User Guide

Microsoft SQL Server on Amazon EC2

Microsoft SQL Server on Amazon EC2: User Guide

Table of Contents

What is Microsoft SQL Server on Amazon EC2?

You can run Microsoft SQL Server on Amazon Elastic Compute Cloud (Amazon EC2). Microsoft SQL Server is a relational database management system (RDBMS) whose primary purpose is to store and retrieve data. SQL Server includes additional services, such as Analysis Services (SSAS), Reporting Services (SSRS), Integration Services (SSIS), and Machine Learning (ML). AWS provides a comprehensive set of services and tools to deploy Microsoft SQL Server on the reliable and secure AWS Cloud infrastructure. The benefits of running SQL Server on AWS include cost savings, scalability, high availability and disaster recovery, improved 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.

Amazon Elastic Compute Cloud (Amazon EC2) supports a self-managed SQL Server. That is, it gives you full control over the setup of the infrastructure and the database environment. Running SQL Server on Amazon EC2 is very similar to running SQL Server 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. You are responsible for data replication and recovery across your instances in the same or different AWS Regions. For more information, refer to the AWS Shared Responsibility Model.

Overview topics

- Options to run SQL Server on the AWS Cloud
- Microsoft SQL Server on Amazon EC2 concepts and terminology
- Microsoft SQL Server on Amazon EC2 features
- Microsoft SQL Server on Amazon EC2 pricing

Options to run SQL Server on the AWS Cloud

AWS provides the option to run Microsoft SQL Server in a cloud environment. For developers and database administrators, running SQL Server in the AWS Cloud is similar to running SQL Server databases in a data center. There are three primary options to run SQL Server on AWS:

- Microsoft SQL Server on Amazon Elastic Compute Cloud (Amazon EC2)
- Amazon RDS for Microsoft SQL Server
- Amazon RDS Custom for Microsoft SQL Server

Your application requirements, database features, functionality, growth capacity, and overall architecture complexity will determine which option to choose. Many AWS customers run multiple SQL Server database workloads across Amazon RDS and Amazon EC2. For more information on how to choose how to run SQL Server on the AWS Cloud, see Decision matrix on the AWS Prescriptive Guidance website.

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 may also want to modernize your SQL Server database running on Windows to run on a Linux operating system to save on cost and licenses.

Options

- Microsoft SQL Server on Amazon EC2
- Amazon RDS for Microsoft SQL Server
- Amazon RDS Custom for SQL Server

Microsoft SQL Server on Amazon EC2

When to choose Microsoft SQL Server on Amazon Elastic Compute Cloud (Amazon EC2):

- You want 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 more information, see Features not supported and features with limited support in the Amazon RDS documentation.
- You require 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 *RDS for Microsoft SQL Server User Guide*.
- Your database size and performance requirements exceed the current RDS for Microsoft SQL Server offerings. For more information, see Amazon RDS DB instance storage in the *Amazon RDS User Guide*.

- 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 RDS for Microsoft SQL Server license-included model.
- You want to achieve higher IOPS and storage capacity than the current limits. For more information, see Amazon RDS DB instance storage in the *Amazon RDS User Guide*.

Amazon RDS for Microsoft SQL Server

RDS for Microsoft SQL Server is a managed database service that simplifies the provisioning and management of SQL Server on AWS. With Amazon RDS, you can quickly deploy multiple versions and editions of SQL Server , with cost-efficient and resizeable compute capacity. You can provision Amazon RDS for SQL Server DB instances with either General Purpose SSD or Provisioned IOPS SSD storage. Provisioned IOPS SSD is optimized for I/O-intensive, transactional (OLTP) database workloads.

Amazon RDS manages database administration tasks, including provisioning, backups, software patching, monitoring, and hardware scaling. Amazon RDS 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, see Amazon RDS for Microsoft SQL Server.

When to choose RDS for Microsoft SQL Server:

- 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 want 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, up front investment.
- Your database size and IOPS requirements 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.

Amazon RDS Custom for SQL Server

Amazon RDS Custom for SQL Server is a managed database service for legacy, custom, and packaged applications that require access to the underlying operating system and database environment. Amazon RDS Custom for SQL Server automates setup, operation, and scaling of databases in the AWS Cloud while granting you access to the database and underlying operating system on Amazon EC2 to configure settings, install patches, and enable native features to meet the dependent application's requirements. For more information, see Working with RDS Custom for SQL Server in the *Amazon Relational Database Service User Guide*.

When to choose Amazon RDS Custom for SQL Server:

- You want the benefits of Amazon RDS, but your requirements include the need to customize the underlying operating system and database environment for legacy, custom, and packaged applications.
- You need administrative rights to the database and underlying operating system.
- You need to install custom database and OS patches and packages.
- You need to configure file systems to share files directly with their applications.

Microsoft SQL Server on Amazon EC2 concepts and terminology

The following concepts introduce you to the fundamental terminology used when working with Microsoft SQL Server on Amazon Elastic Compute Cloud (Amazon EC2) instances:

- Amazon Machine Images (AMIs)
- Backup
- Billing
- High availability and disaster recovery (HADR)
- Instance

- Instance types
- Launching
- Security
- Storage

Amazon Machine Images (AMIs)

SQL Server on Amazon EC2 instances are created from Amazon Machine Images (AMIs). AMIs are similar to templates. SQL Server on Amazon EC2 AMIs are pre-installed with an operating system, typically Microsoft Windows Server, and other software. Together, these determine the operating environment. You can select an AMI provided by AWS, create your own AMI, or select an AMI from the AWS Marketplace. To find a SQL Server on Amazon EC2 AMI, see the options under Find a Windows AMI in the *Amazon EC2 User Guide*.

Backup

Your backup and recovery design for SQL Server on Amazon EC2 is flexible, depending on your RTO and RPO requirements. AWS provides the ability to perform server-level backups using Windows Volume Shadow Copy Service (VSS)-enabled Amazon Elastic Block Store (Amazon EBS) snapshots and with AWS Backup. You can also perform database-level backups using native backup and restore procedures for SQL Server databases. Database-level backups can be stored on Amazon EBS, FSx for Windows File Server, or Amazon Simple Storage Service using AWS Storage Gateway. For more information about backing up SQL Server on Amazon EC2, see Backup and restore options for SQL Server on Amazon EC2 on the AWS Prescriptive Guidance website.

Billing

A SQL Server on Amazon EC2 instance is charged by the second, with a minimum of 1 minute. Applied rates are based on the type and size of the selected instance, the edition of SQL Server when using a license-included instance, along with the cost of any additional services, such as storage or networking. AWS provides a variety of instance families that are favorable to the performance requirements of SQL Server workloads.

You can rent an instance based on your unique CPU, memory, and storage throughput requirements. You can also stop or terminate an instance at any time to pause or stop billing for the instance. The main advantage of the On-Demand model is the ability to save on CAPEX when an instance is no longer required.

> **Warning**
>
> Any data on Amazon EC2 instance store volumes are lost if your instance is stopped or terminated. You'll still incur costs for EBS volumes when your instance is stopped. For more information, see Stop and start your instance in the *Amazon EC2 User Guide*.

High availability and disaster recovery (HADR)

You can take advantage of Windows Server Failover Cluster for high availability and disaster recovery (HADR) with SQL Server on Amazon EC2. SQL Server on Amazon EC2 supports both failover cluster instances (SQL FCIs) and Always On availability groups (AG). For more information see How do I create a SQL Server Always On availability group cluster in the AWS Cloud? in the AWS knowledge center.

Instance

A SQL Server on Amazon EC2 instance is a virtual (or bare metal) server that runs in the AWS Cloud and can be provisioned on demand. The subscriber rents the virtual server by the hour/minute/second, and can use it to deploy specific configurations of SQL Server. For more information about On-Demand instances, see On-Demand instances in the *Amazon EC2 User Guide*.

An Amazon EC2 Dedicated Hosts is a physical server with EC2 instance capacity that is fully dedicated to your use. Dedicated Hosts allow you to use your existing per-socket, per-core, or per-VM Microsoft SQL Server software licenses. For more information about Dedicated Hosts, see Dedicated Hosts in the *Amazon EC2 User Guide*.

Instance types

AWS provides various types of instances with different CPU, memory, storage, and networking configurations to support your application requirements. Each instance type is available in various sizes to address specific workload requirements. Instance types are grouped into families according to target application profiles, such as general purpose, compute-optimized, memory-optimized, and storage-optimized. The memory-optimized family of instances is a popular choice for SQL Server on Amazon EC2 because instances in this family have a high memory to CPU ratio for optimal performance. You can choose bare metal instances to support capabilities such as Always Encrypted with secure enclaves on Amazon EC2 bare metal instances. For more information about individual and families of instance types, see Amazon EC2 Instance Types in the AWS product pages.

Launching SQL Server on Amazon EC2

SQL Server on Amazon EC2 instances can be launched directly from the Amazon EC2 console, with AWS CloudFormation, by using AWS Tools for PowerShell, or by using the AWS CLI. For a guided deployment of Microsoft SQL Server, use AWS Launch Wizard.

Security

AWS supports all security standards and compliance certifications, such as PCI-DSS, HIPAA/HITECH, FedRAMP, GDPR, FIPS, FIPS 140-2, and more. These standards enable you to build a fully compliant application on Amazon EC2. AWS also supports all SQL Server security features such as Transparent Data Encryption (TDE) and Always Encrypted with Secure Enclaves (when using bare metal instances).

Security and compliance is a shared responsibility between you and AWS. This shared model helps to relieve your operational burden because AWS operates, manages, and controls the components from the host operating system and virtualization layer to the physical security of the facilities in which the service operates.

For SQL Server on Amazon EC2, you assume responsibility and management of the guest operating system, including updates and security patches, other associated application software, and the configuration of AWS provided security group firewalls.

For more information about the shared responsibility model, see Shared Responsibility Model.

Storage

AWS provides many storage options to host your database files. In addition to EBS volume types, you can attach volumes to SQL Server on Amazon EC2 instances using an Amazon FSx managed file system service, such as FSx for Windows File Server and Amazon FSx for NetApp ONTAP. Some instance types provide an Amazon EC2 instance store which provides temporary block level storage on NVMe solid state drive (SSD) disks that are physically attached to the host computer. For more information, see Best practices for deploying Microsoft SQL Server on Amazon EC2 on the *AWS Prescriptive Guidance* website.

Microsoft SQL Server on Amazon EC2 features

SQL Server on Amazon EC2 provides the following features:

- **Flexible licensing options** — When you use Amazon EC2 instances with the license included, you are using instances with fully-compliant Windows Server and SQL Server that are licensed

through AWS. Flexible BYOL options include default tenant EC2 for products that are eligible for Microsoft License Mobility through Software Assurance, as well as Amazon EC2 Dedicated Hosts and Amazon EC2 Dedicated Instances. You can use AWS License Manager to track the usage of software licenses and reduce the risk of non-compliance. For more information, see Licensing in the *Amazon Web Services and Microsoft Frequently Asked Questions*.

- **High performance block storage** — Amazon Elastic Block Store provides multiple options for high-performance block storage for Microsoft SQL Server. EC2 Instances using io2 Block Express give you the highest block storage performance with a single storage volume. Other SSD-backed Amazon EBS options include io2 volumes for business-critical applications and gp3 volumes for general purpose applications. Amazon EBS also offers crash-consistent snapshots, and enables application-consistent snapshots through Windows VSS (Volume Shadow Copy Services) to help protect your SQL Server deployments.
- **Fully-managed shared storage** — Amazon FSx for Windows File Server and Amazon FSx for NetApp ONTAP offer fully-managed shared storage for high-availability SQL Server failover cluster instances (FCI) workloads.
- **Windows-based services** — AWS Directory Service offers managed Microsoft Active Directory with identity and access management.
- **Scalable processors** — Intel Xeon Scalable Processors on AWS provide you with better data protection, faster processing of more data volumes, and increased service flexibility for Amazon EC2.
- **Migration programs** — AWS offers programs for migration for customers looking to migrate SQL Server workloads to AWS. AWS Migration Acceleration Program (MAP) for Windows provides services, best practices, and tools to help you save costs and accelerate your migration on AWS.
- **Windows workload optimization** — After you move your SQL Server workloads to AWS, you can continue to optimize costs, usage, and licenses to suit your business requirements. With Cost Explorer Service, you can visualize, understand, and manage your AWS costs and usage over time. AWS Compute Optimizer recommends optimal AWS compute resources for your workloads so that you can reduce costs up to 25% by analyzing historical utilization data. AWS Trusted Advisor can check that your EC2 instances have the required amount of SQL Server licenses and that the EC2 instance vCPU count doesn't exceed what is permitted for the SQL Server edition. AWS Managed Services can help operate your cloud environment post-migration by analyzing alerts and responding to incidents, reducing operational overhead and risk. You can use AWS Systems Manager to automate operational tasks across your AWS resources and better manage your infrastructure at scale.

AWS can help you to modernize you Windows-based applications with AWS open source services if you want to reduce the high cost of commercial licensing. Options include running SQL Server database applications on Linux, moving workloads to Amazon Aurora, containerizing your Windows applications with Amazon EKS, going serverless with AWS Lambda, or taking advantage of micro-services based architecture.

For more features specific to Amazon EC2, see Features of Amazon EC2.

Microsoft SQL Server on Amazon EC2 pricing

For information about pricing for Amazon EC2, see the Amazon EC2 pricing page.

For information about creating a price estimate for Microsoft Windows Server and Microsoft SQL Server, see Tutorial: Using Windows Server and SQL Server on Amazon EC2 calculator in the *AWS Pricing Calculator User Guide*.

Set up Microsoft SQL Server on Amazon EC2

Describes the prerequisites, permissions, and configurations that you should consider when preparing to use Microsoft SQL Server on Amazon Elastic Compute Cloud (Amazon EC2) instances for your SQL Server workloads.

Topics for setting up SQL Server on Amazon EC2

- Prerequisites for using SQL Server on Amazon EC2
- Permissions required to use SQL Server on Amazon EC2

Prerequisites for using SQL Server on Amazon EC2

Complete the tasks in this section to start using SQL Server on Amazon EC2 instances for the first time:

1. Sign up for an AWS account
2. Create a user with administrative access
3. Create a key pair
4. Create a security group

Sign up for an AWS account

If you do not have an AWS account, complete the following steps to create one.

To sign up for an AWS account

1. Open https://portal.aws.amazon.com/billing/signup.
2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

   When you sign up for an AWS account, an *AWS account root user* is created. The root user has access to all AWS services and resources in the account. As a security best practice, assign administrative access to a user, and use only the root user to perform tasks that require root user access.

AWS sends you a confirmation email after the sign-up process is complete. At any time, you can view your current account activity and manage your account by going to https://aws.amazon.com/ and choosing **My Account**.

Create a user with administrative access

After you sign up for an AWS account, secure your AWS account root user, enable AWS IAM Identity Center, and create an administrative user so that you don't use the root user for everyday tasks.

Secure your AWS account root user

1. Sign in to the AWS Management Console as the account owner by choosing **Root user** and entering your AWS account email address. On the next page, enter your password.

   For help signing in by using root user, see Signing in as the root user in the *AWS Sign-In User Guide*.

2. Turn on multi-factor authentication (MFA) for your root user.

   For instructions, see Enable a virtual MFA device for your AWS account root user (console) in the *IAM User Guide*.

Create a user with administrative access

1. Enable IAM Identity Center.

   For instructions, see Enabling AWS IAM Identity Center in the *AWS IAM Identity Center User Guide*.

2. In IAM Identity Center, grant administrative access to a user.

   For a tutorial about using the IAM Identity Center directory as your identity source, see Configure user access with the default IAM Identity Center directory in the *AWS IAM Identity Center User Guide*.

Sign in as the user with administrative access

- To sign in with your IAM Identity Center user, use the sign-in URL that was sent to your email address when you created the IAM Identity Center user.

For help signing in using an IAM Identity Center user, see Signing in to the AWS access portal in the *AWS Sign-In User Guide*.

Assign access to additional users

1. In IAM Identity Center, create a permission set that follows the best practice of applying least-privilege permissions.

   For instructions, see Create a permission set in the *AWS IAM Identity Center User Guide*.

2. Assign users to a group, and then assign single sign-on access to the group.

   For instructions, see Add groups in the *AWS IAM Identity Center User Guide*.

Create a key pair

AWS uses public-key cryptography to secure the login information for your instance. You specify the name of the key pair when you launch your instance, then provide the private key to obtain the administrator password for your Windows instance so you can log in using RDP.

If you haven't created a key pair already, you can create one by using the Amazon EC2 console. Note that if you plan to launch instances in multiple Regions, you'll need to create a key pair in each Region. For more information about Regions, see Regions and Zones in the *User Guide for Windows Instances*.

To create your key pair

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose **Key Pairs**.
3. Choose **Create key pair**.
4. For **Name**, enter a descriptive name for the key pair. Amazon EC2 associates the public key with the name that you specify as the key name. A key name can include up to 255 ASCII characters. It can’t include leading or trailing spaces.
5. For **Key pair type**, choose either **RSA** or **ED25519**. Note that **ED25519** keys are not supported for Windows instances.

6. For **Private key file format**, choose the format in which to save the private key. To save the private key in a format that can be used with OpenSSH, choose **pem**. To save the private key in a format that can be used with PuTTY, choose **ppk**.

   If you chose **ED25519** in the previous step, the **Private key file format** options do not appear, and the private key format defaults to **pem**.
7. Choose **Create key pair**.
8. The private key file is automatically downloaded by your browser. The base file name is the name you specified as the name of your key pair, and the file name extension is determined by the file format you chose. Save the private key file in a safe place.

   > **Important**
   >
   > This is the only chance for you to save the private key file.

For more information, see Amazon EC2 key pairs and Windows instances in the *User Guide for Windows Instances*.

Create a security group

Security groups act as a firewall for associated instances, controlling both inbound and outbound traffic at the instance level. You must add rules to a security group that enable you to connect to your instance from your IP address using RDP. You can also add rules that allow inbound and outbound HTTP and HTTPS access from anywhere.

Note that if you plan to launch instances in multiple Regions, you'll need to create a security group in each Region. For more information about Regions, see Regions and Zones in the *User Guide for Windows Instances*.

Prerequisites

You'll need the public IPv4 address of your local computer. The security group editor in the Amazon EC2 console can automatically detect the public IPv4 address for you. Alternatively, you can use the search phrase "what is my IP address" in an Internet browser, or use the following service: Check IP. If you are connecting through an Internet service provider (ISP) or from behind a firewall without a static IP address, you need to find out the range of IP addresses used by client computers.

You can create a custom security group using one of the following methods.

New Amazon EC2 console

To create a security group with least privilege

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. From the top navigation bar, select a Region for the security group. Security groups are specific to a Region, so you should select the same Region in which you created your key pair.
3. In the left navigation pane, choose **Security Groups**.
4. Choose **Create security group**.
5. For **Basic details**, do the following:
   a. Enter a name for the new security group and a description. Use a name that is easy for you to remember, such as your user name, followed by _SG_, plus the Region name. For example, *me*_SG_*uswest2*.
   b. In the **VPC** list, select your default VPC for the Region.
6. For **Inbound rules**, create rules that allow specific traffic to reach your instance. For example, use the following rules for a web server that accepts HTTP and HTTPS traffic. For more examples, see Security group rules for different use cases in the *User Guide for Windows Instances*.
   a. Choose **Add rule**. For **Type**, choose **HTTP**. For **Source**, choose **Anywhere**.
   b. Choose **Add rule**. For **Type**, choose **HTTPS**. For **Source**, choose **Anywhere**.
   c. Choose **Add rule**. For **Type**, choose **RDP**. For **Source**, do one of the following:
      - Choose **My IP** to automatically add the public IPv4 address of your local computer.
      - Choose **Custom** and specify the public IPv4 address of your computer or network in CIDR notation. To specify an individual IP address in CIDR notation, add the routing suffix `/32`, for example, `203.0.113.25/32`. If your company or your router allocates addresses from a range, specify the entire range, such as `203.0.113.0/24`.

> **Warning**
>
> For security reasons, do not choose **Anywhere** for **Source** with a rule for RDP. This would allow access to your instance from all IP addresses on the internet.

This is acceptable for a short time in a test environment, but it is unsafe for production environments.

7. For **Outbound rules**, keep the default rule, which allows all outbound traffic.
8. Choose **Create security group**.

Old Amazon EC2 console

To create a security group with least privilege

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the left navigation pane, choose **Security Groups**.
3. Choose **Create Security Group**.
4. Enter a name for the new security group and a description. Use a name that is easy for you to remember, such as your user name, followed by _SG_, plus the Region name. For example, *me*_SG_*uswest2*.
5. In the **VPC** list, select your default VPC for the Region.
6. On the **Inbound rules** tab, create the following rules (choose **Add rule** for each new rule):
   - Choose **HTTP** from the **Type** list, and make sure that **Source** is set to **Anywhere** (`0.0.0.0/0`).
   - Choose **HTTPS** from the **Type** list, and make sure that **Source** is set to **Anywhere** (`0.0.0.0/0`).
   - Choose **RDP** from the **Type** list. In the **Source** box, choose **My IP** to automatically populate the field with the public IPv4 address of your local computer. Alternatively, choose **Custom** and specify the public IPv4 address of your computer or network in CIDR notation. To specify an individual IP address in CIDR notation, add the routing suffix `/32`, for example, `203.0.113.25/32`. If your company allocates addresses from a range, specify the entire range, such as `203.0.113.0/24`.

   Warning

   For security reasons, do not allow RDP access from all IP addresses to your instance. This is acceptable for a short time in a test environment, but it is unsafe for production environments.

7. On the **Outbound rules** tab, keep the default rule, which allows all outbound traffic.
8. Choose **Create security group**.

Command line

To create a security group with least privilege

Use one of the following commands:

- create-security-group (AWS CLI)
- New-EC2SecurityGroup (AWS Tools for Windows PowerShell)

For more information, see Amazon EC2 security groups for Windows instances in the *Amazon EC2 User Guide*.

Permissions required to use SQL Server on Amazon EC2

For information about the permissions required to create or modify Amazon EC2 resources, or to perform tasks using the Amazon EC2 API, see IAM policies for Amazon EC2 in the *User Guide for Windows Instances*.

Understand licensing options and considerations for Microsoft SQL Server on Amazon EC2

There are two ways in which you can license Microsoft SQL Server on Amazon Elastic Compute Cloud (Amazon EC2) on the AWS Cloud. You acquire your own existing SQL Server licenses, or those which are provided by AWS. The most cost-effective license strategy for your workload will depend on multiple factors. For more information on comparing the costs of SQL Server editions, see Compare SQL Server editions on the AWS Prescriptive Guidance website.

Topics

Licensing options

You can launch Amazon Elastic Compute Cloud (Amazon EC2) instances with Microsoft SQL Server licenses included from AWS, or you can bring your own SQL Server licenses for use on AWS. You can perform a license type conversion for SQL Server in certain configurations if your needs change. For the most license flexibility, you can import your VM into AWS. For more information, see Eligible license types for license type conversion in the *AWS License Manager User Guide*.

Licensing options topics

License-included

Windows Server with currently supported versions of Microsoft SQL Server AMIs are available from AWS in a variety of combinations. AWS provides these AMIs with SQL Server software and operating system updates already installed. When you purchase an Amazon EC2 instance with a Windows Server AMI, licensing costs and compliance are handled for you. For more information, see *Find a SQL Server license-included AMI*.

Amazon EC2 offers a variety of instance types and sizes that you can configure for your target workload. Amazon EC2 AMIs with Windows Server require no Client Access Licenses (CALs). They also include two Microsoft Remote Desktop Services licenses for administrative purposes.

For SQL Server license-included AMIs, use the installation and setup media included in `C:\SQLServerSetup` to perform in-place SQL Server version upgrades, make changes to the default installation, add new features, or install additional named instances.

BYOL

When you launch a SQL Server instance from an imported AMI, you can bring your existing licenses with the Bring Your Own License model (BYOL), and let AWS manage them to ensure compliance with licensing rules that you set. To import your own licensed image, you can use a service such as VM Import/Export or AWS Application Migration Service. After you import your licensed image, and it is available as a private AMI in your AWS account on the Amazon EC2 console, you can use the AWS License Manager service to create a license configuration.

After you create the license configuration, you must associate the AMI that contains your licensed operating system image with the configuration. Then, you must create a host resource group and associate it with the license configuration. After you associate your host resource group with the configuration, License Manager automatically manages your hosts when you launch instances into a host resource group, and ensures that you do not exceed your configured license count limits. For more information, see the Getting started section of the *License Manager User Guide*.

You can also bring your own SQL Server licenses with Active Software Assurance to default (shared) tenant Amazon EC2 through Microsoft License Mobility through Software Assurance. For information about how to sign up for Microsoft License Mobility, see License Mobility.

Licensing considerations

There are many considerations for cost effectively licensing your Microsoft SQL Server on Amazon Elastic Compute Cloud (Amazon EC2) workload. Your use case, and existing license agreements, will determine whether to bring your own license to AWS with the Bring Your Own License model (BYOL) or to use license included AMIs from AWS. The following topics should help determine which approach you might take. For more information, see Licensing - SQL Server on the *Amazon Web Services and Microsoft Frequently Asked Questions* page.

Licensing considerations topics

- Choose a SQL Server edition

- Purchase SQL Server from AWS
- Use BYOL for SQL Server on AWS
- Quantify the required SQL Server licenses for BYOL
- License Mobility with SQL Server
- Track BYOL license consumption
- SQL Server client access licenses (CALs)
- Licensing for passive failover

Choose a SQL Server edition

The edition of SQL Server that is used will determine the supported features your implementation will have available. For example, the edition determines the maximum compute capacity used by a single instance of the SQL Server Database Engine, and the high availability options you might implement. For a comparison of SQL Server editions and supported features, see Editions and supported features of SQL Server 2022 in the Microsoft documentation.

Purchase SQL Server from AWS

You can utilize Microsoft SQL Server licenses included from AWS. You can choose any of the following editions for your use on Amazon EC2 instances.

- SQL Server Web
- SQL Server Standard
- SQL Server Enterprise

> **Note**
>
> - SQL Server Express AMIs are available for use from AWS. This free edition of SQL Server doesn't incur additional charges as there is no licensing fee.
> - SQL Server Developer edition is eligible for use in non-production, development, and test workloads. Once downloaded from Microsoft, you can bring and install SQL Server Developer edition on Amazon EC2 instances in the AWS Cloud. Dedicated infrastructure is not required for SQL Server Developer edition. For more information, see https://www.microsoft.com/en-us/sql-server/sql-server-downloads.

Use BYOL for SQL Server on AWS

You can use BYOL licenses for SQL Server on AWS. The requirements differ depending on if the licenses have active Software Assurance.

SQL Server licenses with active Software Assurance

You can bring your SQL Server licenses with active Software Assurance to default (shared) tenant Amazon EC2 through License Mobility benefits. Microsoft requires that you complete and send a License Mobility verification form which can be downloaded here. For more information, see License Mobility.

SQL Server licenses without active Software Assurance

SQL Server licenses without Software Assurance can be deployed on Amazon Elastic Compute Cloud Dedicated Hosts if the licenses are purchased prior to 10/1/2019 or added as a true-up under an active Enterprise Enrollment that was effective prior to 10/1/2019. In these specific BYOL scenarios, the licenses can only be upgraded to versions that were available prior to 10/1/2019. For more information, see Dedicated Hosts in the *Amazon EC2 User Guide*, and the Amazon EC2 Dedicated Hosts FAQs.

Quantify the required SQL Server licenses for BYOL

If you are licensing SQL Server under Microsoft License Mobility through Software Assurance, the number of licenses required varies based on the instance type, version of SQL Server, and the Microsoft licensing model you choose. For assistance with virtual core licensing calculations under the Microsoft Product Terms based on the instance type, see SQL License Mobility.

If you are using Dedicated Hosts, Amazon EC2 provides you with the number of physical cores installed on the Dedicated Host. Using this information, you can calculate the number of SQL Server licenses that you need to bring in. For more information, see Amazon EC2 Dedicated Hosts Pricing and the SQL Server 2022 licensing guide.

License Mobility with SQL Server

SQL Server licenses with active Software Assurance are eligible for Microsoft License Mobility and can be deployed on default or dedicated tenant Amazon EC2. For more information on bringing SQL Server licenses with active Software Assurance to default tenant EC2, see Microsoft License Mobility.

It is also possible to bring SQL Server licenses without active Software Assurance to EC2 Dedicated Hosts. To be eligible, the licenses must be purchased prior to October 1, 2019 or added as a true-up under an active Enterprise Enrollment that was effective prior to October 1, 2019. For additional FAQs about Dedicated Hosts, see the Dedicated Hosts section of the *Amazon Web Services and Microsoft FAQ*.

Track BYOL license consumption

You can use AWS License Manager to manage your software licenses for SQL Server. With License Manager, you can create license configurations, take inventory of your license-consuming resources, associate licenses with resources, and track inventory and compliance. For more information, see What is AWS License Manager? in the *AWS License Manager User Guide*.

SQL Server client access licenses (CALs)

When you are using SQL Server on Amazon EC2, license included instances do not require client access licenses (CALs) for SQL Server. An unlimited number of end users can access SQL Server on a license-included instance.

When you bring your own SQL Server licenses to Amazon EC2 through Microsoft License Mobility or BYOL, you must continue to follow the licensing rules in place on-premises. If you purchased SQL Server under the Server/CAL model, you still require CALs to meet Microsoft licensing requirements, but these CALs would remain on-premises and enable end user access SQL Server running on AWS.

Licensing for passive failover

There are various factors to consider when licensing passive failover for SQL Server. The information in this section pertains only to the SQL Server licenses and not the Windows Server licenses. In all cases, you must license Windows Server.

Using instances that include the license for SQL Server

When you purchase SQL Server license included instances on EC2, you must license passive failover instances.

Bringing SQL Server licenses with active Software Assurance to default tenant Amazon EC2

When you bring SQL Server 2014 and later versions with Software Assurance to default tenant EC2, you must license the virtual cores (vCPUs) on the active instance. In return, Software

Assurance permits one passive instance (equal or lesser size) where SQL Server licensing is not required.

Bringing SQL Server to Amazon EC2 Dedicated Instances

SQL Server 2014 and later versions require Software Assurance for SQL Server passive failover benefits on dedicated infrastructure. When you bring SQL Server with Software Assurance, you must license the cores on the active instance/host and are permitted one passive instance/host (equal or lesser size) where SQL Server licensing is not required.

SQL Server 2008 - SQL Server 2012R2 are eligible for passive failover on an Amazon EC2 Dedicated Hosts infrastructure without active Software Assurance. In these scenarios, you will license the active instance/host, and it will be permitted one passive instance/host of equal or lesser size where SQL Server licensing is not required.

There are specific BYOL scenarios that do not require Microsoft License Mobility through Software Assurance. An Amazon EC2 Dedicated Hosts infrastructure is always required in these scenarios. To be eligible, the licenses must be purchased prior to October 1, 2019 or added as a true-up under an active Enterprise Enrollment that was effective prior to October 1, 2019. In these specific BYOL scenarios, the licenses can only be upgraded to versions that were available prior to October 1, 2019.

Find a SQL Server license-included AMI

This topic describes how you can find SQL Server license-included AMIs using the Amazon EC2 console, the AWS Tools for PowerShell, the AWS CLI, or by searching the AWS Marketplace. For SQL Server license-included AMIs, use the installation and setup media included in `C:\SQLServerSetup` to make changes to the default installation, add new features, or install additional named instances.

As you select a SQL Server license-included AMI, consider the following requirements you might have for the instances that you'll launch:

- The AWS Region
- The operating system
- The architecture: 64-bit (x86_64)
- The root device type: Amazon EBS-backed (EBS)
- The provider (for example, Amazon Web Services)
- Additional software (for example, SQL Server)

> **Note**
>
> To view changes to each release of the AWS Windows AMIs, including SQL Server updates, see the AWS Windows AMI version history in the *Amazon EC2 User Guide*.

Methods to find a SQL Server license-included AMI

AWS Marketplace

To view a list of SQL Server AMIs available from AWS in AWS Marketplace, see Windows AMIs.

Console

You can find SQL Server license-included AMIs using the Amazon EC2 console. You can select from the list of AMIs when you use the launch instance wizard to launch an instance, or you can search through all available AMIs using the **Images** page. AMI IDs are unique to each AWS Region.

To find a SQL Server license-included AMI using the launch instance wizard

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. From the navigation bar, select the Region in which to launch your instances. You can select any Region that's available to you, regardless of your location.
3. From the console dashboard, choose **Launch instances**.
4. Under **Application and OS Images (Amazon Machine Image)**, enter SQL in the search bar and choose **Enter**. You will be taken to the **AMIs** page, where you can browse and choose from AMIs with SQL Server included. You can choose from AMIs under the **Quickstart AMIs**, **My AMIs**, **AWS Marketplace AMIs**, and the **Community AMIs** tabs. You can filter by cost, operating system, and architecture.
5. To launch an instance from this AMI, select it and then choose **Launch instance**. For more information about launching an instance using the console, see Launch an instance using the new launch instance wizard. If you're not ready to launch the instance now, take note of the AMI ID for later.

To find a SQL Server AMI using the AMIs page

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. From the navigation bar, select the Region in which to launch your instances. You can select any Region that's available to you, regardless of your location.
3. In the navigation pane, choose **AMI Catalog**.
4. Enter SQL in the search bar and choose **Enter**. You can choose from SQL Server license-included AMIs under the **Quickstart AMIs**, **My AMIs**, **AWS Marketplace AMIs**, and the **Community AMIs** tabs. You can filter by cost, operating system, and architecture.
5. To launch an instance from this AMI, select it and then choose **Launch instance** . For more information about launching an instance using the console, see Launching your instance from an AMI. If you're not ready to launch the instance now, take note of the AMI ID for later.

PowerShell

You can use cmdlets for Amazon EC2 to list only the Windows AMIs that match your requirements. After locating an AMI that matches your requirements, take note of its ID so

that you can use it to launch instances. For more information, see Launch an Instance Using Windows PowerShell in the *AWS Tools for Windows PowerShell User Guide*.

To list the latest SQL Server license-included AMIs provided by Amazon, you can use the `Get-SSMLatestEC2Image` cmdlet. The following command lists the latest Windows AMIs with *SQL* in their image name:

```
Get-SSMLatestEC2Image -Path ami-windows-latest -ImageName *SQL*
```

To list SQL Server license-included AMIs using commands that match specific criteria, you can use the `Get-EC2Image` cmdlet in addition to filters. The following commands filter for AMIs owned by you, or Amazon, with *SQL* in their name:

```
$name_values = New-Object 'collections.generic.list[string]'
$name_values.add("*SQL*")
$filter_name = New-Object Amazon.EC2.Model.Filter -Property @{Name = "name"; Values
 = $name_values}
Get-EC2Image -Owner amazon, self -Filter $filter_name
```

For more information and examples, see Find an AMI Using Windows PowerShell in the *AWS Tools for Windows PowerShell User Guide*.

AWS CLI

You can use AWS CLI commands for Amazon EC2 to list only the SQL Server license-included AMIs that match your requirements. After locating an AMI that matches your requirements, take note of its ID so that you can use it to launch instances. For more information, see Launching an Instance Using the AWS CLI in the *AWS Command Line Interface User Guide*.

The describe-images command supports filtering parameters. For example, use the `--owners` parameter with `amazon` to display public AMIs owned by Amazon or `self` to list AMIs you own. You can specify multiple values for the `--owners` parameter as in the following example:

```
aws ec2 describe-images --owners self amazon
```

You can add the following filter to the previous command to display only SQL Server license-included AMIs:

```
--filters "Name=name,Values=*SQL*"
```

You can use the following filter with the command to display only AMIs backed by Amazon EBS:

```
--filters "Name=root-device-type,Values=ebs"
```

You can combine multiple filters together. For example, this command will list all AMIs owned by you or Amazon with *SQL* in the AMI name and the `--root-device-type` parameter as `ebs`:

```
aws ec2 describe-images --owners self amazon --filters "Name=name,Values=*SQL*"
 "Name=root-device-type,Values=ebs"
```

> **Note**
>
> Omitting the `--owners` flag from the describe-images command will return all images for which you have launch permissions, regardless of ownership.

Deploy SQL Server on Amazon EC2

To launch Microsoft SQL Server using Amazon Elastic Compute Cloud (Amazon EC2) instances with Windows Server, perform the following steps according to your use case.

New SQL environment deployments are classified under three categories:

- SQL Server standalone
- SQL Server Failover Cluster Instances (FCI)
- SQL Server Always On availability groups (AG)

Considerations

Before you launch SQL Server on your instance, consider the following:

- If you use an AWS provided AMI, you must initially manage SQL Server as the local administrator. For more information, see *Connect to SQL Server on Amazon EC2*.
- The built-in availability form of clustering in Windows Server is activated by a feature named Failover Clustering. This feature allows you to build a Windows Server Failover Cluster (WSFC) to use with an availability group or failover cluster instances (FCI).
- Always On is an umbrella term for the availability features in SQL Server, and the term covers both availability groups and FCIs. Always On isn't the name of the Always On availability group (AG) feature.
- The major difference between FCI and AG is that all FCIs require some sort of shared storage, even if it's provided through networking. The FCI's resources can be run and owned by one node at any given time. AG doesn't require that shared storage is also highly available. It's a best practice to have replicas that are local in one data center for high availability, and remote ones in other data centers for disaster recovery, each with separate storage.
- An availability group also has another component called the listener. The listener allows applications and end users to connect without needing to know which SQL Server instance is hosting the primary replica. Each availability group has its own listener.

Deployment options

Use one of the following options to deploy SQL Server on Amazon EC2.

Deploy SQL Server on Amazon EC2 with AWS Launch Wizard

AWS Launch Wizard is a service that guides you through the sizing, configuration, and deployment of enterprise applications following AWS Cloud best practices. AWS Launch Wizard for SQL Server supports both high availability and single instance deployments according to AWS and SQL Server best practices. For more information, see the AWS Launch Wizard for SQL Server User Guide.

Always On availability groups (AG)

Deploy your SQL Server Always On availability groups with primary and secondary replicas for database level protection. Each replica is hosted by a SQL Server instance with its own local storage.

Always On Failover Cluster Instances (SQL FCI)

Deploy SQL Server Always On using Failover Cluster Instances (FCI) for instance-level protection. A single SQL Server instance is installed across Windows Server Failover Clustering (WSFC) nodes to ensure high availability and storage sharing.

Launch Wizard uses Amazon FSx to provide the following shared storage options required for SQL FCI deployments:

- Amazon FSx for NetApp ONTAP using Microsoft iSCSI endpoint
- Amazon FSx for Windows File Server using SMB 3.0 continuously available Windows file share

For more information on how to deploy SQL Server with Launch Wizard, see Deploy an application with AWS Launch Wizard for SQL Server on Windows in the *AWS Launch Wizard User Guide*.

Deploy SQL Server standalone

For a SQL Server standalone deployment, you can use one of the license-include AMIs provided by AWS or by using your own licensed media. For a list of SQL Server AMIs provided by AWS, see Windows AMIs. For more information on licensing options, see Understand licensing options and considerations for Microsoft SQL Server on Amazon EC2.

Deploy SQL Server failover cluster instances (FCIs)

Failover cluster instances (FCIs) provide availability for the entire installation of SQL Server known as an instance. Everything that is included in the instance, such as databases, SQL Server Agent jobs, and linked servers, move to a different server when the underlying server fails.

You can use AWS Launch Wizard to deploy SQL Server FCIs in the AWS Cloud. Launch Wizard identifies the AWS resources to automatically provision the SQL Server databases based on your use case. For more information, see Get started with AWS Launch Wizard for SQL Server.

You can reference the following AWS blogs to manually deploy SQL Server FCIs:

- (Amazon FSx) Deploy a SQL Server FCI using SMB 3.0 Continuously Available File Shares (CAFS) as shared storage
- (Amazon FSx) Deploy a SQL Server FCI using Microsoft iSCSI Initiator as shared storage
- (Amazon EBS) Deploy SQL Server FCI using Amazon EBS Multi-Attach with Persistent Reservations (MAPR)

Deploy SQL Server Always On availability groups (AG)

Always on availability groups provide high availability and disaster recovery of user databases through data replication. Availability groups can also distribute read operations amongst member nodes.

You can use AWS Launch Wizard to deploy a SQL Server Always On availability group in the AWS Cloud. Launch Wizard identifies the AWS resources to automatically provision the SQL Server databases based on your use case. For more information, see Get started with AWS Launch Wizard for SQL Server.

To manually deploy a SQL Server Always On availability group, perform the following steps:

Prerequisites

Before you manually deploy a SQL Server Always On availability group, you must perform the following prerequisites.

- Launch two Amazon EC2 instances with Windows Server 2016 or later and SQL Server 2016 or later Enterprise edition across two Availability Zones within an Amazon VPC. If the deployment is for testing purposes only, you can consider using SQL Server Developer edition.
- Configure secondary Amazon EBS volumes to host SQL Server Master Data File (MDF), Log Data File (LDF), and SQL Backup files (.bak). For more information on the volume types that you can use, see Amazon EBS volume types in the *Amazon Elastic Compute Cloud User Guide*.
- Deploy the cluster nodes in private subnets. You can then use Remote Desktop Protocol (RDP) to connect from a jump server to the cluster node instances.

- Configure inbound security group rules and Windows firewall exceptions to allow the nodes to communicate in a restrictive environment.
- Open all necessary ports for Active Directory domain controllers so that the SQL nodes and witness can join the domain and authenticate against Active Directory. For more information, see Active Directory and Active Directory Domain Services Port Requirements in the Microsoft documentation.
- Join the nodes to the domain before you create the Windows failover cluster. Ensure that you are logged in with domain credentials before you create and configure the cluster.
- Run the SQL Database instances with an Active Directory service account.
- Create a SQL login with `sysadmin` permissions using Windows domain authentication. Consult with your database administrator for details. For more information, see Create a login using SSMS for SQL Server in the Microsoft documentation.
- Properly configure the SQL browser for SQL Server named instances.

Configure the secondary IPs for each cluster node elastic network interface

Two secondary IP addresses are required for each cluster node elastic network interface.

> **Note**
>
> If you do not plan to deploy a SQL Group Listener, add only one secondary IP address for each cluster node elastic network interface.

1. Navigate to the Amazon EC2 console and choose the AWS Region where you want to host your Always On cluster.
2. Choose **Instances** from the left navigation pane, and then select your Amazon EC2 cluster instance.
3. Choose the **Networking** tab.
4. Under **Network interfaces**, select the network interface and then choose **Actions** > **Manage IP addresses**.
5. Choose the network interface Id to open the expandable section, and then choose **Assign new IP address**. You can enter a specific IP address or keep the default entry as `Auto-assign`. Repeat this step to add a second new IP address.

6. Choose **Save** > **Confirm**.
7. Repeat steps 1 through 7 for the other Amazon EC2 instance that will be included in the cluster.

Create a two-node Windows cluster

Perform the following steps to create a two-node Windows cluster.

1. Connect to your Amazon EC2 instance with RDP, using a domain account with local administrator permissions on both nodes.
2. On the Windows **Start** menu, open **Control Panel**, and then choose **Network and Internet** > **Network and Sharing Center**.
3. Choose **Change adapter settings** from the left navigation pane.
4. Choose your network connection, and then choose **Change settings of this connection**.
5. Choose **Internet Protocol Version 4 TCP/IPV4), and then choose Properties.**
6. Choose **Advanced**.
7. Under the **DNS** tab, choose **Append primary and connection specific DNS suffixes.**
8. Choose **OK** > **OK** > **Close**.
9. Repeat steps 1 through 8 for the other Amazon EC2 instance to include in the cluster.
10. On each instance, install the cluster feature on the nodes from the Server Manager, or run the following Windows PowerShell command:

    ```
    Install-WindowsFeature -Name Failover-Clustering -IncludeManagementTools
    ```

11. Open the command line as an administrator and enter `cluadmin.msc` top open the Cluster Manager.
12. Open the context menu (right-click) for **Failover Cluster Manager**, and then choose **Create Cluster**.
13. Choose **Next** > **Browse**.
14. For **Enter the object names to select**, enter the cluster node hostnames, and then choose **OK**.
15. Choose **Next**. You can now choose whether to validate the cluster. We recommend that you perform a cluster validation. If the cluster does not pass validation Microsoft may be unable to provide technical support for your SQL cluster. Choose **Yes** or **No**, and then choose **Next**.
16. Enter a **Cluster Name**, and then choose **Next**.

17. Clear **Add all eligible storage to the cluster**, and then choose **Next**.
18. When the cluster creation is complete, choose **Finish**.

> **Note**
>
> Cluster logs and reports are located at `%systemroot%\cluster\reports`.

19. In the **Cluster Core Resources** section of Cluster Manager, expand the entry for your new cluster.
20. Open the context menu (right-click) for the first IP address entry, and then choose **Properties**. For **IP Address** , choose **Static IP Address**, and then enter one of the secondary IP addresses associated with the `eth0` elastic network interface. Choose **OK**. Repeat this step for the second IP address entry.
21. Open the context menu (right-click) for the cluster name, and then choose **Bring Online**.

> **Note**
>
> We recommend that you configure a File Share Witness (FSW) in addition to your cluster to act as a tie-breaker. You can also use Amazon FSx for Windows File Server with Microsoft SQL Server.

Create Always On availability groups

Perform the following steps to create Always On availability groups.

1. Open SQL Server Configuration Manager.
2. Open the context menu (right-click) for the SQL instance, and then choose **Properties**.
3. On the **AlwaysOn High Availability** tab, select **Enable AlwaysOn Availability Groups**, and then choose **Apply**.
4. Open the context menu (right-click) for the SQL instance, and then choose **Restart**.
5. Repeat steps 1 through 4 on the other cluster node to include in the cluster.
6. Open Microsoft SQL Server Management Studio (SSMS).
7. Log in to one of the SQL instances with your Windows authenticated login that has access to the SQL instance.

> **Note**
>
> We recommend that you use the same MDF and LDF directory file paths across the SQL instances.

8. Create a test database. Open the context menu (right-click) for **Databases**, and then choose **New Database**.

> **Note**
>
> Make sure that you use the **Full** recovery model on the **Options** page.

9. Enter a **Database name**, and then choose **OK**.
10. Open the context menu (right-click) for the new database name, choose **Tasks**, and then choose **Back Up** For **Backup type**, choose **Full**.
11. Choose **OK** > **OK**.
12. Open the context menu (right-click) for **Always On High Availability** and then choose **New Availability Group Wizard**.
13. Choose **Next**.
14. Enter an **Availability group name**, and then choose **Next**.
15. Select your database, and then choose **Next**.
16. A primary replica is already present in the Availability Replicas window. Choose **Add Replica** to create a secondary replica.
17. Enter a **Server name** for the secondary replica and then choose **Connect**.
18. Decide which **Availability Mode** you want to use for each replica, and then choose either **Synchronous commit** or **Asynchronous commit**.
19. Choose **Next**.
20. Choose your data synchronization preference, and then choose **Next**.
21. When the validation is successful, choose **Next**.

> **Note**
>
> You can safely ignore **Checking the listener configuration** because you will add it later.

22. Choose **Finish** > **Close**.

Add a SQL Group Listener

Perform the following steps to add a SQL Group Listener.

1. Open SQL Server Management Studio (SSMS) and expand **Always On High Availability, Availability Groups, <primary replica name>**.
2. Open the context menu (right-click) for **Availability Group Listeners** and then choose **Add Listener**. Enter a **DNS Name**.
3. Enter **Port** 1433.
4. Choose **Static IP** for **Network Mode**.
5. Choose **Add**.

   For the **IPv4 Address**, enter the second secondary IP address from one of the cluster node instances, and then choose **OK**. Repeat this step using the second secondary IP address from the other cluster node instance.
6. Choose **OK**.

> **Note**
>
> If you receive errors when you add a SQL Group Listener, you may be missing permissions. For troubleshooting see:
>
> - Troubleshooting AlwaysOn availability group listener creation in SQL Server 2012
> - Create Availability Group Listener Fails with Message 19471, 'The WSFC cluster could not bring the Network Name resource online'

Test failover

1. From SSMS, open the context menu (right-click) for the primary replica on the navigation menu, and then choose **Failover**.
2. Choose **Next** > **Next**.
3. Choose **Connect** > **Connect**.

4. Choose **Next**, and then choose **Finish**. The primary replica will become the secondary replica after failover.

Connect to Microsoft SQL Server on Amazon EC2

You can connect to your Microsoft SQL Server instance using one of the following tools.

Topics

SQL Server Management Studio (SSMS)

By default, only the built-in local administrator account can access a SQL Server instance launched from an AWS Windows AMI. You can use SQL Server Management Studio (SSMS) to add domain users so that they can access and manage SQL Server.

Perform the following steps to access a SQL Server instance on Amazon EC2 as a domain user.

1. Connect to your instance as a local administrator using Remote Desktop Protocol (RDP).
2. Open SQL Server Management Studio (SSMS).
3. For **Authentication**, choose **Windows Authentication** to log in with the built-in local administrator.
4. (Optional) Allow domain users to log in.
   a. Choose **Connect**.
   b. In Object Explorer, expand **Security**.
   c. Open the context menu (right-click) for **Logins** then select **New Login**.
   d. For **Login name**, select **Windows authentication**. Enter **Domain\username**, replacing **DomainName** with your domain NetBIOS name and **username** with your Active Directory user name.
   e. On the **Server roles** page, select the server roles that you want to grant to the Active Directory user.
   f. Select the **General** page, and then choose **OK**.
   g. Log out from the instance and then log in again as a domain user.
   h. Open SSMS. For **Authentication**, choose **Windows authentication** to log in with your domain user account.

i. Choose **Connect**.

SQL Server Configuration Manager

To connect to SQL Server using SQL Server Configuration Manager, see SQL Server Configuration Manager in the Microsoft documentation.

Evaluate if you can downgrade your Microsoft SQL Server edition

If you find that you aren't using Enterprise edition features, you can consider downgrading to Microsoft SQL Server Standard or Developer edition. By downgrading the edition, you can save on licensing costs.

> **Note**
>
> SQL Server Developer edition is only eligible for use in non-production, development, and test workloads.

Downgrade requirements

Your Microsoft SQL Server on Amazon EC2 must use the Bring Your Own License model (BYOL) and SQL Server Enterprise edition to be eligible for an in-place downgrade. If your instance meets this criteria, you should carefully evaluate which features are being used on your SQL Server instance before performing any changes. You can review the following SQL Server Enterprise edition features and instance level constraints to help evaluate your downgrade eligibility.

> **Tip**
>
> A script is available to help evaluate if you can downgrade your SQL Server edition. For more information, see Downgrade SQL Server Enterprise edition using AWS Systems Manager Document to reduce cost.

Confirm that your instance has **less** than the following resources available:

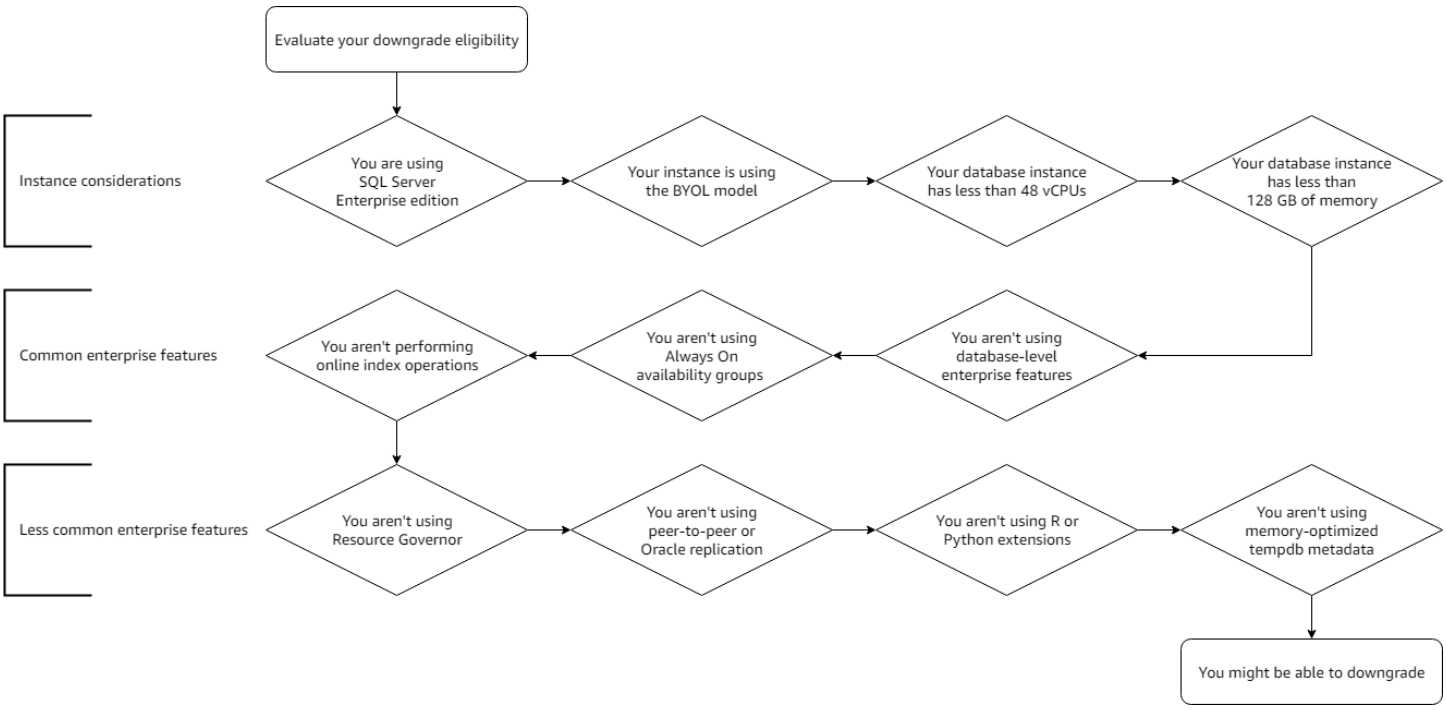
- 48 vCPUs
- 128 GiB of memory

If your instance is under-utilized for your workload, you can change the instance type or size your instance to meet these requirements. For more information, see Change the instance type .

Confirm that you aren't using any of the following SQL Server Enterprise edition features:

- Database-level enterprise features
- Always On availability groups
- Online index operations
- Resource Governor
- Peer-to-peer or Oracle replication
- R or Python extensions
- Memory-optimized tempdb metadata

The following diagram walks through an evaluation for some of the downgrade requirements:



If your workload doesn't utilize any of the previously listed features, you should continue to evaluate if you use any less common SQL Server Enterprise edition features. For more information about SQL Server Enterprise editions and supported features, see the Microsoft documentation for your SQL Server version:

- SQL Server 2022
- SQL Server 2019
- SQL Server 2017

- SQL Server 2016
- SQL Server 2014

Downgrade your SQL Server Enterprise edition

If you determine that you can downgrade your SQL Server Enterprise edition, you can follow this process to convert to SQL Server Standard or Developer edition. For information on how to automate for this process, see Downgrade SQL Server Enterprise edition using AWS Systems Manager Document to reduce cost.

> **Important**
>
> - This process will require downtime for your SQL Server instance. Your database will not be operational until the entire procedure has been completed successfully.
> - Only SQL Server instances using BYOL software support in-place downgrading. For more information, see Licensing options.

To downgrade your SQL Server Enterprise edition

1. Create a `Full` backup of all user and system databases. Ensure that the backup completes successfully before continuing.
2. Note your current SQL Server minor version, service pack, cumulative updates, and the General Distribution Release (GDR). For more information, see Determine which version and edition of SQL Server Database Engine is running in the Microsoft documentation.
3. Detach all user databases.
4. Stop the SQL Server Database Engine service and copy the log and system database data files —`master`, `model`, and `msdb`—to a local backup folder.
5. Uninstall SQL Server Enterprise edition including all components.
6. Reboot the instance.
7. Install SQL Server Standard or Developer edition according to your requirement.
8. Install the same service packs and cumulative updates that you had before the uninstall.
9. Stop the SQL Server Database Engine service.
10. Using the backups you made in step 4, restore the `master`, `model`, and `msdb` databases.

11. Start SQL Server service.
12. Attach the `mdf` and `ldf` user databases that were detached in step 3 to your SQL Server instance.
13. Confirm that your database is operating as expected.

Migrating an on-premises database to Amazon EC2

You can migrate your on-premises Microsoft SQL Server database to Amazon Elastic Compute Cloud (Amazon EC2). If you select a migration method and perform these steps, your on-premises database will reside on an Amazon EC2 instance running Windows Server.

On-premises migration methods

- Automated SQL Server backup and restore
- Manual SQL Server backup and restore
- Server rehost

Automated SQL Server backup and restore

You can use AWS Migration Hub Orchestrator to orchestrate and automate the migration of SQL Server databases to Amazon EC2 using automated native backup and restore. This feature of AWS Migration Hub uses 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. For more information, see Rehost SQL Server on Amazon EC2 in the *Migration Hub Orchestrator User Guide*.

Manual SQL Server backup and restore

You can use native backup files as a way to restore SQL Server databases without additional dependencies. You can back up and restore individual databases, or the entire database instance, from on premises to your EC2 instance.

Manual migration topics

- Prerequisites
- Step 1: Backing up your database
- Step 2: Uploading your database backup files
- Step 3: Downloading your database backup files
- Step 4: Restoring your database backup files

Prerequisites

You must meet the following prerequisites to migrate an on-premises database to Amazon EC2 using Amazon Simple Storage Service (Amazon S3):

- An active AWS account. For more information, see Set up Microsoft SQL Server on Amazon EC2.
- A source SQL Server database running on premises that you'd like to migrate.
- A destination EC2 instance running Windows Server with SQL Server installed on it. It is preferred that the destination instance's SQL Server version is the same or higher than the source SQL Server version running on premises. For more information on how to launch an instance, see Launch your instance in the *Amazon EC2 User Guide*.
- An Amazon Simple Storage Service (Amazon S3) bucket. For more information, see Creating, configuring, and working with Amazon S3 buckets in the *Amazon S3 User Guide*.
- Microsoft SQL Server Management Studio (SSMS) has been installed on the destination EC2 instance. For more information, see Download SQL Server Management Studio (SSMS) in the Microsoft documentation.

Step 1: Backing up your database

You will need to create a full backup of the database as well as back up the Transaction Log for the on-premises SQL Server to capture all of the necessary data for restoration. This procedure generates the backup files can restore your database with in an EC2 instance.

To back up an on-premises database

1. Create a full backup of your database. For more information about how to create a full backup of your database, see Create a Full Database Backup in the Microsoft documentation.
2. Create a backup of the Transaction Log. For more information about how to back up the transaction log, see Back up a Transaction Log in the Microsoft documentation.
3. Make a note of the backup file locations, because you will need to upload them to Amazon S3 in the next step.

Step 2: Uploading your database backup files

With the backup files created, you can now upload them to Amazon S3.

To upload your database backup files

1. Determine size of your backup files to see which upload methods are supported.
2. Use the file locations you noted previously to upload your backup files. For more information about how you can upload your database backup files to Amazon S3, see Uploading objects.

Step 3: Downloading your database backup files

Once the backup files have been uploaded to Amazon S3, you can restore them in an EC2 instance.

To download your backup files from Amazon S3 in the EC2 instance

1. Connect to your SQL Server instance and open SSMS. For more information, see Connect to Microsoft SQL Server on Amazon EC2.
2. Download the backup files in your Amazon EC2 instance running SQL Server. For more information about downloading your files from Amazon S3, see Downloading an object.
3. Make a note of the backup file locations, because you will need them to restore the database in the next step.

Step 4: Restoring your database backup files

After you download the backup files, you can connect to your instance and restore them using SSMS.

To restore your database

1. Connect to your instance and open SSMS.
2. Restore the full database backup using the backup files noted previously. For more information about restoring your database from the backup files, see Restore a Database Backup Using SSMS in the Microsoft documentation.
3. In the EC2 instance, validate that your database has been restored as expected.

Server rehost

You can choose to *rehost* (lift and shift) your entire SQL Server to Amazon EC2 instead of individual databases using AWS Application Migration Service or AWS Migration Hub Orchestrator.

Application Migration Service (MGN)

Application Migration Service (MGN) automates the migration of your servers and applications to the cloud during a cutover window. For more information on how you can rehost SQL Server using Application Migration Service, see Quick start guide in the *Application Migration Service User Guide*.

Migration Hub Orchestrator

Migration Hub Orchestrator orchestrates and further automates the rehost process for servers and applications. For more information on how you can rehost SQL Server using Migration Hub Orchestrator, see Rehost applications on Amazon EC2 in the *Migration Hub Orchestrator User Guide*.

Windows to Linux replatforming assistant for Microsoft SQL Server Databases

The Windows to Linux replatforming assistant for Microsoft SQL Server Databases service is a scripting tool. It helps you move existing Microsoft SQL Server workloads from a Windows to a Linux operating system. You can use the replatforming assistant with any Windows Server virtual machines (VMs) hosted in the cloud, or with on-premises environments running Microsoft SQL Server 2008 and later. The tool checks for common incompatibilities, exports databases from the Windows VM, and imports into an EC2 instance running Microsoft SQL Server 2017 on Ubuntu 16.04. The automated process results in a ready-to-use Linux VM configured with your selected SQL Server databases that can be used for experimenting and testing.

Contents

Concepts

The following terminology and concepts are central to your understanding and use of the Windows to Linux replatforming assistant for Microsoft SQL Server Databases.

Backup

A Microsoft SQL Server backup copies data or log records from a Microsoft SQL Server database or its transaction log to a backup device, such as a disk. For more information, see Backup Overview (Microsoft SQL Server).

Restore

A logical and meaningful sequence for restoring a set of Microsoft SQL Server backups. For more information, see Restore and recovery overview (SQL Server).

Replatform

A Microsoft SQL Server database can be replatformed from an EC2 Windows instance to an EC2 Linux instance running Microsoft SQL Server. It can also be replatformed to the VMware Cloud running Microsoft SQL Server Linux on AWS.

Related services

AWS Systems Manager (Systems Manager) gives you visibility and control of your infrastructure on AWS. The Windows to Linux replatforming assistant for Microsoft SQL Server Databases uses Systems Manager to move your Microsoft SQL databases to Microsoft SQL Server on EC2 Linux. For more information about Systems Manager, see the *AWS Systems Manager User Guide*.

How Windows to Linux replatforming assistant for Microsoft SQL Server works

Windows to Linux replatforming assistant for Microsoft SQL Server Databases allows you to migrate your Microsoft SQL Server databases from an on-premises environment or from an EC2 Windows instance to Microsoft SQL Server 2017 on EC2 Linux using backup and restore. For the destination EC2 Linux instance, you provide either the EC2 instance ID or the EC2 instance type with the subnet ID and EC2 Key Pair.

When you run the PowerShell script for the Windows to Linux replatforming assistant for Microsoft SQL Server Databases on the source Microsoft SQL Server databases, the Windows instance backs up the databases to an encrypted Amazon Simple Storage Service (S3) storage bucket. It then restores the backups to an existing Microsoft SQL Server on EC2 Linux instance, or it launches a new Microsoft SQL Server on EC2 Linux instance and restores the backups to the newly created instance. This process can be used to replatform your 2-tier databases running enterprise applications. It also enables you to replicate your database to Microsoft SQL Server on Linux to test the application while the source Microsoft SQL Server remains online. After testing, you can schedule application downtime and rerun the PowerShell backup script during your final cutover.

The entire replatforming process can also be automated and run unattended. You can run the Systems Manager SSM document AWSEC2-SQLServerDBRestore to import your existing database backup files into Microsoft SQL Server on EC2 Linux without using the PowerShell backup script.

Components

The Windows to Linux replatforming assistant for Microsoft SQL Server Databases script consists of two main components:

1. A PowerShell backup script, which backs up on-premises Microsoft SQL Server databases to an Amazon S3 storage bucket. It then invokes the SSM Automation document AWSEC2-SQLServerDBRestore to restore the backups to a Microsoft SQL Server on EC2 Linux instance.
2. An SSM Automation document named `AWSEC2-SQLServerDBRestore`, which restores database backups to Microsoft SQL Server on EC2 Linux. This automation restores Microsoft SQL Server database backups stored in Amazon S3 to Microsoft SQL Server 2017 running on an EC2 Linux instance. You can provide your own EC2 instance running Microsoft SQL Server 2017 Linux, or the automation launches and configures a new EC2 instance with Microsoft SQL Server 2017 on Ubuntu 16.04. The automation supports the restoration of full, differential, and transactional log backups, and accepts multiple database backup files. The automation automatically restores the most recent valid backup of each database in the files provided. For more information, see AWSEC2-SQLServerDBRestore.

Replatforming script prerequisites

This section covers the steps necessary to run the Windows to Linux replatforming script.

Contents

Prerequisites to run the replatforming script

In order to run the Windows to Linux replatforming assistant for Microsoft SQL Server Databases script, you must do the following:

1. **Install the AWS PowerShell module**

   To install the AWS PowerShell module, follow the steps listed in Installing the AWS Tools for PowerShell on Windows. We recommend that you use PowerShell 3.0 or later for the backup script to work properly.

2. **Install the Windows to Linux replatforming assistant PowerShell backup script**

   To run the Windows to Linux replatforming assistant, download the PowerShell backup script: MigrateSQLServerToEC2Linux.ps1.

3. **Add an AWS user profile to the AWS SDK store**

   To add and configure the AWS user profile, see the steps listed in Managing Profiles in the *AWS Tools for PowerShell User Guide*. Set the following IAM policy for your user profile.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::123456789012:role/DevTeam*"
    },
    {
      "Sid": "VisualEditor1",
      "Effect": "Allow",
      "Action": [
        "ec2:RebootInstances",
        "ssm:SendCommand",
        "ssm:GetAutomationExecution",
        "ec2:DescribeInstances",
        "ssm:ListCommands",
        "ec2:CreateTags",
        "s3:CreateBucket",
        "ec2:RunInstances",
        "s3:ListBucket",
        "ssm:GetCommandInvocation",
        "s3:PutEncryptionConfiguration",
        "ec2:DescribeImages",
        "s3:PutObject",
        "s3:GetObject",
        "ssm:StartAutomationExecution",
        "ssm:DescribeInstanceInformation",
        "s3:DeleteObject",
        "ssm:ListCommandInvocations",
        "s3:DeleteBucket",
        "ec2:DescribeInstanceStatus"
      ],
```

```
      "Resource": "*"
    }
  ]
}
```

4. **Create an IAM instance profile role**

   To create an IAM instance profile role in order to run Systems Manager on EC2 Linux, see the steps listed under Create an IAM instance profile for Systems Manager in the *AWS Systems Manager User Guide*.

Prerequisites for replatforming to an existing EC2 instance

To replatform to an existing instance running Microsoft SQL Server 2017 on Linux, you must:

1. Configure the EC2 instance with an AWS Identity and Access Management (IAM) instance profile and attach the `AmazonSSMManagedInstanceCore` managed policy.

   For information about creating an IAM instance profile for Systems Manager and attaching it to an instance, see the following topics in the *AWS Systems Manager User Guide*:

   - Create an IAM instance profile for Systems Manager
   - Attach an IAM instance profile to an Amazon EC2 instance
2. Verify that SSM Agent is installed on your EC2 instance. For more information, see Working with SSM Agent on EC2 instances for Windows Server in the *AWS Systems Manager User Guide*.
3. Verify that the EC2 instance has enough free disk space to download and restore the Microsoft SQL Server backups.

Run the Windows to Linux replatforming assistant for SQL Server script

This section contains the PowerShell parameter definitions and scripts for replatforming your databases. For more information about how to use PowerShell scripts, see PowerShell.

Topics

- Replatforming script examples
- Replatforming script parameters

Replatforming script examples

The following common scenarios and example PowerShell scripts demonstrate how to replatform your Microsoft SQL Server databases using Windows to Linux replatforming assistant for Microsoft SQL Server Databases.

> **Important**
>
> The Windows to Linux Replatforming Assistant for Microsoft SQL Server Databases resets the SQL Server server administrator (SA) user password on the target instance every time that it is run. After the replatform process is complete, you must set your own SA user password before you can connect to the target SQL Server instance.

Syntax

The Windows to Linux replatforming assistant for Microsoft SQL Server Databases script adheres to the syntax shown in the following example.

```
PS C:\> C:\MigrateSQLServerToEC2Linux.ps1 [[-SqlServerInstanceName] <String>] [[-
DBNames]<Object[]>] [-
MigrateAllDBs] [PathForBackup] <String> [-SetSourceDBModeReadOnly] [-
IamInstanceProfileName] <String>[-
AWSRegion] <String> [[-EC2InstanceId] <String>] [[-EC2InstanceType] <String>] [[-
EC2KeyPair] <String>] [[-
SubnetId] <String>] [[-AWSProfileName] <String>] [[-AWSProfileLocation] <String>] [-
GeneratePresignedUrls]
[<CommonParameters>]
```

Example 1: Move a database to an EC2 instance

The following example shows how to move a database named `AdventureDB` to an EC2 Microsoft SQL Server on Linux instance, with an instance ID of `i-024689abcdef`, from the Microsoft SQL Server Instance named `MSSQLSERVER`. The backup directory to be used is `D:\\Backup` and the AWS Region is `us-east-2`.

```
PS C:\> ./MigrateSQLServerToEC2Linux.ps1 - SQLServerInstanceName MSSQLSERVER -
EC2InstanceId i-
024689abcdef -DBNames AdventureDB -PathForBackup D:\\Backup -AWSRegion us-east-2 -
IamInstanceProfileName AmazonSSMManagedInstanceCore
```

Example 2: Move a database to an EC2 instance using the AWS credentials profile

The following example shows how to move the database in Example 1 using the AWS credentials profile: `DBMigration`.

```
PS C:\> ./MigrateSQLServerToEC2Linux.ps1 - SQLServerInstanceName MSSQLSERVER -
EC2InstanceId i-
024689abcdef -DBNames AdventureDB -PathForBackup D:\\Backup -AWSRegion us-east-2 -
AWSProfileName
DBMigration -IamInstanceProfileName AmazonSSMManagedInstanceCore
```

Example 3: Move a database to a new m5.large type instance

The following example shows how to create an `m5.large` type EC2 Linux instance in `subnet-abc127` using the Key Pair `customer-ec2-keypair` and then moving `AdventureDB` and `TestDB` to the new instance from the database used in Examples 1 and 2.

```
PS C:\> ./MigrateSQLServerToEC2Linux.ps1 -EC2InstanceType m5.large -SubnetId subnet-
abc127 -EC2KeyPair
customer-ec2-keypair -DBNames AdventureDB,TestDB -PathForBackup D:\\Backup -
AWSRegion us-east-2 -
AWSProfileName DBMigration -IamInstanceProfileName AmazonSSMManagedInstanceCore
```

Example 4: Move all databases to a new m5.large type instance

The following example shows how to create an `m5.large` type EC2 Linux instance in `subnet-abc127` using the Key Pair `customer-ec2-keypair` and then migrating all databases to the instance from databases used in Examples 1 and 2.

```
PS C:\> ./MigrateSQLServerToEC2Linux.ps1 -EC2InstanceType m5.large -SubnetId subnet-
abc127 -EC2KeyPair
customer-ec2-keypair -MigrateAllDBs -PathForBackup D:\\Backup -AWSRegion us-east-2 -
AWSProfileName
DBMigration -IamInstanceProfileName AmazonSSMManagedInstanceCore
```

Replatforming script parameters

The following parameters are used by the PowerShell script to replatform your Microsoft SQL Server databases.

`-SqlServerInstanceName`

The name of the Microsoft SQL Server instance to be backed up. If a value for `SqlServerInstanceName` is not provided, `$env:ComputerName` is used by default.

Type: String

Required: No

`-DBNames`

The names of the databases to be backed up and restored. Specify the names of the databases in a comma-separated list (for example, `adventureDB`,`universityDB`). Either the `DBNames` or `MigrateAllDBs` parameter is required.

Type: Object

Required: No

`-MigrateAllDBs`

This switch is disabled by default. If this switch is enabled, the automation migrates all databases except for the system databases (master, msdb, tempdb). Either the `DBNames` or `MigrateAllDBs` parameter is required.

Type: SwitchParameter

Required: No

`-PathForBackup`

The path where the full backup is stored.

Type: String

Required: Yes

`-SetSourceDBModeReadOnly`

This switch is disabled by default. If this switch is enabled, it makes the database read-only during migration.

Type: SwitchParameter

Required: No

-IamInstanceProfileName

Enter the AWS IAM instance role with permissions to run Systems Manager Automation on your behalf. See Getting Started with Automation in the *AWS Systems Manager User Guide*.

Type: String

Required: Yes

-AWSRegion

Enter the AWS Region where your Amazon S3 buckets are created to store database backups.

Type: String

Required: Yes

-EC2InstanceId

To restore Microsoft SQL Server databases to an existing EC2 instance running Microsoft SQL Server Linux, enter the instance ID of the instance. Make sure that the EC2 instance already has the AWS Systems Manager SSM Agent installed and running.

Type: String

Required: No

-EC2InstanceType

To restore Microsoft SQL Server databases to a new EC2 Linux instance, enter the instance type of the instance to be launched.

Type: String

Required: No

-EC2KeyPair

To restore Microsoft SQL Server databases to a new EC2 Linux instance, enter the name of the EC2 Key Pair to be used to access the instance. This parameter is recommended if you are creating a new EC2 Linux instance.

Type: String

Required: No

`-SubnetId`

This parameter is required when creating a new EC2 Linux instance. When creating a new EC2 Linux instance, if `SubnetId` is not provided, the AWS user default subnet is used to launch the EC2 Linux instance.

Type: String

Required: No

`-AWSProfileName`

The name of the AWS profile that the automation uses when connecting to AWS services. For more information on the required user permissions, see Getting Started with Automation in the *AWS Systems Manager User Guide*. If a profile is not entered, the automation uses your default AWS profile.

Type: String

Required: No

`-AWSProfileLocation`

The location of the AWS Profile if the AWS Profile is not stored in the default location.

Type: String

Required: No

`-GeneratePresignedUrls`

This parameter is only used when replatforming to non-EC2 instances, such as to VMware Cloud on AWS or on-premises VMs.

Type: SwitchParameter

Required: No

`<CommonParameters>`

This cmdlet supports the common parameters: `Verbose`, `Debug`, `ErrorAction`, `ErrorVariable`, `WarningAction`, `WarningVariable`, `OutBuffer`, `PipelineVariable`, and `OutVariable`. For more information, see About Common Parameters in the Microsoft PowerShell documentation.

Required: No

Best practices and recommendations for SQL Server clustering on Amazon EC2

You can configure Microsoft SQL Server on Amazon Elastic Compute Cloud (Amazon EC2) instances for high availability. SQL Server Always On availability groups offer high availability without the requirement for shared storage. The list of best practices in this topic, in addition to the prerequisites listed at Prerequisites, Restrictions, and Recommendations for Always On availability groups, can help you optimize operating a SQL Server Always On availability groups on AWS. The practices listed in this topic also offer a method to gather logs.

> **Note**
>
> When nodes are deployed in different Availability Zones, or in different subnets within the same Availability Zone, they should be treated as a multi-subnet cluster. Keep this in mind as you apply these best practices and when you address possible failure scenarios.

Contents

- Set possible owners
- Tune the failover thresholds
- Witness importance and Dynamic Quorum Architecture
- Troubleshoot

Assign IP addresses

Each cluster node should have one elastic network interface assigned that includes three private IP addresses on the subnet: a primary IP address, a cluster IP address, and an Availability Group IP address. The operating system (OS) should have the NIC configured for DHCP. It should not be set for a static IP address because the IP addresses for the cluster IP and Availability Group will be handled virtually in the Failover Cluster Manager. The NIC can be configured for a static IP as long as it is configured to only use the primary IP of **eth0**. If the other IPs are assigned to the NIC, it can cause network drops for the instance during failover events.

When the network drops because the IPs are incorrectly assigned, or when there is a failover event or network failure, it is not uncommon to see the following event log entries at the time of failure.

```
Isatap interface isatap.{9468661C-0AEB-41BD-BB8C-1F85981D5482} is no longer active.
```

```
Isatap interface isatap.{9468661C-0AEB-41BD-BB8C-1F85981D5482} with address
 fe80::5efe:169.254.1.105 has been brought up.
```

Because these messages seem to describe network issues, you could potentially mistake the cause of the outage or failure as a network error. However, these errors describe a symptom, rather than cause, of the failure. ISATAP is a tunneling technology that uses IPv6 over IPv4. When the IPv4 connection fails, the ISATAP adapter also fails. When the network issues are resolved, these entries should no longer appear in the event logs. Alternately, you can reduce network errors by safely disabling ISATAP with the following command.

```
netsh int ipv6 isatap set state disabled
```

When you run this command, the adapter is removed from Device Manager. This command should be run on all nodes. It does not impact the ability of the cluster to function. Instead, when the command has been run, ISATAP is no longer used. However, because this command might cause unknown impacts on other applications that use ISATAP, you should test it.

Cluster properties

To see the complete cluster configuration, run the following PowerShell command.

```
Get-Cluster | Format-List -Property *
```

Cluster quorum votes and 50/50 splits in a multi-site cluster

To learn how the cluster quorum works and what to expect if a failure occurs, see Understanding Cluster and Pool Quorum.

DNS registration

In Windows Server 2012, Failover Clustering, by default, attempts to register each DNS node under the cluster name. This is acceptable for applications that are aware the SQL target is configured for multi-site. However, when the client is not configured this way, it can result in timeouts, delays, and application errors due to attempts to connect to each individual node and failing on the inactive ones. To prevent these problems, the Cluster Resource parameter `RegisterAllProvidersIp` must be changed to **0**. For more information, see RegisterAllProvidersIP Setting and Multi-subnet Clustered SQL + RegisterAllProvidersIP + SharePoint 2013.

The `RegisterAllProvidersIp` can be modified with the following PowerShell script.

```
Import-Module FailoverClusters
$cluster = (Get-ClusterResource | where {($_.ResourceType -eq "Network Name") -and
 ($_.OwnerGroup -ne "Cluster Group")}).Name
Get-ClusterResource $cluster | Set-ClusterParameter RegisterAllProvidersIP 0
Get-ClusterResource $cluster |Set-ClusterParameter HostRecordTTL 300
Stop-ClusterResource $cluster
Start-ClusterResource $cluster
```

In addition to setting the Cluster Resource parameter to **0**, you must ensure that the cluster has permissions to modify the DNS entry for your cluster name.

1. Log in to the Domain Controller (DC) for the domain, or a server that hosts the forward lookup zone for the domain.
2. Launch the DNS Management Console and locate the A record for the cluster.
3. Choose or right-click the A record, and choose **Properties**.

4. Choose **Security**.
5. Choose **Add**.
6. Choose **Object Types...**, select the box for **Computers**, and choose **OK**.
7. Enter the name of the cluster resource object and choose **Check name** and **OK if resolve**.
8. Select the check box for **Full Control**.
9. Choose **OK**.

Elastic Network Adapters (ENAs)

AWS has identified known issues with some clustering workloads running on ENA driver version 1.2.3. We recommend upgrading to the latest version, and adjusting settings on the NIC in the operating system. For the latest versions, see Amazon ENA Driver Versions. The first setting, which applies to all systems, increases Receive Buffers, which you can do with the following example PowerShell command.

```
Set-NetAdapterAdvancedProperty -Name (Get-NetAdapter | Where-Object
 {$_.InterfaceDescription -like '*Elastic*'}).Name -DisplayName "Receive Buffers" -
DisplayValue 8192
```

For instances with more than 16 vCPUs, we recommend preventing RSS from running on CPU 0.

Run the following command.

```
Set-NetAdapterRss -name (Get-NetAdapter | Where-Object {$_.InterfaceDescription -like
 '*Elastic*'}).Name -Baseprocessorgroup 0 -BaseProcessorNumber 1
```

Multi-site clusters and EC2 instance placement

Each cluster is considered a multi-site cluster. The EC2 service does not share IP addresses virtually. Each node must be in a unique subnet. Though not required, we recommend that each node also be in a unique Availability Zone.

Instance type selection

The type of instance recommended for Windows Server Failover Clustering depends on the workload. For production workloads, we recommend instances that support Amazon Elastic Block

Store (Amazon EBS) optimization and enhanced networking. For more information, see EBS optimization and Enhanced networking in the *Amazon EC2 User Guide*.

Assign elastic network interfaces and IPs to the instance

Each node in an EC2 cluster should have only one attached elastic network interface. The network interface should have a minimum of two assigned private IP addresses. However, for workloads that use Availability Groups, such as SQL Always On, you must include an additional IP address for each Availability Group. The primary IP address is used for accessing and managing the server, the secondary IP address is used as the cluster IP address, and each additional IP address is assigned to Availability Groups, as needed.

Heartbeat network

Some Microsoft documentation recommends using a dedicated heartbeat network. However, this recommendation is not applicable to EC2. With EC2, while you can assign and use a second elastic network interface for the heartbeat network, it uses the same infrastructure and shares bandwidth with the primary network interface. Therefore, traffic within the infrastructure cannot be prioritized, and cannot benefit from a dedicated network interface.

Configure the network adapter in the OS

The NIC in the OS can keep using DHCP as long as the DNS servers that are being retrieved from the DHCP Options Set allow for the nodes to resolve each other. You can set the NIC to be configured statically. When completed, you then manually configure only the primary IP address for the elastic network interface. Failover Clustering manages and assigns additional IP addresses, as needed.

For certain instance types, you can increase the maximum transmission unit (MTU) on the network adapter to support Jumbo Frames. This configuration reduces fragmentation of packets wherever Jumbo Frames are supported. For more information, see Network maximum transmission unit (MTU) for your EC2 instance in the *Amazon Elastic Compute Cloud User Guide*.

IPv6

Microsoft does not recommend disabling IPv6 in a Windows Cluster. While Failover Clustering works in an IPv4-only environment, Microsoft tests clusters with IPv6 enabled. See Failover Clustering and IPv6 in Windows Server 2012 R2 for details.

Host record TTL for SQL Availability Group Listeners

Set the host record TTL to **300** seconds instead of the default 20 minutes (1200 seconds). For legacy client comparability, set `RegisterAllProvidersIP` to **0** for SQL Availability Group Listeners. This is not required in all environments. These settings are important because some legacy client applications cannot use `MultiSubnetFailover` in their connection strings. See HostRecordTTL Setting for more information. When you change these settings, the Cluster Resource must be restarted. The Cluster Group for the listener stops when the Cluster Resource is restarted, so it must be started. If you do not start the Cluster Group, the Availability Group remains offline in a `RESOLVING` state. The following are example PowerShell scripts for changing the TTL and `RegisterAllProvidersIP` settings.

```
Get-ClusterResource yourListenerName | Set-ClusterParameter RegisterAllProvidersIP 0
```

```
Get-ClusterResource yourListenerName|Set-ClusterParameter HostRecordTTL 300
```

```
Stop-ClusterResource yourListenerName
```

```
Start-ClusterResource yourListenerName
```

```
Start-ClusterGroup yourListenerGroupName
```

Logging

The default logging level for the cluster log is **3**. To increase the detail of log information, set the logging level to **5**. See Set-ClusterLog for more information about the PowerShell cmdlet.

```
Set-ClusterLog -Level 5
```

NetBIOS over TCP

In Windows Server 2012 R2, you can increase the speed of the failover process by disabling NetBIOS over TCP. This feature was removed from Windows Server 2016. You should test this procedure if you are using earlier operating systems in your environment. For more information,

see Speeding Up Failover Tips-n-Tricks. The following is an example PowerShell command to disable NetBIOS over TCP.

```
Get-ClusterResource "Cluster IP Address" | Set-ClusterParameter EnableNetBIOS 0
```

NetFT Virtual Adapter

For Windows Server versions earlier than 2016 and non-Hyper-V workloads, Microsoft recommends you enable the NetFT Virtual Adapter Performance Filter on the adapter in the OS. When you enable the NetFT Virtual Adapter, internal cluster traffic is routed directly to the NetFT Virtual Adapter. For more information, see NetFT Virtual Adapter Performance Filter. You can enable NetFT Virtual Adapter by selecting the check box in the NIC properties, or by using the following PowerShell command.

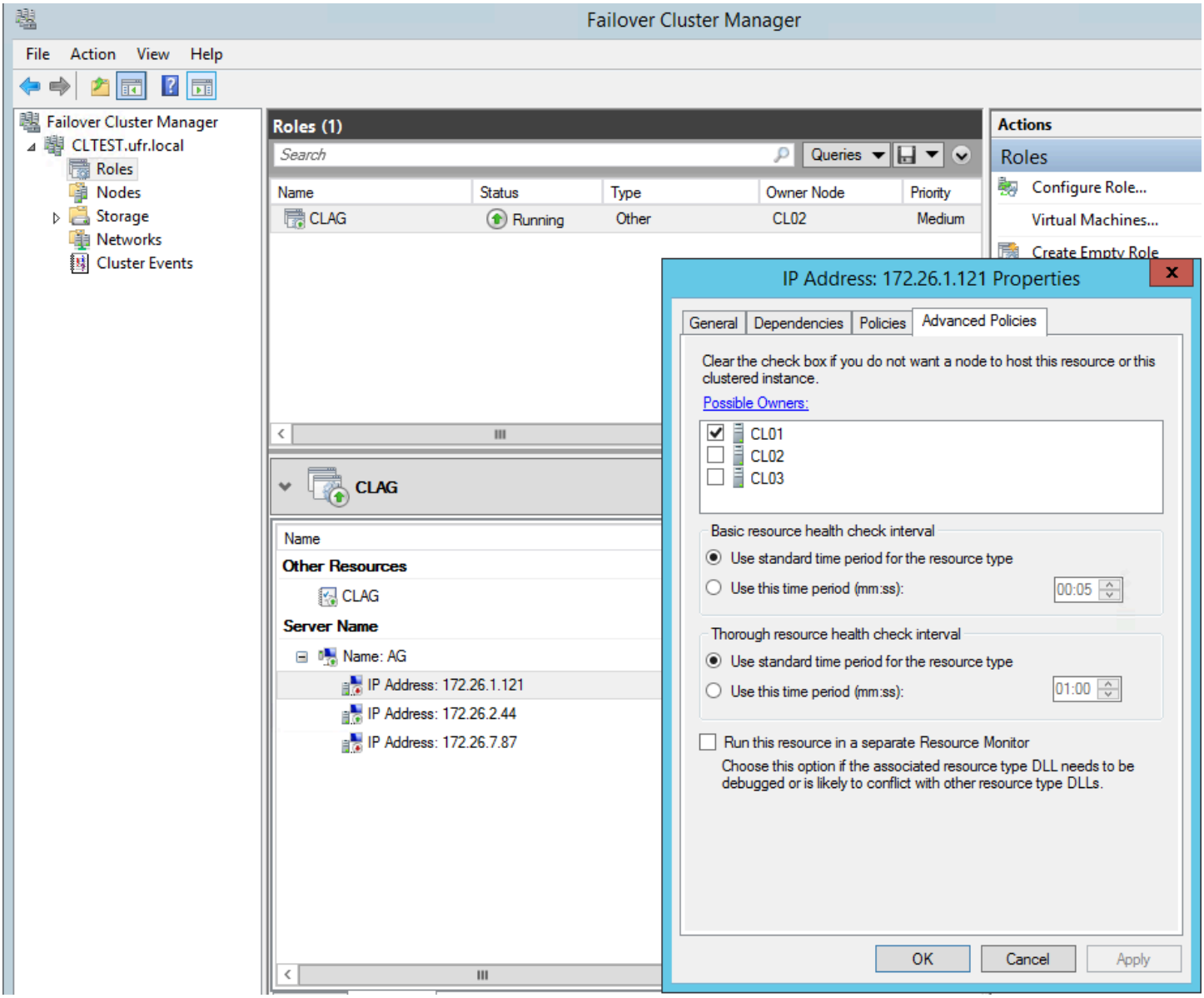
```
Get-NetAdapter | Set-NetAdapterBinding -ComponentID ms_netftflt -Enable $true
```

Set possible owners

The Failover Cluster Manager can be configured so that each IP address specified on the Cluster Core Resources and Availability Group resources can be brought online only on the node to which the IP belongs. When the Failover Cluster Manager is not configured for this and a failure occurs, there will be some delay in failover as the cluster attempts to bring up the IPs on nodes that do not recognize the address. For more information, see SQL Server Manages Preferred and Possible Owner Properties for AlwaysOn Availability Group/Role.

Each resource in a cluster has a setting for Possible Owners. This setting tells the cluster which nodes are permitted to “online” a resource. Each node is running on a unique subnet in a VPC. Because EC2 cannot share IPs between instances, the IP resources in the cluster can be brought online only by specific nodes. By default, each IP address that is added to the cluster as a resource has every node listed as a Possible Owner. This does not result in failures. However, during expected and unexpected failures, you can see errors in the logs about conflicting IPs and failures to bring IPs online. These errors can be ignored. If you set the Possible Owner property, you can eliminate these errors entirely, and also prevent down time while the services are moved to another node.

The following image shows an example of configuring an IP address so that it can only be brought online on the node to which the IP belongs:



Tune the failover thresholds

In Windows Server 2012 R2, the network thresholds for the failover heartbeat network default to high values. See Tuning Failover Cluster Network Thresholds for details. This potentially unreliable configuration, which applies to clusters with some distance between them, was addressed in Server 2016 with an increase in the number of heartbeats. It was discovered that clusters would fail over due to very brief transient network issues. The heartbeat network is maintained with UDP 3343, which is traditionally far less reliable than TCP and more prone to incomplete conversations. Although there are low-latency connections between AWS Availability Zones, there are still geographic separations with a number of "hops" separating resources. Within an Availability Zone,

there may be some distance between clusters unless the customer is using Placement Groups or Dedicated Hosts. As a result, there is a higher possibility for heartbeat failure with UDP than with TCP-based heartbeats.

The only time a cluster should fail over is when there is a legitimate outage, such as a service or node that experiences a hard failover, as opposed to a few UDP packets lost in transit. To ensure legitimate outages, we recommend that you adjust the thresholds to match, or even exceed, the settings for Server 2016 listed in Tuning Failover Cluster Network Thresholds. You can change the settings with the following PowerShell commands.

```
(get-cluster).SameSubnetThreshold = 10
```

```
(get-cluster).CrossSubnetThreshold = 20
```

When you set these values, unexpected failovers should be dramatically reduced. You can fine tune these settings by increasing the delays between heartbeats. However, we recommend that you send the heartbeats more frequently with greater thresholds. Setting these thresholds even higher ensures that failovers occur only for hard failover scenarios, with longer delays before failing over. You must decide how much down time is acceptable for your applications.

After increasing the `SameSubnetThreshold` or `CrossSubnetThreshold`, we recommend that you increase the `RouteHistoryLength` to double the higher of the two values. This ensures that there is sufficient logging for troubleshooting. You can set the `RouteHistoryLength` with the following PowerShell command.

```
(Get-Cluster).RouteHistoryLength = 20
```

Witness importance and Dynamic Quorum Architecture

There is a difference between Disk Witness and File Share Witness. Disk Witness keeps a backup of the cluster database while File Share Witness does not. Both add a vote to the cluster. You can use Disk Witness if you use iSCSI-based storage. For more about witness options, see File Share witness vs Disk witness for local clusters.

Troubleshoot

If you experience unexpected failovers, first make sure that you are not experiencing networking, service, or infrastructure issues.

1. Check that your nodes are not experiencing network-related issues.
2. Check driver updates. If you are using outdated drivers on your instance, you should update them. Updating your drivers might address bugs and stability issues that might be present in your currently installed version.
3. Check for any possible resource bottlenecks that could cause an instance to become unresponsive, such as CPU and disk I/O. If the node cannot service requests, it might appear to be down by the cluster service.

Security in Microsoft SQL Server on Amazon EC2

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security of the cloud and security in the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs. To learn about the compliance programs that apply to SQL Server on EC2, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations

For detailed information about how to configure Amazon EC2 to meet your security and compliance objectives, see Security in Amazon EC2 in the *User Guide for Windows Instances*.

Document history for the Microsoft SQL Server on Amazon EC2 User Guide

The following table describes the documentation releases for Microsoft SQL Server on Amazon EC2.

Change	Description	Date
Downgrade your SQL Server edition	Added a new section about downgrading your Microsoft SQL Server edition.	August 28, 2023
Migration	Added a new section about migrating to Microsoft SQL Server on Amazon EC2.	August 1, 2023
Initial release	Initial release of the Microsoft SQL Server on Amazon EC2 User Guide	August 18, 2022