Deploying Microsoft SQL Server on Amazon Web Services

November 2019

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

Contents

Abstract

This whitepaper explains how you can run SQL Server databases on either Amazon Relational Database Service (Amazon RDS) or Amazon Elastic Compute Cloud (Amazon EC2) and the advantages of each approach. We review in detail how to provision and monitor your SQL Server database, and how to manage scalability, performance, backup and recovery, high availability, and security in both Amazon RDS and Amazon EC2.

We also describe how you can set up a disaster recovery solution between an on-premises SQL Server environment and AWS, using native SQL Server features like log shipping, replication, and Always On availability groups. This whitepaper helps you make an educated decision and choose the solution that best fits your needs.

Introduction

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 discusses these features and controls in greater detail in the following pages.

You can run Microsoft SQL Server versions on AWS using the following services:

- Amazon RDS
- Amazon EC2

Note: Some versions of SQL Server are dependent on Microsoft licensing. For current supported versions, see Amazon RDS for SQL Server and Microsoft SQL Server on AWS.

Amazon RDS for SQL Server

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

Amazon RDS is a fully managed service and your databases run on their own SQL Server instance with the compute and storage resources you specify. Backups, high availability, and failover are fully automated.

Because of these advantages, we recommend customers consider Amazon RDS for SQL Server first.

SQL Server on Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2) is a service that provides computing capacity in the cloud. Using Amazon EC2 is similar to running a SQL Server database on-premises. You are responsible for administering the database, including backups and recovery, patching the operating system and the database, tuning of the operating system and database parameters, managing security, and configuring high availability

or replication. You have full control over the operating system, database installation, and configuration.

With Amazon EC2, you can quickly provision and configure DB instances and storage, and you can scale your instances by changing the size of your instances or amount of storage. You can provision your databases in AWS Regions across the world to provide low latency to your end users worldwide. You are responsible for data replication and recovery across your instances in the same or different Regions.

Running your own relational database on Amazon EC2 is the ideal scenario if you require a maximum level of control and configurability.

Hybrid Scenarios

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

Choosing Between Microsoft SQL Server Solutions on AWS

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

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

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

Based on your answers to the preceding considerations, Amazon RDS might be a better choice if the following is true:

- You want to focus on business growth tasks, such as performance tuning and schema optimization, and outsource the following tasks to AWS: provisioning of the database, management of backup and recovery, management of security patches, upgrades of minor SQL Server versions, and storage management.
- You need a highly available database solution and want to take advantage of the push-button, synchronous Multi-AZ replication offered by Amazon RDS, without having to manually set up and maintain database mirroring, failover clusters, or Always On Availability Groups.
- You don’t want to manage backups and, most importantly, point-in-time recoveries of your database, and prefer that AWS automates and manages these processes.

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

- You need full control over the SQL Server instance, including access to the operating system and software stack.
- Install third party agents on the host
- 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 for SQL Server.
- You need to use SQL Server features or options not currently supported by Amazon RDS.

- You want to run SQL Server 2017 on the Linux operating system.

For a detailed side-by-side comparison of SQL Server features available in the AWS environment, see the Comparison of Microsoft SQL Server Feature Availability on AWS section.

Amazon RDS for Microsoft SQL Server

For the list of currently Amazon RDS currently supported versions and features see Microsoft SQL Server on Amazon RDS

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

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

For a detailed feature comparison between the different SQL Server editions, see Editions and supported features of SQL Server on the Microsoft Developer Network (MSDN) website.

In Amazon RDS for SQL Server, the following features and options are supported, depending on the edition of SQL Server: For the most current supported features, see Amazon RDS for SQL Server features.

- Core database engine features
- SQL Server development tools: Visual Studio integration and IntelliSense
- SQL Server management tools: SQL Server Management Studio (SSMS), sqlcmd, SQL Server Profiles (for client-side traces), SQL Server Migration Assistant (SSMA), Database Engine Tuning Advisor, and SQL Server Agent

- Safe Common Language Runtime (CLR) for SQL Server 2016 and below versions
- Service Broker
- Full-text search (except semantic search)
- Secure Sockets Layer (SSL) connection support
- Transparent Data Encryption (TDE)
- Encryption of storage at rest using the AWS Key Management Service (AWS KMS) for all SQL Server license types
- Spatial and location features
- Change tracking
- Change Data Capture
- Always On or Database mirroring (used to provide the Multi-AZ capability)
- The ability to use an Amazon RDS SQL DB instance as a data source for reporting, analysis, and integration services
- Local Time Zone support
- Custom Server Collations

AWS frequently improves the capabilities of Amazon RDS for SQL Server. For the latest information on supported versions, features, and options, see Version and Feature Support on Amazon RDS.

Starting an Amazon RDS for SQL Server Instance

You can start a SQL Server instance on Amazon RDS in several ways:

- Interactively using the AWS Management Console
- Programmatically using AWS CloudFormation templates
- AWS SDKs and the AWS Command Line Interface (AWS CLI)
- Using the PowerShell

After the instance has been deployed, you can connect to it using standard SQL Server tools. Amazon RDS provides you with a Domain Name Service (DNS) endpoint for the server, as shown in the following figure. To connect to the database, use this DNS

endpoint as the SQL Server hostname, along with the master user name and password configured for the instance. Always use the DNS endpoint to connect to the instance, because the underlying IP address might change. Amazon RDS exposes the Always On AGs availability group listener endpoint for the SQL Server Multi-AZ deployment. The endpoint is visible in the console, and is returned by the `DescribeDBInstances` API as an entry in the endpoints field. You can easily connect to the listener endpoint in order to have faster failover times.

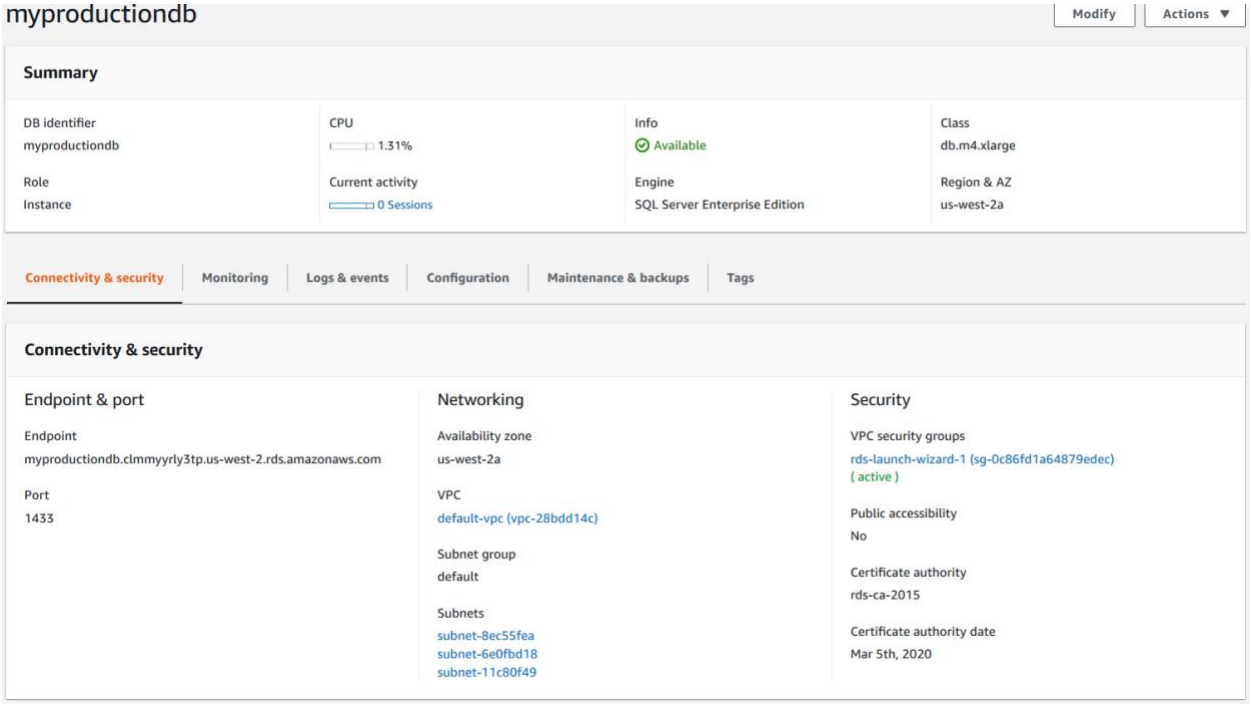

Figure 1: Amazon RDS DB instance properties

Security

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

- Network controls, which determine the network configuration underlying your DB instance
- DB instance access controls, which determine administrative and management access to your RDS resources
- Data access controls, which determine access to the data stored in your RDS DB instance databases

- Data at rest protection, which affects the security of the data stored in your RDS DB instance
- Data in transit protection, which affects the security of data connections to and from your RDS DB instance

Network Controls

At the network layer, controls are on the deployed instance EC2-VPC level

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

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 Microsoft Windows Server Firewall. The rules of a security group define what traffic is allowed to enter the instance (inbound) and what traffic is allowed to exit the instance (outbound). VPC security groups are used for DB instances deployed in a VPC. They can be changed and reassigned without restarting the instances associated with them.

For improved security, we recommend restricting inbound traffic to only database-related traffic (port 1433, unless a custom port number is used) and only traffic from known sources. Security groups can also accept the ID of a different security group (called the source security group) as the source for traffic. This approach makes it easier to manage sources of traffic to your RDS DB instance in a scalable way. In this case, you don’t have to update the security group every time a new server needs to connect to your DB instance; you just have to assign the source security group to it.

Amazon RDS for SQL Server can make DB instances publicly accessible by assigning internet routable IP addresses to the instances. In most use cases, this approach is not needed or desired, and we recommend setting this option to **No** to limit the potential threat. In cases where direct access to the database over the public internet is needed,

we recommend limiting the sources that can connect to the DB instance to known hosts by using their IP addresses. For this option to be effective, the instance must be launched in a subnet that permits public access and the security groups and NACLs must permit inbound traffic from those sources.

DB instances that are exposed publicly over the internet and have open security groups, accepting traffic from any source, might be subject to more frequent patching. Such instances can be force-patched when security patches are made available by the vendors involved. This patching can occur even outside the defined instance maintenance window, to ensure the safety and integrity of customer resources and our infrastructure. Although there are many ways to secure your databases, we recommend using private subnet(s) within a VPC no possible direct internet access.

DB Instance Access Controls

Using AWS Identity and Access Management (IAM), you can manage access to your Amazon RDS for SQL Server instances. For example, you can authorize administrators under your AWS account (or deny them the ability) 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 Authentication and Access Control for Amazon RDS in the *Amazon RDS User Guide.*

Data Access Controls

Amazon RDS for SQL Server supports both SQL Authentication and Windows Authentication, and access control for authenticated users should be configured using the principle of least privilege. A master account is created automatically when an instance is launched. This master user is granted several permissions. For details, see Master User Account Privileges. This login is typically used for administrative purposes only and is granted the roles of processadmin, setupadmin, SQLAgentUser, Alter on SQLAgentOperator and public at the server level. Amazon RDS manages the master user as a login, and creates a user linked to the login in each customer database with the db_owner permission.

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

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

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

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

You can also create SQL Server Windows logins on domain-joined DB instances for users and groups in your directory domain, or the trusted domain if applicable. Logins can be added using a SQL client tool such as SQL Server Management Studio using the following command.

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

More information on configuring Windows Authentication with Amazon RDS for SQL Server can be found in the Using Windows Authentication topic in the *Amazon RDS User Guide.*

Unsupported SQL Server Roles and Permissions in Amazon RDS

The following server-level roles are not currently available in Amazon RDS: bulkadmin, dbcreator, diskadmin, securityadmin, serveradmin and sysadmin. See Features Not Supported and Features with limited support

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 RESOURCES
- ALTER SETTINGS (you can use the DB parameter group API actions to modify parameters)
- AUTHENTICATE SERVER
- CREATE DDL EVENT NOTIFICATION
- CREATE ENDPOINT
- CREATE TRACE EVENT NOTIFICATION
- EXTERNAL ACCESS ASSEMBLY
- SHUTDOWN (you can use the RDS reboot option instead)
- UNSAFE ASSEMBLY
- ALTER ANY AVAILABILITY GROUP
- CREATE ANY AVAILABILITY GROUP

Data at Rest Protection

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

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

Amazon RDS encrypted instances also help secure your data from unauthorized access to the underlying storage. You can use Amazon RDS encryption to increase data protection of your applications deployed in the cloud, and to fulfill compliance requirements for data at rest encryption. To manage the keys used for encrypting and decrypting your Amazon RDS resources, use AWS KMS.

Amazon RDS also supports encryption of data at rest using the 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 such a group doesn't already exist), and then associating the DB instance with that group. You can find more details on Amazon RDS support for TDE on the Options for the Microsoft SQL Server Database Engine topic in the *Amazon RDS User Guide.*

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

Data in Transit Protection

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

Performance Management

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

Instance Sizing

The amount of memory and compute capacity available to your Amazon RDS for SQL Server instance is determined by its instance class. Amazon RDS for SQL Server offers a range of DB instance classes from 1 vCPU and 1 GB of memory to 96 vCPUs and 488 GB of memory. Not all instance classes are available for all SQL Server editions, however. The instance class availability also varies based on the version.

Amazon RDS for SQL Server supports the various DB instance classes for the various SQL Server editions. For the most up-to-date list of supported instance classes, see Amazon RDS for SQL Server instance types.

Previous generation DB instance classes are superseded in terms of both cost-effectiveness and performance by the current generation classes. For the previous generation instance types, see Previous Generation Instances for more information.

Understanding the performance characteristics of your workload is important when identifying the proper instance class. If you are unsure how much CPU you need, we recommend that you start with the smallest appropriate instance class, then monitor CPU utilization using Amazon CloudWatch. You can modify the 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. If you are in a Multi-AZ High Availability configuration, making the change involves a server reboot or a failover.

To modify a SQL Server instance, see Modifying a DB Instance Running the Microsoft SQL Server database engine and for the list of modification setting see setting for Microsoft SQL Server DB Instances. The settings are similar to the ones you configure when launching a new DB instance. By default, changes (including a change to the DB instance class) are applied during the next specified maintenance window. Alternatively, you can use the apply-immediately flag to apply the changes immediately.

Disk I/O Management

Amazon RDS for SQL Server simplifies the allocation and management of database storage for instances. You decide the type and amount of storage to use, and also the level of provisioned I/O performance if applicable. You can change the amount of storage or provisioned I/O on an RDS for SQL Server instance after the instance has been deployed. You can also enable storage auto scaling to enable the Amazon RDS to automatically increase the storage when needed to avoid having your instance run out of storage space.

We recommend that you enable storage auto scaling to handle growth from the onset.

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

- General Purpose (SSD) (also called GP2) is an SSD-backed storage solution with predictable performance and burst capabilities. This option is suitable for workloads that run in larger batches, such as nightly report processing. Credits are replenished while the instance is largely idle, and are then available for bursts of batch jobs.

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

The following table compares the Amazon RDS storage performance characteristics.

Table 1: Amazon RDS storage performance characteristics

Storage Type	Min. Volume Size	Max. Volume Size	Baseline Performance	Burst Capability	Storage Technology	Pricing Criteria
General Purpose	20 GiB (100 GiB recommended)	16 TiB*	3 IOPS/GiB	Yes; up to 3000 IOPS per volume, subject to accrued credits	SSD	Allocated storage
Provisioned IOPS	20 GiB (for Enterprise and Standard editions, 100 GiB for Web and Express Edition)	16 TiB*	10 IOPS/GiB up to max. 64,000 IOPS	No; fixed allocation	SSD	Allocated storage and Provisioned IOPS

* Maximum IOPS of 64,000 is guaranteed only on Nitro-based instances that are on m5 instance types.

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

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

The average queue depth, a metric available through Amazon CloudWatch, tracks the number of I/O requests in the queue that are waiting to be serviced. These requests have been submitted by the application but haven't been sent to the storage device because the device is busy servicing other I/O requests. Time spent in the queue increases I/O latency, and large queue sizes can indicate an overloaded system from a storage perspective.As a result, depending on the storage configuration selected, your overall storage subsystem throughput will be limited either by the maximum IOPS or the maximum channel bandwidth at any time. If your workload is generating a lot of small sized I/O operations (for example, 8 KiB), you are likely to reach maximum IOPS before the overall bandwidth reaches the channel maximum.

However, if I/O operations are large in size (for example, 256 KiB), you might reach the maximum channel bandwidth before maximum IOPS.

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

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

Using small storage sizes (less than 1TB) with General Purpose (GP2) SSD storage can also have a detrimental impact on 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 GP2 storage, small storage sizes will also provide small amounts of baseline IOPS. When created, each instance comes with an initial allocation of I/O credits. This allocation provides for burst capabilities of up to 3,000 IOPS from the start. Once the initial burst credits allocation is exhausted, you must ensure that your ongoing workload needs fit within the baseline I/O performance of the storage size selected.

High Availability

Amazon RDS provides high availability and failover support for DB instances using Multi-AZ deployments. Multi-AZ deployments provide increased availability, data durability, and fault tolerance for DB instances. Multi-AZ high availability option uses SQL Server database mirroring or Always On availability groups configuration options 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 set up SQL Server Multi-AZ, RDS automatically configures all databases on the instance to use database mirroring or availability groups. Amazon RDS handles the primary, the witness, and the secondary DB instance for you.

Because configuration is automatic, RDS selects database mirroring or Always On availability groups based on the version of SQL Server that you deploy. Amazon RDS supports Multi-AZ with database mirroring or availability groups for the following SQL Server versions and editions (exceptions noted): See Multi-AZ Deployments for Microsoft SQL Server for more information.

- SQL Server 2017: Enterprise Editions (Always On availability groups are supported in Enterprise Edition 14.00.3049.1 or later.)
- SQL Server 2016: Enterprise Editions (Always On availability groups are supported in Enterprise Edition 13.00.5216.0 or later.)

Amazon RDS supports Multi-AZ with database mirroring for the following SQL Server versions and editions, except for the versions of Enterprise Edition noted previously:

- SQL Server 2017: Standard and Enterprise Editions
- SQL Server 2016: Standard and Enterprise Editions
- SQL Server 2014: Standard and Enterprise Editions
- SQL Server 2012: Standard and Enterprise Editions

Amazon RDS supports Multi-AZ for SQL Server in all AWS Regions, with the following exceptions:

- US West (N. California): Neither database mirroring nor Always On availability groups are supported.
- South America (São Paulo): Supported on all DB instance classes except m1 or m2.
- EU (Stockholm): Neither database mirroring nor Always On availability groups are supported.

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

If an Availability Zone failure or instance failure occurs, your availability impact is limited to the time that automatic failover takes to complete, typically 60-120 seconds for database mirroring, and 10-15 seconds for availability groups. When failing over, Amazon RDS simply flips the canonical name record (CNAME) for your DB instance to point to the secondary, which is in turn promoted to become the new primary. The canonical name record (or endpoint name) is an entry in DNS.

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

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

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

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

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

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

Monitoring and Management

Amazon CloudWatch collects many Amazon RDS specific metrics. You can look at these metrics using the AWS Management Console, the AWS CLI (using the `mon-get-stats` command), or the AWS API, Or the powershell (using the `Get-CWMetricStatistics` cmdlet). In addition to the system-level metrics collected for Amazon EC2 instances (such as CPU usage, disk I/O, and network I/O), the Amazon RDS metrics include many database-specific metrics, such as database connections, free storage space, read and write I/O per second, read and write latency, read and write throughput, and available RAM. For a full, up-to-date list, see Amazon RDS Dimensions and Metrics in the *Amazon CloudWatch Developer Guide.*

In Amazon CloudWatch, you can also configure alarms on these metrics to trigger notifications when the state changes. An alarm watches a single metric over a time period you specify, and performs one or more actions based on the value of the metric, relative to a given threshold, over a number of time periods.

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

Amazon RDS for SQL Server also supports Enhanced Monitoring. Amazon RDS provides metrics in near-real time for the operating system (OS) that your DB instance runs on. You can view the metrics for your instance using the console, or consume the Enhanced Monitoring JSON output from Amazon CloudWatch Logs in a monitoring system of your choice. Enhanced Monitoring gathers its metrics from an agent on the instance.

Enhanced Monitoring gives you deeper visibility into the health of your Amazon RDS instances in near-real time, providing a comprehensive set of 26 new system metrics and aggregated process information, at a detail level of up to 1 second.

These monitoring metrics cover a wide range of instance aspects, such as the following:

- General metrics, like uptime, instance, and engine version
- CPU utilization, such as idle, kernel, or user time percentage
- Disk subsystem metrics, including utilization, read and write bytes, and number of I/O operations
- Network metrics, like interface throughput and read and write bytes
- Memory utilization and availability, including physical, kernel, commit charge, system cache, and SQL Server footprint
- System metrics, consisting of number of handles, processes, and threads
- Process list information, grouped by OS processes, RDS processes (management, monitoring, diagnostics agents), and RDS child processes (SQL Server workloads)

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

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

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

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

Performance Insights expands on existing Amazon RDS monitoring features to illustrate your database's performance and help you analyze any issues that affect it. With the Performance Insights dashboard, you can visualize the database load and filter the load by waits, SQL statements, hosts, or users. More information can be found at Using

Using Amazon RDS Performance Insights in the *Amazon Relational Database Service User Guide*.

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

- **Backup window:** The backup window is the period of time during which your instance is going to be backed up. Because backups might have a small performance impact on the operation of the instance, we recommend you set the window for a time when this has minimal impact on your workload.
- **Maintenance window:** The maintenance window is the period of time during which instance modifications (such as implementing pending changes to storage or CPU class for the instance) and software patching occur. Your instance might be restarted during this window, if there is a scheduled activity pending and that activity requires a restart, but that is not always the case. We recommend scheduling the maintenance window for a time when your instance has the least traffic, or a potential restart is least disruptive.

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

- **Automated backup and recovery.** Amazon RDS automatically backs up all databases of your instances. You can set the backup retention period when you create an 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. If you select zero days of backup retention, point in time log backups are not taken. Amazon RDS uses these periodic data backups in conjunction with your transaction logs (backed up every 5 minutes) to enable you to restore your DB instance to any second during your retention period, up to the LatestRestorableTime, typically up to the last 5 minutes.
- **Push-button scaling.** With a few clicks, you can change the instance class to increase or decrease the size of your instance's compute capacity, network capacity, and memory. You can choose to make the change immediately, or schedule it for your next maintenance window.
- **Automatic host replacement.** Amazon RDS automatically replaces the compute instance powering your deployment in the event of a hardware failure.

- **Automatic minor version upgrade.** Amazon RDS keeps your database software up-to-date. You have full control on whether Amazon RDS deploys such patching automatically, and you can disable this option to prevent that. Regardless of this setting, publicly accessible instances with open security groups might be force-patched when security patches are made available by vendors, to ensure the safety and integrity of customer resources and our infrastructure. The patching activity occurs during the weekly, 30-minute maintenance window that you specify when you provision your database (and that you can alter at any time). Such patching occurs infrequently, and your database might become unavailable during part of your maintenance window when a patch is applied. You can minimize the downtime associated with automatic patching if you run in Multi-AZ mode. In this case, the maintenance is generally performed on the secondary instance. When it is complete, the secondary instance is promoted to primary. The maintenance is then performed on the old primary, which becomes the secondary.
- **Preconfigured parameters and options.** Amazon RDS provides a default set of DB parameter groups and also option groups for each SQL Server edition and version. These groups contain configuration parameters and options respectively, which allow you to tune the performance and features of your instance. By default, Amazon RDS provides an optimal configuration set suitable for most workloads, based on the class of the instance that you selected. You can create your own parameter and option groups to further tune the performance and features of your instance.

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

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

Commands to create users, rename users, grant, revoke permissions and set passwords work as they do in Amazon EC2 (or on-premises) databases. The administrative commands that RDS doesn't support are listed in Unsupported SQL Server Roles and Permissions in Amazon RDS.

Even though direct, file-system level access to the RDS SQL Server instance is not available, you can always migrate your data out of RDS instances. You can use tools like the Microsoft SQL Server Database Publishing Wizard to download the contents of

your databases into flat T-SQL files. You can then load these files into any other SQL Server instances, or store them as backups in Amazon Simple Storage Service (Amazon S3) or Amazon S3 Glacier, or on-premises. In addition, you can use the AWS Database Migration Service to move data to and from Amazon RDS.

You can also use native backup and restore through S3. You can use native backups to migrate databases to Amazon RDS for SQL Server instances, or back up your RDS for SQL Server instances to S3 to copy to another SQL Server instance, or to retain offline. For more details on how this works and the permissions required, see Importing and Exporting SQL Server Databases.

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 instance depends on the following factors:

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

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

AWS services are available in multiple Regions across the world. In Regions where our costs of operating our services are lower, we pass the savings on to you.

Thus, Amazon RDS hourly prices for the different instance classes vary by the Region. If you have the flexibility to deploy your SQL Server workloads in multiple Regions, the potential savings from operating in one Region as compared to another can be an important factor in choosing the right Region.

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

- On-Demand Instance pricing allows you to pay for Amazon RDS DB instances by the hour, with no term commitments. You incur a charge for each hour a given DB instance is running. If your workload doesn't need to run 24/7, or you are deploying temporary databases for staging, testing, or development purposes, On-Demand Instance pricing can offer significant advantages.
- Reserved Instances (RI) allow you to lower costs and reserve capacity. Reserved Instances can save you up to 60 percent over On-Demand rates when used in steady state, which tend to be the case for many databases. They can be purchased for 1- or 3-year terms. If your SQL Server database is going to be running more than 25 percent of the time each month, you will most likely financially benefit from using a Reserved Instance.

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

The following pricing options are available for RDS Reserved Instances:

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

Note that like in Amazon EC2, in Amazon RDS you can issue a stop command to a standalone DB instance and keep the instance in a stopped state to avoid incurring compute charges. You can't stop an Amazon RDS for SQL Server DB instance in a Multi-AZ configuration, 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, or remove the Multi-AZ configuration first and then stop the instance. Note that after 7 days, your stopped instance will re-start so that any pending maintenance can be applied.

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

- Terminate DB instances with a last snapshot when they are not needed, then reprovision them from that snapshot when they need to be used again. For example, some development and test databases can be terminated at night and on weekends, and reprovisioned on weekdays in the morning. Alternatively, use the stop feature mentioned above to turn off the database for the weekend.
- Scale down the size of your DB instance during off-peak times, by using a smaller instance class.

See the Amazon RDS for SQL Server Pricing webpage for up-to-date pricing information for all pricing models and instance classes.

Microsoft SQL Server on Amazon EC2

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

Starting a SQL Server Instance on Amazon EC2

You can start a SQL Server DB instance on Amazon EC2 in several ways:

- Interactively using the AWS Management Console
- Programmatically using AWS CloudFormation templates
- Using AWS SDKs and the AWS Command Line Interface (AWS CLI)
- Using the PowerShell

For the procedure to launch Amazon EC2 using the AWS Management Console see Launch an Instance. Check the below useful bullets for launching Amazon EC2 for running SQL Server instance.

- You can deploy a SQL Server instance on Amazon EC2 using an Amazon Machine Image (AMI). An AMI is simply a packaged environment that includes all the necessary software to set up and boot your instance. Some AMIs have just the operating system (for example Windows Server 2019), and others have the operating system and a version and edition of SQL Server (Windows Server 2019 and SQL Server 2017 Standard Edition, SQL Server 2017 on Ubuntu, and so on). We recommend that you use the AMIs available at Windows 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.
- Other AMIs install just the Microsoft Windows operating system. This type of AMI allows you the flexibility to perform a separate, custom installation of SQL Server on the Amazon EC2 instance and bring your own license (BYOL) of Microsoft SQL Server if you have qualifying licenses. For additional information on BYOL qualification criteria, see License Mobility.
- Consider all five performance characteristics (vCPU, Memory, Instance Storage, Network Bandwidth and EBS Bandwidth) of Amazon EC2 instances when selecting the EC2 instance. See Amazon EC2 Instance Types for more information.
- Depending on the type of SQL Server deployment, for example stand- alone, Windows Failover Clustering and Always On Availability Groups, SQL Server on Linux and so on, you might decide to assign one or multiple static IP addresses to your Amazon EC2 instance. You can do this assignment in the **Network interface** section of **Configure Instance Details**.
- 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.
- Assign the appropriate tags to the Amazon EC2 instance. We recommend that you assign tags to other Amazon resources, for example Amazon Elastic Block Store (Amazon EBS) volumes, to allow for more control over resource-level permissions and cost allocation. For best practices on tagging AWS resources, see Tagging Your Amazon EC2 Resources in the *Amazon EC2 User Guide.*

Amazon EC2 Security

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

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

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

In an Amazon EC2 deployment of SQL Server, you are also responsible for patching the OS and application stack of your instances when Microsoft or other third-party vendors release new security or functional patches. This patching includes work for additional support services and instances, such as Active Directory servers.

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

EBS volume encryption integrates with the AWS KMS. This integration allows you to use your own customer master key (CMK) for volume encryption. Creating and

leveraging your own CMK gives you more flexibility, including the ability to create, rotate, disable, and define access controls, and to audit the encryption keys used to protect your data.

Performance Management

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

Instance Sizing

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

We recommend that you choose Amazon EC2 instances that best fit your workload requirements and have a good balance of CPU, memory, and IO performance. SQL Server workloads are typically memory-bound, so look at the r5 or r5d instances, also referred to as memory-optimized instances. If your workload is more CPU-bound, look at the latest compute-optimized instances of the c5 instance family. See Amazon EC2 Instance types for more information.You can customize the number of CPU cores for the instance. You might do this to potentially optimize the licensing costs of your software with an instance that has sufficient amounts of RAM for memory-intensive workloads but fewer CPU cores. See Optimizing CPU Options for more information.

One of the differentiators among all these instance types is that the m5, r5 and c5 instance types are EBS-optimized by default, whereas older instance types, such as the r3 family, can be optionally EBS-optimized. You can find a detailed explanation of EBS-optimized instances in the Disk I/O Management section following. If your workload is network-bound, again look at instance families that support 25 gigabit network performance, because these instance families also support Enhanced Networking. These include the r5, z1d, m5, and c5 instance families. The i3en and c5n instance types even support 100 gigabit network performance.

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. See Enhanced Networking on Windows in the *Amazon EC2 User Guide.*

Disk I/O Management

The same storage types available for Amazon RDS are also available when deploying SQL Server on Amazon EC2. Additionally, you also have access to instance storage. Because you have fine-grained control over the storage volumes and strategy to use, you can deploy workloads that require more than 4 TiB in size or 64,000 IOPS in Amazon EC2. Multiple EBS volumes or instance storage disks can even be striped together in a software RAID configuration to aggregate both the storage size and usable IOPS, beyond the capabilities of a single volume.

The two main Amazon EC2 storage options are as follows:

- **Instance store volumes:** Several Amazon EC2 instance types come with a certain amount of local (directly attached) storage, which is ephemeral. These include R5d, M5d, i3, i3en, and x1e instance types.
- Any data saved on instance storage is no longer available after you stop and restart that instance, or if the underlying hardware fails, which causes an instance restart to happen on a different host server. This characteristic makes instance storage a challenging option for database persistent storage. However, Amazon EC2 instances can have the following benefits:
    - Instance store volumes offer good performance for sequential disk access, and don't have a negative impact on your network connectivity. Some customers have found it useful to use these disks to store temporary files to conserve network bandwidth.
    - Instance types with large amounts of instance storage offer unmatched I/O performance and are recommended for database workloads, as long as you implement a backup or replication strategy that addresses the ephemeral nature of this storage.

- **EBS volumes:** Similar to Amazon RDS, you can use EBS for persistent block-level storage volumes. Amazon EBS volumes are off-instance storage that persists independently from the life of an instance. Amazon EBS volume data is mirrored across multiple servers in an Availability Zone to prevent the loss of data from the failure of any single component. You can back them up to Amazon S3 by using snapshots. These attributes make EBS volumes suitable for data files, log files, and the flash recovery area. Although the maximum size of an EBS volume is 16 TB, you can address larger database sizes by striping your data across multiple volumes. See EBS volume characteristics for more information.

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 depending on the instance type. When attached to EBS-optimized instances, Provisioned IOPS volumes are designed to deliver within 10 percent of their provisioned performance 99.9 percent of the time. The combination of EBS-optimized instances and Provisioned IOPS volumes helps to ensure that instances are capable of consistent and high EBS I/O performance. See EBS optimized by default for more information.

Most databases with high I/O requirements should benefit from this feature. You can also use EBS-optimized instances with standard EBS volumes if you need predictable bandwidth between your instances and EBS. For up-to-date information about the availability of EBS-optimized instances, see Amazon 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 eight 100 GB EBS volumes instead of one 800 GB EBS volume. However, remember that using striping generally reduces the operational durability of the logical volume by a degree inversely proportional to the number of EBS volumes in the stripe set. The more volumes you include in a stripe, the larger the pool of data that can get corrupted if a single volume fails, because the data on all other volumes of the stripe gets invalidated also.

EBS volume data is natively replicated, so using RAID 0 (striping) might provide you with sufficient redundancy and availability. No other RAID mechanism is supported for EBS volumes.

Data, logs, and temporary files benefit from being stored on independent EBS volumes or volume aggregates because they present different I/O patterns. To take advantage of additional EBS volumes, be sure to evaluate the network load to help ensure that your instance size is sufficient to provide the network bandwidth required.

I3 and i3en 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.

Amazon EC2 offers many options to optimize and tune your I/O subsystem. We encourage you to benchmark your application on several instance types and storage configurations to select the most appropriate configuration. For EBS volumes, we recommend that you monitor the CloudWatch average queue length metric of a given volume, and target an average queue length of 1 for every 500 IOPS for volumes up to 2000 IOPS, and a length between 4 and 8 for volumes with 2,000 to 4,000 IOPS. Lower metrics indicate overprovisioning, and higher numbers usually indicate your storage system is overloaded.

High Availability

High availability is a design and configuration principle to help protect services or applications from single points of failure. The goal is for services and applications to continue to function even if underlying physical hardware fails or is removed or replaced. We will review three native SQL Server features that improve database high availability and ways to deploy these features on AWS.

Log Shipping

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

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

We recommend running the primary and secondary DB instances in separate Availability Zones, and optionally deploying an optional monitor instance to track all the

details of log shipping. Backup, copy, restore, and failure events for a log shipping group are available from the monitor instance.

Database Mirroring

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

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

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

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

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

SQL Server Always On Availability Groups

Always On availability groups is an enterprise-level feature that provides high availability and disaster recovery to SQL Server databases. Always On availability groups uses advanced features of Windows Failover Cluster and the Enterprise Edition of all versions of SQL Server from SQL Server 2012. Starting in SQL Server 2016 SP1, basic availability groups are available for Standard Edition SQL Server as well (as a replacement for database mirroring).

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

You can implement SQL Server Always On availability groups on Amazon Web Services using services like Windows Server Failover Clustering (WSFC), Amazon EC2, Amazon VPC, Active Directory, and DNS. Always On clusters require multiple subnets and need the `MultiSubnetFailover=True` parameter in the connection string to work correctly.

See How do I create a SQL Server Always On availability group cluster in the AWS Cloud? for how to deploy SQL Server Always On availability Groups. For details on how to deploy SQL Server Always On availability groups in AWS using CloudFormation, see the SQL Server on the AWS Cloud: Quick Start Reference Deployment.

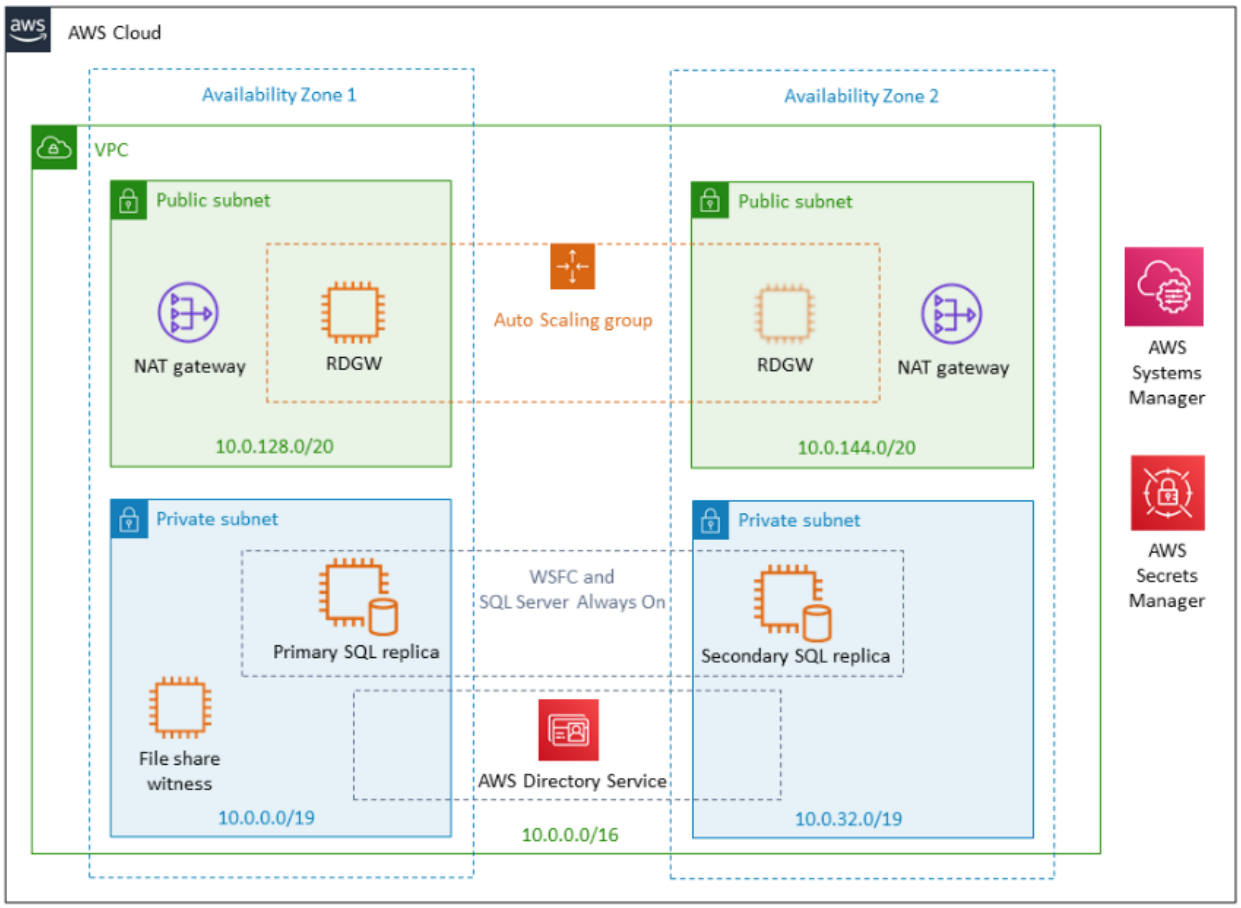

Figure 2: SQL Server Always On availability group

Monitoring and Management

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

Additionally, Amazon CloudWatch supports custom metrics, such as memory utilization or disk utilizations, which are metrics visible only from within the instance. You can publish your own relevant metrics to the service, to consolidate monitoring information. You can also push custom logs to CloudWatch Logs to monitor, store, and access your log files for Amazon EC2 SQL Server instances.

You can then retrieve the associated log data from CloudWatch Logs using the Amazon CloudWatch console, the CloudWatch Logs commands in the AWS CLI, or the

CloudWatch Logs SDK. This approach allows you to track log events in real time for your SQL Server instances.

As with Amazon RDS, you can configure alarms on Amazon EC2, Amazon EBS, and custom metrics to trigger notifications when the state changes. An alarm tracks a single metric over a time period you specify, and performs one or more actions based on the value of the metric, relative to a given threshold, over a number of time periods. Notifications are sent to Amazon SNS topics or AWS Auto Scaling policies. You can configure these alarms to notify database administrators by email or SMS text message when they get triggered.

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

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

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

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

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

Managing Cost

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

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

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

Different instance types have different hourly charges. Generally, current generation instance types have lower hourly charges compared to previous generation instance types, along with better performance due to newer hardware architectures. We recommend that you test your workloads on new instance types as these become available, and plan to migrate your workloads to new instance types, if the cost vs. performance ratio makes sense for your use case.

Many EC2 instance types are available with the EBS-optimized option. This option is available for an additional hourly surcharge, and provides additional, dedicated networking capacity for EBS I/O. This dedicated capacity ensures a predictable amount of networking capacity to sustain predictable EBS I/O. Some current generation

instance types, such as the C4, M4 and D2 instance types, are EBS-optimized by default and don't have an additional surcharge for the optimization.

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

Up-to-date pricing information is available on the Amazon EC2 Dedicated Instances pricing page.

You also have the option to provision EC2 Dedicated Hosts. These are physical servers with EC2 instance capacity fully dedicated to your use. Dedicated Hosts can help you address compliance requirements and reduce costs by allowing you to use your existing server-bound software licenses. For more information, see Amazon EC2 Dedicated Hosts and Bring license to AWS

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

The following pricing options are available for EC2 Reserved Instances:

- 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.
- 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.

- No Upfront Reserved Instances: you don't make any upfront payments, but will be charged a discounted hourly rate for the instance for the duration of the Reserved Instance term. This option still provides you with a significant discount compared to On-Demand Instance pricing, but the discount is usually less than for the other two Reserved Instance pricing options.

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

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

Caching

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

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

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

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

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

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

Hybrid Scenarios and Data Migration

Some AWS customers already have SQL Server running in their on-premises or colocated data center but want to use the AWS Cloud to enhance their architecture to provide a more highly available solution or one that offers disaster recovery. Other customers are looking to migrate workloads to AWS without incurring significant downtime. These efforts often can stretch over a significant amount of time. AWS offers several services and tools to assist customers in these use cases, and SQL Server has several replication technologies that offer high availability and disaster recovery solutions. These features differ depending on the SQL Server version and edition.

Amazon RDS on VMware lets you deploy managed databases in on-premises VMware environments using the Amazon RDS technology enjoyed by hundreds of thousands of AWS customers. Amazon RDS provides cost-efficient and resizable capacity while automating time-consuming administration tasks including hardware provisioning, database setup, patching, and backups, freeing you to focus on your applications. RDS on VMware brings these same benefits to your on-premises deployments, making it

easy to set up, operate, and scale databases in VMware vSphere private data centers, or to migrate them to AWS.

RDS on VMware allows you to utilize the same simple interface for managing databases in on-premises VMware environments as you would use in AWS. You can easily replicate RDS on VMware databases to RDS instances in AWS, enabling low-cost hybrid deployments for disaster recovery, read replica bursting, and optional long-term backup retention in Amazon Simple Storage Service (S3).

Amazon RDS on VMware is supporting Microsoft SQL Server, PostgreSQL, MySQL, and MariaDB databases, with Oracle to follow in the future.

Backups to the Cloud

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

Amazon Simple Storage Service (Amazon S3)

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

AWS Storage Gateway

AWS Storage Gateway is a service connecting an on-premises software appliance with cloud-based storage to provide seamless and secure integration between an organization's on-premises IT environment and AWS's storage infrastructure.

The service allows you to securely store data in the AWS Cloud for scalable and cost-effective storage. AWS Storage Gateway supports open standard storage protocols that work with your existing applications. It provides low-latency performance by maintaining frequently accessed data on-premises while securely storing all of your data encrypted in Amazon S3. AWS Storage Gateway enables your existing on-premises–to–cloud

backup applications to store primary backups on Amazon S3's scalable, reliable, secure, and cost-effective storage service.

SQL Server Log Shipping Between On-Premises and Amazon EC2

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

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

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

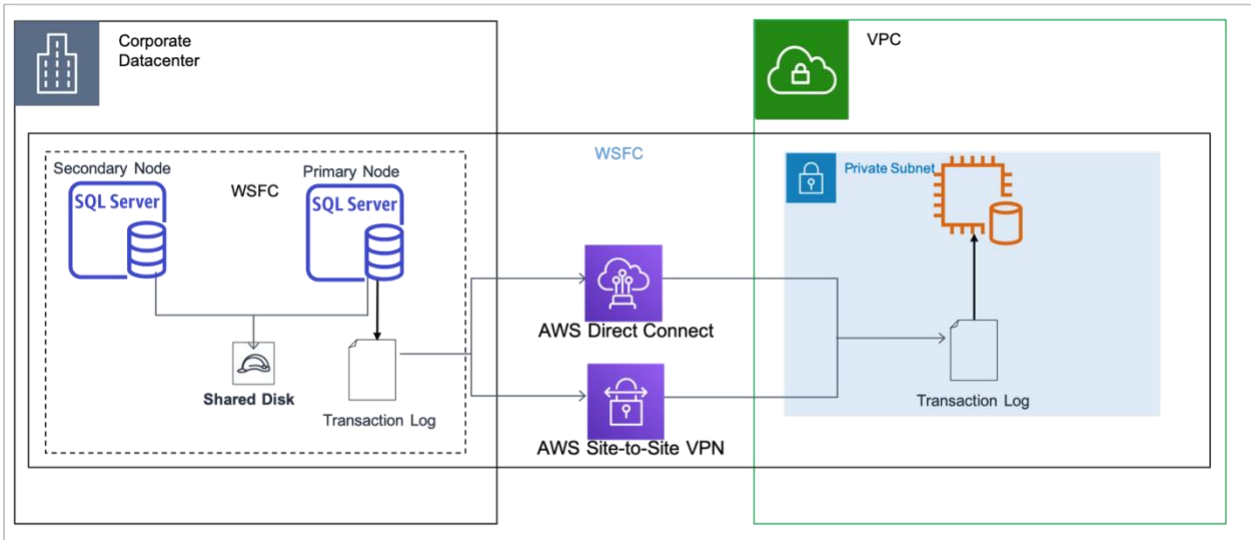

Figure 3: Hybrid SQL Server Log Shipping

SQL Server Always On Availability Groups Between On-Premises and Amazon EC2

SQL Server Always On availability groups is an advanced, enterprise-level feature to provide high availability and disaster recovery solutions. This feature is available when deploying the Enterprise Edition of SQL Server 2012, 2014, 2016, or 2017 within the AWS Cloud on Amazon EC2 or on physical or virtual machines deployed in on-premises or colocated data centers. SQL Server 2016 and SQL Server 2017 standard edition provides basic high availability two node single database failover non-readable secondary. You can also setup the Always On availability groups on Linux based SQL Server by using PaceMaker for clustering instead of using the Windows Server Failover Clustering (WSFC). If you have existing on-premises deployments of SQL Server Always On availability groups, you might want to use the AWS Cloud to provide an even higher level of availability and disaster recovery. To do so, you can extend your data center into a VPC by using a dedicated network connection like AWS Direct Connect or setting secure VPN tunnels between these two environments.

Consider the following points when planning a hybrid implementation of SQL Server Always On availability groups:

- Establish secure, reliable, and consistent network connection between on-premises and AWS (using AWS Direct Connect or VPN).
- Create a VPC based on the Amazon VPC service.

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

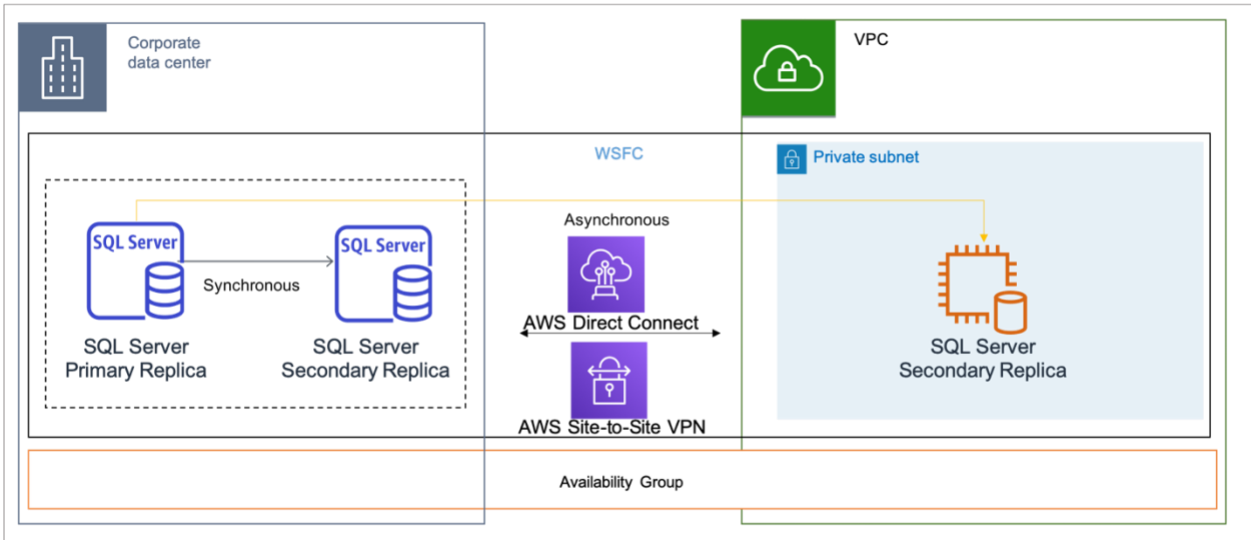

Figure 4: Always On availability groups

You can also use the distributed availability groups. This type of availability group is supported in SQL Server 2016 and later versions. Distributed availability groups span two separate availability groups and you can use them for AWS as a DR solution or migrating on premises Amazon EC2.

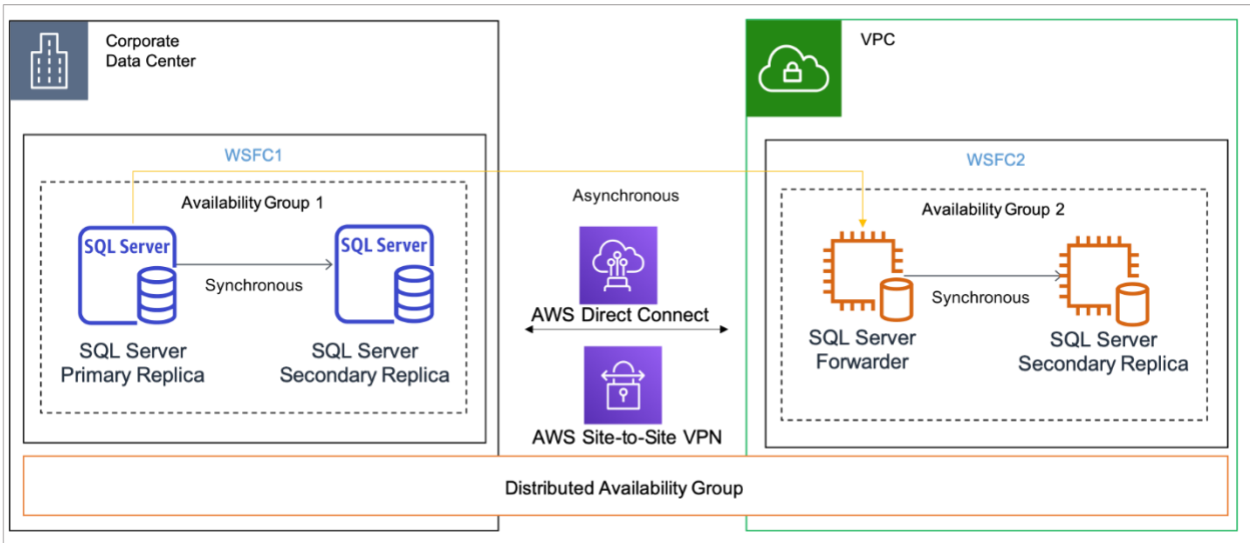

Figure 5: Hybrid Windows Server Failover Cluster

AWS Database Migration Service

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

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

Comparison of Microsoft SQL Server Feature Availability on AWS

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

Table 2: SQL Server features on AWS

	Amazon RDS	Amazon EC2
SQL Server Editions	**Supported Versions**	**Supported Versions**
Express	2012, 2014, 2016, 2017	2012, 2014, 2016, 2017
Web	2012, 2014, 2016, 2017	2012, 2014, 2016, 2017
Standard	2012, 2014, 2016, 2017	2012, 2014, 2016, 2017
Enterprise	2012, 2014, 2016, 2017	2012, 2014, 2016, 2017
SQL Server Editions	**Installation Method**	**Installation Method**
Express	N/A	AMI, Manual install
Web	N/A	AMI, Manual install
Standard	N/A	AMI, Manual install
Enterprise	N/A	AMI, Manual install
Manageability Benefits	**Supported**	**Supported**
Managed Automated Backups	Yes	No (need to configure and manage maintenance plans, or use third-party solutions)
Multi-AZ with Automated Failover	Yes	Enterprise Edition only (with manual configuration of Always On Availability Groups)
Built-in Instance and Database Monitoring and Metrics	Yes	No (push your own metrics to CloudWatch or use third-party solution)
Automatic Software Patching	Yes	No
Preconfigured Parameters	Yes	No (default SQL Server installation only)
DB Event Notifications	Yes	No (manually track and manage DB events)
SQL Server Feature	**Supported**	**Supported**
SQL Authentication	Yes	Yes
Windows Authentication	Yes	Yes

	Amazon RDS	Amazon EC2
TDE (encryption at rest)	Yes (Enterprise Edition only)	Yes (Enterprise Edition only)
Encrypted Storage using AWS KMS	Yes (all editions except Express)	Yes
SSL (encryption in transit)	Yes	Yes
Database Replication	No (Limited Push Subscription)	Yes
Log Shipping	No	Yes
Database Mirroring	Yes (Multi-AZ)	Yes
Always On Availability Groups	Yes	Yes
Max Number of DBs per Instance	Depends on the instance size and Multi-AZ configuration	None
Rename existing databases	Yes (Single-AZ only)	Yes (not available for databases in Availability Groups or enabled for mirroring)
Max Size of DB Instance	16 TiB	None
Min Size of DB Instance	20 GB (Web, Express) 200 GB (Standard, Enterprise)	None
Increase Storage Size	Yes	Yes
BACKUP Command	Yes	Yes
RESTORE Command	Yes	Yes
SQL Server Analysis Services	Data source only*	Yes
SQL Server Integration Services	Data source only*	Yes
SQL Server Reporting Services	Data source only*	Yes
Data Quality Services	No	Yes
Master Data Services	No	Yes
Custom Set Time Zones	Yes	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

	Amazon RDS	Amazon EC2
SQL Server Agent	Yes	Yes
Safe CLR	Yes	Yes
Full-text search	Yes (except semantic search)	Yes
Spatial and location features	Yes	Yes
Change Data Capture	Yes (Enterprise Edition –All versions, 2016/2017 Standard edition)	Yes
Change Tracking	Yes	Yes
Columnstore Indexes	2012 and later (Enterprise)	2012 and later (Standard, Enterprise)
Flexible Server Roles	2012 and later	2012 and later
Partially Contained Databases	2012 and later	2012 and later
Sequences	2012 and later	2012 and later
THROW statement	2012 and later	2012 and later
UTF-16 Support	2012 and later	2012 and later
New Query Optimizer	2014 and later	2014 and later
Delayed Transaction Durability (lazy commit)	2014 and later	2014 and later
Maintenance Plans	No**	Yes
Database Mail	Yes	Yes
Linked Servers	Yes	Yes
MSDTC	No	Yes
Service Broker	Yes (except Endpoints)	Yes
Performance Data Collector	No	Yes
WCF Data Services	No	Yes
FILESTREAM	No	Yes
Policy-Based Management	No	Yes
SQL Server Audit	Yes	Yes
BULK INSERT	No	Yes
OPENROWSET	Yes	Yes
Data Quality Services	No	Yes
Buffer Pool Extensions	No	Yes
Stretch Database	No	Yes

	Amazon RDS	Amazon EC2
Resource Governor	No	Yes
Polybase	No	Yes
Machine Learning & R Services	No	Yes
File Tables	No	Yes

* Amazon RDS SQL Server DB instances can be used as data sources for SSRS,

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

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

For detailed list of features supported by the editions of SQL Server, see High Availability in Microsoft Documentation.

Conclusion

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

Contributors

Contributors to this document include:

- Jugal Shah, Solutions Architect, Amazon Web Services
- Richard Waymire, Outbound Principal Architect, Amazon Web Services
- Russell Day, Solutions Architect, Amazon Web Services
- Darryl Osborne, Solutions Architect, Amazon Web Services
- Vlad Vlasceanu, Solutions Architect, Amazon Web Services

Further Reading

For additional information, see:

- Microsoft Products on AWS
- Active Directory Reference Architecture: Implementing Active Directory Domain Services on AWS
- Remote Desktop Gateway on AWS
- Securing the Microsoft Platform on AWS
- Implementing Microsoft Windows Server Failover Clustering and SQL Server Always On Availability Groups in the AWS Cloud
- AWS Directory Service
- SQL Server Database Restore to Amazon EC2 Linux

Document Revisions

Date	Description
November 2019	Updated with information on new features and changes: release of SQL Server 2016 and 2017 in RDS, RDS Backup, and SQL Server on EC2 Linux new instance classes; updated screen captures, architecture diagrams, optimize CPU, Hybrid Scenarios and other minor corrections and content updates.
June 2016	Updated with information on new features and changes: release of Amazon RDS SQL Server Windows Authentication; availability of SQL Server 2014 in Amazon RDS; new RDS Reserved DB Instance pricing model, availability of the AWS Database Migration Service; other minor corrections and content updates.
May 2015	First publication