Migrating Microsoft SQL Server databases to the AWS Cloud

AWS Prescriptive Guidance

AWS Prescriptive Guidance: Migrating Microsoft SQL Server databases to the AWS Cloud

Table of Contents

Migrating Microsoft SQL Server databases to the AWS Cloud

Sagar Patel, Amazon Web Services (AWS)

April 2024 (document history)

Amazon Web Services (AWS) provides a comprehensive set of services and tools for deploying Microsoft SQL Server databases on the reliable and secure AWS Cloud infrastructure. Benefits of running SQL Server on AWS include cost savings, scalability, high availability and disaster recovery, better performance, and ease of management. For more information, see Learn why AWS is the best cloud to run Microsoft Windows Server and SQL Server workloads on the AWS Compute blog.

This guide describes the options available for migrating SQL Server databases from on premises to the AWS Cloud, to Amazon Relational Database Service (Amazon RDS), Amazon Elastic Compute Cloud (Amazon EC2), or VMware Cloud on AWS. It dives into the best practices and recommendations for using these migration options. It also provides information about how to set up a high availability and 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 guide is for program or project managers, product owners, database administrators, database engineers, and operations or infrastructure managers who are planning to migrate their on-premises SQL Server databases to AWS.

Overview

Before you migrate your SQL Server databases to AWS, you should understand and evaluate your migration strategy by using the framework discussed in Migration strategy for relational databases.

The first step is to perform an analysis of your application and SQL Server database workloads by understanding the complexity, compatibility, and cost of migration. Here are some of the top points you should consider when you plan to migrate:

- **Database size** – Check the current size and overall capacity growth of your database. For example, if you're planning to migrate your SQL Server database to Amazon RDS or Amazon RDS

Custom, you can create DB instances with up to 16 TiB of storage. You can request more storage by opening a support ticket with AWS Support. For the latest information, see Amazon RDS DB instance storage in the Amazon RDS documentation.

- **IOPS** – Determine the IOPS and throughput of your databases. If you're planning to migrate to Amazon RDS, consider the I/O performance of Amazon RDS DB instances.
- **Dependencies** – Check current database dependencies. If your database is dependent on other databases, you can either migrate them together or create dependencies after you migrate your main database.

  If your database supports legacy, custom, or packaged applications, Amazon RDS Custom for SQL Server might be a good choice. This service lets you retain control over database configurations, shared file systems, and operating system patches.

  Inventory all SQL Server dependencies. Find out which web servers (for example, reporting servers or business intelligence servers) interface with SQL Server. When it's time to migrate, this information helps you determine what will be impacted and how you can minimize the impact.
- **Compliance** – Review your current architecture and auditing or compliance needs, to make sure you can satisfy these requirements after moving to Amazon RDS or Amazon EC2.
- **HA/DR** – Do you need high availability (HA) and automated failover capabilities? If you are running a production workload, high availability and disaster recovery (DR) are recommended best practices.

  Understand your HA/DR requirements to determine whether you need a multi-Region architecture. If so, migrate your SQL Server database to Amazon EC2. Amazon RDS doesn't support a multi-Region configuration.
- **Version support** – Check the version and edition of your SQL Server software if you're planning to move to Amazon RDS for SQL Server (see currently supported versions for Amazon RDS and Amazon RDS).
- **Network connectivity** – Check the network connectivity between your on-premises environment and AWS, to make sure that it provides enough bandwidth for fast transfers of data between on premises and AWS.
- **Migration downtime** – Determine the amount of downtime available for migration so you can plan your migration approach and decide whether you want to use online or offline migration.
- **RTO, RPO, SLA requirements** – Identify your recovery time objective (RTO), recovery point objective (RPO), and service-level agreement (SLA) requirements for your existing database workloads.

- **Licensing** – Understand your licensing options. You can choose license-included options on Amazon EC2 and Amazon RDS, or choose to bring your own license (BYOL) on Amazon EC2.
- **Feature support** – Identify the database features and functionality that your application uses, whether it was developed in-house or it's commercial-off-the-shelf (COTS) software. This information can help you determine whether you can reduce your licensing costs by switching from SQL Server Enterprise edition to Standard edition. However, review Standard edition resource limitations before you switch. For example, Standard edition supports only 128 GB of RAM.

  Does your workload fit within the features and capabilities offered by Amazon RDS for SQL Server? For more information, see SQL Server features on Amazon RDS. If you need features that aren't supported, migrating to Amazon EC2 is an option.

SQL Server database migration strategies

At a high level, there are two options for migrating a SQL Server database from on premises to the AWS Cloud: either stay on SQL Server (*homogeneous migration*) or move off SQL Server (*heterogeneous migration*). In a homogeneous migration, you don't change the database engine. That is, your target database is also a SQL Server database. In a heterogeneous migration, you switch your SQL Server databases either to an open-source database engine such as MySQL, PostgreSQL, or MariaDB, or to an AWS Cloud-native database such as Amazon Aurora, Amazon DynamoDB, or Amazon Redshift.

There are three common strategies for migrating your SQL Server databases to AWS: rehost, replatform, and re-architect (refactor). These are part of the 7 Rs of application migration strategies and described in the following table.

Strategy	Type	When to choose	Example
Rehost	Homogeneous	You want to migrate your SQL Server database as is, with or without changing the operating system, database software, or configuration.	SQL Server to Amazon EC2 (Browse rehost patterns)
Replatform	Homogeneous	You want to reduce the time you spend managing database instances by using a fully managed database offering.	SQL Server to Amazon RDS for SQL Server (Browse Replatform patterns)
Re-architect (refactor)	Heterogeneous	You want to restructure, rewrite, and re-architect your database and application to take advantage of open-	SQL Server to Amazon Aurora PostgreSQL, MySQL, or MariaDB

Strategy	Type	When to choose	Example
		source and cloud-nat ive database features.	Browse Re-architect patterns)

If you're trying to decide between rehosting or replatforming your SQL Server databases, see Choosing between Amazon EC2 and Amazon RDS later in this guide for a side-by-side comparison of supported features.

Choosing the right migration strategy

Choosing the correct strategy depends on your business requirements, resource constraints, migration timeframe, and cost considerations. The following diagram shows the effort and complexity involved in migrations, including all seven strategies.

Refactoring your SQL Server database and migrating to an open-source or AWS Cloud-native database such as Amazon Aurora PostgreSQL-Compatible Edition or Aurora MySQL-Compatible Edition can help you modernize and optimize your database. By moving to an open-source database, you can avoid expensive licenses (resulting in lower costs), vendor lock-in periods, and audits. However, depending on the complexity of your workload, refactoring your SQL Server database can be a complicated, time-consuming, and resource-intensive effort.

To reduce complexity, instead of migrating your database in a single step, you might consider a phased approach. In the first phase, you can focus on core database functionality. In the next phase, you can integrate additional AWS services into your cloud environment, to reduce costs, and to optimize performance, productivity, and compliance. For example, if your goal is to replace your on-premises SQL Server database with Aurora MySQL-Compatible, you might consider rehosting your database on Amazon EC2 or replatforming your database on Amazon RDS for SQL Server in the first phase, and then refactor to Aurora MySQL-Compatible in a subsequent phase. This approach helps reduce costs, resources, and risks during the migration phase and focuses on optimization and modernization in the second phase.

Online and offline migration

You can use two methods to migrate your SQL Server database from an on-premises or another cloud environment to the AWS Cloud, based on your migration timeline and how much downtime you can allow: offline migration or online migration.

- **Offline migration:** This method is used when your application can afford a planned downtime. In offline migration, the source database is offline during the migration period. While the source database is offline, it is migrated over to the target database on AWS. After the migration is complete, validation and verification checks are performed to ensure data consistency with the source database. When the database passes all validation checks, you perform a cutover to AWS by connecting your application to the target database on AWS.
- **Online migration:** This method is used when your application requires near zero to minimal downtime. In online migration, the source database is migrated in multiple steps to AWS. In the initial steps, the data in the source database is copied to the target database while the source database is still running. In subsequent steps, all changes from the source database are propagated to the target database. When the source and target databases are in sync, they are ready for cutover. During the cutover, the application switches its connections over to the target database on AWS, leaving no connections to the source database. You can use AWS Database Migration Service (AWS DMS) or tools available from AWS Marketplace (such as Attunity) to synchronize the source and target databases.

SQL Server database migration methods

There are various methods to migrate your SQL Server databases to AWS. You can choose from AWS services and SQL Server native features based on your assessment and requirements. This section describes some of the most common methods, which are summarized in the following two tables. Detailed discussions of some of these methods are included in the sections on Amazon EC2 and Amazon RDS later in this guide.

AWS services

Migration method	Target	Features and limitations	More information
AWS DMS	Amazon EC2 Amazon RDS Amazon RDS Custom Amazon Aurora	• Supports full load and CDC • Supports all database sizes	AWS DMS section
AWS Migration Hub Orchestrator	Amazon EC2 Amazon RDS	• Provides predefined, step-by-step workflow templates • Automates native backup and restore • Supports all SQL Server editions and versions • Can be applied to one or many databases at one time • Supports all database sizes	AWS Migration Hub Orchestrator section

Migration method	Target	Features and limitations	More information
AWS Application Migration Service	Amazon EC2	• Highly automated lift-and-shift solution • Agent-based, block-level replication	Not covered in this guide (see Application Migration Service documentation)
AWS Snowball Edge	Amazon EC2 Amazon RDS Amazon RDS Custom	• Supports very large databases (up to 210 TB) • Uses Amazon Simple Storage Service (Amazon S3) for storing and restoring data	Snowball Edge section

SQL Server native methods

Migration method	Target	Features and limitations	More information
Native backup and restore	Amazon EC2 Amazon RDS Amazon RDS Custom	• Can be applied to one or many databases at one time • Requires downtime • Supports all database sizes	Native SQL Server backup/restore section (you can use AWS Migration Hub Orchestrator to automate native backup and restore)
Log shipping	Amazon EC2 Amazon RDS	• Applied per database • Can be delayed	Log shipping section

Migration method	Target	Features and limitations	More information
	Amazon RDS Custom		
Database mirroring	Amazon EC2	• Applied per database • Can be synchrono us or asynchronous, based on the SQL Server edition • Secondary database isn't readable; it acts as a standby • Supports both automatic and manual failover	Database mirroring section

Migration method	Target	Features and limitations	More information
Always On availability groups	Amazon EC2 Amazon RDS Custom	• Applied to a set of user databases • Can be synchronous or asynchronous • Secondary database is readable (SQL Server Enterprise edition only) • Supports both automatic and manual failover • Failover can be initiated for multiple databases at a time, at the database group level	Always On availability groups section

Migration method	Target	Features and limitations	More information
Basic Always On availability groups	Amazon EC2 Amazon RDS Custom	• Supported in SQL Server Standard edition • Applied to a single user database per availability group • Can be synchronous or asynchronous • Supports both automatic and manual failover • Failover can be initiated at the availability group level • Can be used as a hybrid environment between on premises and AWS	Not covered in this guide (see Basic Always On availability groups for a single database in the Microsoft documentation)

Migration method	Target	Features and limitations	More information
Distributed availability groups	Amazon EC2 Amazon RDS Custom	• Can be used for multi-Region SQL Server deployments • Can fail over to a later version of SQL Server • Doesn't require Windows Server Failover Clustering (WSFC) to be extended to the target AWS environment • Can be used between Windows-based (source) and Linux-based (target) SQL Server databases • Can be used as a hybrid SQL Server deployment between on premises and AWS	Distributed availability groups section

Migration method	Target	Features and limitations	More information
Transactional replication	Amazon EC2 Amazon RDS Amazon RDS Custom	• Supports migration of a set of objects (tables, view, stored procedures) • Supports asynchronous replication with near real-time data • Subscriber database is readable • Requires close monitoring of SQL Server replication jobs that perform the replication	Transactional replication section
Bulk copy program (bcp)	Amazon EC2 Amazon RDS Custom	• Supports small databases • Requires downtime • Schema is pre-created at the destination • Used for moving data, but not metadata	Not covered in this guide (see Importing and exporting SQL Server data using other methods, *Bulk copy* section in the Amazon RDS documentation)

Migration method	Target	Features and limitations	More information
Detach and attach	Amazon EC2 Amazon RDS Custom	• No backup needed • Requires downtime • Involves stopping, detaching, copying files, and attaching to Amazon EC2	Not covered in this guide (see Database Detach and Attach in the Microsoft documentation)
Import/export	Amazon EC2 Amazon RDS Custom	• Supports small databases • Requires downtime • Schema is pre-created at the destination • Used for moving data, but not metadata	Not covered in this guide (see Importing and exporting SQL Server data using other methods in the Amazon RDS documentation)

Native SQL Server backup/restore

Amazon RDS supports native backup and restore operations for Microsoft SQL Server databases using full and differential backup (.bak) files. It also supports differential restore and log restore options on an Amazon RDS for SQL Server DB instance or Amazon EC2 SQL Server instance, to minimize downtime for your application.

> **Note**
>
> You can perform full, differential, and log restore operations on Amazon RDS for SQL Server. However, you can perform only full and differential backup (not log backup) at this time.

Using native .bak files is the simplest way to back up and restore SQL Server databases. You can use this method to migrate databases to or from Amazon RDS. You can back up and restore single

databases instead of entire DB instances. You can also move databases between Amazon RDS for SQL Server DB instances.

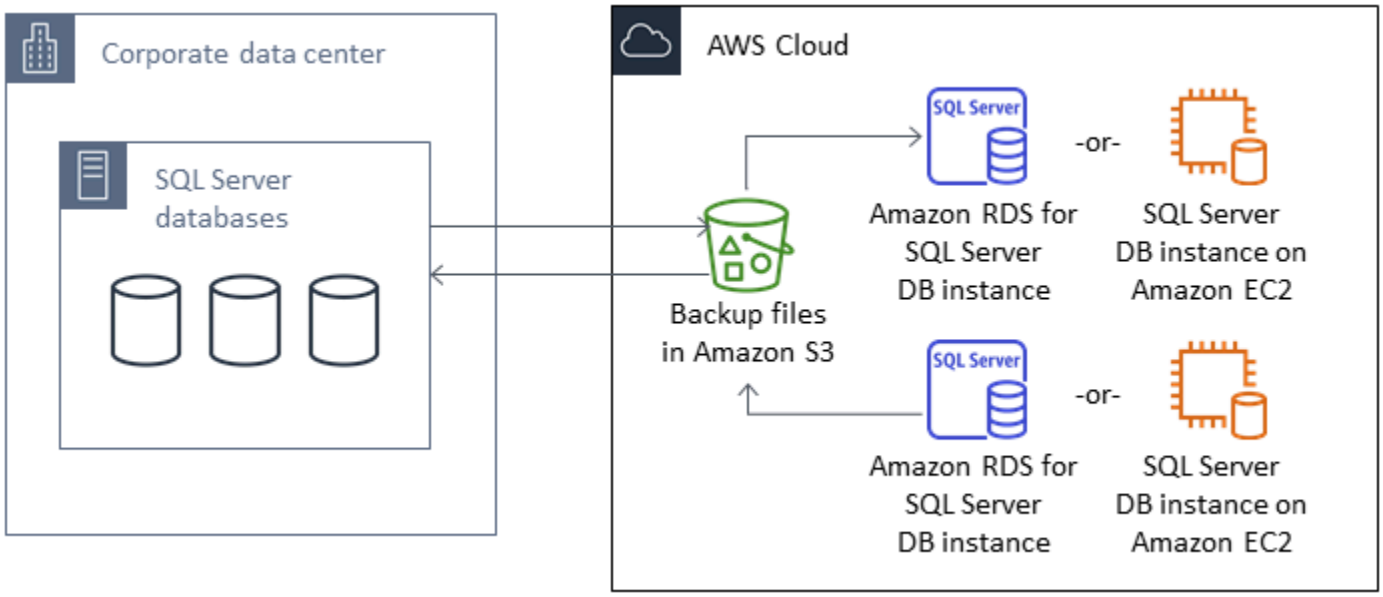
When you use Amazon RDS, you can store and transfer backup files in Amazon Simple Storage Service (Amazon S3), for an added layer of protection for disaster recovery. For example:

- You can create a full backup of your database from your local server, copy it to an S3 bucket, and then restore it onto an existing Amazon RDS SQL Server DB instance.
- You can take backups from an Amazon RDS for SQL Server DB instance, store them in Amazon S3, and then restore them wherever you want.
- You can implement Amazon S3 Lifecycle configuration rules to archive or delete long-term backups.

Amazon RDS for SQL Server supports restoring SQL Server native backups onto SQL Server DB instances that have read replicas configured. This means that you don't have to remove the read replica before restoring the native backup file onto your Amazon RDS for SQL Server DB instance.

> **Note**
>
> You can use Migration Hub Orchestrator to automate and orchestrate your SQL Server database migrations to Amazon EC2 or Amazon RDS by using native backup and restore. For more information, see the AWS Migration Hub Orchestrator section.

The following diagram shows the native SQL Server backup/restore process. You can use Migration Hub Orchestrator to automate this process. You can use this process to back up and restore SQL Server databases to Amazon EC2 as well.



To automate backup and restore, see the Migration Hub Orchestrator documentation.

To set up native backup/restore using Amazon S3, see the Amazon RDS documentation.

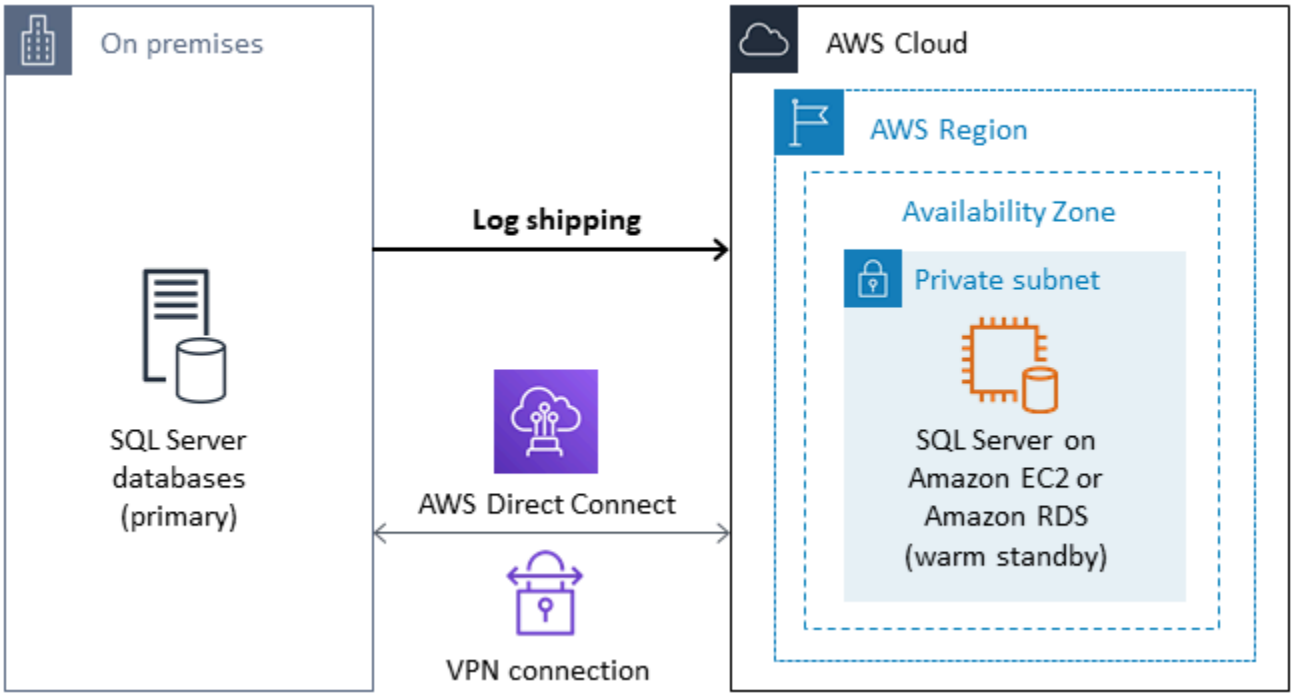
For limitations when using SQL Server native backup and restore, see Limitations and recommendations in the Amazon RDS documentation.

Log shipping

You can use log shipping to send transaction log backups from your primary, on-premises SQL Server database to one or more secondary (warm standby) SQL Server databases that are deployed on EC2 instances or Amazon RDS for SQL Server DB instances in the AWS Cloud. To set up log shipping on Amazon RDS for SQL Server, you have to use your own custom scripts.

In this scenario, you configure a warm standby SQL Server database on an EC2 instance or Amazon RDS for SQL Server DB instance, and send transaction log backups asynchronously between your on-premises database and the warm standby server in the AWS Cloud. The transaction log backups are then applied to the warm standby database. When all the logs have been applied, you can perform a manual failover and cut over to the cloud.

This option supports all versions and editions of SQL Server. After you have migrated the database to the AWS Cloud, you can add a secondary replica by using an Always On availability group for high availability and resiliency purposes.



For more information about using this method to achieve high availability, data protection, and disaster recovery for your SQL Server databases on Amazon EC2, see Log shipping in the *Amazon EC2 for SQL Server* section.

Database mirroring

You can use database mirroring to set up a hybrid cloud environment for your SQL Server databases. This option requires SQL Server Enterprise edition. In this scenario, your principal SQL Server database runs on premises, and you create a warm standby in the cloud. You replicate your data asynchronously, and perform a manual failover when you're ready for cutover. After you have migrated the database to the AWS Cloud, you can add a secondary replica by using an Always On availability group for high availability and resiliency purposes.

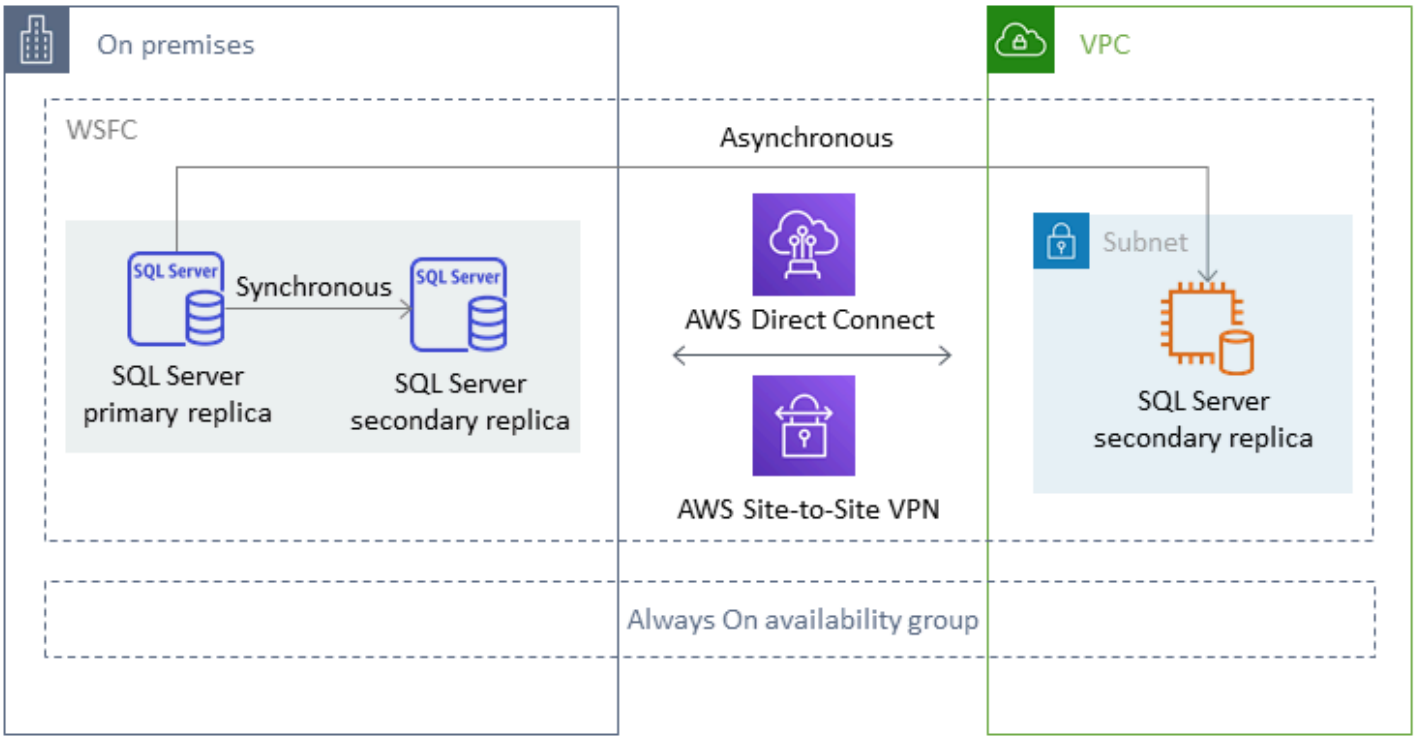
For more information about using this method to achieve high availability, data protection, and disaster recovery for your SQL Server databases on Amazon EC2, see Database mirroring in the *Amazon EC2 for SQL Server* section.

Always On availability groups

SQL Server Always On availability groups is an advanced, enterprise-level feature to provide high availability and disaster recovery solutions. This feature is available if you are using SQL Server

2014 and later versions. You can also use an Always On availability group to migrate your on-premises SQL Server databases to Amazon EC2 on AWS. This approach enables you to migrate your databases with minimal to no downtime.

If you have an existing on-premises deployment of SQL Server Always On availability groups, your primary replica and secondary replica will be synchronously replicating data within the availability group. So, to migrate your database to AWS Cloud, you can extend your Windows Server Failover Clustering (WSFC) cluster to the cloud. This can be temporary, just for migration purposes. You then create a secondary replica in the AWS Cloud and use asynchronous replication, as shown in the following diagram. After the secondary replica is synchronized with the primary on-premises database, you can perform a manual failover whenever you are ready for cutover.



For more information about using this method to achieve high availability, data protection, and disaster recovery for your SQL Server databases on Amazon EC2, see Always On availability groups in the *Amazon EC2 for SQL Server* section.

Distributed availability groups

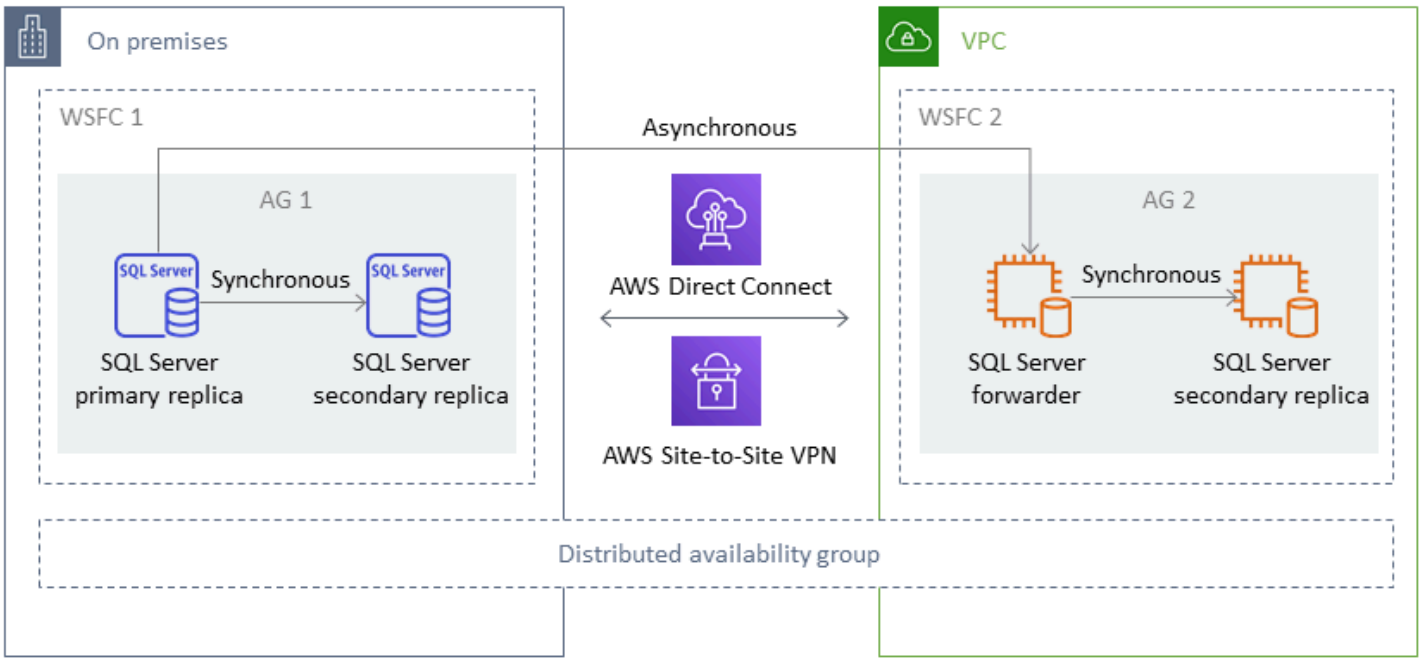
A distributed availability group spans two separate availability groups. You can think of it as an availability group of availability groups. The underlying availability groups are configured on two

different WSFC clusters. The availability groups that participate in a distributed availability group do not need to share the same location. They can be physical or virtual, on premises or in the public cloud. The availability groups in a distributed availability group don't have to run the same version of SQL Server. The target DB instance can run a later version of SQL Server than the source DB instance.

A distributed availability group architecture gives you a flexible way to rehost a mission-critical SQL Server instance or database on AWS. It provides a hybrid solution for lifting and shifting (or lifting and transforming) your critical SQL Server databases on AWS.

Using a distributed availability group architecture is more efficient than extending existing on-premises WFSC clusters to AWS. Data is transferred only from the on-premises primary to one of the AWS replicas (the *forwarder*). The forwarder is responsible for sending data to other secondary read replicas in AWS.

In the following diagram, the first WSFC cluster (WSFC 1) is hosted on premises and has an on-premises availability group (AG 1). The second WSFC cluster (WSFC 2) is hosted on AWS and has an AWS availability group (AG 2). AWS Direct Connect is used as a dedicated network connection between the on-premises environment and AWS. The on-premises availability group (AG 1) has two replicas (*nodes*). The data transfer between the nodes is synchronous, with automatic failover. Similarly, the AWS availability group (AG 2) also has two replicas, and the data transfer between them is synchronous with automatic failover. The distributed availability group keeps the databases in sync in an asynchronous manner. Data is transferred from the SQL Server primary replica in AG 1 (which is on premises) to the primary replica (the forwarder) in AG 2 (which is on AWS). The forwarder is responsible for sending data to other read replicas on AWS and keeping them updated. After the on-premises and AWS databases are synchronized, you can perform a manual failover of the distributed availability group to AWS. The AWS database becomes the primary database for read/write access from applications.



> **Note**
>
> At any given point of time, there is only one database that is available for write operations. You can use the remaining secondary replicas for read operations. To scale out your read workloads, you can add more read replicas in multiple Availability Zones on AWS.

For more information about distributed availability groups, see:

- Microsoft SQL Server documentation
- How to architect a hybrid Microsoft SQL Server solution using distributed availability groups on the AWS Database blog
- Migrate SQL Server to AWS using distributed availability groups on the AWS Prescriptive Guidance website

Transactional replication

Transactional replication is a SQL Server technology that is used to replicate changes between two databases. These changes can include database objects like tables (primary key is required), stored procedures, views, and so on, as well as data. The replication process involves a *publisher* (the primary database that publishes data), a *subscriber* (a secondary database that receives

replicated data), and a *distributor* (a server that stores metadata and transactions for transactional replication). You can use transactional replication for SQL Server on Amazon EC2 and Amazon RDS for SQL Server DB instances.

Transactional replication creates a snapshot of the objects and data in your on-premises (publication) database and sends it to the subscriber database. After the snapshot is applied to the subscriber, all subsequent data changes and schema modifications made at the publisher are sent over to the subscriber as they occur. The data changes are then continuously applied to the subscriber in the same order as they occurred at the publisher.

After synchronization is complete, you perform validation on the target SQL Server DB instance. When the two databases are in sync, you stop stop the activity on the on-premises database, ensure that replication has completed, and then perform the cutover to the target SQL Server DB instance. You can then stop the push subscription, delete it, and start using Amazon RDS for SQL Server.

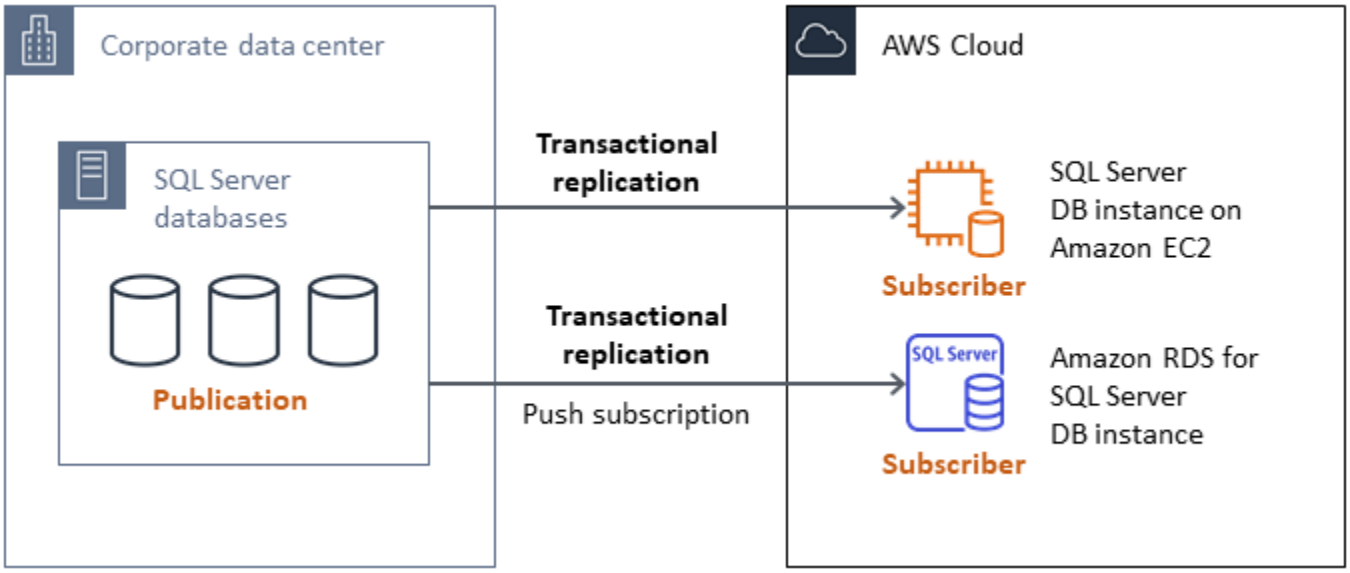
Subscriber databases can also be used as read-only databases. The distributor, which records synchronization jobs, is recommended to be on a separate server. If your target database is on Amazon RDS for SQL Server, you can set up a push subscription to propagate changes to the subscriber.

We recommend that you use transactional replication when you want to:

- Perform a one-time migration of your data to Amazon RDS or Amazon EC2.
- Migrate schema-level or table-level objects to AWS.
- Migrate a portion of a database to AWS.
- Migrate with minimal downtime using existing SQL Server replication strategies by adding additional subscribers.

If you're planning to use transactional replication for one-time migration of your data to Amazon RDS for SQL Server, we recommend that you set up a Single-AZ configuration for the replication. After the replication process is complete, you can convert your environment into a Multi-AZ architecture for high availability.

The following diagram shows the transactional replication process for databases on Amazon RDS and Amazon EC2.



For more information about transactional replication, see the Microsoft SQL Server documentation and the post How to migrate to Amazon RDS for SQL Server using transactional replication on the AWS Database blog.

AWS Migration Hub Orchestrator

AWS Migration Hub Orchestrator helps you orchestrate and automate the migration of SQL Server databases to Amazon EC2 or Amazon RDS. This feature of AWS Migration Hub helps you get started quickly by using predefined workflow templates that are built based on best practices. Migration Hub Orchestrator automates error-prone manual tasks involved in the migration process, such as checking environment readiness and connections. You can also use Migration Hub Orchestrator to orchestrate and accelerate migrations for .NET applications, SAP workloads, and virtual machine images, in addition to your SQL Server databases. You can access this tool through the Migration Hub Orchestrator console.

For SQL Server migration, Migration Hub Orchestrator supports three use cases:

- Rehost SQL Server on Amazon EC2. You can choose specific SQL servers and rehost them on Amazon EC2 by using automated native backup and restore in Migration Hub Orchestrator. To learn more, see Rehost SQL server on Amazon EC2 in the Migration Hub Orchestrator documentation.
- Replatform SQL Server on Amazon RDS. You can choose specific SQL Server databases and replatform them on Amazon RDS by using automated native backup and restore in Migration

Hub Orchestrator. To learn more, see Replatform SQL server on Amazon RDS in the Migration Hub Orchestrator documentation.

- Rehost Windows and SQL Server applications on Amazon EC2. You can lift and shift your Windows servers running .NET and SQL Server to Amazon EC2 by using the *Rehost applications on Amazon EC2 template*. To learn more, see Rehost applications on Amazon EC2 in the Migration Hub Orchestrator documentation.

Migration Hub Orchestrator helps avoid schedule and budget overruns in your SQL Server migrations. Other key benefits include:

- Migrate applications by using a prescriptive methodology. You can get started quickly with the predefined workflow templates, which are based on proven migration best practices. You can also customize your migration workflow by adding, reordering, and removing steps based on your needs. For example, you can add a step for cutover approval.
- Automate manual steps. Migration Hub Orchestrator automates manual tasks such as installing agents, importing on-premises images, provisioning your target environment on AWS, and verifying source and target environments. Automation saves you time and costs while reducing errors.
- Orchestrate migration workflow. Migration Hub Orchestrator orchestrates the tools used in migration steps by reusing the inventory metadata, configuration specification, and environment context to minimize the number of inputs that these tools require.

For additional information, see the following resources:

- Migration Hub Orchestrator console
- Rehost applications on Amazon EC2 (*Migration Hub Orchestrator User Guide*)
- Replatform SQL server on Amazon RDS (*Migration Hub Orchestrator User Guide*)
- Migration workflows (*Migration Hub Orchestrator User Guide*)
- Using Migration Hub Orchestrator to simplify and accelerate Microsoft SQL Server migrations (AWS blog post)
- Simplify migrating your Windows Server images with AWS Migration Hub Orchestrator (AWS blog post)

AWS Snowball Edge

You can use AWS Snowball Edge to migrate very large databases (up to 210 TB in size). Snowball has a 10 Gb Ethernet port that you plug into your on-premises server and place all database backups or data on the Snowball device. After the data is copied to Snowball, you send the appliance to AWS for placement in your designated S3 bucket. You can then download the backups from Amazon S3 and restore them on SQL Server on an EC2 instance, or run the `rds_restore_database` stored procedure to restore the database to Amazon RDS. You can also use AWS Snowcone for databases up to 8 TB in size. For more information, see the AWS Snowball Edge documentation and Importing and exporting SQL Server databases, *Restoring a database* section, in the Amazon RDS documentation.

Homogeneous database migration for SQL Server

AWS offers you the ability to run SQL Server databases in a cloud environment. For developers and database administrators, running SQL Server database in the AWS Cloud is very similar to running SQL Server database in a data center. This section describes options for migrating your SQL Server database from an on-premises environment or a data center to the AWS Cloud.

AWS offers three options for running SQL Server on AWS, as described in the following table.

Option	Highlights	More information
SQL Server on Amazon RDS	Managed service, provides easy provisioning and licensing, cost-effective, easy to set up, manage, and maintain.	Amazon RDS for SQL Server section
SQL Server on Amazon RDS Custom	Managed service, but you retain administrative rights to the database and the underlying operating system.	Amazon RDS Custom for SQL Server section
SQL Server on Amazon EC2	Self-managed, provides full control and flexibility.	Amazon EC2 for SQL Server section
SQL Server on VMware Cloud on AWS	Set up, scale, and operate your SQL Server workloads on VMware Cloud on AWS and integrate with AWS Directory Service, Active Directory Connector, and Amazon S3.	VMware Cloud on AWS for SQL Server section

Your application requirements, database features, functionality, growth capacity, and overall architecture complexity will determine which option to choose. If you are migrating multiple SQL Server databases to AWS, some of them might be a great fit for Amazon RDS, whereas others might be better suited to run directly on Amazon EC2. You might have databases that are running on SQL Server Enterprise edition but are a good fit for SQL Server Standard edition. You might

also want to modernize your SQL Server database running on Windows to run on a Linux operating system to save on cost and licenses. Many AWS customers run multiple SQL Server database workloads across Amazon RDS, Amazon EC2, and VMware Cloud on AWS.

> **Note**
>
> You can use Migration Hub Orchestrator to automate and orchestrate your SQL Server database migrations to Amazon EC2 or Amazon RDS by using native backup and restore. For more information, see the AWS Migration Hub Orchestrator section.

Amazon RDS for SQL Server

Amazon RDS for SQL Server is a managed database service that simplifies the provisioning and management of SQL Server on AWS. Amazon RDS makes it easy to set up, operate, and scale SQL Server deployments in the cloud. With Amazon RDS, you can deploy multiple versions of SQL Server (2014, 2016, 2017, 2019, and 2022) and editions (including Express, Web, Standard and Enterprise) in minutes, with cost-efficient and resizable compute capacity. You can provision Amazon RDS for SQL Server DB instances with either General Purpose SSD or Provisioned IOPS SSD storage. (For details, see Amazon RDS Storage Types in the AWS documentation.) Provisioned IOPS SSD is designed to deliver fast, predictable, and consistent I/O performance, and is optimized for I/O-intensive, transactional (OLTP) database workloads.

Amazon RDS frees you to focus on application development, because it manages time-consuming database administration tasks, including provisioning, backups, software patching, monitoring, and hardware scaling. Amazon RDS for SQL Server also offers Multi-AZ deployments and read replicas (for SQL Server Enterprise edition) to provide high availability, performance, scalability, and reliability for production workloads.

For more information about migrating from SQL Server to Amazon RDS, see the replatform patterns on the AWS Prescriptive Guidance website.

When to choose Amazon RDS

Amazon RDS for SQL Server is a migration option when:

- You want to focus on your business and applications, and you want AWS to take care of undifferentiated heavy lifting tasks such as the provisioning of the database, management of

backup and recovery tasks, management of security patches, minor SQL Server version upgrades, and storage management.

- You need a highly available database solution, and you 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 want to pay for the SQL Server license as part of the instance cost on an hourly basis instead of making a large, upfront investment.
- Your database size and IOPS needs are supported by Amazon RDS for SQL Server. See Amazon RDS DB Instance Storage in the AWS documentation for the current maximum limits.
- You don't want to manage backups or point-in-time recoveries of your database.
- You want to focus on high-level tasks, such as performance tuning and schema optimization, instead of the daily administration of the database.
- You want to scale the instance type up or down based on your workload patterns without being concerned about licensing complexities.

After assessing your database and project requirements, if you decide to migrate to Amazon RDS for SQL Server, see the details provided in the following sections, and review the migration best practices we discuss later in this guide.

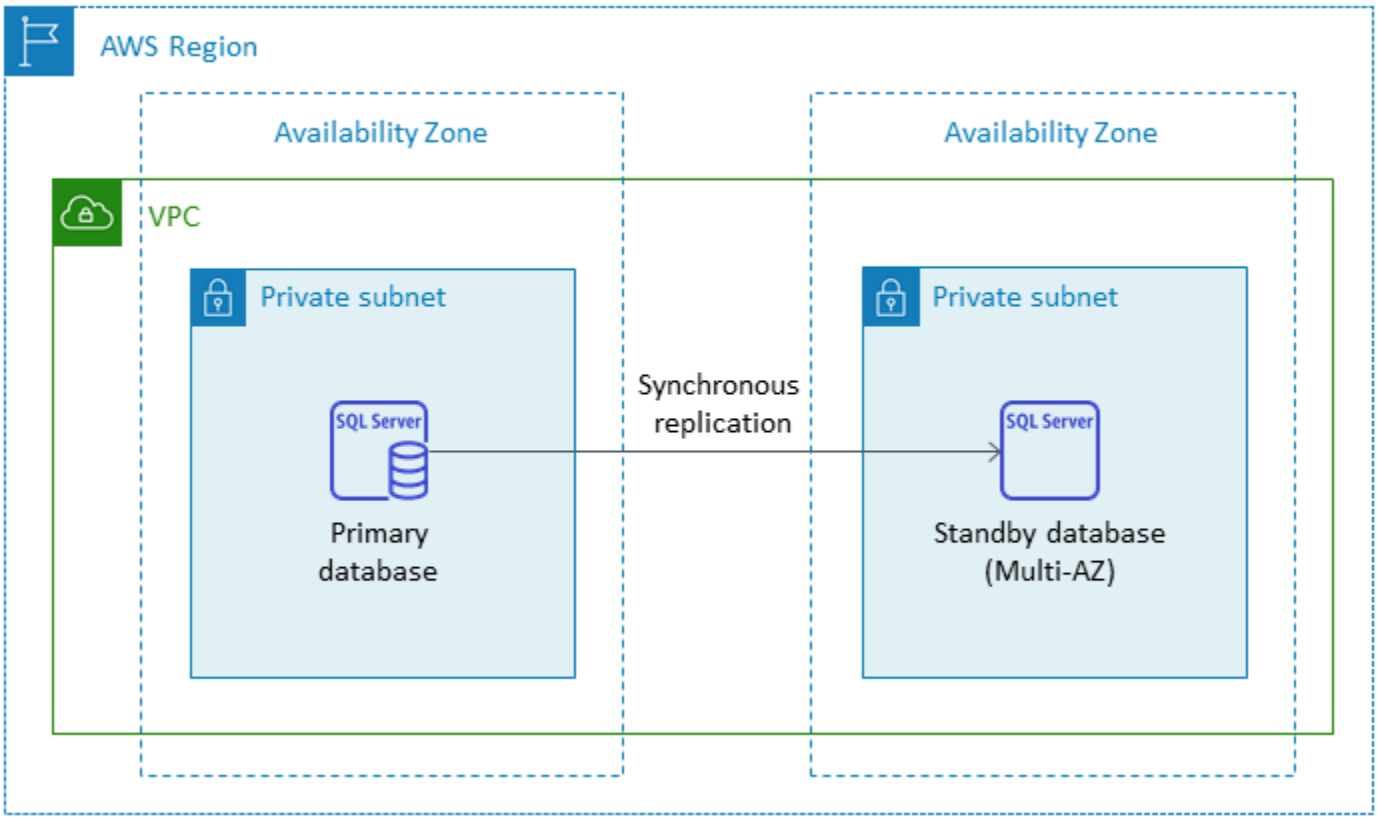
For currently supported SQL Server features, versions, and options, see Amazon RDS for SQL Server features on the AWS website, Choosing between Amazon EC2 and Amazon RDS later in this guide, and Microsoft SQL Server on Amazon RDS in the AWS documentation. If you're moving to Amazon RDS Custom, make sure to review the requirements and limitations for Amazon RDS Custom for SQL Server.

High availability

Amazon RDS provides high availability and failover support for databases that are deployed with the Multi-AZ option. When you provision your database with the Multi-AZ option, Amazon RDS automatically provisions and maintains a synchronous standby instance in a different Availability Zone. The primary database synchronously replicates the data to the standby instance. If problems occur, Amazon RDS automatically repairs the unhealthy instance and re-establishes synchronization. In case of infrastructure failure or Availability Zone disruption, Amazon RDS performs an automatic failover to the standby instance. Failover occurs only if the standby and primary databases are fully synchronized. Because the endpoint remains the same for the primary

and standby instances, you can resume database operations as soon as the failover is complete, without performing a manual intervention. The failover time depends on the time it takes to complete the recovery process. Large transactions increase the failover time.

The following diagram illustrates the Amazon RDS for SQL Server Multi-AZ deployment option.



When you set up SQL Server in a Multi-AZ configuration, Amazon RDS automatically configures standby database instance using database mirroring or Always On availability groups, based on the version of SQL Server that you deploy. The specific SQL Server versions and editions are listed in the Amazon RDS documentation.

In Multi-AZ deployments, operations such as instance scaling or system upgrades such as operating system (OS) patching are applied first on the standby instance, before the automatic failover of the primary instance, for enhanced availability.

Because of failover optimization of SQL Server, certain workloads can generate greater I/O load on the standby instance than they do on the primary instance, particularly in database mirroring deployments. This functionality can result in higher IOPS on the standby instance. We recommend that you consider the maximum IOPS needs of both the primary and standby instances when you provision the storage type and IOPS of your Amazon RDS for SQL Server DB instance. You can also

specify `MultiSubnetFailover=True`, if your client driver supports it, to significantly reduce the failover time.

Limitations

- The Multi-AZ option isn't available for SQL Server Express and Web editions. It's available only for SQL Server Standard and Enterprise editions.
- You can't configure the standby DB instance to accept database read activity.
- Cross-Region Multi-AZ isn't supported.
- 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 before termination, and recreate a new Amazon RDS instance from the snapshot when you need it. Or, you can remove the Multi-AZ configuration first and then stop the instance. After seven days, your stopped instance will restart so that any pending maintenance can be applied.

For additional limitations, see Microsoft SQL Server Multi-AZ deployment notes and recommendations in the Amazon RDS documentation.

Read replicas

Read replicas provide scalability and load balancing. A SQL Server read replica is a physical copy of an Amazon RDS for SQL Server DB instance that is used for read-only purposes. Amazon RDS helps reduces the load on the primary DB instance by offloading read-only workloads to the read replica DB instance. Updates made to your primary DB instance are asynchronously copied to the read replica instance.

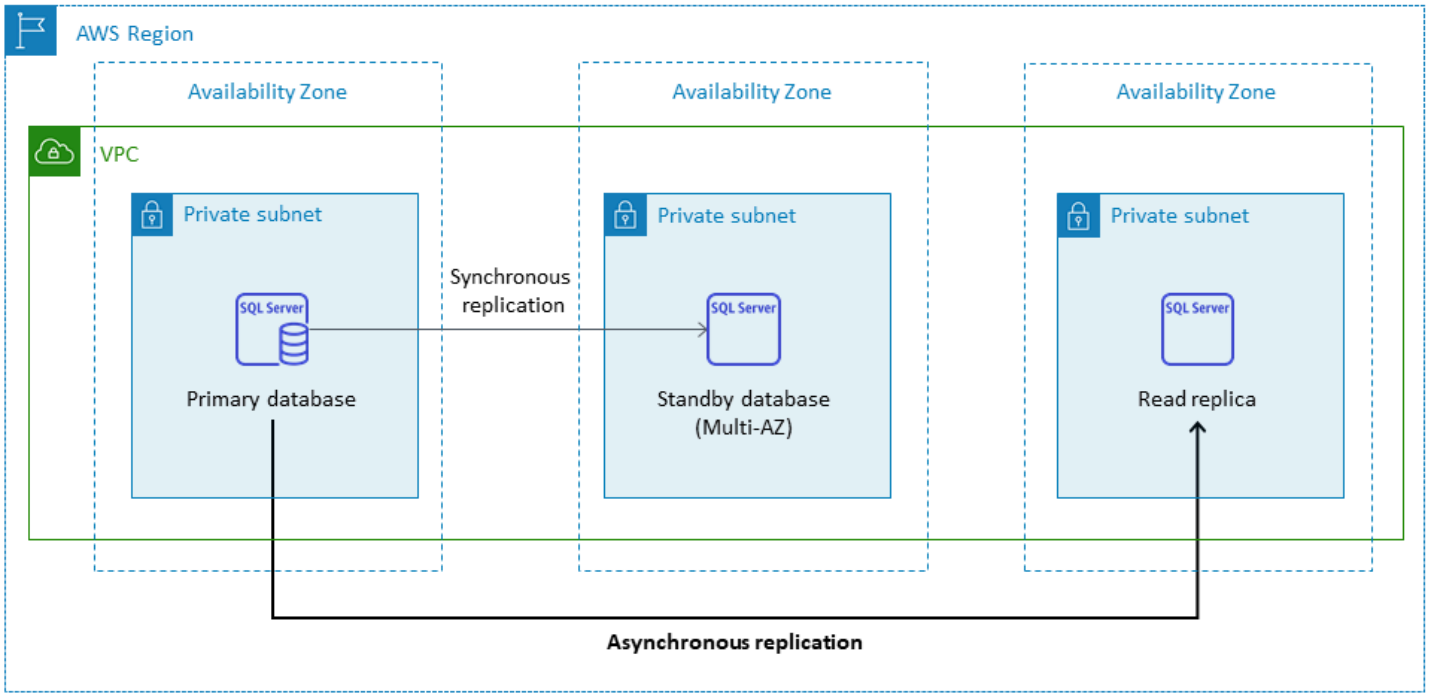
When you request a read replica, Amazon RDS takes a snapshot of the source DB instance, and this snapshot becomes the read replica. There is no outage while creating and deleting a read replica. Amazon RDS for SQL Server upgrades the primary database immediately after upgrading the read replicas, regardless of the maintenance window. Every read replica comes with a separate endpoint that you use to connect to the read replica database.

Amazon RDS for SQL Server makes it easy to create read replicas by configuring Always On availability groups, and maintaining secure network connections between a primary DB instance and its read replicas.

You can set up a read replica in the same AWS Region as your primary database. Amazon RDS for SQL Server doesn't support cross-Region read replicas. You can create up to five read replicas for one source DB instance.

> **Note**
>
> Read replicas are available only with the following SQL Server versions and editions:
>
> - SQL Server 2017 Enterprise edition 14.00.3049.1 or later
> - SQL Server 2016 Enterprise edition 13.00.5216.0 or later
>
> SQL Server versions and editions that support database mirroring for Multi-AZ environments do not offer read replicas.

The following diagram illustrates an Amazon RDS for SQL Server DB instance in a Multi-AZ environment with a read replica in another Availability Zone within the same AWS Region. Not all AWS Regions offer more than two Availability Zones, so you should check the Region you're planning to use before adopting this strategy.



A SQL Server read replica doesn't allow write operations. However, you can promote the read replica to make it writable. After you promote it, you cannot revert it back to a read replica. It

will become a single, standalone DB instance that has no relationships with its original primary database instance. The data in the promoted read replica will match the data in the source DB instance up to the point when the request was made to promote it. The SQL Server DB engine version of the source DB instance and all of its read replicas will be the same.

For efficient replication, we recommend the following:

- Set up each read replica with the same compute and storage resources as the source DB instance.
- You must enable automatic backups on the source DB instance by setting the backup retention period to a value other than 0 (zero).
- The source DB instance must be deployed in a Multi-AZ environment with Always On availability groups.

For SQL Server version support, editions, and limitations, see Read replica limitations with SQL Server in the Amazon RDS documentation.

For more information about using read replicas, see Working with read replicas and Working with SQL Server read replicas for Amazon RDS in the AWS documentation. For more information about data transfer pricing, see Amazon RDS pricing.

Disaster recovery

With Amazon RDS for SQL Server you can create a reliable, cross-Region disaster recovery (DR) strategy. The main reasons for creating a DR solution are business continuity and compliance:

- An effective DR strategy helps you keep your systems up and running with minimal or no interruptions during a catastrophic event. A reliable and effective cross-Region DR strategy keeps your business in operation even if an entire Region goes offline.
- A cross-Region DR solution helps you meet auditing and compliance requirements.

Recovery point objective (RPO), recovery time objective (RTO), and cost are three key metrics to consider when developing your DR strategy. For other options for providing cross-Region replicas, see AWS Marketplace. For more information about these approaches, see Cross-Region disaster recovery of Amazon RDS for SQL Server on the AWS Database blog.

Amazon RDS Custom for SQL Server

If you're unable to move to a fully managed service such as Amazon RDS because of customization requirements for third-party applications, you can migrate to Amazon RDS Custom for SQL Server. With Amazon RDS Custom, you can retain administrative rights to the database and its underlying operating system to enable dependent applications.

When to choose Amazon RDS Custom for SQL Server

Amazon RDS Custom for SQL Server is a good migration option when:

- You have legacy, custom, and packaged applications that require access to the underlying operating system and database environment.
- You need administrative user access to meet vendor-based application deployment requirements.
- You need access to the underlying operating system to configure settings, install patches, and enable native features to meet the dependent application's requirements.
- You want to access and customize the database environment (by applying custom database patches or modifying OS packages) to meet your database and application needs.

How it works

To use Amazon RDS Custom for SQL Server, review the requirements in the Amazon RDS Custom for SQL Server documentation. You must first set up your environment for Amazon RDS Custom for SQL Server, as explained in the Amazon RDS documentation. After the environment is set up, follow these steps, which are illustrated in the following diagram:

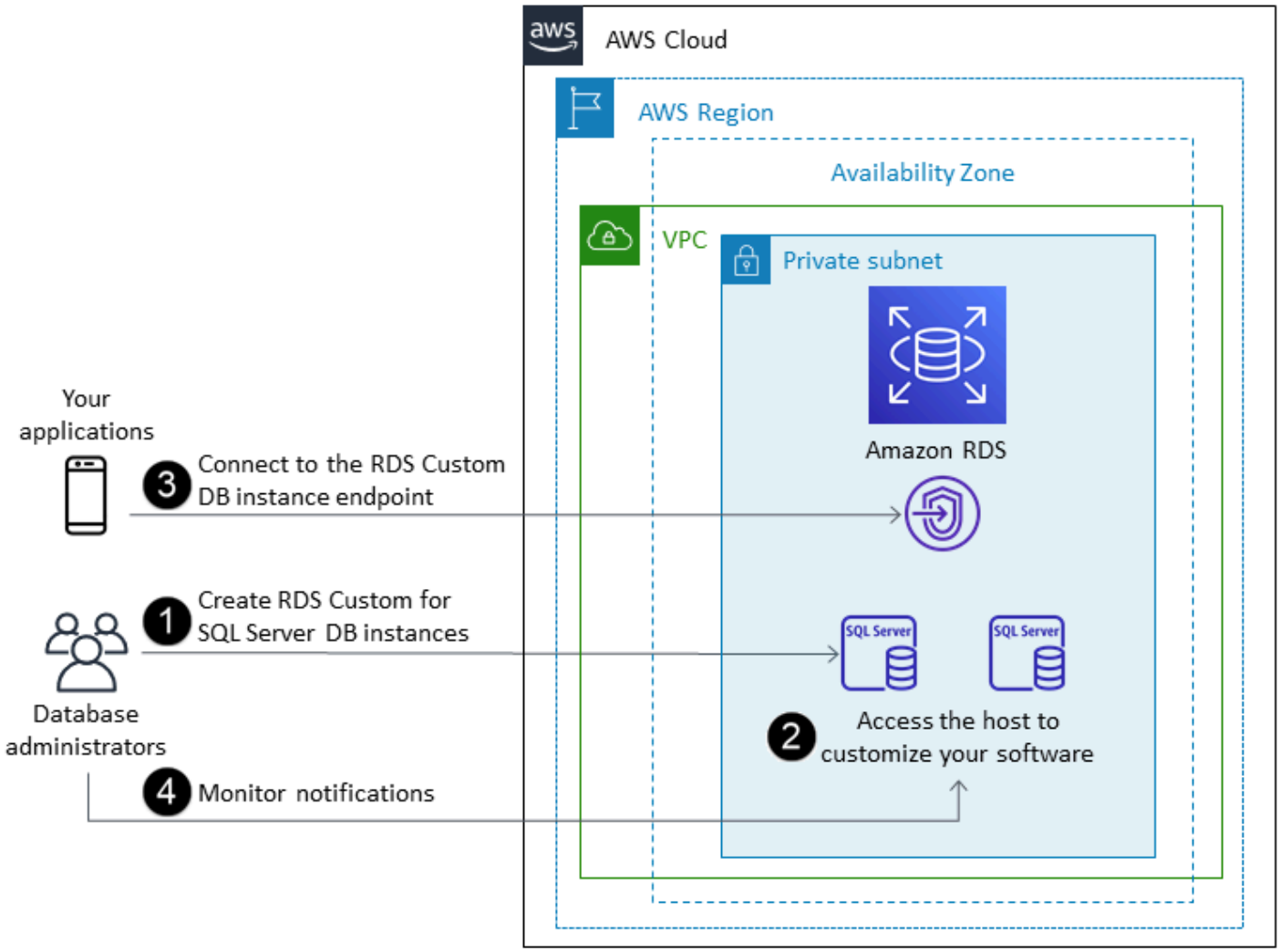
1. Create an Amazon RDS Custom for SQL Server DB instance from a SQL Server engine version offered by Amazon RDS Custom.

   Amazon RDS Custom for SQL Server currently supports SQL Server 2022 or 2019 Enterprise, Standard, or Developer Edition with the supported DB instance classes listed in the documentation. For more information, see Creating an RDS Custom for SQL Server DB instance.
2. Connect your application to the Amazon RDS Custom DB instance endpoint.

   For more information, see Connecting to your RDS Custom DB instance using AWS Systems Manager and Connecting to your RDS Custom DB instance using RDP.

3. (Optional) Access the host to customize your software.
4. Monitor notifications and messages generated by Amazon RDS Custom automation.

For more information about these steps, see the Amazon RDS Custom documentation.



In Amazon RDS Custom for SQL Server, you can install software to run custom applications and agents. Because you have privileged access to the host, you can modify file systems to support legacy applications. You can also apply custom database patches or modify OS packages on your Amazon RDS Custom DB instances.

Amazon RDS Custom automatically provides monitoring, backups, and instance recovery, and ensures that your DB instance uses a supported AWS infrastructure, operating system, and database. If you want to customize your instance, you can pause Amazon RDS Custom automation

for up to 24 hours and then resume it when your customization work is complete. Pausing the automation prevents Amazon RDS automation from directly interfering with your customizations.

When you resume automation, the support perimeter determines whether your customization of the database or operating system environment interferes with, or breaks, Amazon RDS Custom automation. Amazon RDS Custom supports your customization of the host and database environment as long as your changes don't put the DB instance outside the support perimeter. The support perimeter checks are performed every 30 minutes by default, and also occur after events such as snapshot deletions or uninstalling the Amazon RDS Custom agent, which monitors the DB instance. The Amazon RDS Custom agent is a critical component for ensuring Amazon RDS Custom functionality. If you uninstall the agent, Amazon RDS Custom runs the support perimeter check after one minute and moves the DB instance outside the support perimeter.

When you set up an Amazon RDS Custom for SQL Server DB instance, the software license is included. That is, you don't need to purchase SQL Server licenses separately. For more information about licensing, see section 10.5 in AWS service terms. If you have an active AWS Premium Support account, you can contact AWS Premium Support for Amazon RDS Custom for SQL Server-specific issues.

Amazon RDS Custom for SQL Server is supported in a limited selection of AWS Regions and with limited DB instance classes. For these and other limitations, see the requirements and limitations page in the Amazon RDS Custom for SQL Server documentation.

If you have an on-premises SQL Server database, you can follow the process described in the Amazon RDS documentation to migrate it to Amazon RDS Custom for SQL Server by using native backup and restore utilities.

For additional information, see the following resources:

- New – Amazon RDS Custom for SQL Server Is Generally Available (AWS News blog)
- Configure SQL Server replication between Amazon RDS Custom for SQL Server and Amazon RDS for SQL Server (AWS Database blog)
- Automate on-premises or Amazon EC2 SQL Server to Amazon RDS for SQL Server migration using custom log shipping (AWS Database blog)
- Configure high availability with Always On Availability Groups on Amazon RDS Custom for SQL Server (AWS Database blog)
- Get started with Amazon RDS Custom for SQL Server using an AWS CloudFormation template (Network setup) (AWS Database blog)

- Migrate on-premises SQL Server workloads to Amazon RDS Custom for SQL Server using distributed availability groups (AWS Database blog)
- Optimize your SQL Server costs by using bring your own media (BYOM) on Amazon RDS Custom for SQL Server (AWS Database blog)

Amazon EC2 for SQL Server

Amazon EC2 supports a self-managed SQL Server database. That is, it gives you full control over the setup of the infrastructure and the database environment. Running the database on Amazon EC2 is very similar to running the database on your own server. You have full control of the database and operating system-level access, so you can use your choice of tools to manage the operating system, database software, patches, data replication, backup, and restoration. This migration option requires you to set up, configure, manage, and tune all the components, including EC2 instances, storage volumes, scalability, networking, and security, based on AWS architecture best practices. You are responsible for data replication and recovery across your instances in the same or different AWS Regions.

For more information about migrating from SQL Server to Amazon EC2, see the rehost patterns on the AWS Prescriptive Guidance website.

When to choose Amazon EC2

Amazon EC2 is a good migration option for your SQL Server database when:

- You need full control over the database and access to its underlying operating system, database installation, and configuration.
- You want to administer your database, including backups and recovery, patching the operating system and the database, tuning the operating system and database parameters, managing security, and configuring high availability or replication.
- You want to use features and options that aren't currently supported by Amazon RDS. For details, see Features not supported and features with limited support in the Amazon RDS documentation.
- You need a specific SQL Server version that isn't supported by Amazon RDS. For a list of supported versions and editions, see SQL Server versions on Amazon RDS in the Amazon RDS documentation.
- Your database size and performance needs exceed the current Amazon RDS for SQL Server offerings. For details, see Amazon RDS DB instance storage in the Amazon RDS documentation.

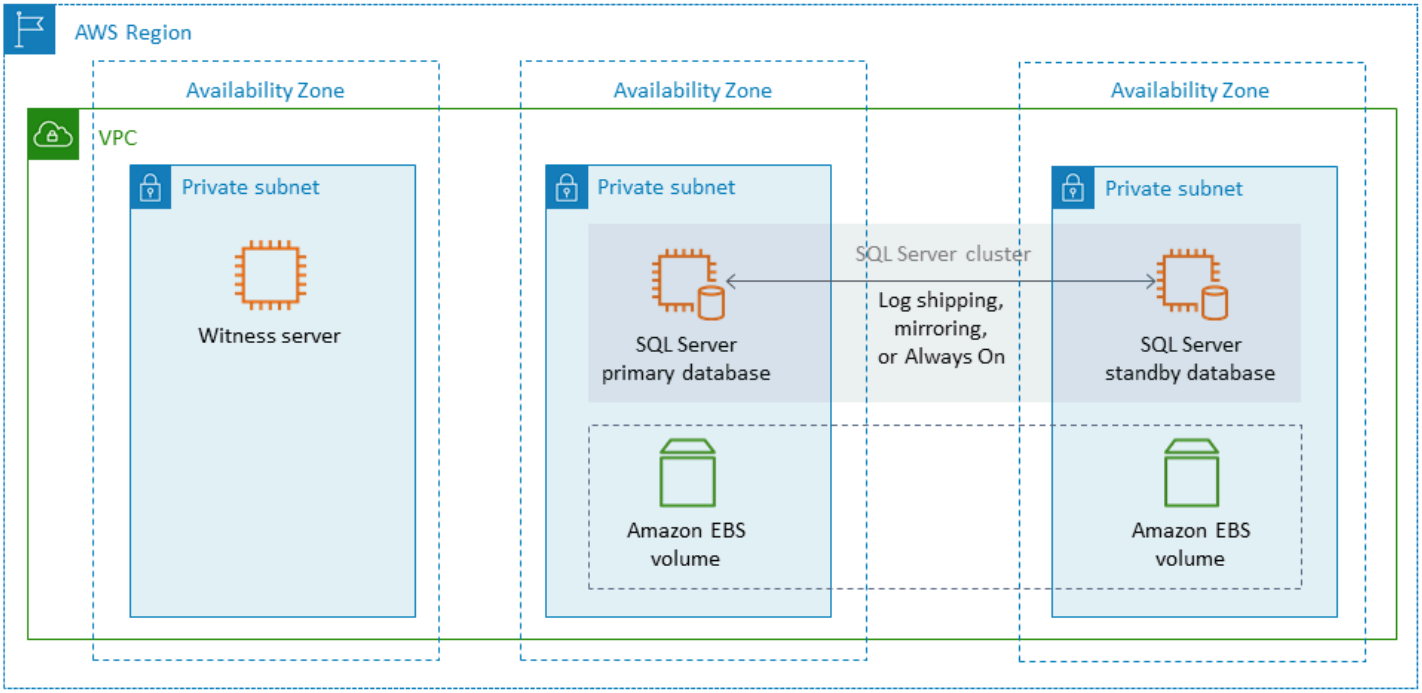
- You want to avoid automatic software patches that might not be compliant with your applications.
- You want to bring your own license instead of using the Amazon RDS for SQL Server license-included model.
- You want to achieve higher IOPS and storage capacity than the current limits. For details, see Amazon RDS DB instance storage in the Amazon RDS documentation.

For a list of currently supported SQL Server features and versions on Amazon EC2, see Choosing between Amazon EC2 and Amazon RDS later in this guide.

High availability

You can use any SQL Server-supported replication technology with your SQL Server database on Amazon EC2 to achieve high availability, data protection, and disaster recovery. Some of the common solutions are log shipping, database mirroring, Always On availability groups, and Always On Failover Cluster Instances.

The following diagram shows how you can use SQL Server on Amazon EC2 across multiple Availability Zones within a single AWS Region. The primary database is a read-write database, and the secondary database is configured with log shipping, database mirroring, or Always On availability groups for high availability. All the transaction data from the primary database is transferred and can be applied to the secondary database asynchronously for log shipping, and asynchronously for Always On availability groups and mirroring.



Log shipping

Log shipping lets you automatically send transaction log backups from a primary database instance to one or more secondary databases (also known as *warm standby*) on separate DB instances. Log shipping uses SQL Server Agent jobs to automate the process of backing up, copying, and applying the transaction log backups. Although log shipping is typically considered a disaster recovery feature, it can also provide high availability by allowing secondary DB instances to be promoted if the primary DB instance fails. If your RTO and RPO are flexible, or your databases aren't considered highly mission-critical, consider using log shipping to provide better availability for your SQL Server databases.

Log shipping increases the availability of databases by providing access to secondary databases to use as read-only copies of the primary database when needed. You can configure a lag delay (a longer delay time) during which you can 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 deploying a 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. A log shipping configuration doesn't automatically fail over from the primary server to the secondary server. However, any of the secondary databases can be brought online manually if the primary database becomes unavailable.

Log shipping is often used as a disaster recovery solution but also can be used as a high availability solution, depending on your application requirements. Use log shipping when:

- You have flexible RTO and RPO requirements. Log shipping provides an RPO of minutes, and an RTO of minutes to hours.
- You do not need an automatic failover to the secondary database.
- You want to read from the secondary database, but you don't require readability during a restore operation.

For more information about log shipping, see the Microsoft SQL Server documentation.

Database mirroring

Database mirroring takes a database that's on an EC2 instance and provides a complete or almost complete read-only copy (mirror) of it on a separate DB instance. Amazon RDS uses database mirroring to provide Multi-AZ support for Amazon RDS for SQL Server. This feature increases the availability and protection of databases, and provides a mechanism to keep databases available during upgrades.

> **Note**
>
> According to the Microsoft documentation, database mirroring will be removed in a future version of SQL Server. You should plan to use Always On availability groups instead.

In database mirroring, SQL servers can take one of three roles:

- The principal server, which hosts the primary read/write version of the database.
- The mirror server, which hosts a copy of the principal database.
- An optional witness server. This server is available only in high-safety mode. It 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. During mirroring, all database changes that are performed in the principal database are also performed on the mirror database. Database mirroring can be either a synchronous or an asynchronous operation. This is determined by two mirroring operating modes: high-safety mode and high-performance mode.

- **High-safety mode:** This mode uses synchronous operations. In this mode, the database mirroring session synchronizes the insert, update, and delete operations from the principal database to the mirror database as quickly as possible. As soon as the database is synchronized, the transaction is committed in both the principal and the mirror databases. We recommend that you use this operating mode when the mirror databases are in the same or different Availability Zones, but hosted within the same AWS Region.
- **High-performance mode:** This mode uses asynchronous operations. In this mode, the database mirroring session synchronizes the insert, update, and delete operations from the principal database to the mirror database, but there can be a lag between the time the principal database commits transactions and the time the mirror database commits transactions. We recommend that you use this mode when the mirror databases are in different AWS Regions.

Use database mirroring when:

- You have strict RTO and RPO requirements, and cannot have delays between the primary and secondary databases. Database mirroring provides an RPO of zero seconds (with synchronous commit) and an RTO of seconds to minutes.
- You do not have a requirement to read from the secondary database.
- You want to perform automatic failover when you have a witness server configured in synchronization mode.
- You cannot use Always On availability groups, which is the preferred option.

Limitations:

- Only one-to-one failover is supported. You cannot have multiple database destinations sync with the primary database.

For more information about mirroring, see the Microsoft SQL Server documentation.

Always On availability groups

SQL Server Always On availability groups provide high availability and disaster recovery solutions for SQL Server databases. An availability group consists of a set of user databases that fail over together. It includes a single set of primary read/write databases and multiple (one to eight) sets of related, secondary databases. You can make the secondary databases available to the application tier as read-only copies of the primary databases (SQL Server Enterprise edition only), to provide

a scale-out architecture for read workloads. You can also use the secondary databases for backup operations.

SQL Server Always On availability groups support both synchronous and asynchronous commit modes. In synchronous mode, the primary replica commits database transactions after the changes are committed or written to the log of the secondary replica. Using this mode, you can perform planned manual failover and automatic failover if the replicas are in sync. You can use synchronous commit mode between SQL Server instances within the same environment (for example, if all instances are on-premises or all instances are in AWS).

In asynchronous commit mode, the primary replica commits database transactions without waiting for the secondary replica. You can use asynchronous commit mode between SQL Server instances that are in different environments (for example, if you have instances on premises and in AWS).

You can use Always On availability groups for high availability or disaster recovery. Use this method when:

- You have strict RTO and RPO requirements. Always On availability groups provide an RPO of seconds, and an RTO of seconds to minutes.
- You want to manage and fail over a group of databases. Always On availability groups support 0-4 secondary replicas in synchronous commit mode for SQL Server 2019.
- You want to use automatic failover in synchronous commit mode, and you don’t need a witness server.
- You want to read from the secondary database.
- You want to synchronize multiple database destinations with your primary database.

Starting with SQL Server 2016 SP1, SQL Server Standard edition provides basic high availability for a single, non-readable secondary database and listener per availability group. It also supports a maximum of two nodes per availability group.

Always On Failover Cluster Instances

SQL Server Always On Failover Cluster Instances (FCIs) use Windows Server Failover Clustering (WSFC) to provide high availability at the server instance level. An FCI is a single instance of SQL Server that is installed across WSFC nodes to provide high availability for the entire installation of SQL Server. If the underlying node experiences hardware, operating system, application, or service failures, everything inside the SQL Server instance is moved to another WSFC node. This includes system databases, SQL Server logins, SQL Server Agent jobs, and certificates.

An FCI is generally preferable over an Always On availability group when:

- You're using SQL Server Standard edition instead of Enterprise edition.
- You have a large number of small databases per instance.
- You're constantly modifying instance-level objects such as SQL Server Agent jobs, logins, and so on.
- You want to optimize costs (both Amazon EC2 infrastructure costs and SQL Server license costs) for running the SQL Server instances. When you use an FCI, you remove the dependency of the SQL Server host EC2 instance on Amazon Elastic Block Store (Amazon EBS) throughput. As a result, you can right-size (that is, without the need to over-provision) the SQL Server EC2 instance and reduce your Amazon EC2 and SQL Server license costs, without affecting I/O performance.
- You want to optimize database performance on the SQL Server instances. Using an FCI provides about 30 percent better performance for typical workloads, because data replication is performed at the block level instead of the database level.

FCIs require some form of shared storage—Amazon EBS Provisioned IOPS SSD (`io2`) volumes, disks on a storage area network (SAN), file shares on Server Message Blocks (SMBs), or locally attached storage with Storage Spaces Direct (S2D), SIOS Datakeeper, or Amazon FSx—to provide resiliency and high availability.

There are four options for deploying FCIs on AWS:

- Amazon EBS Multi-Attach with persistent reservations
- Amazon FSx for Windows File Server
- Amazon FSx for NetApp ONTAP
- Solutions from AWS Partners

Using Amazon EBS Multi-Attach with persistent reservations

Amazon EBS Multi-Attach with NVMe reservations supports the creation of SQL Server FCIs with Amazon EBS `io2` volumes as the shared storage on Windows Server failover clusters. This feature simplifies the failover cluster setup process by enabling you to build a failover cluster by using Amazon EBS `io2` volumes. These volumes can be attached only to instances that are in the same Availability Zone. To deploy Windows Server failover clusters by using Amazon EBS `io2` volumes, you must use the latest AWS NVMe drivers.

Amazon EBS volumes and instance store volumes are exposed as NVMe block devices on Nitro-based instances. You must have the AWS NVMe driver installed with the SCSI persistent reservation feature configured when you use Amazon EBS `io2` volumes to form WSFC and SQL Server FCIs.

For more information about this feature, see the AWS blog post How to deploy a SQL Server failover cluster with Amazon EBS Multi-Attach on Windows Server.

Using Amazon FSx for Windows File Server

Amazon FSx for Windows File Server provides fully managed shared file storage. It automatically replicates the storage synchronously across two Availability Zones to provide high availability. Using FSx for Windows File Server for file storage helps simplify and optimize SQL Server high availability deployments on Amazon EC2.

With Microsoft SQL Server, high availability is typically deployed across multiple database nodes in an WSFC, and each node has access to shared file storage. You can use FSx for Windows File Server as shared storage for SQL Server high availability deployments in two ways: as storage for active data files and as an SMB file share witness.

For information about how you can reduce the complexity and cost of running SQL Server FCI deployments by using FSx for Windows File Server, see the blog post Simplify your Microsoft SQL Server high availability deployments using Amazon FSx for Windows File Server. The blog post also provides step-by-step instructions for deploying SQL Server FCIs by using an Amazon FSx Multi-AZ file system as the shared storage solution. For more information, see the Amazon FSx for Windows File Server documentation.

Using Amazon FSx for NetApp ONTAP

Amazon FSx for NetApp ONTAP is a fully managed service that provides highly reliable, scalable, high-performing, and feature-rich file storage that's built on the NetApp ONTAP file system. FSx for ONTAP combines the familiar features, performance, capabilities, and API operations of NetApp file systems with the agility, scalability, and simplicity of a fully managed AWS service.

FSx for ONTAP provides multi-protocol access to data over the NFS, SMB, and iSCSI protocols for Windows and Linux systems. You can build a highly available SQL Server Always On FCI architecture, as explained in detail in the blog post SQL Server High Availability Deployments Using Amazon FSx for NetApp ONTAP. FSx for ONTAP can also provide a quick way to fail over your SQL Server environment to a different AWS Region in order to meet recovery time objective (RTO) and recovery point objective (RPO) requirements. For more information, see the blog post Implementing HA and DR for SQL Server Always-On Failover Cluster Instance using FSx for ONTAP.

You can also use AWS Launch Wizard to deploy SQL Server solutions on AWS, with support for Always On Availability Groups and single-node deployments. Launch Wizard supports the deployment for SQL Server Always on FCIs on Amazon EC2 with FSx for ONTAP as the shared storage. This service saves you time and effort by replacing a complex manual deployment process with a guided, console-based wizard that accelerates the migration of your on-premises SQL Server workloads that rely on shared storage. For more information about how Launch Wizard can help you provision and configure SQL Server FCIs in hours, see the blog post Simplify SQL Server Always On deployments with AWS Launch Wizard and Amazon FSx. Launch Wizard also supports deployment for SQL Server Always On FCIs by using Amazon FSx for Windows File Server as the shared storage solution.

Using solutions from AWS Partners

- SIOS DataKeeper provides high availability cluster failover support across AWS Regions and Availability Zones. SIOS DataKeeper is available in AWS Marketplace.
- DxEnterprise from DH2i enables fully automatic failover of SQL Server Availability Groups in Kubernetes and unified instance failover for Windows and Linux. D2HI is available in AWS Marketplace.

FSx for Windows File Server

FSx for Windows File Server provides fully managed, highly reliable, and scalable file storage that is accessible by using the Server Message Block (SMB) protocol. It is built on Windows Server and delivers a wide range of administrative features such as user quotas, end-user file restore, and Microsoft Active Directory (AD) integration. It offers Single-AZ and Multi-AZ deployment options, fully managed backups, and encryption of data at rest and in transit. You can optimize cost and performance for your workloads with solid-state drives (SSD) and hard disk drives (HDD) storage options, and you can scale storage and change the throughput performance of your file system at any time. Amazon FSx file storage is accessible from Windows, Linux compute instances running on AWS, and on premises.

Amazon FSx makes it easier to deploy shared Windows storage for high availability SQL Server deployments through its support for continuously available (CA) file shares and smaller file systems. This option is suitable for these use cases:

- As shared storage used by SQL Server nodes in a WSFC instance.
- As an SMB file share witness that can be used with any SQL Server cluster with WSFC.

Amazon FSx provides fast performance with baseline throughput up to 2 GB/second per file system, hundreds of thousands of IOPS, and consistent sub-millisecond latencies.

To provide the right performance for your SQL instances, you can choose a throughput level that is independent of your file system size. Higher levels of throughput capacity also come with higher levels of IOPS that the file server can serve to the SQL Server instances accessing it.

The storage capacity determines not only how much data you can store, but also how many IOPS you can perform on the storage. Each gigabyte of storage provides 3 IOPS. You can provision each file system to be up to 64 TB in size.

For information about configuring and using Amazon FSx to reduce the complexity and costs of your SQL Server high availability deployments, see Simplify your Microsoft SQL Server high availability deployments using FSx for Windows File Server on the AWS Storage blog. To learn more about creating a new CA share, see the FSx for Windows File Server documentation.

Disaster recovery

Many organizations implement high availability for their SQL Server databases, but that isn't sufficient for organizations that require true IT resilience. We recommend that you implement a disaster recovery solution to avoid data loss and downtime of mission-critical databases. Adopting a multi-Region disaster recovery architecture for your SQL Server deployments help you:

- Achieve business continuity
- Improve latency for your geographically distributed customer base
- Satisfy your auditing and regulatory requirements

Options for disaster recovery include log shipping, Always On availability groups, Amazon EBS snapshots that are stored in Amazon S3 and replicated across AWS Regions, Always On Failover Cluster Instances (FCIs) combined with Always On availability groups, and distributed availability groups.

Distributed availability groups

An architecture with distributed availability groups is an optimal approach for multi-Region SQL Server deployment. A distributed availability group is a special type of availability group that spans two separate availability groups. You can think of it as an availability group of availability groups. The underlying availability groups are configured on two different WSFC clusters.

Distributed availability groups are loosely coupled, which means that they don't require a single WSFC cluster and they're maintained by SQL Server. Because the WSFC clusters are maintained individually and transmissions are primarily asynchronous between two availability groups, it's easier to configure disaster recovery at another site. The primary replicas in each availability group synchronize their own secondary replicas.

Distributed availability groups support only manual failover at this time. To ensure that no data is lost, stop all transactions on the global primary databases (that is, on the databases of the primary availability group). Then set the distributed availability group to synchronous commit.

VMware Cloud on AWS for SQL Server

VMware Cloud on AWS is an integrated cloud offering jointly developed by AWS and VMware. SQL Server easily integrates with VMware Cloud on AWS. This migration option makes it possible for you to build on your existing investment in virtualization.

You can access VMware Cloud on AWS on an hourly, on-demand basis or in subscription form. It includes the same core VMware technologies that you run in your data centers, including vSphere Hypervisor (ESXi), Virtual SAN (vSAN), and the NSX network virtualization platform, and is designed to provide an efficient, seamless experience for managing your SQL Server databases. You can scale the storage, compute and memory of your SQL Server databases on VMware Cloud on AWS within minutes.

VMware Cloud on AWS runs directly on the physical hardware, but takes advantage of network and hardware features that were designed to support the AWS security-first infrastructure model. This means that the VMware virtualization stack runs on AWS infrastructure without having to use nested virtualization.

VMware Cloud on AWS makes it is easy to set up, scale, and operate your SQL Server databases workloads on AWS. It provides high availability solutions, integrates with on-premises Active Directory, and provides access to AWS services like AWS Directory Service for Microsoft Active Directory and AD Connector, Amazon Route 53, Amazon CloudWatch, and Amazon S3. You can store your backups in Amazon S3, and modernize and simplify your disaster recovery process.

When to choose VMware Cloud on AWS

VMware Cloud on AWS is an option for your SQL Server database when:

- Your SQL Server databases are already running in an on-premises data center in a vSphere virtualized environment.
- You have a large number of databases and you need fast migration (for example, only a few hours) to the cloud for one of the following reasons, without requiring any additional work from the migration team:
  - Data center extension. You need on-demand capacity to run virtualized desktops, to publish applications, or to provide a development/testing environment.
  - Disaster recovery. You want to set up a new disaster recovery system or replace your existing system.
  - Cloud migration. You want to migrate your entire data center to the cloud, or refresh your infrastructure.

If your SQL Server database requires more than 80K IOPS, you can use vSAN.

For more information, see In the Works – VMware Cloud on AWS on the AWS News blog, and Deploy Microsoft SQL Server on VMware Cloud on AWS on the AWS website.

Heterogeneous database migration for SQL Server

Because of the innovations and improvements in open-source databases and cloud computing platforms like AWS, many organizations are moving from proprietary (online transaction processing or OLTP) database engines such as SQL Server to open-source engines. SQL Server databases are mission-critical systems for any organization, but being locked into a particular vendor is a risky and costly situation. Low operating cost and no licensing fees are compelling reasons to consider switching the underlying database technology to open-source or AWS Cloud-native databases.

Other reasons for migrating off SQL Server are vendor lock-in periods, licensing audits, expensive licensing, and cost. For this reason, many organizations choose to migrate their SQL Server databases to either open-source databases (such as PostgreSQL, MySQL, or MariaDB) or AWS Cloud-native databases (such as Amazon Aurora or Amazon DynamoDB) when they migrate to AWS.

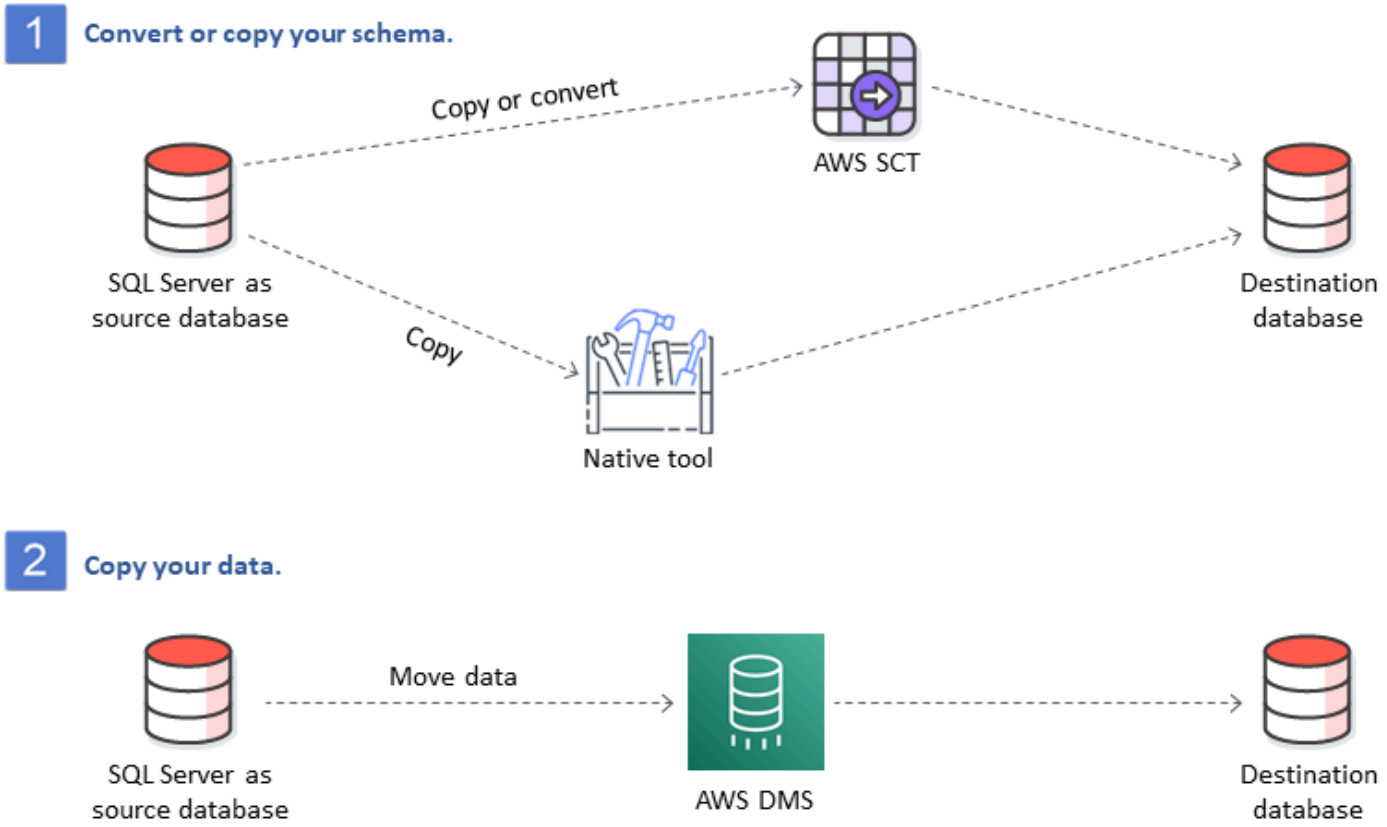
You can also migrate your SQL Server data warehouse database to Amazon Redshift, which is a fast, fully managed cloud data warehouse. Amazon Redshift is integrated with your data lake, offers up to three times faster performance than any other data warehouse, and costs up to 75 percent less than any other cloud data warehouse. For more information, see the pattern Migrate an on-premises Microsoft SQL Server database to Amazon Redshift using AWS DMS on the AWS Prescriptive Guidance website.

To migrate to an open-source or AWS Cloud-native database, choose the right database depending on the type of data you have, the access model, scalability, application practicalities, and complexity. Migrating from SQL Server to PostgreSQL and to other open-source databases has often been difficult and time-consuming, and requires careful assessment, planning, and testing.

This process becomes easier with services like AWS Database Migration Service (AWS DMS) and AWS Schema Conversion Tool (AWS SCT), which help you migrate your commercial database to an open-source database on AWS with minimal downtime.

In heterogeneous database migrations, the source and target databases engines are different, as in SQL Server to Aurora, or SQL Server to MariaDB migrations. The schema structure, data types, and database code in the source and target databases can be quite different, so the schema and code must be transformed before the data migration starts. For this reason, heterogeneous migration is a two-step process:

- Step 1. Convert the source schema and code to match that of the target database. You can use AWS SCT for this conversion.
- Step 2. Migrate data from the source database to the target database. You can use AWS DMS for this process.



AWS DMS handles the major data type conversions automatically during migration. The source database can be located in your own premises outside AWS, it can be a database that's running on an EC2 instance, or it can be an Amazon RDS database (see Sources for Data Migration in the AWS DMS documentation). The target can be a database in Amazon EC2, Amazon RDS, or Aurora. For information about using MySQL as a target database, see Migrating a SQL Server Database to a MySQL-Compatible Database Engine on the AWS Database blog.

For more information about refactoring your SQL Server database on AWS, see the re-architect patterns on the AWS Prescriptive Guidance website.

Tools for heterogeneous database migrations

The following chart provides a list of tools that you can use to migrate from SQL Server to another database engine.

Migration tool	Target database support	Used for
AWS SCT	Amazon RDS for MySQL Amazon RDS for PostgreSQL Amazon Aurora MySQL Amazon Aurora PostgreSQL	Schema conversion
AWS DMS	Amazon RDS for MySQL Amazon RDS for PostgreSQL Amazon Aurora MySQL Amazon Aurora PostgreSQL	Data migration
Babelfish	Amazon Aurora PostgreSQL	Data access and migration

The following subsections provide more information about each tool.

AWS SCT

AWS Schema Conversion Tool (AWS SCT) converts your existing commercial database schemas to an open-source engine or to an AWS Cloud-native database. AWS SCT makes heterogeneous database migrations predictable by automatically converting the source database schema and a majority of the database code objects, including views, stored procedures, and functions, to a format that's compatible with the target database.

When you convert your database schema from one engine to another, you also need to update the SQL code in your applications to interact with the new database engine instead of the old one. AWS SCT also converts the SQL code in C++, C#, Java or other application code. Any objects that can't be automatically converted are clearly marked for manual conversion. AWS SCT can also

scan your application source code for embedded SQL statements and convert them as part of a database schema conversion project. For more information, see Using Microsoft SQL Server as a source for AWS SCT in the AWS documentation.

AWS DMS

AWS Database Migration Service (AWS DMS) migrates your data rapidly and securely to AWS. During migration, the source database remains fully operational, minimizing application downtime. AWS DMS supports homogeneous migrations such as migrating data from one SQL Server database to another. It also supports heterogeneous migrations between different database platforms, such as migrating your SQL Server database to an open-source database or to an AWS cloud-native database. AWS DMS manages the complexities of the migration process, including automatically replicating data changes that occur in the source database to the target database. After the database migration is complete, the target database remains synchronized with the source database for as long as you choose, and you can switch over to the target database at a convenient time. For more information, see Using a Microsoft SQL Server database as a source for AWS DMS in the AWS documentation.

Babelfish

Babelfish is a built-in capability of Amazon Aurora. Babelfish for Aurora PostgreSQL enables your Aurora PostgreSQL-Compatible Edition databases to understand commands from applications that were written for Microsoft SQL Server. Modifying SQL Server applications that have SQL Server database code written in Transact-SQL (T-SQL), SQL Server's proprietary SQL dialect, takes effort and is time-consuming. Babelfish for Aurora PostgreSQL makes this process simpler and easier. Using Babelfish, you do not have to make changes to your application code. Instead, you can use Babelfish for Aurora PostgreSQL to migrate an SQL Server database to an Aurora PostgreSQL-Compatible DB cluster.

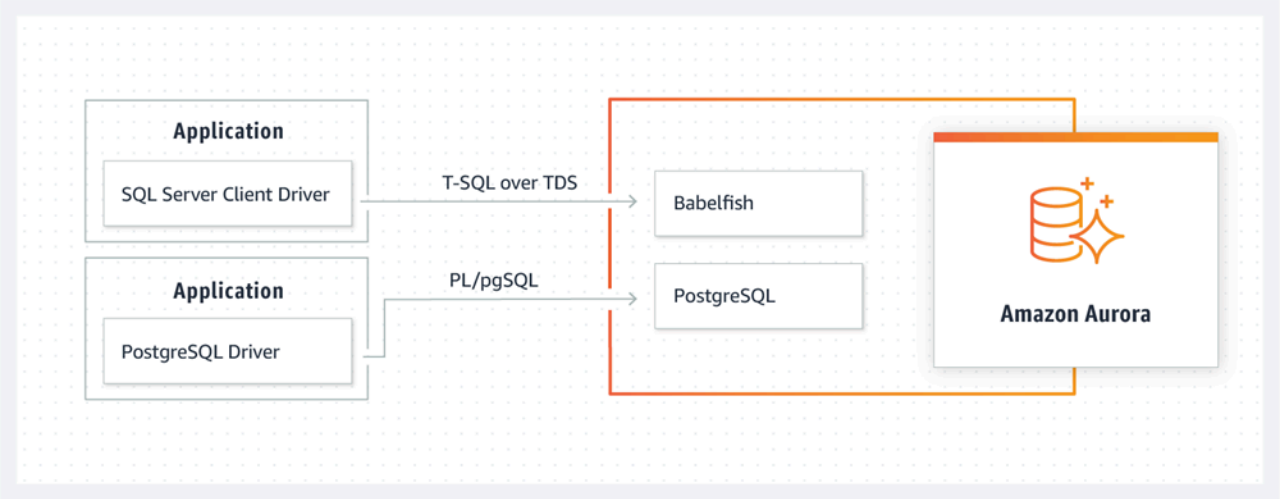
With Babelfish, Aurora PostgreSQL understands T-SQL and supports the same communications protocol, so you don't have to switch database drivers or rewrite your application queries. Your applications that were originally written for SQL Server can now work with Aurora with fewer code changes. This reduces the effort required to modify and move applications running on SQL Server or newer to Aurora, leading to faster, lower-risk, and more cost-effective migrations.

If you're migrating from legacy SQL Server databases, you can use Babelfish to run SQL Server code side by side with new functionality you built by using native PostgreSQL APIs. Babelfish enables Aurora PostgreSQL to work with commonly used SQL Server tools, commands, and drivers.

Babelfish also provides access to data by using the native PostgreSQL connection. By default, both SQL dialects supported by Babelfish are available through their native wire protocols at the following ports:

- For SQL Server dialect (T-SQL), connect to port 1433.
- For PostgreSQL dialect (PL/pgSQL), connect to port 5432.

Babelfish enables your legacy SQL Server applications to communicate with Aurora without extensive code rewrites, by providing connections from the SQL Server or PostgreSQL port. The following diagram illustrates this architecture.



You can enable Babelfish on your Aurora cluster from the Amazon RDS management console. For instructions, see Creating a Babelfish for Aurora PostgreSQL DB cluster in the Amazon RDS documentation.

For more information about migrating, see Migrating a SQL Server database to Babelfish for Aurora PostgreSQL in the Aurora documentation.

For additional information, see the following resources:

- Get started with Babelfish for Aurora PostgreSQL (AWS Database blog)
- Migrate from SQL Server to Amazon Aurora using Babelfish (AWS Database blog)
- Migrate from SQL Server to Aurora PostgreSQL using SSIS and Babelfish (AWS Database blog)
- Modify SSIS packages from SQL Server to Babelfish for Aurora PostgreSQL (AWS Database blog)
- Run SQL Server Reporting Services reports against Babelfish for Aurora PostgreSQL (AWS Database blog)

- Prepare for Babelfish migration with the AWS SCT assessment report (AWS Database blog)

Hybrid migration scenarios for SQL Server

You can also run SQL Server workloads in a hybrid environment that includes AWS. For example, you might already be running SQL Server in your on-premises or co-located data center but want to use the AWS Cloud to enhance your architecture to provide a high availability or disaster recovery solution. You can also use hybrid solutions to store long-term SQL Server backups on AWS, to roll back your migration, in case of issues, or to run a secondary replica using SQL Server Always On availability groups in the AWS Cloud. SQL Server has several replication technologies that offer high availability and disaster recovery solutions.

Backing up your SQL Server databases to the AWS Cloud

Amazon Simple Storage Service (Amazon S3) enables you to take advantage of the flexibility and pricing of cloud storage. It gives you the ability to back up your SQL Server databases to a secure, highly available, highly durable, reliable storage system. You can securely store your SQL Server backups in Amazon S3. You can also use Amazon S3 Lifecycle policies to store your backups for the long term. Amazon S3 allows you to store large amounts of data at a very low cost. You can use AWS DataSync to transfer backup files to Amazon S3.

You can use Storage Gateway to store your on-premises SQL Server backups and archive data on Amazon S3 or Amazon S3 Glacier. You can create cached storage volumes and mount them as Internet Small Computer System Interface (iSCSI) devices from your on-premises backup application servers. All data is securely transferred to AWS over SSL and stored in encrypted format in Amazon S3. Using gateway cached volumes saves the upfront cost of maintaining and scaling costly storage hardware on premises. If you want to keep your primary data or backups on premises, you can use gateway stored volumes to keep this data locally, and back up the data off-site to Amazon S3.

Extending high availability and disaster recovery solutions

You can extend your existing on-premises high availability practices and provide a disaster recovery solution in AWS by using the native log shipping feature in SQL Server. You can transfer your SQL Server transaction logs from your on-premises or co-located data centers to a SQL Server instance that is running on an EC2 instance or an Amazon RDS for SQL Server DB instance in a virtual private cloud (VPC). You can transmit this data securely over a dedicated network connection by using AWS Direct Connect, or transmit it over a secure VPN tunnel. The transaction log backups are sent to the EC2 instance, and they are applied to secondary database instances.

You can use the AWS Cloud to provide a higher level of high availability and disaster recovery by using SQL Server Always On availability groups between your on-premises data center and Amazon EC2. This can be done by extending your data center into a VPC on AWS by using a dedicated network connection like AWS Direct Connect, or by setting secure VPN tunnels between these two environments.

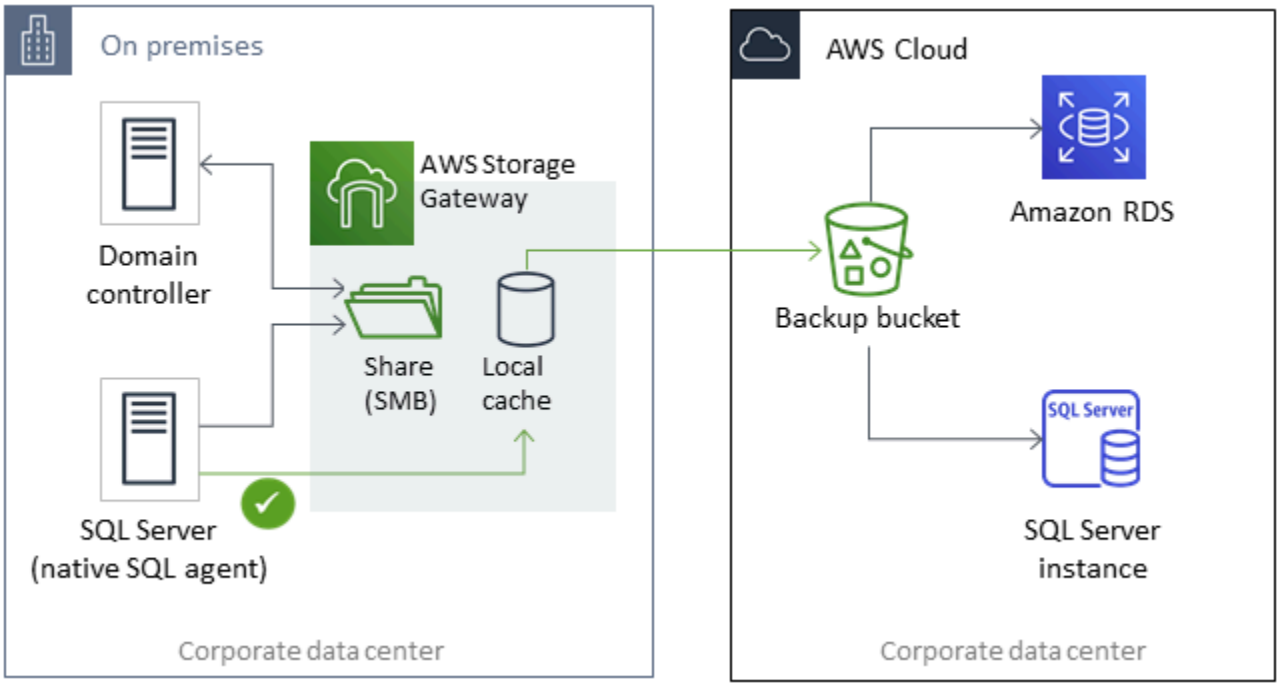
Here are a few things to consider when planning a hybrid implementation of SQL Server Always On availability groups:

- Establish secure, reliable, and consistent network connections between your on-premises environment and AWS through AWS Direct Connect or VPN.
- Create a VPC by using the Amazon Virtual Private Cloud (Amazon VPC) service. Use Amazon VPC route tables and security groups to enable the appropriate communications between the two environments.
- Extend Active Directory domains into the VPC by deploying domain controllers as EC2 instances, or by using AWS Directory Service for Microsoft Active Directory. You can also use AWS Managed Microsoft AD for Amazon RDS for SQL Server. For more information, see the Amazon RDS documentation.

Storage Gateway

Storage Gateway enables you to store and retrieve files by using a Server Message Block (SMB) share for Windows. You can join the storage gateway to your on-premises Active Directory domain. By having your SQL Server database and storage gateway in the same domain, you can take the backups directly to the SMB network share instead of storing them locally and then uploading them to the network share. The storage gateway is configured to use an S3 bucket, so all your backups will be available in the S3 bucket on AWS. You can restore your database by downloading the backup files to SQL Server on an EC2 instance, or restore the database directly to Amazon RDS.

The following diagram shows how to store and access backups by using Storage Gateway and Amazon S3. For more information, see the Storage Gateway documentation.



Using AWS DMS and AWS SCT

You can use AWS DMS in hybrid SQL Server environments, to migrate data from your on-premises database to the cloud, or the other way around. You can migrate your SQL Server database to MySQL or PostgreSQL by using AWS DMS with AWS SCT. For migration steps, see the AWS SCT documentation. Before you migrate your data, you can run a migration assessment report that flags any additional manual work that might be required.

You can also use AWS DMS for ongoing replication (change data capture or CDC). For more information, see Using ongoing replication (CDC) from a SQL Server source in the AWS DMS documentation.

Modernizing your SQL Server database

This section describes how to modernize your SQL Server workloads on AWS by switching from the Windows operating system to Linux. This change enables you to take advantage of open-source technologies and to save on Windows licensing costs without drastically altering your system architecture or retraining your users.

Migrating your SQL Server workloads from Windows to Linux

Starting with SQL Server 2017, SQL Server is available to run on Linux operating systems. Moving your SQL Server workloads to Linux provides both cost savings and performance improvements.

Almost all SQL Server functions, applications, statements, and scripts that you use on Microsoft Windows are supported on Linux as well. You can also use tools such as SQL Server Management Studio (SSMS), SQL Server Data Tools (SSDT), and PowerShell module (sqlps) to manage SQL Server on Linux from a Windows instance.

You can use one of these three options to migrate your SQL Server workloads to Linux:

- Native SQL Server backup and restore feature (see the Microsoft SQL Server documentation)
- Distributed availability groups (to change your operating system while you migrate to AWS)
- The AWS replatforming assistant, which is a PowerShell-based scripting tool

The AWS replatforming assistant helps you migrate from your existing SQL Server workloads from Windows to a Linux operating system. When you run the PowerShell script for the replatforming assistant on a source SQL Server database, the Windows instance backs up the database to an encrypted Amazon S3 storage bucket. It then restores the backup to new or existing SQL Server database on an EC2 Linux instance. You can replicate your database and test your applications while your source SQL Server database remains online. After testing, you can schedule application downtime and rerun the PowerShell backup script to perform your final cutover.

For more information about using the replatforming assistant, see Migrate your on-premises SQL Server Windows Workloads to Amazon EC2 Linux on the AWS Database blog, and the Amazon EC2 documentation.

High availability on Linux

SQL Server 2017 supports Always On availability groups between Windows and Linux to create read-scale workloads without high availability. Unfortunately, you cannot achieve high availability between Windows and Linux, because there is no clustered solution that can manage that cross-platform configuration.

To use high availability with Always On availability groups, consider using a Windows Server Failover Cluster (WSFC) or Pacemaker on Linux. This solution is suitable for a migration path from SQL Server on Windows to Linux and vice versa, or for disaster recovery using manual failover. For more information about this scenario, see Deploying Always On availability groups between Amazon EC2 Windows and Amazon Linux 2 instances on the AWS Database blog.

AWS Launch Wizard for SQL Server

AWS Launch Wizard is a service that guides you through the sizing, configuration, and deployment of Microsoft SQL Server on Amazon EC2. It supports both SQL Server single instance and high availability (HA) deployments on Amazon EC2.

Launch Wizard is a free service. You pay only for the AWS resources provisioned to run your application, such as Amazon EC2, Amazon EBS, and Amazon VPC resources.

You input your application requirements, including performance, number of nodes, and connectivity, on the Launch Wizard console. Launch Wizard identifies the right AWS resources to deploy and run your SQL Server application. It also provides an estimated cost of deployment, and you can modify your resources and instantly view the updated cost assessment. When you confirm your selections and initiate deployment, Launch Wizard provisions and configures the selected resources in a few hours to create a fully functioning, production-ready SQL Server application. You can access your deployed SQL Server application from the Amazon EC2 console.

Here are some of benefits of using Launch Wizard for SQL Server:

- **Simple deployment** – You can simplify the provisioning of your SQL Server resources on AWS by answering questions based on your requirements. A Launch Wizard deployment is faster than a manual deployment, so it eliminates the time to provision and configure your application on AWS.
- **Automated sizing and cost estimation** – Launch Wizard provides built-in instance selection based on your requirements. It selects the instance type, EBS volumes, and other resources that best suit your SQL Server requirements. Launch Wizard also provides you with a cost estimate before it provisions the AWS resources.
- **Time savings with repeatable automation templates** – You can redeploy SQL Server with reusable AWS CloudFormation templates that are created by Launch Wizard. These templates serve as a baseline and save you time.

Launch Wizard supports the following operating systems, SQL Server versions, and features. For the latest information, see the AWS Launch Wizard documentation.

Category	Use case or feature	Launch Wizard support	Quick Start support	Amazon EC2 console support
Deployment on Windows	Single SQL node deployment	 Y	 N	 Yes
	HA deployment: Always On availability groups	 Y	 Y	 No
	HA deployment: Always On Failover Cluster Instances (FCIs) with FSx for Windows File Server	 Y	 Y	 No
	HA deployment: Dedicated Hosts	 Y	 Y	 No
	Reusable code templates	 Y	 Y	 No
Deployment on Linux	Single SQL node deployment	 Y	 N	 Yes
	HA deployment on Ubuntu	 Y	 N	 No

Category	Use case or feature	Launch Wizard support	Quick Start support	Amazon EC2 console support
	Reusable code templates	 Y	 N	 No
Sizing	Instance type recommendation	 Y	 N	 No
	Cost estimation	 Y	 N	 No
Configuration	Automatically created AWS Systems Manager resource group	 Y	 N	 No
	One-click Amazon SNS notification	 Y	 N	 No
	One-click Amazon CloudWatch monitoring	 Y	 N	 No
	Connecting to existing Active Directory (both on-premises and managed)	 Y	 N	 No

Category	Use case or feature	Launch Wizard support	Quick Start support	Amazon EC2 console support
	Early input validation	 Y	 N	 No
	Managed IAM policy	 Y	 N	 No

For more information about Launch Wizard for SQL Server, see the following:

- AWS Launch Wizard for SQL Server documentation
- Simplify SQL Server Always On deployments with AWS Launch Wizard and Amazon FSx blog post
- Accelerate SQL Server Always On Deployments with AWS Launch Wizard blog post

Best practices for migrating to Amazon RDS for SQL Server

Based on the assessment of your database and your project requirements, if your goal is to migrate to Amazon RDS for SQL Server, follow the best practices in this section to provision your target database, perform the migration, and test, operate, and optimize your Amazon RDS for SQL Server database.

> **Important**
>
> Make sure that you have a rollback plan before you migrate your database.

> **Note**
>
> You can use Migration Hub Orchestrator to automate and orchestrate your SQL Server database migrations to Amazon EC2 or Amazon RDS by using native backup and restore. For more information, see the AWS Migration Hub Orchestrator section.

Provisioning your target database

After you finish assessing, planning, and preparing your database migration strategy, follow these best practices when provisioning your Amazon RDS for SQL Server database:

- Right-size the Amazon RDS for SQL Server DB instance based on your requirements for CPU, memory, IOPS, and storage type. (If you're using SQL Server Standard edition, provision CPU and memory within the limitations of Standard edition.)
- Set the correct time zone and collation.
- Make sure to launch Amazon RDS in the correct virtual private cloud (VPC).
- Create the security groups with correct port and IP addresses.
- Provision your Amazon RDS database in a private subnet for security.
- If possible, provision the SQL Server instance with the latest version of SQL Server.
- Create a separate option group and parameter group for each Amazon RDS database.
- Collect and extract logins, users, and roles for migration.

- Review SQL Server Agent jobs for maintenance and applications that need to be migrated.

Backing up from your source database

There are many tools for migrating a SQL Server database to an Amazon RDS for SQL Server database. The most commonly used method is using SQL Server native backup and restore if your requirements allow downtime.

If you have limited downtime, you can use native SQL Server backup/restore with differential backup and log backup. Or you can use AWS DMS, which provides three options: full-load, full-load and CDC, or CDC only.

Transferring data dump files to AWS

- If you're using AWS Direct Connect, which provides high bandwidth connectivity between your on-premises environment and AWS, you can copy your SQL Server backups to Amazon S3 and set up Amazon S3 integration.
- If you don't have high bandwidth through AWS Direct Connect, use AWS Snowball to transfer large database backup files. You can also use AWS DMS to transfer the data when replication is required.

Restoring data to your target database

- If you're migrating a very large database, we recommend that you provision a bigger Amazon RDS instance type initially, for the duration of the migration, for faster data loads.
- Disable Multi-AZ. (This can be re-enabled after migration.)
- Disable backup retention. (This can be re-enabled after migration.)
- Restore the database by using the native SQL Server **restore** command.
- Create logins and users, and fix orphaned users, if required.
- Create SQL Server Agent jobs and review the schedule, as needed.

Post-migration steps

After the migration is complete, you can:

- Change the DB instance to the right-sized instance type.
- Enable Multi-AZ and backup retention.
- Make sure that all jobs are created on secondary nodes (for Multi-AZ configuration).
- Publish SQL Server error and agent logs to Amazon CloudWatch Logs, and use CloudWatch to view metrics and create alarms. For more information, see the Amazon RDS documentation.
- Enable enhanced monitoring to get metrics for your DB instance in real time.
- Set up Amazon Simple Notification Service (Amazon SNS) topics for alerts.

Testing the migration

We recommend the following tests to validate your application against your new Amazon RDS for SQL Server database:

- Perform functional testing.
- Compare the performance of SQL queries in your source and target databases, and tune the queries as needed. Some queries might perform more slowly in the target database, so we recommend that you capture the baselines of the SQL queries in the source database.

For additional validation during the proof-of-concept (POC) phase, we recommend the following supplemental tests:

- Run performance tests to ensure that they meet your business expectations.
- Test database failover, recovery, and restoration to make sure that you're meeting RPO and RTO requirements.
- List all critical jobs and reports, and run them on Amazon RDS to evaluate their performance against your service-level agreements (SLAs).

Operating and optimizing your Amazon RDS database

When your database is on AWS, make sure that you are following best practices in areas such as monitoring, alerting, backups, and high availability in the cloud. For example:

- Set up CloudWatch monitoring, and enable detailed monitoring.
- Use Amazon RDS Performance Insights and other third-party monitoring solutions like SentryOne or Foglight for SQL Server to monitor your database.

- Set up alerts by using SNS topics.
- Set up automatic backups by using AWS Backup or native SQL Server backups, and copy to Amazon S3.
- For high availability, set up the Amazon RDS Multi-AZ feature.
- If you need read-only databases, set up a read replica within the same or across AWS Regions according to your needs.

Choosing between Amazon EC2 and Amazon RDS

Amazon EC2 and Amazon RDS offer unique benefits that may be beneficial for your specific use case. You have the flexibility to use one or both services for your SQL Server database, depending on your needs. This section provides detailed information to help with your choice.

Decision matrix

The following table provides a side-by-side comparison of SQL Server features supported on Amazon RDS, Amazon RDS Custom for SQL Server, and Amazon EC2. Use this information to understand their differences and to choose the best approach for your use case.

For the most current information for Amazon RDS, see Microsoft SQL Server on Amazon RDS in the AWS documentation.

Development

Development feature	Amazon RDS	Amazon RDS Custom	Amazon EC2	Notes
Buffer pool extensions	N	Y	Y	If this feature is critical to your workload, consider choosing Amazon RDS Custom or Amazon EC2.
`BULK INSERT`	Y	Y	Y	See Integrating an Amazon RDS for SQL Server DB instance with Amazon S3 in the Amazon RDS documentation.

Development feature	Amazon RDS	Amazon RDS Custom	Amazon EC2	Notes
Change data capture (CDC)	 Y (Enterprise Edition: all versions; Standard Edition: 2016 SP1 and later)	 Y	 Y	See Using change data capture in the Amazon RDS documentation.
Change tracking	 Y	 Y	 Y	
Columnstore indexes	 Y (Enterprise Edition: 2014 and later)	 Y (Enterprise Edition: 2019)	 Y (Enterprise Edition: 2014 and later)	
Data Quality Services	 N	 Y	 Y	If this feature is critical to your workload, consider choosing Amazon RDS Custom or Amazon EC2.

Development feature	Amazon RDS	Amazon RDS Custom	Amazon EC2	Notes
Database Mail	 Y	 Y	 Y	See the blog post Using Database Mail on Amazon RDS for SQL Server. We encourage you to use the Amazon Simple Email Service (Amazon SES) to send outbound email originating from AWS resources, to ensure a high degree of deliverability.
Database Engine Tuning Advisor	 Y	 Y	 Y	
DB event notifications	 Y	 Y	 N (manually track and manage DB events)	See Using Amazon RDS event notification in the Amazon RDS documentation.

Development feature	Amazon RDS	Amazon RDS Custom	Amazon EC2	Notes
DDL event notifications	 N	 Y	 Y	If this feature is critical to your workload, consider choosing Amazon RDS Custom or Amazon EC2.
Delayed transaction durability (lazy commit)	 Y (SQL Server 2014 and later)	 Y (SQL Server 2019)	 Y (SQL Server 2014 and later)	
Distributed queries	 Y (SQL Server targets)	 Y (SQL Server targets)	 Y (SQL Server targets)	See the Implementing linked servers with Amazon RDS for SQL Server blog post.
Extended events	 Y	 Y	 Y	

Development feature	Amazon RDS	Amazon RDS Custom	Amazon EC2	Notes
Extended stored procedures, including `xp_cmdshell`	 N	 Y	 Y	If this feature is critical to your workload, consider choosing Amazon RDS Custom or Amazon EC2.
File tables	 N	 Y	 Y	If this feature is critical to your workload, consider choosing Amazon RDS Custom or Amazon EC2.
`FILESTREAM`	 N	 Y	 Y	`FILESTREAM` isn't compatible with Amazon RDS. However, you can configure the in-memory database.
Full-text search	 Y (except semantic search)	 Y	 Y	

Development feature	Amazon RDS	Amazon RDS Custom	Amazon EC2	Notes
In-memory database	 Y (SQL Server 2014 and later)	 Y (SQL Server 2019)	 Y (SQL Server 2014 and later)	
Linked servers	 Y (SQL Server and Oracle targets)	 Y	 Y	See the Implementing linked servers with Amazon RDS for SQL Server blog post and Support for linked servers with Oracle OLEDB in Amazon RDS for SQL Server in the Amazon RDS documentation.

Development feature	Amazon RDS	Amazon RDS Custom	Amazon EC2	Notes
Machine Learning Services (with R scripts)	 Y	 Y	 Y	Machine Learning Services must be installed separately on a Windows or Linux machine. It's supported on an Always On Failover Cluster Instance (FCI) only in SQL Server 2019 and later. Although R isn't supported on Amazon RDS, you can use it on AWS (see the blog post Getting started with R on AWS).

Development feature	Amazon RDS	Amazon RDS Custom	Amazon EC2	Notes
Maintenance plans	 N	 Y	 Y	Amazon RDS provides a separate set of features to facilitate backup and recovery of databases. For backup, you can configure automated backup.
Master Data Services	 N	 Y	 Y	If this feature is critical to your workload, consider choosing Amazon RDS Custom or Amazon EC2.
Microsoft Distributed Transaction Coordinator (MSDTC)	 Y	 Y	 Y	See the blog post Enabling distributed transaction support for domain-joined Amazon RDS for SQL Server instances.

Development feature	Amazon RDS	Amazon RDS Custom	Amazon EC2	Notes
`OPENROWSET`	 Y	 Y	 Y	
Partially contained databases	 Y (SQL Server 2014 and later)	 Y (SQL Server 2019)	 Y (SQL Server 2014 and later)	
Performance Data Collector	 N	 Y	 Y	On Amazon RDS, you can use Amazon CloudWatch, AWS CloudTrail, and Performance Insights to monitor your SQL Server performance (see Overview of monitoring Amazon RDS in the Amazon RDS documentation).

Development feature	Amazon RDS	Amazon RDS Custom	Amazon EC2	Notes
Policy-Based Management	 N	 Y	 Y	If this feature is critical to your workload, consider choosing Amazon RDS Custom or Amazon EC2.
PolyBase	 N	 Y	 Y	If this feature is critical to your workload, consider choosing Amazon RDS Custom or Amazon EC2.
Preconfigured parameters	 Y	 N	 N	
Resource Governor	 N	 Y	 Y	If this feature is critical to your workload, consider choosing Amazon RDS Custom or Amazon EC2.

Development feature	Amazon RDS	Amazon RDS Custom	Amazon EC2	Notes
Safe CLR	 Y (SQL Server 2014 and 2016)	 Y	 Y	
Sequences	 Y (SQL Server 2014 and later)	 Y (SQL Server 2019)	 Y (SQL Server 2014 and later)	
Server-level triggers	 N	 Y	 Y	If this feature is critical to your workload, consider choosing Amazon RDS Custom or Amazon EC2.
Service Broker	 Y (except endpoints)	 Y	 Y	
Spatial and location features	 Y	 Y	 Y	
SQL Server Agent	 Y	 Y	 Y	

Development feature	Amazon RDS	Amazon RDS Custom	Amazon EC2	Notes
SQL Server Analysis Services (SSAS)	 (SQL Server 2016 and later)			See Support for SSAS in Amazon RDS for SQL Server in the Amazon RDS documentation.
SQL Server Integration Services (SSIS)	 (SQL Server 2016 and later)			See Support for SSIS in Amazon RDS for SQL Server in the Amazon RDS documentation.
SQL Server Management Studio (SSMS)				
SQL Server Migration Assistant (SSMA)				
SQL Server Profiler	 (server-side and client-side traces)			

Development feature	Amazon RDS	Amazon RDS Custom	Amazon EC2	Notes
SQL Server Reporting Services (SSRS)	 Y (SQL Server 2016 and later)	 Y	 Y	See Support for SSRS in Amazon RDS for SQL Server in the Amazon RDS documentation.
`sqlcmd`	 Y	 Y	 Y	
Stretch Database	 N	 Y	 Y	If this feature is critical to your workload, consider choosing Amazon RDS Custom or Amazon EC2.
`THROW` statement	 Y (SQL Server 2014 and later)	 Y (SQL Server 2019)	 Y (SQL Server 2014 and later)	

Development feature	Amazon RDS	Amazon RDS Custom	Amazon EC2	Notes
Transact-SQL endpoints	 N	 Y	 Y	All operations that use `CREATE ENDPOINT` are unavailable on Amazon RDS. We recommend that you install SQL Server on an EC2 instance for these operations.
UTF-16 support	 Y (SQL Server 2014 and later)	 Y	 Y (SQL Server 2014 and later)	
WCF Data Service	 N	 Y	 Y	If this feature is critical to your workload, consider choosing Amazon RDS Custom or Amazon EC2.

HA/DR

Development feature	Amazon RDS	Amazon RDS Custom	Amazon EC2	Notes
Always On availability groups	 Y	 Y (both synchronous and asynchronous)	 Y	If you need a self-managed Always On availability group, we recommend that you use AWS Launch Wizard to simplify SQL Server HA deployment on an EC2 instance. See AWS Launch Wizard for SQL Server in the AWS documentation.
Always On Failover Cluster Instances (FCIs)	 N	 Y	 Y	You can use AWS Launch Wizard to simplify your SQL Server FCI deployment on Amazon EC2. See AWS Launch Wizard for SQL Server in the AWS documentation.

Development feature	Amazon RDS	Amazon RDS Custom	Amazon EC2	Notes
Backing up to Amazon S3	 Y	 Y	 Y	Amazon RDS supports native backup and restore for SQL Server databases by using full backup files (.bak files) and Amazon S3 as a repository. See Importing and Exporting SQL Server databases in the Amazon RDS documentation.
BACKUP command	 N	 Y	 Y	See How do I perform native backups of an Amazon RDS DB instance that's running SQL Server? in AWS Knowledge Center.
Database mirroring	 Y (Multi-AZ)	 Y	 Y	

Development feature	Amazon RDS	Amazon RDS Custom	Amazon EC2	Notes
Database replication	 N (limited push subscription)	 Y	 Y	If you want to replicate a single table on Amazon RDS, you can also use AWS DMS or set up read replicas.
Distributed availability groups	 N	 Y	 Y	If this feature is critical to your workload, consider choosing Amazon RDS Custom or Amazon EC2.
Log shipping	 N	 Y	 Y	For disaster recovery purposes, you can use read replicas or AWS DMS.
Managed automated backups	 Y	 Y	 N (requires configuring and managing maintenance plans, or using third-party solutions)	See Working with backups in the Amazon RDS documentation.

Development feature	Amazon RDS	Amazon RDS Custom	Amazon EC2	Notes
Multi-AZ with automated failover	 Y	 Y (with manual configuration of Always On availability groups)	 Y (Enterprise Edition only, with manual configuration of Always On availability groups)	See Multi-AZ deployments for Amazon RDS for SQL Server in the Amazon RDS documentation.
Read replicas	 Y (SQL Server 2016 and later)	 Y (with manual configuration of Always On availability groups)	 Y (with manual configuration of Always On availability groups)	
RESTORE command	 Y	 Y	 Y	See AWS Knowledge Center.

Scalability

Development feature	Amazon RDS	Amazon RDS Custom	Amazon EC2	Notes
Built-in instance and database monitoring and metrics	 Y	 N	 N (export your	See the blog post Monitor your SQL Server database by

Development feature	Amazon RDS	Amazon RDS Custom	Amazon EC2	Notes
			own metrics to CloudWatch or use a third-par ty solution)	using custom metrics with Amazon CloudWatc h and AWS Systems Manager.
Configurable storage size	 Y	 Y	 Y	
Maximum number of databases per instance	Depends on the instance size and Multi-AZ configuration	SQL Server maximum (5000)	 limitation N	See Maximum capacity specifications for SQL Server in the Microsoft SQL Server documentation.

Development feature	Amazon RDS	Amazon RDS Custom	Amazon EC2	Notes
Maximum storage size of a DB instance	16 TiB	16 TiB	 N limitation	Amazon RDS also supports tempdb databases on local disks by using Non-Volatile Memory Express (NVMe) instance storage. See Instance store support for the tempdb database on Amazon RDS for SQL Server in the Amazon RDS documentation.
Minimum storage size of a DB instance	20 GiB (Enterprise, Standard, Web, and Express Editions)	20 GiB (Enterprise, Standard, Web, and Express Editions)	 N limitation	
New Query Optimizer	 Y (SQL Server 2014 and later)	 Y (SQL Server 2014 and later)	 Y (SQL Server 2014 and later)	

Development feature	Amazon RDS	Amazon RDS Custom	Amazon EC2	Notes
Read replicas	 Y (SQL Server 2016 and later)	 Y (with manual configuration of Always On availability groups)	 Y (with manual configuration of Always On availability groups)	

Security

Development feature	Amazon RDS	Amazon RDS Custom	Amazon EC2	Notes
Automatic software patching	 Y	 N	 N	
Encrypted storage using AWS KMS	 Y (all SQL Server editions except Express)	 Y	 Y	See the blog post Securing data in Amazon RDS using AWS KMS encryption.
Flexible server roles	 Y (SQL Server 2014 and later)	 Y (SQL Server 2019)	 Y (SQL Server 2014 and later)	

Development feature	Amazon RDS	Amazon RDS Custom	Amazon EC2	Notes
SQL authentication	 Y	 Y	 Y	
SQL Server audit	 Y	 Y	 Y	
SSL (encryption in transit)	 Y	 Y	 Y	See Using SSL with a Microsoft SQL Server DB instance in the Amazon RDS documentation.

Development feature	Amazon RDS	Amazon RDS Custom	Amazon EC2	Notes
sysadmin role	 N	 Y	 Y	For unsupported server-level roles, see Microsoft SQL Server security in the Amazon RDS documentation. When you create a new RDS DB instance, the default master user that you use gets certain privileges for that DB instance (see Account privileges in the Amazon RDS documentation).
TDE (encryption at rest)	 Y (Enterprise Edition: 2014-2019; Standard Edition: 2019)	 Y (SQL Server 2019 Enterprise, Standard, Web, and Developer Editions)	 Y (Enterprise Edition: 2014-2019; Standard Edition: 2019)	See information about TDE support in the Amazon RDS and Amazon RDS Custom documentation.

Development feature	Amazon RDS	Amazon RDS Custom	Amazon EC2	Notes
Windows Authentication	 Y	 Y	 Y	

Other features

Development feature	Amazon RDS	Amazon RDS Custom	Amazon EC2	Notes
Ability to install a third-party agent	 N	 Y	 Y	
Ability to rename existing databases	 Y (Single-AZ only)	 Y (not available for databases in availability groups or enabled for mirroring)	 Y (not available for databases in availability groups or enabled for mirroring)	For Multi-AZ deployments on Amazon RDS, see Renaming a Microsoft SQL Server database in a Multi-AZ deployment in the Amazon RDS documentation.
Control over DB instance and operating system	 N	 Y	 Y	If this feature is critical to your workload, consider choosing Amazon RDS Custom or Amazon EC2.

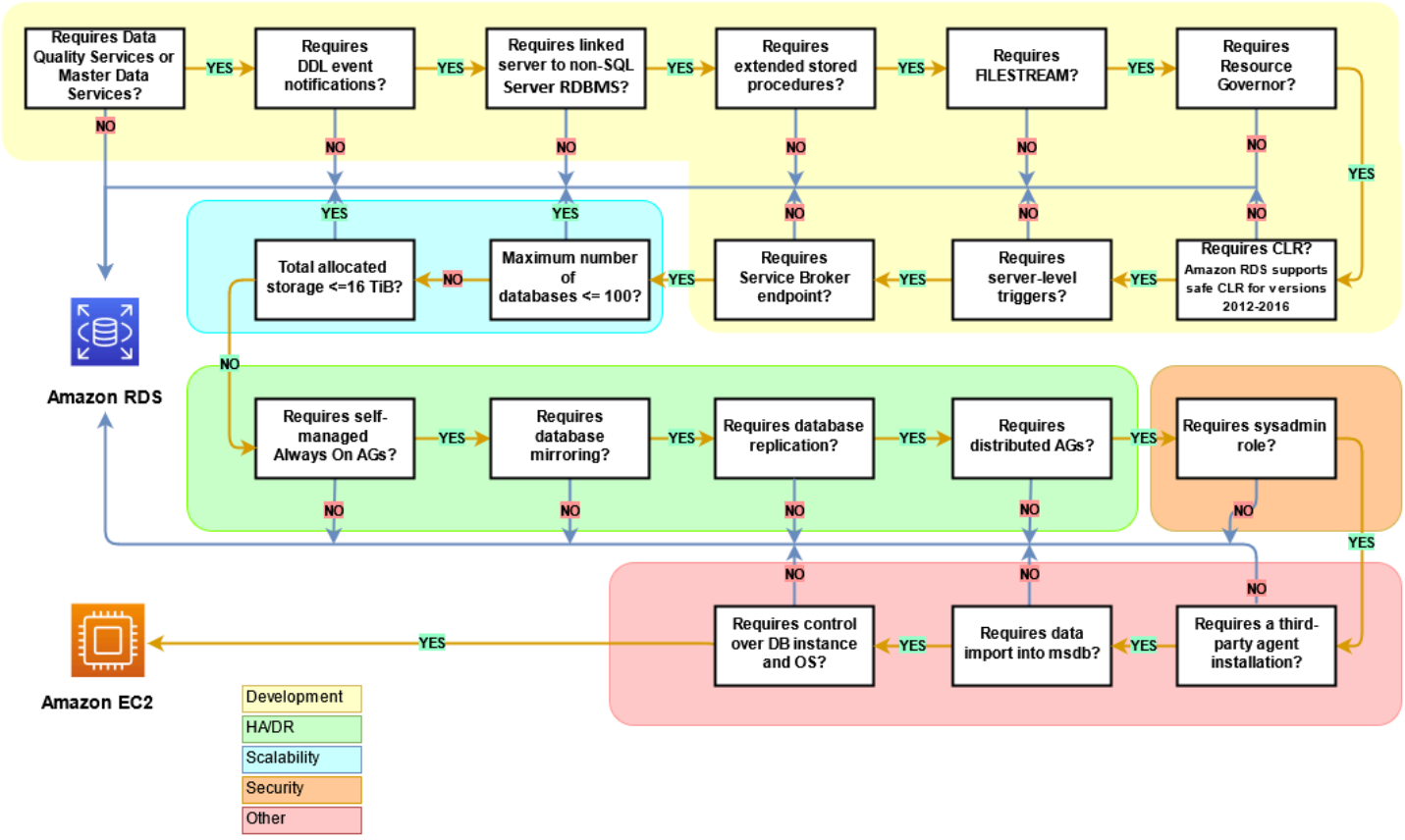
Development feature	Amazon RDS	Amazon RDS Custom	Amazon EC2	Notes
Custom set time zones	 Y	 Y	 Y	
Distributed Replay	 N	 Y	 Y	The SQL Server Distributed Replay client service requires sysadmin permissions, which is why it isn't supported in Amazon RDS.
Import data into the msdb database	 N	 Y	 Y	If this feature is critical to your workload, consider choosing Amazon RDS Custom or Amazon EC2.
Installation methods	N/A	N/A	Amazon Machine Image (AMI) or manual installation	
SQL Server editions	Enterprise, Standard, Web, Express	Enterprise, Standard, Developer	Enterprise, Standard, Web, Developer, Express	

Development feature	Amazon RDS	Amazon RDS Custom	Amazon EC2	Notes
SQL Server versions	2014, 2016, 2017, 2019, 2022	2019, 2022	2014, 2016, 2017, 2019, 2022	

For detailed information about these features, see the following:

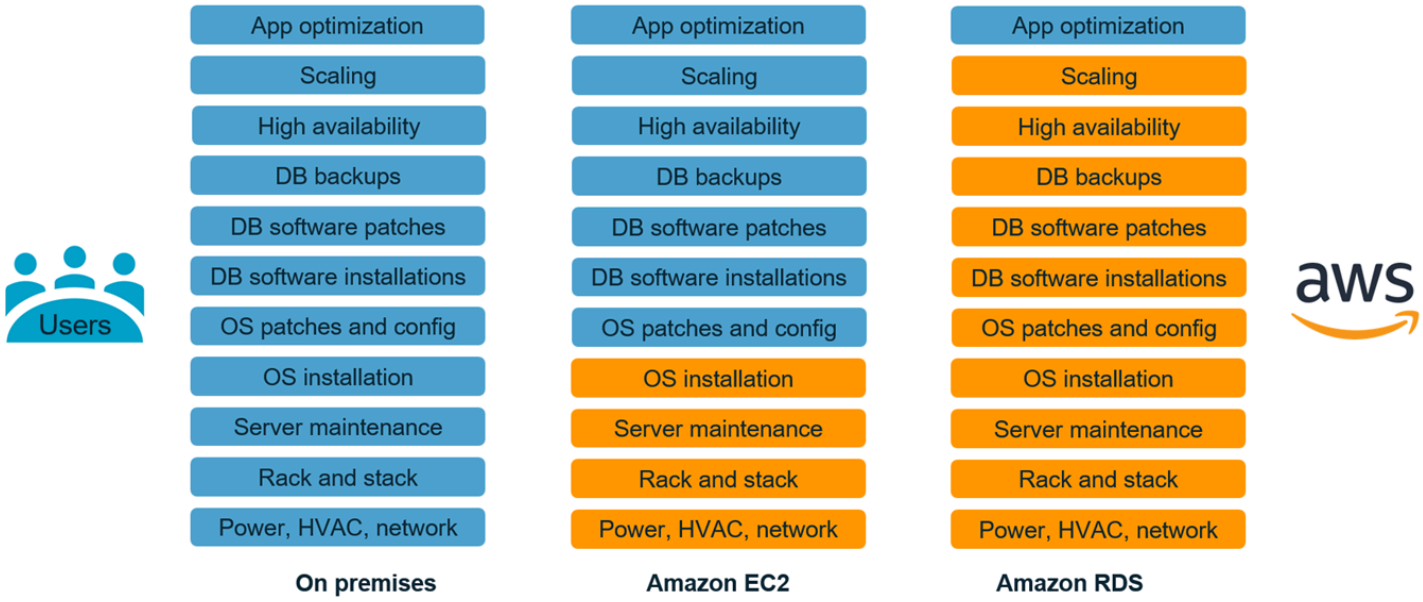
- Microsoft Products on AWS
- Active Directory Reference Architecture: Implementing Active Directory Domain Services on AWS
- Remote Desktop Gateway on AWS (AWS Quick Start)
- Securing the Microsoft Platform on AWS
- SQL Server with Always On replication on AWS (AWS Quick Start)
- AWS Directory Service
- `AWSEC2-SQLServerDBRestore` (AWS Systems Manager Automation runbook that restores SQL Server database backups stored in Amazon S3 to SQL Server 2017 running on an EC2 Linux instance)

The following diagram helps visualize the information in the previous table, to assist in your decision-making process.



Shared responsibility

The following diagram shows the division of responsibilities between AWS and the user in the management of SQL Server features and operations.



With AWS services, you don't have to worry about administration tasks such as server provisioning, patching, setup, configuration, backups, or recovery. AWS continuously monitors your clusters to keep your workloads up and running with self-healing storage and automated scaling. You focus on high-value application development tasks such as schema design, query construction, and optimization, while AWS takes care of operational tasks on your behalf.

You never have to over-provision or under-provision infrastructure to accommodate application growth, intermittent spikes, and performance requirements, or incur fixed capital costs, including software licensing and support, hardware refresh, and resources to maintain hardware. AWS manages these, so you can spend time innovating and building new applications, not managing infrastructure.

For more information, see Shared Responsibility Model on the AWS website.

SQL Server database migration patterns

Use the following links to see the AWS Prescriptive Guidance patterns for migrating SQL Server databases to AWS:

- Rehost patterns (from SQL Server to Amazon EC2)
- Replatform patterns (from SQL Server to Amazon RDS for SQL Server)
- Re-architect patterns (from SQL Server to open-source and AWS Cloud-native databases)

If you're looking for patterns that cover the use of a specific tool, type in the tool name in the search box or choose it from a filter. For example, you can use this query to see all SQL Server migration patterns that use AWS DMS.

Partners

Database migration can be a challenging project that requires expertise and tools. You can accelerate your migration and time to results through partnership. AWS Database Migration Service partners have the required expertise to help customers migrate to the cloud easily and securely. These partners have the expertise for both homogeneous migrations such as SQL Server to SQL Server, and heterogeneous migrations between different database platforms, such as SQL Server to Amazon Aurora or Amazon RDS for MySQL.

Based on your requirements and preferences, you can use the partner to handle the complete migration or to help with only some aspects of the migration. In addition, you can use tools and solutions provided by AWS Partner Network (APN) Partners to help with the migration. For a complete catalog of migration tools and solutions, see AWS Partner tools and solutions.

Additional resources

Blog posts

- Cross-Region disaster recovery of Amazon RDS for SQL Server
- Database Migration—What Do You Need to Know Before You Start?
- Deploying Always On availability groups between Amazon EC2 Windows and Amazon Linux 2 instances
- How to architect a hybrid Microsoft SQL Server solution using distributed availability groups
- How to migrate to Amazon RDS for SQL Server using transactional replication
- Introducing Ongoing Replication from Amazon RDS for SQL Server Using AWS Database Migration Service
- Learn why AWS is the best cloud to run Microsoft Windows Server and SQL Server workloads
- Migrate your on-premises SQL Server Windows Workloads to Amazon EC2 Linux
- Migrating a SQL Server Database to a MySQL-Compatible Database Engine
- Migrating your on-premises SQL Server Windows workloads to Amazon EC2 Linux
- Simplify your Microsoft SQL Server high availability deployments using FSx for Windows File Server
- Store SQL Server backups in Amazon S3 using Storage Gateway

AWS documentation

- Amazon Aurora
- Amazon EC2
- Amazon RDS
- Amazon RDS Custom
- AWS DMS
- AWS SCT
- SQL Server Licensing

Acknowledgments

The author acknowledges the following specialists for contributing to this guide:

- Marcelo Fernandes, AWS Migrations Senior Consultant – Choosing between Amazon EC2 and Amazon RDS section
- Tarun Chawla, Database Migrations Consultant – Choosing between Amazon EC2 and Amazon RDS section
- Alex Zuo, Senior Technical Product Manager, SQL Server on Amazon EC2 – AWS Migration Hub Orchestrator section

Appendix: SQL Server database migration questionnaire

Use the questionnaire in this section as a starting point to gather information for the assessment and planning phases of your migration project. You can download this questionnaire in Microsoft Excel format and use it to record your information.

Download questionnaire

General information

1. What is the name of your SQL Server instance?
2. What is the version of your SQL Server?
3. What is the edition of your SQL Server database: Standard, Developer, or Enterprise?
4. What is the database type (OLTP, DW, reporting, batch processing)?
5. How many databases do you have on the SQL Server instance?
6. What is the size of your database?
7. What is the database collation?
8. What is the time zone of the database?
9. What are the average and maximum I/O transactions per second (TPS)?
10. What is the IOPS (on average and maximum) for this database for read/write operations?
11. How many transaction logs do you generate per hour (with average and maximum size)?
12. Does the database have linked servers pointing to other databases?
13. What are the SLA requirements for your database?
14. What is the RTO and RPO requirements for your database?
15. How much database downtime can you allow for migration purposes?
16. Do you have any compliance, regulatory, or auditing requirements?
17. What tool do you use to monitor your SQL Server databases?

Infrastructure

1. What is the hostname of the database?

2. What is the operating system used for this database?
3. How many CPU cores does the server have?
4. What is the memory size on the server?
5. Is the database on a virtualized machine or a physical server?
6. Are you using local storage?
7. Do you use network-attached storage (NAS) or storage area network (SAN) storage types?
8. Do you have a cluster or single instances?

Database backups

1. How do you back up your database? How often?
2. What is your retention period for transaction logs and backups?
3. Where do you store your backup?

Database features

1. Do you use automatic tuning for your SQL Server instance?
2. Do you use parallel-indexed operations?
3. Do you use partitioned table parallelism features?
4. Do you use table and index partitioning?

Database security

1. Do you use dynamic data masking?
2. Do you use security features such as Transparent Database Encryption (TDE)?
3. Do you use server or database audits?
4. Do you use advanced compression?

Database high availability and disaster recovery

1. What are your high availability requirements?

2. Do you use transactional replication?
3. Do you use peer-to-peer transactional replication?
4. What type of high availability solutions (for example, failover clustering, Always On availability groups, database mirroring) do you use for your SQL Server environment?
5. Where are your primary and standby database regions?
6. What do you use as a disaster recovery solution (for example, log shipping, Always On availability groups, a SAN-based virtualized environment)?
7. Do you use a Domain Name System (DNS) alias for database connectivity?

Document history

The following table describes significant changes to this guide. If you want to be notified about future updates, you can subscribe to an RSS feed.

Change	Description	Date
Updated section	Added information about Amazon EBS Multi-Attach with persistent reservations and other options for deploying FCIs on AWS to the Always On Failover Cluster Instances section.	April 1, 2024
Updated information	Updated the decision matrix for SQL Server 2022.	March 18, 2024
Updated information	Updated the decision matrix to reflect that Amazon RDS Custom now supports TDE.	November 16, 2023
Added section	Added information about AWS Migration Hub Orchestrator.	June 29, 2023
Removed section	Removed information about CloudEndure Migration, which is being discontinued. AWS Application Migration Service is the primary migration service recommended for lift-and-shift migrations to the AWS Cloud.	September 23, 2022
Updated section	Added more information to the decision matrix.	August 3, 2022

Added and updated sections	Added information about migrating SQL Server databases to Amazon RDS Custom and using Babelfish to migrate SQL Server databases to Aurora PostgreSQL. Updated the decision matrix with the latest information.	July 29, 2022
Corrected text	Corrected the CLR informati on in the decision matrix.	June 21, 2022
Updated section	Updated the *CloudEndure Migration* section with the latest information about product availability.	May 10, 2022
Removed section	Removed information about Amazon RDS for VMware, which is no longer supported.	April 19, 2022
Added section	Added information about AWS Launch Wizard for SQL Server.	July 15, 2021
Added decision matrix	In the Choosing between Amazon EC2 and Amazon RDS section, provided a side-by-s ide comparison of SQL Server support.	June 28, 2021
Updated log shipping information	In the Log shipping section, clarified that log shipping on Amazon RDS for SQL Server requires custom scripts.	March 25, 2021
Initial publication	—	October 9, 2020

AWS Prescriptive Guidance glossary

The following are commonly used terms in strategies, guides, and patterns provided by AWS Prescriptive Guidance. To suggest entries, please use the **Provide feedback** link at the end of the glossary.

Numbers

7 Rs

Seven common migration strategies for moving applications to the cloud. These strategies build upon the 5 Rs that Gartner identified in 2011 and consist of the following:

- Refactor/re-architect – Move an application and modify its architecture by taking full advantage of cloud-native features to improve agility, performance, and scalability. This typically involves porting the operating system and database. Example: Migrate your on-premises Oracle database to the Amazon Aurora PostgreSQL-Compatible Edition.
- Replatform (lift and reshape) – Move an application to the cloud, and introduce some level of optimization to take advantage of cloud capabilities. Example: Migrate your on-premises Oracle database to Amazon Relational Database Service (Amazon RDS) for Oracle in the AWS Cloud.
- Repurchase (drop and shop) – Switch to a different product, typically by moving from a traditional license to a SaaS model. Example: Migrate your customer relationship management (CRM) system to Salesforce.com.
- Rehost (lift and shift) – Move an application to the cloud without making any changes to take advantage of cloud capabilities. Example: Migrate your on-premises Oracle database to Oracle on an EC2 instance in the AWS Cloud.
- Relocate (hypervisor-level lift and shift) – Move infrastructure to the cloud without purchasing new hardware, rewriting applications, or modifying your existing operations. This migration scenario is specific to VMware Cloud on AWS, which supports virtual machine (VM) compatibility and workload portability between your on-premises environment and AWS. You can use the VMware Cloud Foundation technologies from your on-premises data centers when you migrate your infrastructure to VMware Cloud on AWS. Example: Relocate the hypervisor hosting your Oracle database to VMware Cloud on AWS.
- Retain (revisit) – Keep applications in your source environment. These might include applications that require major refactoring, and you want to postpone that work until a later

time, and legacy applications that you want to retain, because there's no business justification for migrating them.

- Retire – Decommission or remove applications that are no longer needed in your source environment.

A

ABAC

See attribute-based access control.

abstracted services

See managed services.

ACID

See atomicity, consistency, isolation, durability.

active-active migration

A database migration method in which the source and target databases are kept in sync (by using a bidirectional replication tool or dual write operations), and both databases handle transactions from connecting applications during migration. This method supports migration in small, controlled batches instead of requiring a one-time cutover. It's more flexible but requires more work than active-passive migration.

active-passive migration

A database migration method in which in which the source and target databases are kept in sync, but only the source database handles transactions from connecting applications while data is replicated to the target database. The target database doesn't accept any transactions during migration.

aggregate function

A SQL function that operates on a group of rows and calculates a single return value for the group. Examples of aggregate functions include `SUM` and `MAX`.

AI

See artificial intelligence.

AIOps

See artificial intelligence operations.

anonymization

The process of permanently deleting personal information in a dataset. Anonymization can help protect personal privacy. Anonymized data is no longer considered to be personal data.

anti-pattern

A frequently used solution for a recurring issue where the solution is counter-productive, ineffective, or less effective than an alternative.

application control

A security approach that allows the use of only approved applications in order to help protect a system from malware.

application portfolio

A collection of detailed information about each application used by an organization, including the cost to build and maintain the application, and its business value. This information is key to the portfolio discovery and analysis process and helps identify and prioritize the applications to be migrated, modernized, and optimized.

artificial intelligence (AI)

The field of computer science that is dedicated to using computing technologies to perform cognitive functions that are typically associated with humans, such as learning, solving problems, and recognizing patterns. For more information, see What is Artificial Intelligence?

artificial intelligence operations (AIOps)

The process of using machine learning techniques to solve operational problems, reduce operational incidents and human intervention, and increase service quality. For more information about how AIOps is used in the AWS migration strategy, see the operations integration guide.

asymmetric encryption

An encryption algorithm that uses a pair of keys, a public key for encryption and a private key for decryption. You can share the public key because it isn't used for decryption, but access to the private key should be highly restricted.

atomicity, consistency, isolation, durability (ACID)

A set of software properties that guarantee the data validity and operational reliability of a database, even in the case of errors, power failures, or other problems.

attribute-based access control (ABAC)

The practice of creating fine-grained permissions based on user attributes, such as department, job role, and team name. For more information, see ABAC for AWS in the AWS Identity and Access Management (IAM) documentation.

authoritative data source

A location where you store the primary version of data, which is considered to be the most reliable source of information. You can copy data from the authoritative data source to other locations for the purposes of processing or modifying the data, such as anonymizing, redacting, or pseudonymizing it.

Availability Zone

A distinct location within an AWS Region that is insulated from failures in other Availability Zones and provides inexpensive, low-latency network connectivity to other Availability Zones in the same Region.

AWS Cloud Adoption Framework (AWS CAF)

A framework of guidelines and best practices from AWS to help organizations develop an efficient and effective plan to move successfully to the cloud. AWS CAF organizes guidance into six focus areas called perspectives: business, people, governance, platform, security, and operations. The business, people, and governance perspectives focus on business skills and processes; the platform, security, and operations perspectives focus on technical skills and processes. For example, the people perspective targets stakeholders who handle human resources (HR), staffing functions, and people management. For this perspective, AWS CAF provides guidance for people development, training, and communications to help ready the organization for successful cloud adoption. For more information, see the AWS CAF website and the AWS CAF whitepaper.

AWS Workload Qualification Framework (AWS WQF)

A tool that evaluates database migration workloads, recommends migration strategies, and provides work estimates. AWS WQF is included with AWS Schema Conversion Tool (AWS SCT). It analyzes database schemas and code objects, application code, dependencies, and performance characteristics, and provides assessment reports.

B

BCP

See business continuity planning.

behavior graph

A unified, interactive view of resource behavior and interactions over time. You can use a behavior graph with Amazon Detective to examine failed logon attempts, suspicious API calls, and similar actions. For more information, see Data in a behavior graph in the Detective documentation.

big-endian system

A system that stores the most significant byte first. See also endianness.

binary classification

A process that predicts a binary outcome (one of two possible classes). For example, your ML model might need to predict problems such as “Is this email spam or not spam?" or "Is this product a book or a car?"

bloom filter

A probabilistic, memory-efficient data structure that is used to test whether an element is a member of a set.

branch

A contained area of a code repository. The first branch created in a repository is the *main branch*. You can create a new branch from an existing branch, and you can then develop features or fix bugs in the new branch. A branch you create to build a feature is commonly referred to as a *feature branch*. When the feature is ready for release, you merge the feature branch back into the main branch. For more information, see About branches (GitHub documentation).

break-glass access

In exceptional circumstances and through an approved process, a quick means for a user to gain access to an AWS account that they don't typically have permissions to access. For more information, see the Implement break-glass procedures indicator in the AWS Well-Architected guidance.

brownfield strategy

The existing infrastructure in your environment. When adopting a brownfield strategy for a system architecture, you design the architecture around the constraints of the current systems and infrastructure. If you are expanding the existing infrastructure, you might blend brownfield and greenfield strategies.

buffer cache

The memory area where the most frequently accessed data is stored.

business capability

What a business does to generate value (for example, sales, customer service, or marketing). Microservices architectures and development decisions can be driven by business capabilities. For more information, see the Organized around business capabilities section of the Running containerized microservices on AWS whitepaper.

business continuity planning (BCP)

A plan that addresses the potential impact of a disruptive event, such as a large-scale migration, on operations and enables a business to resume operations quickly.

C

CAF

See AWS Cloud Adoption Framework.

CCoE

See Cloud Center of Excellence.

CDC

See change data capture.

change data capture (CDC)

The process of tracking changes to a data source, such as a database table, and recording metadata about the change. You can use CDC for various purposes, such as auditing or replicating changes in a target system to maintain synchronization.

chaos engineering

Intentionally introducing failures or disruptive events to test a system's resilience. You can use AWS Fault Injection Service (AWS FIS) to perform experiments that stress your AWS workloads and evaluate their response.

CI/CD

See continuous integration and continuous delivery.

classification

A categorization process that helps generate predictions. ML models for classification problems predict a discrete value. Discrete values are always distinct from one another. For example, a model might need to evaluate whether or not there is a car in an image.

client-side encryption

Encryption of data locally, before the target AWS service receives it.

Cloud Center of Excellence (CCoE)

A multi-disciplinary team that drives cloud adoption efforts across an organization, including developing cloud best practices, mobilizing resources, establishing migration timelines, and leading the organization through large-scale transformations. For more information, see the CCoE posts on the AWS Cloud Enterprise Strategy Blog.

cloud computing

The cloud technology that is typically used for remote data storage and IoT device management. Cloud computing is commonly connected to edge computing technology.

cloud operating model

In an IT organization, the operating model that is used to build, mature, and optimize one or more cloud environments. For more information, see Building your Cloud Operating Model.

cloud stages of adoption

The four phases that organizations typically go through when they migrate to the AWS Cloud:

- Project – Running a few cloud-related projects for proof of concept and learning purposes
- Foundation – Making foundational investments to scale your cloud adoption (e.g., creating a landing zone, defining a CCoE, establishing an operations model)
- Migration – Migrating individual applications
- Re-invention – Optimizing products and services, and innovating in the cloud

These stages were defined by Stephen Orban in the blog post The Journey Toward Cloud-First & the Stages of Adoption on the AWS Cloud Enterprise Strategy blog. For information about how they relate to the AWS migration strategy, see the migration readiness guide.

CMDB

See configuration management database.

code repository

A location where source code and other assets, such as documentation, samples, and scripts, are stored and updated through version control processes. Common cloud repositories include GitHub or AWS CodeCommit. Each version of the code is called a *branch*. In a microservice structure, each repository is devoted to a single piece of functionality. A single CI/CD pipeline can use multiple repositories.

cold cache

A buffer cache that is empty, not well populated, or contains stale or irrelevant data. This affects performance because the database instance must read from the main memory or disk, which is slower than reading from the buffer cache.

cold data

Data that is rarely accessed and is typically historical. When querying this kind of data, slow queries are typically acceptable. Moving this data to lower-performing and less expensive storage tiers or classes can reduce costs.

computer vision

A field of AI used by machines to identify people, places, and things in images with accuracy at or above human levels. Often built with deep learning models, it automates extraction, analysis, classification, and understanding of useful information from a single image or a sequence of images.

configuration management database (CMDB)

A repository that stores and manages information about a database and its IT environment, including both hardware and software components and their configurations. You typically use data from a CMDB in the portfolio discovery and analysis stage of migration.

conformance pack

A collection of AWS Config rules and remediation actions that you can assemble to customize your compliance and security checks. You can deploy a conformance pack as a single entity in

an AWS account and Region, or across an organization, by using a YAML template. For more information, see Conformance packs in the AWS Config documentation.

continuous integration and continuous delivery (CI/CD)

The process of automating the source, build, test, staging, and production stages of the software release process. CI/CD is commonly described as a pipeline. CI/CD can help you automate processes, improve productivity, improve code quality, and deliver faster. For more information, see Benefits of continuous delivery. CD can also stand for *continuous deployment*. For more information, see Continuous Delivery vs. Continuous Deployment.

D

data at rest

Data that is stationary in your network, such as data that is in storage.

data classification

A process for identifying and categorizing the data in your network based on its criticality and sensitivity. It is a critical component of any cybersecurity risk management strategy because it helps you determine the appropriate protection and retention controls for the data. Data classification is a component of the security pillar in the AWS Well-Architected Framework. For more information, see Data classification.

data drift

A meaningful variation between the production data and the data that was used to train an ML model, or a meaningful change in the input data over time. Data drift can reduce the overall quality, accuracy, and fairness in ML model predictions.

data in transit

Data that is actively moving through your network, such as between network resources.

data minimization

The principle of collecting and processing only the data that is strictly necessary. Practicing data minimization in the AWS Cloud can reduce privacy risks, costs, and your analytics carbon footprint.

data perimeter

A set of preventive guardrails in your AWS environment that help make sure that only trusted identities are accessing trusted resources from expected networks. For more information, see Building a data perimeter on AWS.

data preprocessing

To transform raw data into a format that is easily parsed by your ML model. Preprocessing data can mean removing certain columns or rows and addressing missing, inconsistent, or duplicate values.

data provenance

The process of tracking the origin and history of data throughout its lifecycle, such as how the data was generated, transmitted, and stored.

data subject

An individual whose data is being collected and processed.

data warehouse

A data management system that supports business intelligence, such as analytics. Data warehouses commonly contain large amounts of historical data, and they are typically used for queries and analysis.

database definition language (DDL)

Statements or commands for creating or modifying the structure of tables and objects in a database.

database manipulation language (DML)

Statements or commands for modifying (inserting, updating, and deleting) information in a database.

DDL

See database definition language.

deep ensemble

To combine multiple deep learning models for prediction. You can use deep ensembles to obtain a more accurate prediction or for estimating uncertainty in predictions.

deep learning

An ML subfield that uses multiple layers of artificial neural networks to identify mapping between input data and target variables of interest.

defense-in-depth

An information security approach in which a series of security mechanisms and controls are thoughtfully layered throughout a computer network to protect the confidentiality, integrity, and availability of the network and the data within. When you adopt this strategy on AWS, you add multiple controls at different layers of the AWS Organizations structure to help secure resources. For example, a defense-in-depth approach might combine multi-factor authentication, network segmentation, and encryption.

delegated administrator

In AWS Organizations, a compatible service can register an AWS member account to administer the organization's accounts and manage permissions for that service. This account is called the *delegated administrator* for that service. For more information and a list of compatible services, see Services that work with AWS Organizations in the AWS Organizations documentation.

deployment

The process of making an application, new features, or code fixes available in the target environment. Deployment involves implementing changes in a code base and then building and running that code base in the application's environments.

development environment

See environment.

detective control

A security control that is designed to detect, log, and alert after an event has occurred. These controls are a second line of defense, alerting you to security events that bypassed the preventative controls in place. For more information, see Detective controls in *Implementing security controls on AWS*.

development value stream mapping (DVSM)

A process used to identify and prioritize constraints that adversely affect speed and quality in a software development lifecycle. DVSM extends the value stream mapping process originally designed for lean manufacturing practices. It focuses on the steps and teams required to create and move value through the software development process.

digital twin

A virtual representation of a real-world system, such as a building, factory, industrial equipment, or production line. Digital twins support predictive maintenance, remote monitoring, and production optimization.

dimension table

In a star schema, a smaller table that contains data attributes about quantitative data in a fact table. Dimension table attributes are typically text fields or discrete numbers that behave like text. These attributes are commonly used for query constraining, filtering, and result set labeling.

disaster

An event that prevents a workload or system from fulfilling its business objectives in its primary deployed location. These events can be natural disasters, technical failures, or the result of human actions, such as unintentional misconfiguration or a malware attack.

disaster recovery (DR)

The strategy and process you use to minimize downtime and data loss caused by a disaster. For more information, see Disaster Recovery of Workloads on AWS: Recovery in the Cloud in the AWS Well-Architected Framework.

DML

See database manipulation language.

domain-driven design

An approach to developing a complex software system by connecting its components to evolving domains, or core business goals, that each component serves. This concept was introduced by Eric Evans in his book, *Domain-Driven Design: Tackling Complexity in the Heart of Software* (Boston: Addison-Wesley Professional, 2003). For information about how you can use domain-driven design with the strangler fig pattern, see Modernizing legacy Microsoft ASP.NET (ASMX) web services incrementally by using containers and Amazon API Gateway.

DR

See disaster recovery.

drift detection

Tracking deviations from a baselined configuration. For example, you can use AWS CloudFormation to detect drift in system resources, or you can use AWS Control Tower to detect changes in your landing zone that might affect compliance with governance requirements.

DVSM

See development value stream mapping.

E

EDA

See exploratory data analysis.

edge computing

The technology that increases the computing power for smart devices at the edges of an IoT network. When compared with cloud computing, edge computing can reduce communication latency and improve response time.

encryption

A computing process that transforms plaintext data, which is human-readable, into ciphertext.

encryption key

A cryptographic string of randomized bits that is generated by an encryption algorithm. Keys can vary in length, and each key is designed to be unpredictable and unique.

endianness

The order in which bytes are stored in computer memory. Big-endian systems store the most significant byte first. Little-endian systems store the least significant byte first.

endpoint

See service endpoint.

endpoint service

A service that you can host in a virtual private cloud (VPC) to share with other users. You can create an endpoint service with AWS PrivateLink and grant permissions to other AWS accounts or to AWS Identity and Access Management (IAM) principals. These accounts or principals

can connect to your endpoint service privately by creating interface VPC endpoints. For more information, see Create an endpoint service in the Amazon Virtual Private Cloud (Amazon VPC) documentation.

envelope encryption

The process of encrypting an encryption key with another encryption key. For more information, see Envelope encryption in the AWS Key Management Service (AWS KMS) documentation.

environment

An instance of a running application. The following are common types of environments in cloud computing:

- development environment – An instance of a running application that is available only to the core team responsible for maintaining the application. Development environments are used to test changes before promoting them to upper environments. This type of environment is sometimes referred to as a *test environment*.
- lower environments – All development environments for an application, such as those used for initial builds and tests.
- production environment – An instance of a running application that end users can access. In a CI/CD pipeline, the production environment is the last deployment environment.
- upper environments – All environments that can be accessed by users other than the core development team. This can include a production environment, preproduction environments, and environments for user acceptance testing.

epic

In agile methodologies, functional categories that help organize and prioritize your work. Epics provide a high-level description of requirements and implementation tasks. For example, AWS CAF security epics include identity and access management, detective controls, infrastructure security, data protection, and incident response. For more information about epics in the AWS migration strategy, see the program implementation guide.

exploratory data analysis (EDA)

The process of analyzing a dataset to understand its main characteristics. You collect or aggregate data and then perform initial investigations to find patterns, detect anomalies, and check assumptions. EDA is performed by calculating summary statistics and creating data visualizations.

F

fact table

The central table in a star schema. It stores quantitative data about business operations. Typically, a fact table contains two types of columns: those that contain measures and those that contain a foreign key to a dimension table.

fail fast

A philosophy that uses frequent and incremental testing to reduce the development lifecycle. It is a critical part of an agile approach.

fault isolation boundary

In the AWS Cloud, a boundary such as an Availability Zone, AWS Region, control plane, or data plane that limits the effect of a failure and helps improve the resilience of workloads. For more information, see AWS Fault Isolation Boundaries.

feature branch

See branch.

features

The input data that you use to make a prediction. For example, in a manufacturing context, features could be images that are periodically captured from the manufacturing line.

feature importance

How significant a feature is for a model's predictions. This is usually expressed as a numerical score that can be calculated through various techniques, such as Shapley Additive Explanations (SHAP) and integrated gradients. For more information, see Machine learning model interpretability with :AWS.

feature transformation

To optimize data for the ML process, including enriching data with additional sources, scaling values, or extracting multiple sets of information from a single data field. This enables the ML model to benefit from the data. For example, if you break down the "2021-05-27 00:15:37" date into "2021", "May", "Thu", and "15", you can help the learning algorithm learn nuanced patterns associated with different data components.

FGAC

See fine-grained access control.

fine-grained access control (FGAC)

The use of multiple conditions to allow or deny an access request.

flash-cut migration

A database migration method that uses continuous data replication through change data capture to migrate data in the shortest time possible, instead of using a phased approach. The objective is to keep downtime to a minimum.

G

geo blocking

See geographic restrictions.

geographic restrictions (geo blocking)

In Amazon CloudFront, an option to prevent users in specific countries from accessing content distributions. You can use an allow list or block list to specify approved and banned countries. For more information, see Restricting the geographic distribution of your content in the CloudFront documentation.

Gitflow workflow

An approach in which lower and upper environments use different branches in a source code repository. The Gitflow workflow is considered legacy, and the trunk-based workflow is the modern, preferred approach.

greenfield strategy

The absence of existing infrastructure in a new environment. When adopting a greenfield strategy for a system architecture, you can select all new technologies without the restriction of compatibility with existing infrastructure, also known as brownfield. If you are expanding the existing infrastructure, you might blend brownfield and greenfield strategies.

guardrail

A high-level rule that helps govern resources, policies, and compliance across organizational units (OUs). *Preventive guardrails* enforce policies to ensure alignment to compliance standards. They are implemented by using service control policies and IAM permissions boundaries. *Detective guardrails* detect policy violations and compliance issues, and generate alerts

for remediation. They are implemented by using AWS Config, AWS Security Hub, Amazon GuardDuty, AWS Trusted Advisor, Amazon Inspector, and custom AWS Lambda checks.

H

HA

See high availability.

heterogeneous database migration

Migrating your source database to a target database that uses a different database engine (for example, Oracle to Amazon Aurora). Heterogeneous migration is typically part of a re-architecting effort, and converting the schema can be a complex task. AWS provides AWS SCT that helps with schema conversions.

high availability (HA)

The ability of a workload to operate continuously, without intervention, in the event of challenges or disasters. HA systems are designed to automatically fail over, consistently deliver high-quality performance, and handle different loads and failures with minimal performance impact.

historian modernization

An approach used to modernize and upgrade operational technology (OT) systems to better serve the needs of the manufacturing industry. A *historian* is a type of database that is used to collect and store data from various sources in a factory.

homogeneous database migration

Migrating your source database to a target database that shares the same database engine (for example, Microsoft SQL Server to Amazon RDS for SQL Server). Homogeneous migration is typically part of a rehosting or replatforming effort. You can use native database utilities to migrate the schema.

hot data

Data that is frequently accessed, such as real-time data or recent translational data. This data typically requires a high-performance storage tier or class to provide fast query responses.

hotfix

An urgent fix for a critical issue in a production environment. Due to its urgency, a hotfix is usually made outside of the typical DevOps release workflow.

hypercare period

Immediately following cutover, the period of time when a migration team manages and monitors the migrated applications in the cloud in order to address any issues. Typically, this period is 1–4 days in length. At the end of the hypercare period, the migration team typically transfers responsibility for the applications to the cloud operations team.

I

IaC

See infrastructure as code.

identity-based policy

A policy attached to one or more IAM principals that defines their permissions within the AWS Cloud environment.

idle application

An application that has an average CPU and memory usage between 5 and 20 percent over a period of 90 days. In a migration project, it is common to retire these applications or retain them on premises.

IIoT

See industrial Internet of Things.

immutable infrastructure

A model that deploys new infrastructure for production workloads instead of updating, patching, or modifying the existing infrastructure. Immutable infrastructures are inherently more consistent, reliable, and predictable than mutable infrastructure. For more information, see the Deploy using immutable infrastructure best practice in the AWS Well-Architected Framework.

inbound (ingress) VPC

In an AWS multi-account architecture, a VPC that accepts, inspects, and routes network connections from outside an application. The AWS Security Reference Architecture recommends

setting up your Network account with inbound, outbound, and inspection VPCs to protect the two-way interface between your application and the broader internet.

incremental migration

A cutover strategy in which you migrate your application in small parts instead of performing a single, full cutover. For example, you might move only a few microservices or users to the new system initially. After you verify that everything is working properly, you can incrementally move additional microservices or users until you can decommission your legacy system. This strategy reduces the risks associated with large migrations.

infrastructure

All of the resources and assets contained within an application's environment.

infrastructure as code (IaC)

The process of provisioning and managing an application's infrastructure through a set of configuration files. IaC is designed to help you centralize infrastructure management, standardize resources, and scale quickly so that new environments are repeatable, reliable, and consistent.

industrial Internet of Things (IIoT)

The use of internet-connected sensors and devices in the industrial sectors, such as manufacturing, energy, automotive, healthcare, life sciences, and agriculture. For more information, see Building an industrial Internet of Things (IIoT) digital transformation strategy.

inspection VPC

In an AWS multi-account architecture, a centralized VPC that manages inspections of network traffic between VPCs (in the same or different AWS Regions), the internet, and on-premises networks. The AWS Security Reference Architecture recommends setting up your Network account with inbound, outbound, and inspection VPCs to protect the two-way interface between your application and the broader internet.

Internet of Things (IoT)

The network of connected physical objects with embedded sensors or processors that communicate with other devices and systems through the internet or over a local communication network. For more information, see What is IoT?

interpretability

A characteristic of a machine learning model that describes the degree to which a human can understand how the model's predictions depend on its inputs. For more information, see Machine learning model interpretability with AWS.

IoT

See Internet of Things.

IT information library (ITIL)

A set of best practices for delivering IT services and aligning these services with business requirements. ITIL provides the foundation for ITSM.

IT service management (ITSM)

Activities associated with designing, implementing, managing, and supporting IT services for an organization. For information about integrating cloud operations with ITSM tools, see the operations integration guide.

ITIL

See IT information library.

ITSM

See IT service management.

L

label-based access control (LBAC)

An implementation of mandatory access control (MAC) where the users and the data itself are each explicitly assigned a security label value. The intersection between the user security label and data security label determines which rows and columns can be seen by the user.

landing zone

A landing zone is a well-architected, multi-account AWS environment that is scalable and secure. This is a starting point from which your organizations can quickly launch and deploy workloads and applications with confidence in their security and infrastructure environment. For more information about landing zones, see Setting up a secure and scalable multi-account AWS environment.

large migration

A migration of 300 or more servers.

LBAC

See label-based access control.

least privilege

The security best practice of granting the minimum permissions required to perform a task. For more information, see Apply least-privilege permissions in the IAM documentation.

lift and shift

See 7 Rs.

little-endian system

A system that stores the least significant byte first. See also endianness.

lower environments

See environment.

M

machine learning (ML)

A type of artificial intelligence that uses algorithms and techniques for pattern recognition and learning. ML analyzes and learns from recorded data, such as Internet of Things (IoT) data, to generate a statistical model based on patterns. For more information, see Machine Learning.

main branch

See branch.

managed services

AWS services for which AWS operates the infrastructure layer, the operating system, and platforms, and you access the endpoints to store and retrieve data. Amazon Simple Storage Service (Amazon S3) and Amazon DynamoDB are examples of managed services. These are also known as *abstracted services*.

MAP

See Migration Acceleration Program.

mechanism

A complete process in which you create a tool, drive adoption of the tool, and then inspect the results in order to make adjustments. A mechanism is a cycle that reinforces and improves itself as it operates. For more information, see Building mechanisms in the AWS Well-Architected Framework.

member account

All AWS accounts other than the management account that are part of an organization in AWS Organizations. An account can be a member of only one organization at a time.

microservice

A small, independent service that communicates over well-defined APIs and is typically owned by small, self-contained teams. For example, an insurance system might include microservices that map to business capabilities, such as sales or marketing, or subdomains, such as purchasing, claims, or analytics. The benefits of microservices include agility, flexible scaling, easy deployment, reusable code, and resilience. For more information, see Integrating microservices by using AWS serverless services.

microservices architecture

An approach to building an application with independent components that run each application process as a microservice. These microservices communicate through a well-defined interface by using lightweight APIs. Each microservice in this architecture can be updated, deployed, and scaled to meet demand for specific functions of an application. For more information, see Implementing microservices on AWS.

Migration Acceleration Program (MAP)

An AWS program that provides consulting support, training, and services to help organizations build a strong operational foundation for moving to the cloud, and to help offset the initial cost of migrations. MAP includes a migration methodology for executing legacy migrations in a methodical way and a set of tools to automate and accelerate common migration scenarios.

migration at scale

The process of moving the majority of the application portfolio to the cloud in waves, with more applications moved at a faster rate in each wave. This phase uses the best practices and lessons learned from the earlier phases to implement a *migration factory* of teams, tools, and

processes to streamline the migration of workloads through automation and agile delivery. This is the third phase of the AWS migration strategy.

migration factory

Cross-functional teams that streamline the migration of workloads through automated, agile approaches. Migration factory teams typically include operations, business analysts and owners, migration engineers, developers, and DevOps professionals working in sprints. Between 20 and 50 percent of an enterprise application portfolio consists of repeated patterns that can be optimized by a factory approach. For more information, see the discussion of migration factories and the Cloud Migration Factory guide in this content set.

migration metadata

The information about the application and server that is needed to complete the migration. Each migration pattern requires a different set of migration metadata. Examples of migration metadata include the target subnet, security group, and AWS account.

migration pattern

A repeatable migration task that details the migration strategy, the migration destination, and the migration application or service used. Example: Rehost migration to Amazon EC2 with AWS Application Migration Service.

Migration Portfolio Assessment (MPA)

An online tool that provides information for validating the business case for migrating to the AWS Cloud. MPA provides detailed portfolio assessment (server right-sizing, pricing, TCO comparisons, migration cost analysis) as well as migration planning (application data analysis and data collection, application grouping, migration prioritization, and wave planning). The MPA tool (requires login) is available free of charge to all AWS consultants and APN Partner consultants.

Migration Readiness Assessment (MRA)

The process of gaining insights about an organization's cloud readiness status, identifying strengths and weaknesses, and building an action plan to close identified gaps, using the AWS CAF. For more information, see the migration readiness guide. MRA is the first phase of the AWS migration strategy.

migration strategy

The approach used to migrate a workload to the AWS Cloud. For more information, see the 7 Rs entry in this glossary and see Mobilize your organization to accelerate large-scale migrations.

ML

See machine learning.

MPA

See Migration Portfolio Assessment.

modernization

Transforming an outdated (legacy or monolithic) application and its infrastructure into an agile, elastic, and highly available system in the cloud to reduce costs, gain efficiencies, and take advantage of innovations. For more information, see Strategy for modernizing applications in the AWS Cloud.

modernization readiness assessment

An evaluation that helps determine the modernization readiness of an organization's applications; identifies benefits, risks, and dependencies; and determines how well the organization can support the future state of those applications. The outcome of the assessment is a blueprint of the target architecture, a roadmap that details development phases and milestones for the modernization process, and an action plan for addressing identified gaps. For more information, see Evaluating modernization readiness for applications in the AWS Cloud.

monolithic applications (monoliths)

Applications that run as a single service with tightly coupled processes. Monolithic applications have several drawbacks. If one application feature experiences a spike in demand, the entire architecture must be scaled. Adding or improving a monolithic application's features also becomes more complex when the code base grows. To address these issues, you can use a microservices architecture. For more information, see Decomposing monoliths into microservices.

multiclass classification

A process that helps generate predictions for multiple classes (predicting one of more than two outcomes). For example, an ML model might ask "Is this product a book, car, or phone?" or "Which product category is most interesting to this customer?"

mutable infrastructure

A model that updates and modifies the existing infrastructure for production workloads. For improved consistency, reliability, and predictability, the AWS Well-Architected Framework recommends the use of immutable infrastructure as a best practice.

O

OAC

See origin access control.

OAI

See origin access identity.

OCM

See organizational change management.

offline migration

A migration method in which the source workload is taken down during the migration process. This method involves extended downtime and is typically used for small, non-critical workloads.

OI

See operations integration.

OLA

See operational-level agreement.

online migration

A migration method in which the source workload is copied to the target system without being taken offline. Applications that are connected to the workload can continue to function during the migration. This method involves zero to minimal downtime and is typically used for critical production workloads.

operational-level agreement (OLA)

An agreement that clarifies what functional IT groups promise to deliver to each other, to support a service-level agreement (SLA).

operational readiness review (ORR)

A checklist of questions and associated best practices that help you understand, evaluate, prevent, or reduce the scope of incidents and possible failures. For more information, see Operational Readiness Reviews (ORR) in the AWS Well-Architected Framework.

operations integration (OI)

The process of modernizing operations in the cloud, which involves readiness planning, automation, and integration. For more information, see the operations integration guide.

organization trail

A trail that's created by AWS CloudTrail that logs all events for all AWS accounts in an organization in AWS Organizations. This trail is created in each AWS account that's part of the organization and tracks the activity in each account. For more information, see Creating a trail for an organization in the CloudTrail documentation.

organizational change management (OCM)

A framework for managing major, disruptive business transformations from a people, culture, and leadership perspective. OCM helps organizations prepare for, and transition to, new systems and strategies by accelerating change adoption, addressing transitional issues, and driving cultural and organizational changes. In the AWS migration strategy, this framework is called *people acceleration*, because of the speed of change required in cloud adoption projects. For more information, see the OCM guide.

origin access control (OAC)

In CloudFront, an enhanced option for restricting access to secure your Amazon Simple Storage Service (Amazon S3) content. OAC supports all S3 buckets in all AWS Regions, server-side encryption with AWS KMS (SSE-KMS), and dynamic PUT and DELETE requests to the S3 bucket.

origin access identity (OAI)

In CloudFront, an option for restricting access to secure your Amazon S3 content. When you use OAI, CloudFront creates a principal that Amazon S3 can authenticate with. Authenticated principals can access content in an S3 bucket only through a specific CloudFront distribution. See also OAC, which provides more granular and enhanced access control.

ORR

See operational readiness review.

outbound (egress) VPC

In an AWS multi-account architecture, a VPC that handles network connections that are initiated from within an application. The AWS Security Reference Architecture recommends setting up your Network account with inbound, outbound, and inspection VPCs to protect the two-way interface between your application and the broader internet.

P

permissions boundary

An IAM management policy that is attached to IAM principals to set the maximum permissions that the user or role can have. For more information, see Permissions boundaries in the IAM documentation.

personally identifiable information (PII)

Information that, when viewed directly or paired with other related data, can be used to reasonably infer the identity of an individual. Examples of PII include names, addresses, and contact information.

PII

See personally identifiable information.

playbook

A set of predefined steps that capture the work associated with migrations, such as delivering core operations functions in the cloud. A playbook can take the form of scripts, automated runbooks, or a summary of processes or steps required to operate your modernized environment.

policy

An object that can define permissions (see identity-based policy), specify access conditions (see resource-based policy), or define the maximum permissions for all accounts in an organization in AWS Organizations (see service control policy).

polyglot persistence

Independently choosing a microservice's data storage technology based on data access patterns and other requirements. If your microservices have the same data storage technology, they can encounter implementation challenges or experience poor performance. Microservices are more easily implemented and achieve better performance and scalability if they use the data store best adapted to their requirements. For more information, see Enabling data persistence in microservices.

portfolio assessment

A process of discovering, analyzing, and prioritizing the application portfolio in order to plan the migration. For more information, see Evaluating migration readiness.

predicate

A query condition that returns `true` or `false`, commonly located in a `WHERE` clause.

predicate pushdown

A database query optimization technique that filters the data in the query before transfer. This reduces the amount of data that must be retrieved and processed from the relational database, and it improves query performance.

preventative control

A security control that is designed to prevent an event from occurring. These controls are a first line of defense to help prevent unauthorized access or unwanted changes to your network. For more information, see Preventative controls in *Implementing security controls on AWS*.

principal

An entity in AWS that can perform actions and access resources. This entity is typically a root user for an AWS account, an IAM role, or a user. For more information, see *Principal* in Roles terms and concepts in the IAM documentation.

Privacy by Design

An approach in system engineering that takes privacy into account throughout the whole engineering process.

private hosted zones

A container that holds information about how you want Amazon Route 53 to respond to DNS queries for a domain and its subdomains within one or more VPCs. For more information, see Working with private hosted zones in the Route 53 documentation.

proactive control

A security control designed to prevent the deployment of noncompliant resources. These controls scan resources before they are provisioned. If the resource is not compliant with the control, then it isn't provisioned. For more information, see the Controls reference guide in the AWS Control Tower documentation and see Proactive controls in *Implementing security controls on AWS*.

production environment

See environment.

pseudonymization

The process of replacing personal identifiers in a dataset with placeholder values. Pseudonymization can help protect personal privacy. Pseudonymized data is still considered to be personal data.

Q

query plan

A series of steps, like instructions, that are used to access the data in a SQL relational database system.

query plan regression

When a database service optimizer chooses a less optimal plan than it did before a given change to the database environment. This can be caused by changes to statistics, constraints, environment settings, query parameter bindings, and updates to the database engine.

R

RACI matrix

See responsible, accountable, consulted, informed (RACI).

ransomware

A malicious software that is designed to block access to a computer system or data until a payment is made.

RASCI matrix

See responsible, accountable, consulted, informed (RACI).

RCAC

See row and column access control.

read replica

A copy of a database that's used for read-only purposes. You can route queries to the read replica to reduce the load on your primary database.

re-architect

See 7 Rs.

recovery point objective (RPO)

The maximum acceptable amount of time since the last data recovery point. This determines what is considered an acceptable loss of data between the last recovery point and the interruption of service.

recovery time objective (RTO)

The maximum acceptable delay between the interruption of service and restoration of service.

refactor

See 7 Rs.

Region

A collection of AWS resources in a geographic area. Each AWS Region is isolated and independent of the others to provide fault tolerance, stability, and resilience. For more information, see Managing AWS Regions in *AWS General Reference*.

regression

An ML technique that predicts a numeric value. For example, to solve the problem of "What price will this house sell for?" an ML model could use a linear regression model to predict a house's sale price based on known facts about the house (for example, the square footage).

rehost

See 7 Rs.

release

In a deployment process, the act of promoting changes to a production environment.

relocate

See 7 Rs.

replatform

See 7 Rs.

repurchase

See 7 Rs.

resource-based policy

A policy attached to a resource, such as an Amazon S3 bucket, an endpoint, or an encryption key. This type of policy specifies which principals are allowed access, supported actions, and any other conditions that must be met.

responsible, accountable, consulted, informed (RACI) matrix

A matrix that defines the roles and responsibilities for all parties involved in migration activities and cloud operations. The matrix name is derived from the responsibility types defined in the matrix: responsible (R), accountable (A), consulted (C), and informed (I). The support (S) type is optional. If you include support, the matrix is called a *RASCI matrix*, and if you exclude it, it's called a *RACI matrix*.

responsive control

A security control that is designed to drive remediation of adverse events or deviations from your security baseline. For more information, see Responsive controls in *Implementing security controls on AWS*.

retain

See 7 Rs.

retire

See 7 Rs.

rotation

The process of periodically updating a secret to make it more difficult for an attacker to access the credentials.

row and column access control (RCAC)

The use of basic, flexible SQL expressions that have defined access rules. RCAC consists of row permissions and column masks.

RPO

See recovery point objective.

RTO

See recovery time objective.

runbook

A set of manual or automated procedures required to perform a specific task. These are typically built to streamline repetitive operations or procedures with high error rates.

S

SAML 2.0

An open standard that many identity providers (IdPs) use. This feature enables federated single sign-on (SSO), so users can log into the AWS Management Console or call the AWS API operations without you having to create user in IAM for everyone in your organization. For more information about SAML 2.0-based federation, see About SAML 2.0-based federation in the IAM documentation.

SCP

See service control policy.

secret

In AWS Secrets Manager, confidential or restricted information, such as a password or user credentials, that you store in encrypted form. It consists of the secret value and its metadata. The secret value can be binary, a single string, or multiple strings. For more information, see Secret in the Secrets Manager documentation.

security control

A technical or administrative guardrail that prevents, detects, or reduces the ability of a threat actor to exploit a security vulnerability. There are four primary types of security controls: preventative, detective, responsive, and proactive.

security hardening

The process of reducing the attack surface to make it more resistant to attacks. This can include actions such as removing resources that are no longer needed, implementing the security best practice of granting least privilege, or deactivating unnecessary features in configuration files.

security information and event management (SIEM) system

Tools and services that combine security information management (SIM) and security event management (SEM) systems. A SIEM system collects, monitors, and analyzes data from servers,

networks, devices, and other sources to detect threats and security breaches, and to generate alerts.

security response automation

A predefined and programmed action that is designed to automatically respond to or remediate a security event. These automations serve as detective or responsive security controls that help you implement AWS security best practices. Examples of automated response actions include modifying a VPC security group, patching an Amazon EC2 instance, or rotating credentials.

server-side encryption

Encryption of data at its destination, by the AWS service that receives it.

service control policy (SCP)

A policy that provides centralized control over permissions for all accounts in an organization in AWS Organizations. SCPs define guardrails or set limits on actions that an administrator can delegate to users or roles. You can use SCPs as allow lists or deny lists, to specify which services or actions are permitted or prohibited. For more information, see Service control policies in the AWS Organizations documentation.

service endpoint

The URL of the entry point for an AWS service. You can use the endpoint to connect programmatically to the target service. For more information, see AWS service endpoints in *AWS General Reference*.

service-level agreement (SLA)

An agreement that clarifies what an IT team promises to deliver to their customers, such as service uptime and performance.

service-level indicator (SLI)

A measurement of a performance aspect of a service, such as its error rate, availability, or throughput.

service-level objective (SLO)

A target metric that represents the health of a service, as measured by a service-level indicator.

shared responsibility model

A model describing the responsibility you share with AWS for cloud security and compliance. AWS is responsible for security *of* the cloud, whereas you are responsible for security *in* the cloud. For more information, see Shared responsibility model.

SIEM

See security information and event management system.

single point of failure (SPOF)

A failure in a single, critical component of an application that can disrupt the system.

SLA

See service-level agreement.

SLI

See service-level indicator.

SLO

See service-level objective.

split-and-seed model

A pattern for scaling and accelerating modernization projects. As new features and product releases are defined, the core team splits up to create new product teams. This helps scale your organization's capabilities and services, improves developer productivity, and supports rapid innovation. For more information, see Phased approach to modernizing applications in the AWS Cloud.

SPOF

See single point of failure.

star schema

A database organizational structure that uses one large fact table to store transactional or measured data and uses one or more smaller dimensional tables to store data attributes. This structure is designed for use in a data warehouse or for business intelligence purposes.

strangler fig pattern

An approach to modernizing monolithic systems by incrementally rewriting and replacing system functionality until the legacy system can be decommissioned. This pattern uses the analogy of a fig vine that grows into an established tree and eventually overcomes and replaces its host. The pattern was introduced by Martin Fowler as a way to manage risk when rewriting monolithic systems. For an example of how to apply this pattern, see Modernizing legacy

Microsoft ASP.NET (ASMX) web services incrementally by using containers and Amazon API Gateway.

subnet

A range of IP addresses in your VPC. A subnet must reside in a single Availability Zone.

symmetric encryption

An encryption algorithm that uses the same key to encrypt and decrypt the data.

synthetic testing

Testing a system in a way that simulates user interactions to detect potential issues or to monitor performance. You can use Amazon CloudWatch Synthetics to create these tests.

T

tags

Key-value pairs that act as metadata for organizing your AWS resources. Tags can help you manage, identify, organize, search for, and filter resources. For more information, see Tagging your AWS resources.

target variable

The value that you are trying to predict in supervised ML. This is also referred to as an *outcome variable*. For example, in a manufacturing setting the target variable could be a product defect.

task list

A tool that is used to track progress through a runbook. A task list contains an overview of the runbook and a list of general tasks to be completed. For each general task, it includes the estimated amount of time required, the owner, and the progress.

test environment

See environment.

training

To provide data for your ML model to learn from. The training data must contain the correct answer. The learning algorithm finds patterns in the training data that map the input data attributes to the target (the answer that you want to predict). It outputs an ML model that

captures these patterns. You can then use the ML model to make predictions on new data for which you don't know the target.

transit gateway

A network transit hub that you can use to interconnect your VPCs and on-premises networks. For more information, see What is a transit gateway in the AWS Transit Gateway documentation.

trunk-based workflow

An approach in which developers build and test features locally in a feature branch and then merge those changes into the main branch. The main branch is then built to the development, preproduction, and production environments, sequentially.

trusted access

Granting permissions to a service that you specify to perform tasks in your organization in AWS Organizations and in its accounts on your behalf. The trusted service creates a service-linked role in each account, when that role is needed, to perform management tasks for you. For more information, see Using AWS Organizations with other AWS services in the AWS Organizations documentation.

tuning

To change aspects of your training process to improve the ML model's accuracy. For example, you can train the ML model by generating a labeling set, adding labels, and then repeating these steps several times under different settings to optimize the model.

two-pizza team

A small DevOps team that you can feed with two pizzas. A two-pizza team size ensures the best possible opportunity for collaboration in software development.

U

uncertainty

A concept that refers to imprecise, incomplete, or unknown information that can undermine the reliability of predictive ML models. There are two types of uncertainty: *Epistemic uncertainty* is caused by limited, incomplete data, whereas *aleatoric uncertainty* is caused by the noise and randomness inherent in the data. For more information, see the Quantifying uncertainty in deep learning systems guide.

undifferentiated tasks

Also known as *heavy lifting*, work that is necessary to create and operate an application but that doesn't provide direct value to the end user or provide competitive advantage. Examples of undifferentiated tasks include procurement, maintenance, and capacity planning.

upper environments

See environment.

V

vacuuming

A database maintenance operation that involves cleaning up after incremental updates to reclaim storage and improve performance.

version control

Processes and tools that track changes, such as changes to source code in a repository.

VPC peering

A connection between two VPCs that allows you to route traffic by using private IP addresses. For more information, see What is VPC peering in the Amazon VPC documentation.

vulnerability

A software or hardware flaw that compromises the security of the system.

W

warm cache

A buffer cache that contains current, relevant data that is frequently accessed. The database instance can read from the buffer cache, which is faster than reading from the main memory or disk.

warm data

Data that is infrequently accessed. When querying this kind of data, moderately slow queries are typically acceptable.

window function

A SQL function that performs a calculation on a group of rows that relate in some way to the current record. Window functions are useful for processing tasks, such as calculating a moving average or accessing the value of rows based on the relative position of the current row.

workload

A collection of resources and code that delivers business value, such as a customer-facing application or backend process.

workstream

Functional groups in a migration project that are responsible for a specific set of tasks. Each workstream is independent but supports the other workstreams in the project. For example, the portfolio workstream is responsible for prioritizing applications, wave planning, and collecting migration metadata. The portfolio workstream delivers these assets to the migration workstream, which then migrates the servers and applications.

WORM

See write once, read many.

WQF

See AWS Workload Qualification Framework.

write once, read many (WORM)

A storage model that writes data a single time and prevents the data from being deleted or modified. Authorized users can read the data as many times as needed, but they cannot change it. This data storage infrastructure is considered immutable.

Z

zero-day exploit

An attack, typically malware, that takes advantage of a zero-day vulnerability.

zero-day vulnerability

An unmitigated flaw or vulnerability in a production system. Threat actors can use this type of vulnerability to attack the system. Developers frequently become aware of the vulnerability as a result of the attack.

zombie application

An application that has an average CPU and memory usage below 5 percent. In a migration project, it is common to retire these applications.