xvdc	Search (case-insensitive)	2048	Provisioned IOPS (	20000	☐	☐
EBS	xvdd	Search (case-insensitive)	2048	Provisioned IOPS (	20000	☐	☐

Figure 10: Add Storage

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 below.



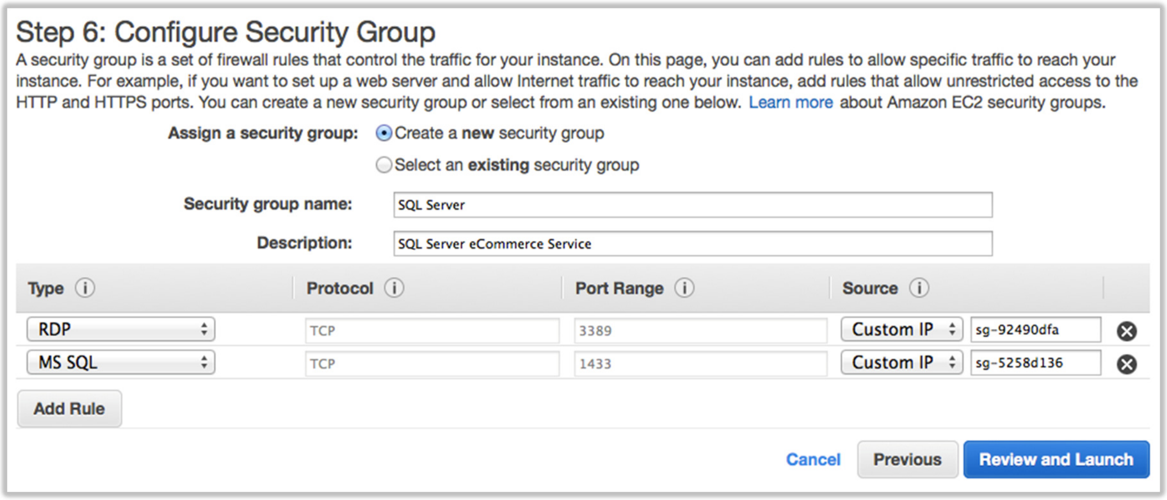
Step 5: Tag Instance

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

Figure 11: Tag Instance

Assign the appropriate tags to the Amazon EC2 instance. It is also recommended to assign tags to other Amazon resources (e.g. EBS volumes, etc.) to allow for more control over resource level permissions and cost allocation. For more information see Tagging Your Amazon EC2 Resources in the Amazon Elastic Compute Cloud User Guide for Microsoft Windows.



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

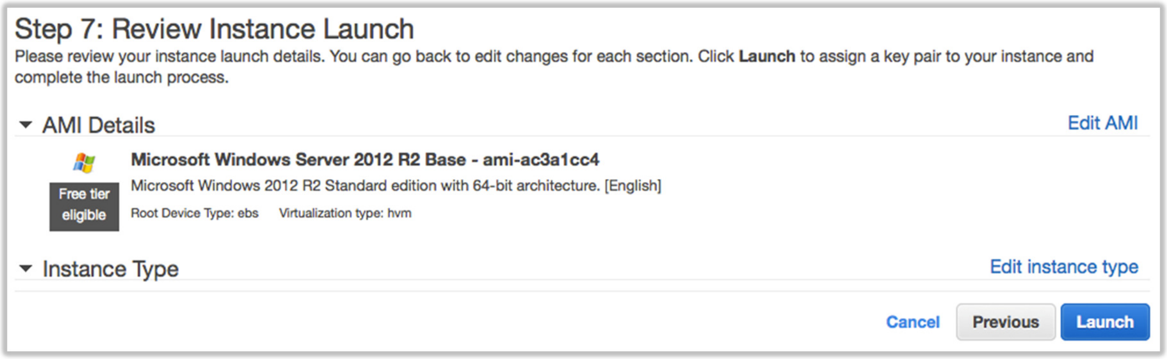
Security group name: SQL Server

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

Figure 12: Configure Security Group

Security Groups for Windows instances act as virtual firewalls that control the network traffic for one or more instances.



Step 7: Review Instance Launch

Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.

AMI Details

Microsoft Windows Server 2012 R2 Base - ami-ac3a1cc4

Microsoft Windows 2012 R2 Standard edition with 64-bit architecture. [English]

Root Device Type: ebs Virtualization type: hvm

Figure 13: Review Instance Launch

Review the instance details and click Launch, or make any necessary changes by selecting the Edit link next to the appropriate section.

Security

When running 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, please review the Amazon RDS Security section above.

Using AWS Identity and Access Management (IAM), you can control access to your Amazon EC2 resources, and authorize (or deny) some users the ability to manage your instances running the SQL Server database and the corresponding EBS volumes. For example, you could 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 http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/UsingIAM.html.

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 3rd party vendors release new security or functional patches. This includes 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 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. Please 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 Amazon Key Management Service (KMS). This allows you to leverage 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, define access controls, and audit the encryption keys used to protect your data.

Performance Management

The performance of a relational database 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, 48TB 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 the 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 IO bound look at 8xlarge instances, available in a number of instances families, like the c4, r3, i2, or d2, which gives you 10 Gigabit network performance.

One of the differentiators between all these 10 Gigabit instance types is that the c4 and d2 instance types are EBS-Optimized. A detailed explanation of EBS-Optimized instances is presented in the Disk I/O Management section. If your workload is network bound, again look instance families that support 10 Gigabit network performance as these instance families also support Enhanced Networking. These include 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. In order to take advantage of this feature, you should launch an HVM AMI in Amazon VPC, and install the appropriate driver. Driver installation may be necessary for Windows instances, see the Amazon Elastic Compute Cloud User Guide for Windows for detail instructions on enabling Enhanced Networking for Windows instances.

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 granular control over the storage volumes and strategy to use, it is possible to deploy workloads that require more than 8TiB 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:

- **Instance Storage**: Several Amazon EC2 instance types come with a certain amount of “local” (directly attached) storage, which is ephemeral. Any data saved on instance storage will not be available after you stop and restart that instance, or if the underlying hardware fails, which would cause 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:
    - Ephemeral disks offer good performance for sequential disk access, and don’t impact your network connectivity. Some customers have found it useful to use ephemeral 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**: Similarly to Amazon RDS, you can leverage 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 (data center) to prevent the loss of data from the failure of any single component. It is easy to back them up to Amazon Simple Storage Service (S3) using snapshots. These attributes make EBS volumes suitable for data files, log files and for the flash recovery area. While the maximum size of an EBS volume is 16TB, 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% of their provisioned performance 99.9% of the time. The combination of EBS-optimized instances and Provisioned IOPS volumes helps to maintain 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, please see: http://aws.amazon.com/ec2/instance-types/.

To scale up random I/O performance, you can increase the number of EBS volumes your data resides on, for example by using 8 x 100GB EBS volumes instead of 1 x 800GB EBS volume. However, remember that utilizing striping techniques 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, since the data on all other volumes of the stripe gets invalidated as well.

EBS volume data is natively replicated, so using RAID 0 (striping) might provide you with sufficient redundancy and availability. You could 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. In order to take advantage of additional EBS volumes, be sure to evaluate the network load so 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 6.4TB 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 in order 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 while higher numbers usually indicate your storage system is overloaded.

High Availability

High Availability (HA) is a design and configuration principle to help protect services or applications from single points of failure. By understanding the quote by Werner Vogels, CTO of Amazon.com, when he says, "everything fails all the time", 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 database instance to one or more secondary databases on separate database instances. While Log shipping is typically considered a disaster recovery feature, it can also provide high availability by allowing secondary database instances to be promoted as the primary in the event of a failure of the primary database 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, it 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' type delay, or a longer delay time, which could 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 database 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 'mirror' or almost complete 'mirror' of a database, depending on the operating mode, on a separate database instance. Database mirroring is the technology used by Amazon RDS to provide Multi-AZ support for Amazon RDS for SQL Server. It increases the availability and protection of mirrored databases, and provides a mechanism to keep mirrored databases available during upgrades.

SQL Servers can be 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 provides a mechanism to monitor 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 will redo all inserts, updates, and deletes that are executed against the principal database onto the mirror database. Database mirroring can either be a synchronous or asynchronous operation. These operations are used in the two mirroring operating modes.

- **High-safety mode** leverages synchronous operation. Using 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** leverages 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 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. This leverages advanced features of Windows Failover Cluster and the Enterprise Edition of SQL Server 2012 and 2014. 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. They can also be used 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 Domain Name Services.

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, please see the links below:

- **Whitepaper**:

  http://aws.amazon.com/windows/resources/whitepapers/alwayson/

- **Webinars:**

  http://aws.amazon.com/windows/events/

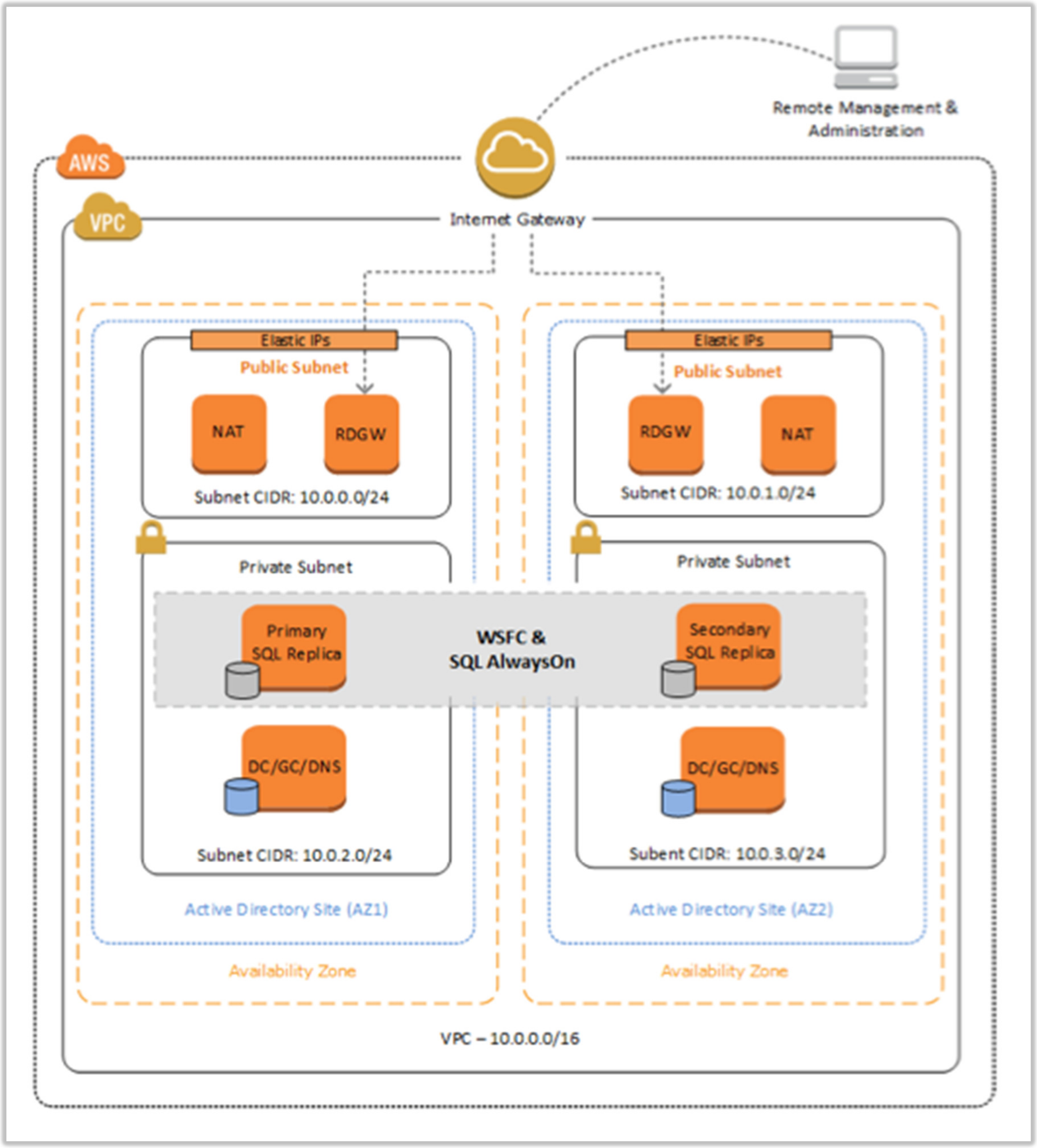


Figure 13: SQL Server AlwaysOn Availability Group

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, which allows for detailed reporting and management. This data is available in the web-based AWS Management Console as well as the API. This 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 only visible 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 allows you to track log events in real time for your SQL Server instances.

Similarly 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 Simple Notification Service (SNS) topics or Auto Scaling policies. You can configure these alarms to notify database administrators via email or even SMS when they get triggered.

In addition, you can use Microsoft ecosystem 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-premise database. You can use SQL Server Management Studio, SQL Server Configuration Manager, SQL Server Profiler, as well as other ecosystem 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-premise resources. These add-ins are currently available at no additional cost for System Center Operations Manager (SCOM) versions 2007 and 2012 and System Center Virtual Machine Manager (SCVMM). More information can be found at: http://aws.amazon.com/windows/system-center/.

While you can use AWS's EBS Snapshots as a mechanism to backup 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 for the disk I/O of SQL Server to be quiesced 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 snapshot, will be excluded from the EBS snapshot. Lacking the 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 leveraging 3rd 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. Similarly 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 **Region** the instance is deployed in
- **Instance Type** and **EBS Optimization**
- The type **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 in regards to where to deploy your workloads geographically, we recommend deploying your workload in the region with the cheapest EC2 costs for your particular use case.

Similarly, 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 provides a predictable amount of networking capacity to sustain predictable EBS I/O. At this time, the newly released C4 instance types are EBS Optimized out of the box, and do not have an additional surcharge for the optimization. For up to date pricing information for EBS Optimized instance types visit: http://aws.amazon.com/ec2/pricing/#EBS-Optimized_Instances

Dedicated Instances are Amazon EC2 instances that run in a virtual private cloud (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. Deploying EC2 SQL Server instances, in dedicated tenancy is recommended for customers with 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 here: http://aws.amazon.com/ec2/purchasing-options/dedicated-instances/

Amazon EC2 reserved instances allow you to lower costs and reserve capacity. Reserved Instances can save you up to 70% 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% of the time, you will most likely financially benefit from using a reserved instance. Unlike 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 do not 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 Amazon Machine Images 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 is especially effective for databases that are not running 24/7 and for short projects.
- Shutdown database instances when they are not needed. For example, some development and test databases could be shutdown 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 do not 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 read traffic and connections against 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 against the database, the more effective caching can be.
- Commonly, the more repetitive query patterns are, with queries returning infrequently changing data sets, the more you can benefit from caching.

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

There are several tools that 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 above, ElastiCache allows you to provision fully managed caching clusters supporting both Memcached and Redis (details below). 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. It 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. It 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

Some AWS customers already have SQL Server running in their one-premises or co-located data center but want to leverage the AWS cloud to enhance their architecture to provide a more highly available (HA) solution or one that offers disaster recovery (DR). SQL Server has several replication technologies that offer HA and DR 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 up-front 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 (S3)

Leveraging Amazon Simple Storage Service (S3) allows you to take advantage of the flexibility and pricing of cloud storage. This gives you ability to backup SQL Server databases to a secure, highly available, highly durable, reliable storage solution like Amazon S3. There are many third-party backup solutions designed to secure store SQL Server backups in Amazon S3. You can also design and develop a SQL Server backup solution yourself by leverage AWS tools like the AWS CLI, AWS Tools for Windows PowerShell, or a wide variety of SDKs for .NET or Java as well as the AWS Toolkit for Visual Studio.

Customers are using the AWS cloud to enable faster disaster recovery of their critical IT systems without incurring the infrastructure expense of a second physical site. By leveraging Amazon S3 for SQL Server backups, customers are able to take advantage of the following benefits of Amazon S3:

- **Secure**: Amazon S3 supports data transfer over SSL and automatic encryption of your data once it is uploaded. Encrypt the data at rest in Amazon S3 by managing your own encryption keys using the AWS Key Management Service.
- **Available**: Amazon S3 is designed for 99.99% availability of objects over a given year. Amazon S3 is also backed by the Amazon S3 Service Level Agreement, ensuring that you can rely on it when you need it. And you can choose an AWS region to optimize for latency, minimize costs, or address regulatory requirements.
- **Scalable**: With Amazon S3, you can store as much data as you want and access it when you need it. You can stop guessing your future storage needs and scale up and down as required, dramatically increasing business agility.
- **Durable**: Amazon S3 provides durable infrastructure to store important data and is designed for durability of 99.999999999% of objects. Your data is redundantly stored across multiple facilities and multiple devices in each facility.
- **Low Cost**: Amazon S3 allows you to store large amounts of data at a very low cost. You pay for what you need, with no minimum commitments or up-front fees.

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 organizations 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. The 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 supports three configurations:

- **Gateway-Cached Volumes**: You can store your primary data in Amazon S3, and retain your frequently accessed data locally. Gateway-Cached volumes provide substantial cost savings on primary storage, minimize the need to scale your storage on-premises, and retain low-latency access to your frequently accessed data.
- **Gateway-Stored Volumes**: In the event you need low-latency access to your entire data set, you can configure your on-premises data gateway to store your primary data locally, and asynchronously back up point-in-time snapshots of this data to Amazon S3. Gateway-Stored volumes provide durable and inexpensive off-site backups that you can recover locally or from Amazon EC2 if, for example, you need replacement capacity for disaster recovery.
- **Gateway-Virtual Tape Library** (Gateway-VTL): With Gateway-VTL you can have a limitless collection of virtual tapes. Each virtual tape can be stored in a Virtual Tape Library backed by Amazon S3 or a Virtual Tape Shelf backed by Amazon Glacier. The Virtual Tape Library exposes an industry standard iSCSI interface, which provides your backup application with on-line access to the virtual tapes. When you no longer require immediate or frequent access to data contained on a virtual tape, you can use your backup application to move it from its Virtual Tape Library to your Virtual Tape Shelf in order to further reduce your storage costs.

The AWS Storage Gateway enables your existing on-premise to cloud backup applications to store primary backups on Amazon S3's scalable, reliable, secure, and cost-effective storage service. You can create Gateway-Cached storage volumes and mount them as iSCSI devices to your on-premises backup application servers. All data is securely transferred to AWS over SSL and stored encrypted in Amazon S3 using AES 256-bit encryption. Using Gateway-Cached volumes provides an attractive alternative to the traditional choice of maintaining and scaling costly storage hardware on-premises.

For scenarios where you want to keep your primary data or backups on-premises, you can use Gateway-Stored volumes to keep this data locally, and backup this data off-site to Amazon S3. Gateway-Stored volumes provide an attractive alternative to dealing with the longer recovery times and operational burden of managing off-site tape storage for backups.

There are many benefits of using AWS Storage Gateway for SQL Server backups:

- **Secure**: The AWS Storage Gateway securely transfers your data to AWS over SSL and stores data encrypted at rest in Amazon S3 and Amazon Glacier using Advanced Encryption Standard (AES) 256, a secure symmetric-key encryption standard using 256-bit encryption keys.
- **Durably backed by Amazon S3 and Amazon** Glacier: The AWS Storage Gateway durably stores your on-premises application data by uploading it to Amazon S3 and Amazon Glacier. Amazon S3 and Amazon Glacier redundantly store data in multiple facilities and on multiple devices within each facility. Amazon S3 and Amazon Glacier also perform regular, systematic data integrity checks and are built to be automatically self-healing.
- **Compatible**: There is no need to re-architect your on-premises applications. Gateway-Cached volumes and Gateway-Stored volumes expose a standard iSCSI block disk device interface and Gateway-VTL presents a standard iSCSI virtual tape library interface.
- **Cost-Effective**: By making it easy for your on-premises applications to store data on Amazon S3 or Amazon Glacier, AWS Storage Gateway reduces the cost, maintenance, and scaling challenges associated with managing primary, backup and archive storage environments. You pay only for what you use with no long-term commitments.
- **Designed for use with other Amazon Web Services**: Gateway-Stored volumes and Gateway-Cached volumes are designed to seamlessly integrate with Amazon S3, Amazon EBS, and Amazon EC2 by enabling you to store point-in-time snapshots of your on-premises application data in Amazon S3 as Amazon EBS snapshots for future recovery on-premises or in Amazon EC2. This integration allows you to easily mirror data from your on-premises applications to applications running on Amazon EC2 in disaster recovery (DR) and on-demand compute capacity cases. Gateway-VTL integrates with Amazon Glacier and allows you to cost effectively and durably store your archive and long-term backup data.

- **Optimized for Network Efficiency**: The AWS Storage Gateway efficiently uses your Internet bandwidth to speed up the upload of your on-premises application data to AWS. The AWS Storage Gateway only uploads data that has changed, minimizing the amount of data sent over the Internet. You can also use AWS Direct Connect to further increase throughput and reduce your network costs by establishing a dedicated network connection between your on-premises gateway and AWS.

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 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.

Other AWS customers have been using SQL Server synchronous mirroring to provide an HA 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 HA solution and provide a DR solution in AWS by using the native SQL Server feature of log shipping. SQL Server transaction logs can be shipping from the on-premises or co-located data centers to a SQL Server instance running on an Amazon EC2 instance within an Amazon Virtual Private Cloud (VPC). This data can be securely transmitted over a dedicated network connection using Amazon Direct Connect or over a secure VPN tunnel. Once shipped to the Amazon EC2 instance, these transaction log backups are applied to secondary database instances. One or multiple databases could be configured as secondary databases. An optional third Amazon EC2 instance could 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.

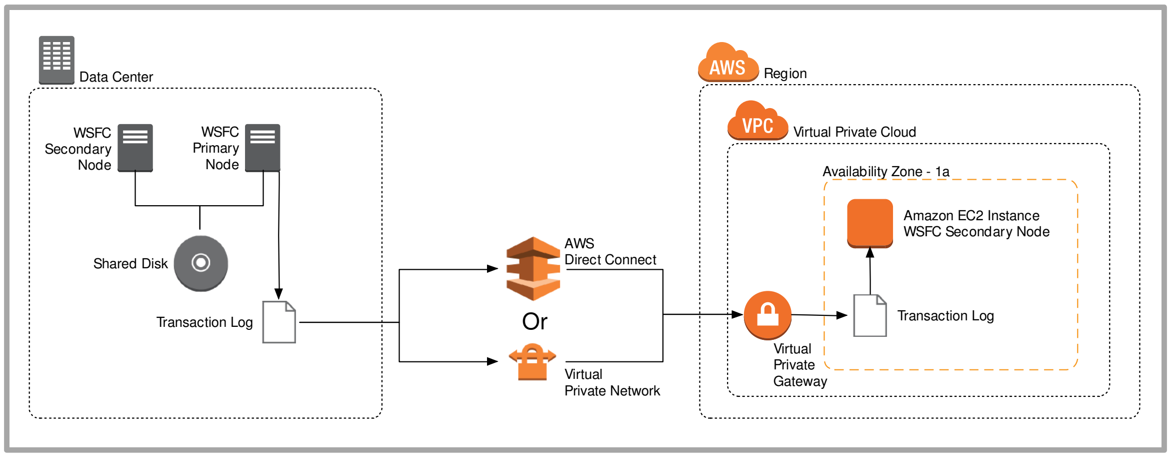


Figure 14: Hybrid Windows Server Failover Cluster

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 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. Some customers with existing on-premises deployments of SQL Server AlwaysOn Availability Groups may want to leverage the AWS cloud to provide an even higher level of availability and disaster recovery. This can be done by extending their data center into an Amazon VPC using dedicated network connection like AWS Direct Connect or setting secure VPN tunnels between these two environments.

Below are a few things you may consider when planning a hybrid implementation of SQL Server AlwaysOn Availability Groups.

- Establish secure, reliable and consistent network connection between on-premises and AWS (AWS Direct Connect or VPN)
- Create an Amazon VPC

- Leverage Amazon VPC Route Tables and Security Groups to enable the appropriate communicate between the new environments
- Extend Active Directory domains into the Amazon VPC by deploying domain controllers as Amazon EC2 instances or leveraging the AWS Directory Services AD Connector service.
- Use synchronous mode between SQL Server instances within the same environment (e.g. all instances on-premises or all instances in AWS)
- Use asynchronous mode between SQL Server instances in different environments (e.g. instances in AWS and on-premises)

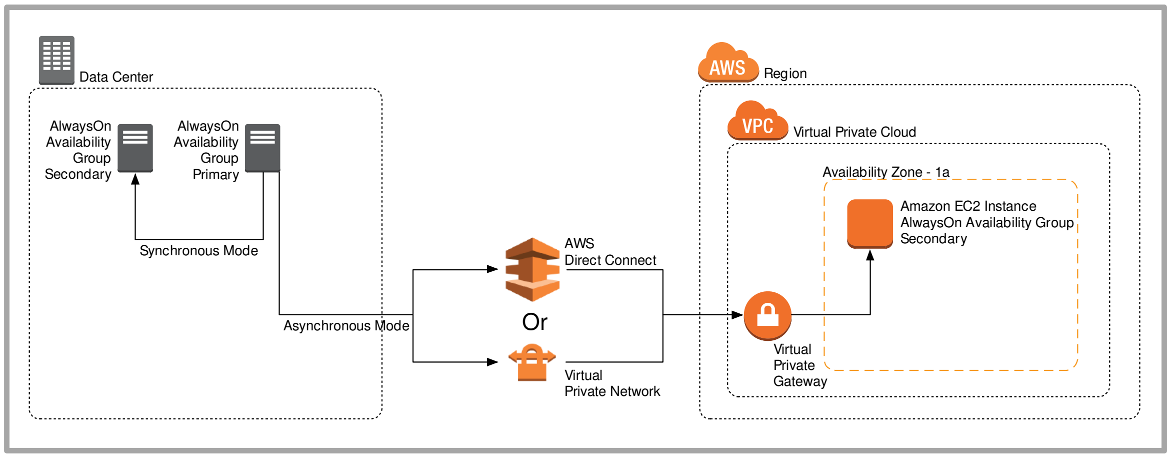

Data Center
AlwaysOn Availability Group Secondary
AlwaysOn Availability Group Primary
Synchronous Mode
Asynchronous Mode
AWS Direct Connect
Or
Virtual Private Network
AWS Region
VPC Virtual Private Cloud
Availability Zone - 1a
Amazon EC2 Instance AlwaysOn Availability Group Secondary
Virtual Private Gateway

Figure 14: Hybrid Windows Server Failover Cluster

Amazon RDS Migration Tool

Recently, AWS released the Amazon RDS Migration Tool for migrating data with minimal downtime from on-premise and EC2-based SQL Server, as well as MySQL and Oracle, to Amazon RDS, Amazon Redshift and Amazon Aurora. The tool supports not only like-to-like migrations, e.g. SQL Server-to-SQL Server, but also migrations between different database platforms, e.g. SQL Server-to-MySQL. The tool consists of a Web-based console and a replication server to replicate data across heterogeneous data sources. These capabilities provide users with instant visibility to current and historical exceptions, status, performance, and resource usage information. Please refer to the User Guide for more details on the product.

Conclusion

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

Further Reading

- Microsoft on AWS: http://aws.amazon.com/microsoft/
- Microsoft SQL Server on Amazon RDS User Guide: http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_SQLServer.html
- Amazon EC2 Windows Guide: http://docs.amazonwebservices.com/AWSEC2/latest/WindowsGuide/Welcome.html?r=7870
- Microsoft License Mobility: http://aws.amazon.com/windows/mslicensemobility
- 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/

Appendix

I. 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	2005 | 2008 | 2008 R2 | 2012 | 2014
Web	2008 R2 | 2012	2008 R2 | 2012 | 2014
Standard	2008 R2 | 2012	2005 | 2008 | 2008 R2 | 2012 | 2014
Enterprise	2008 R2 | 2012	2005 | 2008 | 2008 R2 | 2012 | 2014
SQL Server Editions	**Installation Method**	**Installation Method**
Express	N/A	AMI | Self
Web	N/A	AMI | Self
Standard	N/A	AMI | Self
Enterprise	N/A	Self
Manageability Benefits	**Supported**	**Supported**
Managed Automated Backups	Yes	No (need to configure and manage Maintenance Plans, or leverage 3rd party solutions)
Multi-AZ with Automated Failover	Yes	No (failover managed manually)
Built-in Instance and Database Monitoring and Metrics	Yes	No (push your own metrics to CloudWatch or leverage 3rd party solution)
Automatic Software Patching	Yes	No
Pre-configured Parameters	Yes	No (default SQL Server installation only)
DB Event Notifications	Yes	No (manually track and manage DB events)
SQL Server Feature	**Supported**	**Supported**

	Amazon RDS	Amazon EC2
SQL Authentication	Yes	Yes
Windows Authentication	No	Yes
TDE (encryption at rest)	Yes (Enterprise Edition only)	Yes (Enterprise Edition only)
Encrypted Storage using Amazon 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	No	Yes
Max Size of DB Instance	8 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 Timezones	No (UTC Only)	Yes
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

	Amazon RDS	Amazon EC2
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 (Ent)	2012 | 2014 (Std | Ent)
Flexible Server Roles	2012	2012 | 2014
Partially Contained Databases	2012	2012 | 2014
Sequences	2012	2012 | 2014
THROW statement	2012	2012 | 2014
UTF-16 Support	2012	2012 | 2014
Maintenance Plans	No**	Yes
Database Mail	No***	Yes
Linked Servers	No	Yes
MSDTC	No	Yes
Service Broker	No	Yes
Performance Data Collector	No	Yes
WCF Data Services	No	Yes
FILESTREAM	No	Yes
Policy-Based Management	No	Yes
SQL Server Audit	No	Yes
BULK INSERT	No	Yes
OPENROWSET	No	Yes
Data Quality Services	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 leverage the Amazon Simple Email Service (SES) to send outbound emails originating from AWS resources for a high degree of deliverability.

For detailed list of features supported by the Editions of SQL Server, see http://msdn.microsoft.com/en-us/library/cc645993%28v=sql.120%29.aspx#High_availability.

II. SQL Server Roles and Permissions Not Supported in Amazon RDS

Please note that the following server-level roles are not currently available in Amazon RDS:

- bulkadmin
- dbcreator
- diskadmin
- securityadmin
- serveradmin
- 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 APIs 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)

Notes

[1] VPC ClassicLink:
http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/vpc-classiclink.html

[2] Encrypting Amazon RDS Resources:
http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Overview.Encryption.html

[3] General Purpose (SSD) Volumes:
http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSVolumeTypes.html#EBSVolumeTypes_gp2

[4] Facts about Amazon RDS Storage:
http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Storage.html#d0e5959

[5] Microsoft SQL Server I/O Patterns:
http://blogs.msdn.com/b/psssql/archive/2010/03/24/how-it-works-bob-dorr-s-sql-server-i-o-presentation.aspx

[6] Domain Name System (DNS):
http://en.wikipedia.org/wiki/Domain_Name_System