

Domain 1: Plan and implement data platform resources

Prepare to maintain SQL Server-based databases on Azure

1. Introduction

<https://learn.microsoft.com/en-us/training/modules/prepare-to-maintain-sql-databases-azure/1-introduction>

Introduction

Completed

Understanding Azure Database Administrator responsibilities, and how they relate to other Microsoft Intelligent Data Platform roles, is important to performing the tasks and duties expected for this function.

In general, cloud services can be broken down into two sets of services: Infrastructure as a Service (IaaS) and Platform as a Service (PaaS).

IaaS services typically consist of virtual machines, storage, and virtual networking components, and are largely managed by the user in terms of patching and software. Alternatively, PaaS services have a larger percentage of management tasks, which are handled by the cloud provider.

Learning objectives

At the end of this module, you will be able to:

- Understand the role of Azure Database Administrator, and other data platform roles
- Describe the key differences between the SQL Server-based database options in Azure
- Describe other features for Azure SQL platforms available

2. Describe Microsoft Intelligent Data Platform roles

Describe Microsoft Intelligent Data Platform roles

Completed

The role of Azure Database Administrator is just one of the roles that can be filled by professionals working with the Microsoft Intelligent Data Platform services. There are five different role-based certifications offered by Microsoft for people whose main job responsibilities deal with cloud-focussed data.

Role	Definition
Azure Data Engineer	Design and implement the management, monitoring, security, and privacy of data using the full stack of Azure data services to satisfy business needs.
Azure Data Analyst	Enable businesses to maximize the value of their data assets by using Microsoft Power BI. As a subject matter expert, Data Analysts are responsible for designing and building scalable data models, cleaning and transforming data, and enabling advanced analytic capabilities that provide meaningful business value through easy-to-comprehend data visualizations.
Azure Data Scientist	Applies their knowledge of data science and machine learning to implement and run machine learning workloads on Azure; in particular, using Azure Machine Learning Service.
Azure Artificial Intelligence Engineer	Use Cognitive Services, Machine Learning, and Knowledge Mining to architect and implement Microsoft AI solutions involving natural language processing, speech, computer vision, bots, and agents.
Azure Database Administrator	Implements and manages the operational aspects of cloud-native and hybrid data platform solutions built on Microsoft Azure data services and Microsoft SQL Server. The Azure Database Administrator uses a variety of methods and tools to perform day-to-day operations, including applying knowledge of using T-SQL for administrative management purposes.

3. Understand SQL Server in an Azure virtual machine

Understand SQL Server in an Azure virtual machine

Completed

A SQL Server running in an Azure virtual machine (IaaS) is equivalent to an on-premises SQL Server. You'll notice that several features described for SQL Server on Azure virtual machine are applicable to all your on-premises SQL Servers.

Many applications require SQL Server running on a virtual machine. The reasons include:

- **General application support and incompatibility** - For applications requiring an older version of SQL Server for vendor support. In addition, some application services may have a requirement to be installed with the database instance in a manner that isn't compatible with a PaaS offering.
- **Use of other SQL Server Services** - In order to maximize licensing, many users choose to run SQL Server Analysis Services (SSAS), SQL Server Integration Services (SSIS), and/or SQL Server Reporting Services (SSRS) on the same machine as the database engine.

Versions of SQL Server available

Microsoft keeps images of all supported versions of SQL Server available in the Azure Marketplace. If you require an older version that is covered by an extended support contract, you'll need to install your own SQL Server binaries.

Backup solutions

In recent releases of SQL Server, Microsoft has introduced several features to support running SQL Server in an Azure virtual machine. We're going to focus on two key backup features:

- Back up to URL
- Azure Backup

The back up to URL option enables you to back up your databases to Azure Blob Storage service. Azure Backup for SQL Server Virtual Machines provides a comprehensive enterprise backup solution that automatically manages your backups across your entire infrastructure.

Deployment options

All resources in Azure are managed and deployed through a common provider known as Azure Resource Manager. While there are various methods to deploy Azure resources, they ultimately converge into JSON documents called Azure Resource Manager templates, which serve as one of the deployment options for Azure resources.

The key distinction between these methods is that Azure Resource Manager templates use a declarative deployment approach, which defines the desired structure and state of the resources to be deployed. In contrast, other methods are imperative, using procedural models to explicitly specify the steps to be executed. For large-scale deployments, the declarative approach is preferable and should be adopted.

Overview of Azure storage

Azure offers a fully redundant object-based storage model, and there are a few things to be aware of in designing and deploying Virtual Machines architecture. In terms of virtual machines, there are four types of storage you can use:

- Standard
- Standard SSD
- Premium SSD
- Ultra Disk

For production SQL Server data and transaction log files, you should only use Premium SSD storage and Ultra Disk. With premium storage, you see latencies in the range of 5-10 ms on a properly configured system. Alternatively, with Ultra Disk you may have sub millisecond latency but will likely see 1-2 ms workloads in the real world. You can use Standard storage for your database backups, as the performance is adequate for most backup and restore workloads.

High availability in Azure

The Azure platform is designed to be fault tolerant and provides quick recovery from service disruptions and transient errors. In fact, many organizations see higher levels of availability in single virtual machines deployments than they previously experienced in their on-premises environments. Microsoft guarantees uptime of at least 99.9% for single instance Azure virtual machine, when using Premium SSD or Ultra Disk for all disks.

Azure offers several features to support high availability including availability sets, availability zones, and load-balancing techniques that provide high availability by distributing incoming traffic among Virtual Machines.

SQL Server enabled by Azure Arc

Azure Arc extends Azure management capabilities to SQL Server instances running outside of Azure, whether they're on-premises, in other clouds, or at the edge. By enabling SQL Server with Azure Arc, you can bring the benefits of Azure's cloud management and governance to your existing SQL Server

deployments without needing to move them to Azure. This includes applying consistent policies, ensuring compliance, and using Azure services such as Azure Monitor and Azure Security Center for enhanced security and performance monitoring.

With Azure Arc, you can centrally manage and monitor your SQL Server instances through the Azure portal just like you would with native Azure services. This unified management experience simplifies operations and reduces the complexity of managing disparate environments. Additionally, Azure Arc enables advanced features like automated updates, backup and restore, and disaster recovery for your SQL Server instances, ensuring they're always up-to-date, secure, and resilient against failures. By connecting your SQL Server instances to Azure Arc, you can also take advantage of Azure's machine learning and artificial intelligence capabilities, enabling you to build and deploy intelligent applications that use your existing data.

To learn more about enabling SQL Server with Azure Arc, see [SQL Server enabled by Azure Arc](#).

4. Design Azure SQL Database for cloud-native applications

<https://learn.microsoft.com/en-us/training/modules/prepare-to-maintain-sql-databases-azure/4-design-azure-sql-database-for-cloud-native-applications>

Design Azure SQL Database for cloud-native applications

Completed

Azure SQL Database is a Platform as a Service (PaaS) that provides high scalability capabilities, and it can be a great solution for certain workloads, and requires minimal maintenance efforts.

Azure SQL Database is aimed at new application development as it gives developers a great deal of flexibility in building new application services, and granular deployment options at scale. SQL Database offers a low maintenance solution that can be a great option for certain workloads.

Purchasing model

SQL Database comes in two main purchasing models: vCore-based model and DTU-based model. Each purchasing model offers the following service tiers:

vCore-based

In this purchasing model, compute and storage resources are decoupled. It means you can scale storage and compute resources independently from one another. Here are listed the service tiers

available:

Service tier	Capability
General Purpose	This service tier is designed for less intensive operations, and offers budget oriented balanced compute and storage options. It provides both provisioned compute tier and serverless compute tier.
Business Critical	This service tier supports In-Memory OLTP, and built-in read-only replica. It also includes more memory per core, and uses local SSD storage, which is designed for performance-sensitive workloads.
Hyperscale	Hyperscale introduces horizontal scaling features that use advanced techniques to add compute nodes as the data sizes grow. It's only supported on single SQL database. Hyperscale allows you to scale storage and compute resources significantly over the limits available for the General Purpose and Business Critical service tiers.

DTU-based

In the DTU model, there are three service tiers available: Basic, Standard, and Premium. Compute and storage resources are dependent on the DTU level, and they provide a range of performance capabilities at a fixed storage limit, backup retention, and cost.

For example, if your database grows to the point it reaches the maximum storage limit, you would need to increase your DTU capacity, even if the compute utilization is low.

The scaling operation on SQL Database may incur in a brief connection interruption at the end of the scaling operation. There are two main changes that trigger this behavior:

- Once you initiate a scaling operation that requires an internal failover.
- When adding or removing databases to the elastic pool.

You can use a proper [retry logic](#) in your application to handle connection errors.

Note

Azure SQL Managed Instance doesn't support DTU-based purchasing model.

Serverless compute tier

Despite its name, the serverless compute tier does require you to have a server with your database. The serverless option can best be thought of as an autoscaling and autopause solution for SQL Database. It's effective for lowering the costs in development and testing environments. For example, you can set up a minimum and a maximum vCores configuration for your database, where it scales dynamically based on your workload.

The autopause delay feature allows you to define the period of time the database will be inactive before it's automatically paused. The autopause delay feature can be set up from 1 hour to seven

days. Alternatively, autopause delay feature can be disabled.

The resume operation is triggered when the next attempt to access the database occurs, and only storage charges are applicable when the database is paused.

The screenshot displays the 'Configure' page for an Azure SQL Database. The 'Service and compute tier' section is active, showing the 'General Purpose (Scalable compute and storage options)' service tier. Under 'Compute tier', the 'Serverless' option is selected and highlighted with a red box. The 'Compute Hardware' section shows 'Gen5' hardware configuration. Below this, the 'Max vCores' and 'Min vCores' sliders are visible, with 'Max vCores' set to 4 and 'Min vCores' set to 0.5. The 'Auto-pause delay' section is also highlighted with a red box, showing 'Enable auto-pause' checked, with 'Days' set to 0, 'Hours' set to 1, and 'Minutes' set to 0. The 'Data max size (GB)' slider is set to 32. The '9.6 GB LOG SPACE ALLOCATED' is displayed at the bottom. On the right, a 'Cost summary' box shows the 'Gen5 - General Purpose (GP S Gen5 4)' tier with a cost of \$41.6 per month. A 'NOTES' section at the bottom right explains that serverless databases are billed in vCores based on a combination of CPU and memory utilization.

Configure

Feedback

Service and compute tier

Select from the available tiers based on the needs of your workload. The vCore model provides a wide range of configuration controls and offers Hyperscale and Serverless to automatically scale your database based on your workload needs. Alternately, the DTU model provides set price/performance packages to choose from for easy configuration. [Learn more](#)

Service tier: General Purpose (Scalable compute and storage options) [Compare service tiers](#)

Compute tier:

- ☐ Provisioned - Compute resources are pre-allocated. Billed per hour based on vCores configured.
- ☒ **Serverless** - Compute resources are auto-scaled. Billed per second based on vCores used.

Compute Hardware

Select the hardware configuration based on your workload requirements. Availability of compute optimized, memory optimized, and confidential computing hardware depends on the region, service tier, and compute tier.

Hardware Configuration: Gen5
up to 40 vCores, up to 120 GB memory
[Change configuration](#)

Max vCores: 4 vCores

Min vCores: 0.5 vCores

2.1 GB MIN MEMORY 12 GB MAX MEMORY

Auto-pause delay

The database automatically pauses if it is inactive for the time period specified here, and automatically resumes when database activity recurs. Alternatively, auto-pausing can be disabled.

☒ Enable auto-pause

Days: 0 Hours: 1 Minutes: 0

Data max size (GB): 32

9.6 GB LOG SPACE ALLOCATED

Apply

Cost summary

Gen5 - General Purpose (GP S Gen5 4)
Cost per GB (in USD)

Max storage selected (in GB) x 41.6

ESTIMATED STORAGE COST / MONTH USD

COMPUTE COST / VCORE / SECOND ¹ USD

NOTES

¹ Serverless databases are billed in vCores based on a combination of CPU and memory utilization. [Learn more about serverless billing](#)

The image above shows where you can change the autoscaling and autopause properties for serverless compute tier.

Deployment model

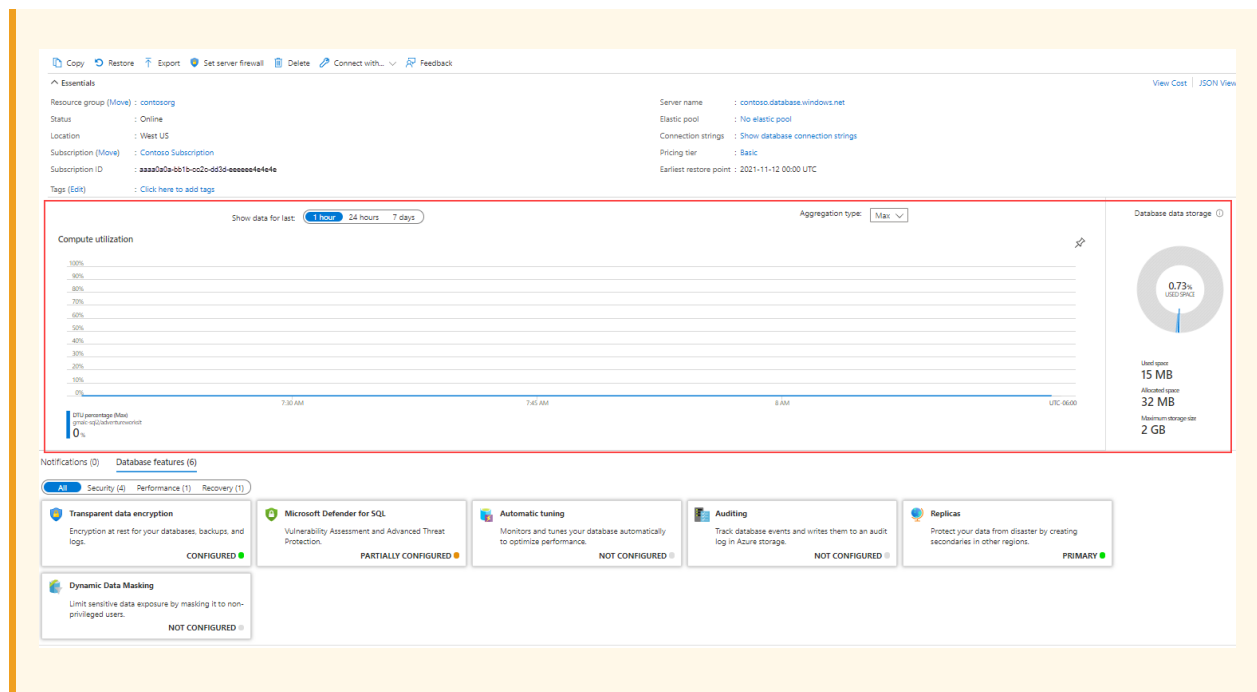
There are two main deployment models when deploying SQL Database on Azure: **single database** and **elastic pool**. Elastic pools share resources with other databases part of the same pool, while single databases resources are managed independently.

SQL Database, like virtual machines, can be deployed using Azure Resource Manager templates, PowerShell, Azure CLI, or the Azure portal.

Single database

Single database is the simplest deployment model of Azure SQL Database. You manage each of your databases individually from scale and data size perspectives. Each database deployed in this model has its own dedicated resources, even if deployed to the same logical server.

You can monitor database resources utilization through Azure portal. This feature allows you to easily identify how the database is performing, as shown in the image below:



Elastic pool

Elastic pools allow you to allocate storage and compute resources to a group of databases, rather than having to manage resources for each database individually. Additionally, elastic pools are easier to scale than single databases, where scaling individual databases is no longer needed due to changes made to the elastic pool.

Elastic pools provide a cost-effective solution for software as a service application model, since resources are shared between all databases. You can configure resources based either on the DTU-based purchasing model or the vCore-based purchasing model.

Due to the nature of this feature, it's recommended to monitor your resources continually to identify concurrent performance spikes that could affect other databases part of the same elastic pool. Often, you may need to revisit your allocation strategy to make sure there's enough resource available for all databases sharing the same elastic pool.

Elastic pool is a good fit for multitenant architecture with low average utilization, where each tenant has its own copy of the database.

Network options

Azure SQL Database by default has a public internet endpoint. Access to this endpoint can be controlled via firewall rules, or limited to specific Azure networks, using features like Virtual Network endpoints or Private Link.

Backup and restore

Azure provides seamless backup and restore capabilities for SQL Database and SQL Managed Instance. Let's learn about some of the most important features.

Continuous backup

With SQL Database, you can increase administration efficiency by knowing that databases are backed up regularly, and that they're continuously copied to a read-access geo-redundant storage (RA-GRS).

Full backups are performed every week, differential backups every 12 to 24 hours, and transaction log backups every 5 to 10 minutes.

Geo-restore

As backups are geo-redundant by default for SQL Database and SQL Managed Instance, you can easily restore databases to a different geographical region, which is especially useful for less strict disaster recovery scenarios.

Backup storage is billed apart from regular database files storage. However, when provisioning a SQL Database, the backup storage is created with the maximum size of the data tier selected for your database at no extra cost.

The duration of a geo-restore operation can be affected by several underlying components including the size of the database, the number of transaction logs involved in a restore operation, and the amount of simultaneous restore requests being processed in the target region.

Note

Geo-restore is available when the backup storage redundancy property is set to ***geo-redundant backup storage***.

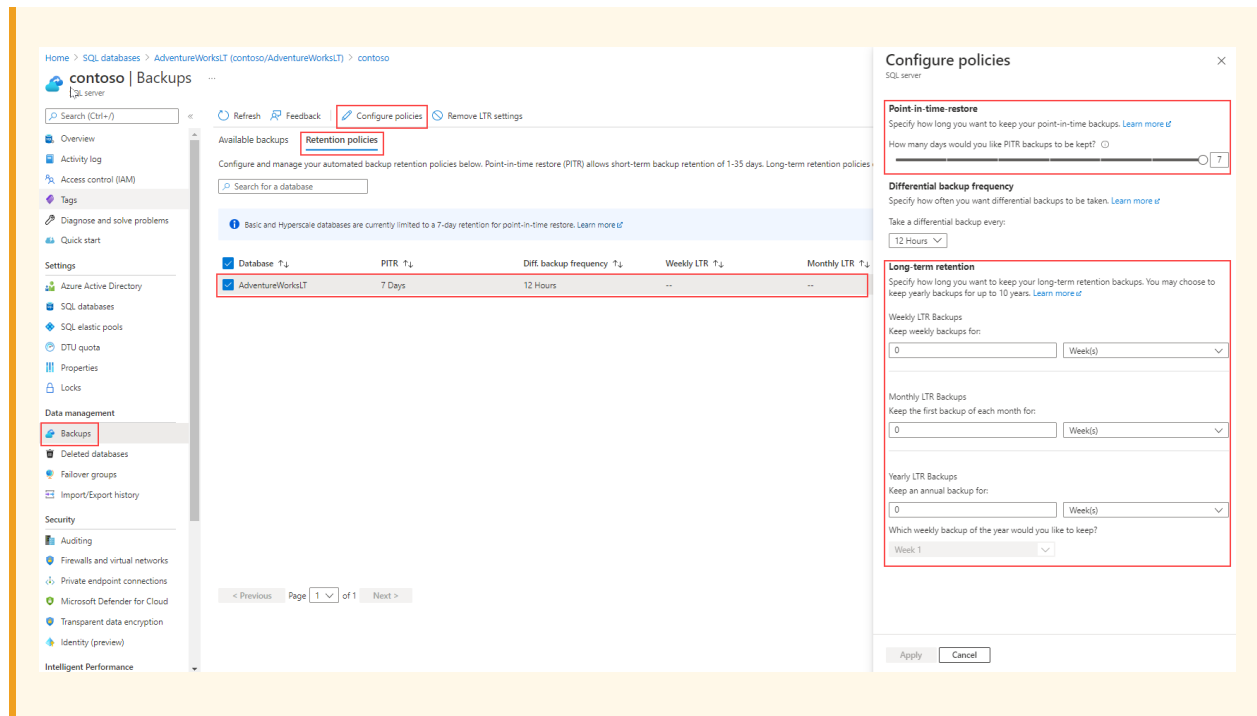
Point-in-time restore (PITR)

You can configure a specific point in time retention policy for each database running on a SQL Database offering. SQL Database retention period can be set from 1 up to 35 days. In fact, if not specified, the default configuration is seven days.

You can restore your databases to a specific point in time according to the retention defined, but PITR is only supported if you're restoring a database in the same server the backup was originated. You can use either Azure portal, Azure PowerShell, Azure CLI, or REST API to restore a SQL Database.

Long-term retention (LTR)

Long-term retention is useful for scenarios that require you to set the retention policy beyond what is offered by Azure. You can set a retention policy for up to 10 years, and this option is disabled by default.



In the image above, you can configure long-term retention policies through Azure portal. Once the database is selected, a new panel opens on the right side of the screen, where you can override the default properties.

For more information about automated backups, see [Automated backups - Azure SQL Database & Azure SQL Managed Instance](#).

Automatic Tuning

Automatic tuning is a built-in feature that relies on machine learning regression capabilities, and automatically identify tuning opportunities based on your query performance.

Automatic tuning currently includes the following features:

- Identify Expensive Queries
- Forcing Last Good Execution Plan
- Adding Indexes
- Removing Indexes

The Azure services use a combination of built-in advanced features to determine the best indexes for your query pattern. Initially, these indexes are tested on a copy of your database, and finally applied to your database.

All databases inherit configuration from their parent server, and you can easily disable this feature at any time.

Elastic query (preview)

Elastic query allows you to run T-SQL queries that bridge multiple databases in SQL Database. This feature is useful for applications using three- and four-part names that can't be changed. It also increases portability as it allows for migration.

Elastic queries support the following partitioning strategies:

Service tier	Capability
Vertical partitioning	Also called cross-database queries. The data is partitioned vertically between many databases. Schemas are different for each database. For example, you may have a database used for customer data, and a different one used for payment information. With the help of vertical partitioning, you can now run a cross-database query between both databases.
Horizontal Partitioning	Also called sharding. The data is partitioned horizontally to distribute rows across several scaled-out databases. In this topology, the schema remains the same among all sharding databases. It supports either single-tenant model or multiple tenant models.

Elastic jobs

The elastic job feature is the SQL Server Agent replacement for Azure SQL Database. To some extent, elastic job is equivalent to the Multi Server Administration feature available on an on-premises SQL Server instance.

You can execute T-SQL commands across several target deployments like SQL Databases, SQL Database elastic pools, and SQL Databases in shard maps. Database resources can run on different Azure subscriptions, and/or regions. The execution runs in parallel, which is useful when automating database maintenance tasks.

Note

Azure SQL Managed Instance doesn't support elastic jobs.

SQL Database in Fabric

SQL Database in Microsoft Fabric is a powerful and versatile solution designed to integrate seamlessly with the broader Microsoft ecosystem. It allows users to use the capabilities of SQL databases within the Microsoft Fabric environment, providing a unified platform for managing and analyzing data. This integration enables organizations to streamline their data workflows, enhance collaboration, and drive more informed decision-making processes.

Moreover, SQL Database in Microsoft Fabric supports a wide range of use cases, from operational and transactional workloads to advanced analytics. This flexibility makes it an ideal choice for organizations looking to consolidate their data infrastructure and use the full potential of their data assets. With features like seamless integration with other Microsoft services, enhanced security, and scalability, SQL Database in Microsoft Fabric empowers users to build and deploy data-driven applications with ease.

The **Databases** workload in Fabric is designed to address many aspects of operational SQL databases, enabling data engineers, analysts, and data scientists to work together to create and query data that is optimized for their specific needs.

5. Explore Azure SQL Managed Instance

<https://learn.microsoft.com/en-us/training/modules/prepare-to-maintain-sql-databases-azure/5-explore-azure-sql-database-managed-instance>

Explore Azure SQL Managed Instance

Completed

Most of the features available on Azure SQL Database will also work for Azure SQL Managed Instance as they share the same base code. The fully managed platform as a service (PaaS) provides some of the following benefits:

- Automatic backups
- Automatic patching
- Built-in high availability
- Security and performance tools
- Embedded auditing capabilities

Another key benefit when migrating to one of the PaaS offerings on Azure is that you no longer have to install or patch SQL Server, which can increase application uptime, and reduce maintenance efforts.

Unlike Azure SQL Database, which is designed around single database structures, SQL Managed Instance provides several other features including cross-database queries, common language runtime (CLR), access to the system databases, and use of the SQL Agent features.

For a complete list of the features available on Azure SQL Managed Instance, see [Features of SQL Database and SQL Managed Instance](#).

Hybrid licensing options

Microsoft offers several benefits to SQL Server licenses. For both SQL Database and SQL Managed Instance, taking advantage of your existing licenses can reduce the cost of running the PaaS offering.

- For each core of Enterprise Edition with Active Software Assurance, you're eligible for one vCore of SQL Database or SQL Managed Instance Business Critical, and eight vCores of General Purpose.
- For each core of Standard Edition with Active Software Assurance, you're eligible for one vCore of General Purpose.

This model can reduce the total license costs by up to 40%. Effectively, you'll only be paying for the compute and storage costs, and not the software licensing costs.

For more information on bring-your-own licensing model, see [License Mobility through Software Assurance on Azure](#).

Connectivity architecture

Connections to SQL Managed Instance are made through TDS endpoints. While the routing and security on these connections differ, there's a gateway component that handles and routes connections to the database service. This gateway component is also deployed in a highly available fashion.

Backup and restore

Automated database backup provides a fully managed backup service that takes full, differential, and log backups regularly for both SQL Managed Instance and SQL Database offerings. Automated backups are geo-redundant, and replicated to a paired-region automatically, which protects your data against localized outages in the primary region.

Similarly, SQL Managed Instance allows easy migration of existing applications, enabling restores from on-premises backups.

There are some important considerations when running backup and restore operations on SQL Managed Instance databases:

- It isn't possible to overwrite an existing database during the restore process. Before restoring a database, you must ensure that it doesn't exist.
- For SQL Managed Instance, backups can only be restored to another managed instance. It isn't possible to restore a managed instance database backup to a SQL Server running on a virtual machine or SQL Database.
- Copy-only backup to Azure blob storage is available for SQL Managed Instance. SQL Database doesn't support this feature.

For more information about automated backups, see [Automated backups - Azure SQL Database & Azure SQL Managed Instance](#).

High availability architecture

SQL Database and SQL Managed Instance have similar high availability architectures, which guarantee 99.99% uptime. Windows and SQL Server updates are handled by the backend infrastructure, generally without any effect to your application, though it's important to place a [retry logic](#) into your application.

The auto-failover groups feature allows you to fail over a group of replicated databases on a server to another region. This feature is designed on top of the existing active geo-replication capability, which simplifies deployment and management of geo-replicated databases.

A failover group can include one or multiple databases, often used by the same application. Additionally, you can use the readable secondary databases to offload read-only query workloads.

Note

The auto-failover groups feature is supported on both SQL Managed Instance and SQL Database.

For more information about auto-failover groups, see [Use auto-failover groups to enable transparent and coordinated geo-failover of multiple databases](#).

Migration options

In general, migrating to SQL Managed Instance is often simple given the large set of available features. There are a couple of ways to migrate on-premises databases:

- **Log Replay Service.** It's an online migration option, and used when you need more control of your database migration project.
- **Managed Instance link.** The Managed Instance link, using distributed availability groups, securely extends your data estate by replicating data almost instantly (online) between any hosted SQL Server and Azure SQL Managed Instance, and vice versa.
- **Native backup and restore.** Backup and restore are a simple migration method favored by many SQL Server professionals. It's the easiest migration option for customers who can provide full database backups to Azure Storage.
- **Transactional replication.** Transactional replication is a way to move data between continuously connected database servers. It's best to be used for online or offline migration of large and complex databases.

Machine Learning Services

Machine Learning Services provides machine learning operations within your relational database structure. This feature supports Python and R packages, ideal for high-intensive predictive capabilities. This option is available on SQL Managed Instance, SQL Server on Azure virtual machine, and on-premises SQL Server.

Applications can use relational database on Azure combined with machine learning high-performance capabilities, where you can:

- Train machine learning models based on either sampled dataset or population dataset.
- Reduce complexity in security and compliance, where you don't need to relocate your data to build and train your machine learning models.
- Deploy machine learning models using T-SQL stored procedures that support Python or R programming language.
- Use of open-source libraries like scikit-learn, PyTorch, and TensorFlow.

For busy environments, you can use the [T-SQL PREDICT function](#), which allows you to accelerate predictions based on your stored model.

The Machine Learning Services feature can be enabled by running the following command:

```
EXEC sp_configure 'external scripts enabled', 1;  
RECONFIGURE WITH OVERRIDE;
```

The command above allows the execution of external scripts in your managed instance, and it should be enabled before you attempt to use **sp_execute_external_script** to execute Python or R scripts in your database.

Note

SQL Database doesn't support Machine Learning Services feature.

For more information about Machine Learning Services, see [Machine Learning Services in Azure SQL Managed Instance](#).

6. Module assessment

<https://learn.microsoft.com/en-us/training/modules/prepare-to-maintain-sql-databases-azure/6-knowledge-check>

Module assessment

Completed

7. Summary

<https://learn.microsoft.com/en-us/training/modules/prepare-to-maintain-sql-databases-azure/7-summary>

Summary

Completed

Microsoft provides a great deal of options for deploying SQL Server to Azure. Migrating on-premises databases to Azure SQL Server on Virtual Machine offers portability, while Azure SQL Database and Azure SQL Managed Instance provide scalability and flexibility.

Now that you've reviewed this module, you should be able to:

- Understand the role of Azure Database Administrator, and other data platform roles
- Describe the key differences between the SQL Server-based database options in Azure
- Describe other features for Azure SQL platforms available

Note

If you're ready to get started with Azure SQL Database, [try Azure SQL Database free of charge](#) for the life of your subscription.

Deploy IaaS solutions with Azure SQL

1. Introduction

<https://learn.microsoft.com/en-us/training/modules/deploy-iaas-solutions-with-azure-sql/1-introduction>

Introduction

Completed

The most common reason for deploying SQL Server in an Azure Virtual Machine (VM) is because you want an easy, straightforward method to migrate an existing on-premises SQL Server into the cloud.

Understanding the options and methods for deploying SQL Server in an Azure VM is critical to ensure a successful migration. Infrastructure as a Service (IaaS) allows for greater flexibility in configuration. This flexibility means that you, as a database administrator, must plan your configuration carefully. Choosing the proper VM sizing, storage, and networking options are crucial to ensure adequate performance for your workloads.

Learning objectives

At the end of this module, you'll be able to:

- Explore the basics of SQL Server in an Infrastructure as a Service (IaaS) offering
- Learn how hybrid scenario works
- Explore performance and security options available
- Understand the high availability and disaster recovery options

2. Explain IaaS options to deploy SQL Server in Azure

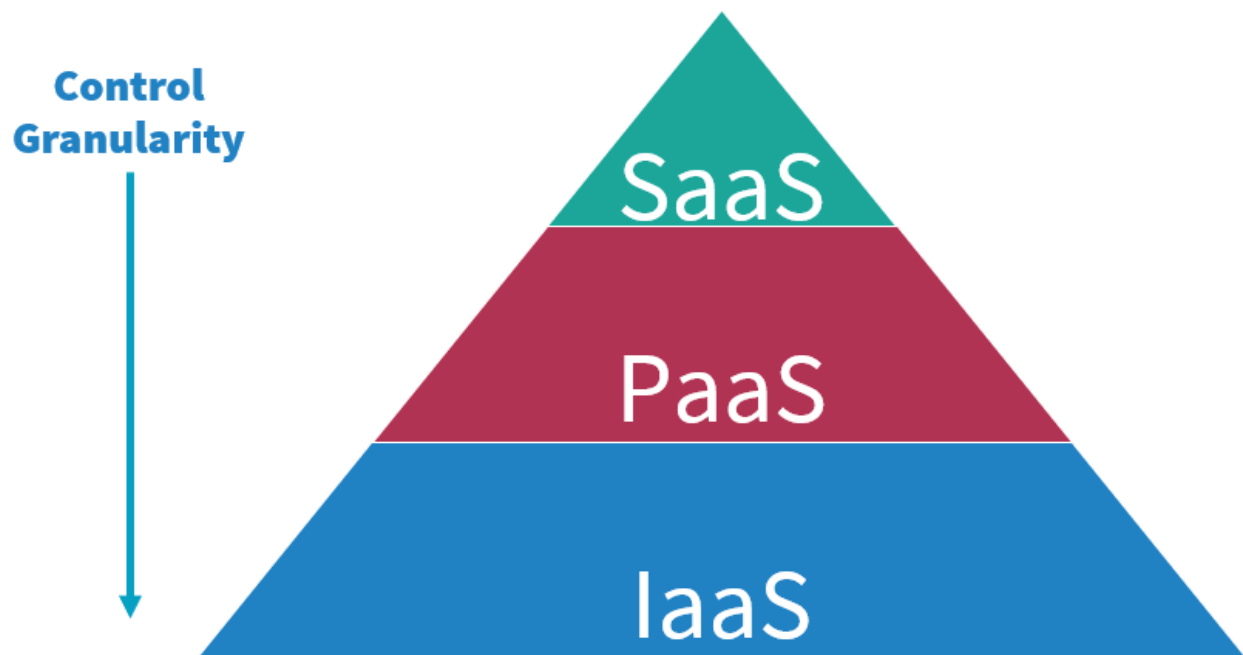
Explain IaaS options to deploy SQL Server in Azure

Completed

Many applications require a VM running SQL Server. Some reasons for this option include:

- **Older versions of SQL Server**—If an application requires an older version of SQL Server for vendor support, running inside a VM is the best option for those applications, because it allows for the application to be supported by that vendor.
- **Use of other SQL Server services**—While Analysis Services and to an extent Integration Services (by using Azure Data Factory) are available as PaaS offerings, many users maximize their licensing by running SQL Server Analysis Services, Integration Services, or Reporting Services on the same machine as the database engine.
- **General application incompatibility**—This reason is a somewhat of a catch-all. For example, Azure SQL Database doesn't support cross-database querying, while managed instance does. Some applications require other services to coexist with the database instance in a manner that isn't compatible with a PaaS offering.

Infrastructure as a Service (IaaS) allows the administrator to have more granular access over specific settings of the underlying infrastructure than the other Azure offerings. While the Azure platform manages the underlying server and network hardware, you still have access to the virtual storage, virtual networking configuration, and any other software you might install within the virtual machine, including SQL Server.



The image illustrates the increased control you have using IaaS, compared to PaaS offerings. Note that while this diagram shows SaaS as a general cloud service model, Azure SQL offerings are either PaaS (such as Azure SQL Database and Azure SQL Managed Instance) or IaaS (SQL Server on Azure Virtual Machines).

In Azure SQL PaaS offerings, the administrator is responsible for database management, user security, and data management, while Microsoft manages the operating system and SQL Server software. When you use Azure SQL PaaS services, the operating system (OS) and SQL Server software are managed by the cloud provider. A good example of this is Azure SQL Database where the operating system and database engine are installed and configured by Microsoft, allowing you to start building database applications quickly. IaaS solutions are the most open-ended; you're responsible for OS patching, SQL Server installation and configuration, and optimal configuration of your network and storage options. With an IaaS deployment, you're also responsible for software configuration.

Some of your applications may not be suited for other Azure offerings, such as Azure SQL Database, because they require specific operating conditions. These conditions could include a specific combination of SQL Server and Windows versions for vendor support purposes, or other software that needs to be installed alongside of SQL Server. SQL Server paired with the Azure IaaS platform provides the required control options for many organizations, whether it be specific feature like CLR or replication, or the use of Active Directory (as opposed to Microsoft Entra ID) authentication. Another requirement is that some applications install software alongside SQL Server, which requires direct access to the underlying operating system. Direct access to the OS isn't supported in a PaaS model. These organizations and their applications can obtain the advantages of moving to a cloud service without losing critical capabilities that their organization requires.

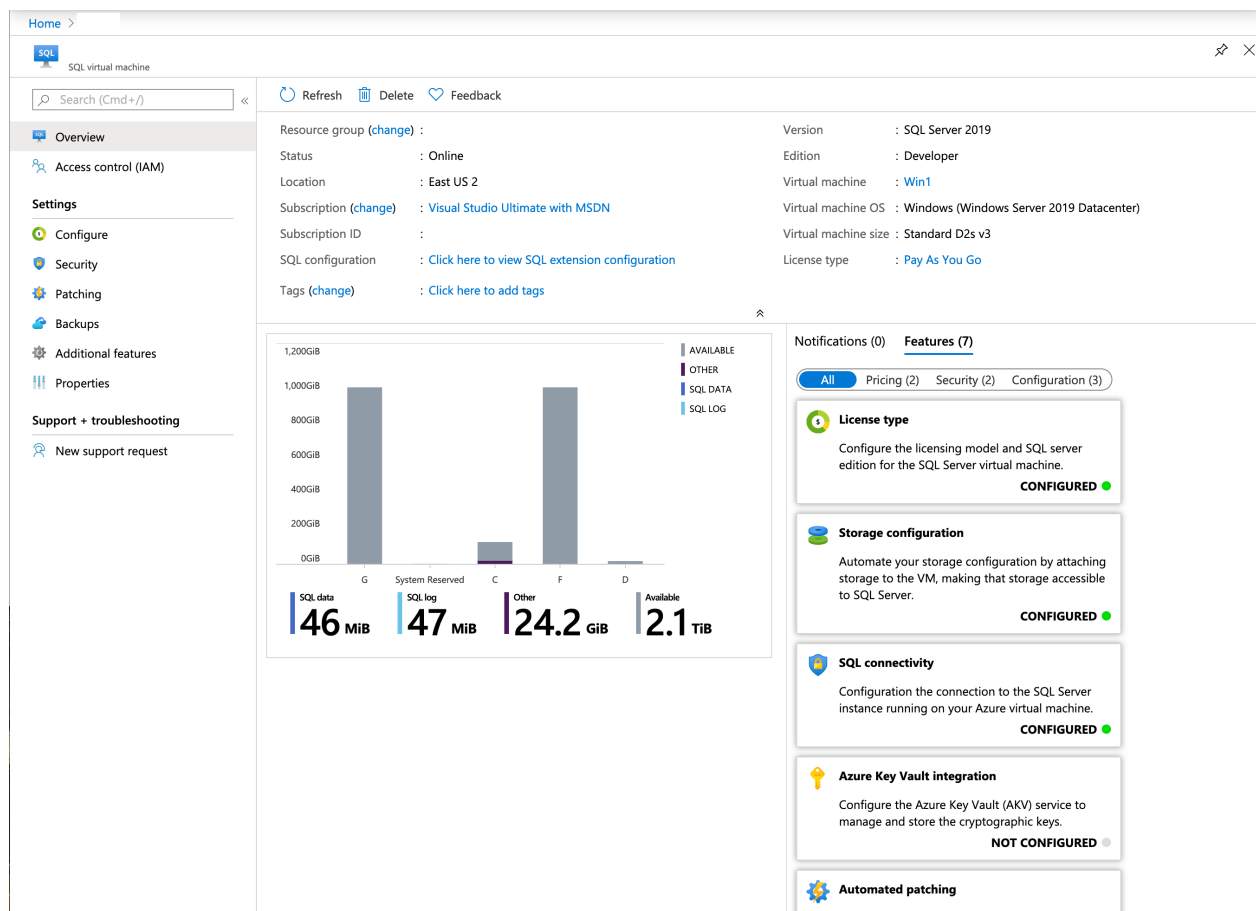
SQL Server IaaS Agent Extension

When you deploy an SQL Server VM from Azure Marketplace, part of the process installs the IaaS Agent Extension.

Extensions are code that is executed on your VM post-deployment, typically to perform post-deployment configurations. Some examples include installing anti-virus software or enabling a Windows feature. The SQL Server IaaS Agent Extension provides the following main features that can reduce your administrative overhead.

- **Automated backup**
- **Automated patching**
- **Azure Key Vault integration**
- **Microsoft Defender for Cloud integration**
- **View Disk utilization in the portal**
- **Flexible licensing**
- **Flexible version or edition**
- **SQL best practices assessment**

In addition to these features, the extension allows you to view information about your SQL Server's configuration and storage utilization.



SQL Server licensing models

There are several different options related to how SQL Server is licensed when using the Azure IaaS offering.

If you aren't participating in the Microsoft Software Assurance (SA) program, you can deploy an image from Azure Marketplace containing a preconfigured SQL Server, and pay-per-minute for the use of SQL Server. This option is referred to as the Pay as you Go model and the cost of the SQL Server license is included with the cost of the virtual machine.

If you're participating in the Microsoft Software Assurance (SA) program, you have more flexibility in how you license your SQL Server:

- You can use the previous method and pay-per-minute by deploying a virtual machine image containing a SQL Server from Azure Marketplace
- You can Bring Your Own License (BYOL) when deploying the virtual machine that doesn't contain a preconfigured SQL Server instance. This option is possible when you already have purchased a valid SQL Server license for your on-premises infrastructure. This license can be applied to the virtual machine to ensure that you're properly licensed. You must report the usage of licenses to Microsoft by using the License Mobility verification form within 10 days of implementing the virtual machine.

When choosing this method, you can manually install SQL Server through media you have obtained, or you can choose to upload a virtual machine image to Azure.

In addition to flexible licensing options for SQL Server, there are also Windows Server licensing options that can be taken advantage of. These Windows Server options are known as the Azure Hybrid Benefit (AHB). Similar to applying a SQL Server license you already have purchased, you're able to take advantage of Windows Server licenses you already own.

Reserving a virtual machine for one to three years provides another option for cost savings. This commitment doesn't require an upfront payment and can be billed monthly. Using the reservation option can be beneficial if you know the workloads are going to be persisted. The cost savings can be significant, especially for larger VMs.

Virtual machine families

There are several series, or 'families,' of virtual machine sizes that you can choose. Each series is a combination of memory, CPU, and storage that meets certain requirements. For example, the series that are compute optimized have a higher CPU-to-memory ratio. Having multiple options allows you to select an appropriate hardware configuration for the expected workload. The following six series each has various sizes available, the details of which are fully described in the Azure portal when you choose the option to select your VM size.

General purpose - These VMs provide a balanced ratio of CPU to memory. This VM class is ideal for testing and development, small to medium-sized database servers, and web servers with a low to medium amount of traffic.

Compute optimized - Compute optimized VMs have a high CPU-to-memory ratio and are good for web servers with a medium amount of traffic, network appliances, batch processes, and application servers. These VMs can also support machine learning workloads that can't benefit from GPU-based VMs.

Memory optimized - These VMs provide a high memory-to-CPU ratio. These VMs cover a broad range of CPU and memory options (all the way up to 4 TB of RAM) and are well suited for most database workloads.

Storage optimized - Storage optimized VMs provide fast, local, NVMe storage that is ephemeral. They're good candidates for scale-out data workloads such as Cassandra. It's possible to use them with SQL Server; however, since the storage is ephemeral, you need to ensure you configure data protection using a feature like Always On Availability Groups or Log Shipping.

GPU - Azure VMs with GPUs are targeted at two main types of workloads—naturally graphics processing operations like video rendering and processing, but also massively parallel machine learning workloads that can take advantage of GPUs.

FPGA accelerated - These sizes are designed for compute-intensive workloads. Storage throughput and network bandwidth are also included for each size in this group.

High performance compute - High Performance Compute workloads support applications that can scale horizontally to thousands of CPU cores. This support is provided by high-performance CPU and remote direct memory access (RDMA) networking that provides low latency communications between VMs.

The easiest way to see the sizing options within each series is through the Azure portal. From the blade for creating a VM, you select the option **See all sizes** to see the list.

Select a VM size ...

vCPUs: All RAM (GiB): All Display cost: Monthly Add filter

Showing 791 VM sizes. | Subscription: WWL-JuPadrao | Region: West US 2 | Current size: Standard_D16as_v4 | Image: Free SQL Server License: SQL Server 2022 Developer on Windows Server 2022 [Learn more about VM sizes](#)

SKU	Type	vCPUs	RAM (GiB)	Data disks	Max IOPS	Local storage (GiB)	Premium disk	Cost per month
Most used by Azure users								
The most used sizes by users in Azure								
B1s ⓘ	General purpose	1	1	2	320	4	Supported	\$10.51 Popular
B2ms ⓘ	General purpose	2	8	4	1920	16	Supported	\$66.58 Popular
B2s ⓘ	General purpose	2	4	4	1280	8	Supported	\$36.21 Popular
B4ms ⓘ	General purpose	4	16	8	2880	32	Supported	\$132.86 Popular
D2as_v4 ⓘ	General purpose	2	8	4	3200	16	Supported	\$137.24 Popular
D2s_v3 ⓘ	General purpose	2	8	4	3200	16	Supported	\$137.24 Popular
D4s_v3 ⓘ	General purpose	4	16	8	6400	32	Supported	\$274.48 Popular
D8s_v3 ⓘ	General purpose	8	32	16	12800	64	Supported	\$548.96 Popular
DS1_v2 ⓘ	General purpose	1	3.5	4	3200	7	Supported	\$85.41 Popular
DS2_v2 ⓘ	General purpose	2	7	8	6400	14	Supported	\$170.82 Popular
DS3_v2 ⓘ	General purpose	4	14	16	12800	28	Supported	\$341.64 Popular
D-Series v5								
The 5th generation D family sizes recommended for your general purpose needs								
D-Series v4								
The 4th generation D family sizes recommended for your general purpose needs								

The image above shows just a small set of the series and size possibilities. For each option, you can see the number of Virtual CPUs, the amount of RAM, the number of Data disks, the Max IOPS, the

temporary storage provided and whether Premium storage is supported.

For more information about VM size best practices, see [Best practices for SQL Server on Azure VMs](#).

Azure Marketplace

Azure Marketplace is essentially a centralized location that provides the ability to create Azure resources based on a predesigned template. For example, you can quickly create a SQL Server 2022 instance on Windows Server 2019 with a couple of clicks, along with some basic information, such as the virtual machine name and SQL Server configuration details. Once provided, Azure Resource Manager initiates the creation of the virtual machine, and within minutes it's up and running.

The blade for SQL Server 2019 on Windows Server 2019 in Azure Marketplace is shown below. This blade gives you the option of preset configurations that support OLTP or Data Warehouse workloads and allows you to specify storage, patching, and backup options.

SQL Server 2022 on Windows Server 2022  ...

Microsoft

**SQL Server 2022 on Windows Server 2022**  [Add to Favorites](#)

Microsoft | Virtual Machine

Plan

Free SQL Server License: SQL Server 2...  [Create](#) [Start with a pre-set configuration](#)

Overview **Plans** Usage Information + Support Ratings + Reviews

Software plan	Description
Free SQL Server License: SQL Server 2022 Developer on Windows Server 2022	This image contains the Developer edition of SQL Server 2022 on Windows Server 2022. This free edition (no SQL Server licensing cost) includes all the functionality of Enterprise edition, but it is licensed for development and testing only, not production. It provides comprehensive capabilities for mission-critical transactional processing, data warehousing, and real-time business intelligence. It includes the database engine with support for key features from previous versions including in-memory transactional processing, intelligent query processing, Always On availability groups for +99.99% high availability and read scale-out capabilities, as well as new features such as Azure Synapse Link for SQL, Link to Azure SQL Managed Instance, Ledger, and parameter sensitive plan optimization. Includes Integration Services for moving and transforming data, Analysis Services for data mining, and Reporting Services for web and mobile reports. We recommend that you use a virtual machine size of D8ds-v5 or higher for development and functional testing, E16bds-v5 or higher for performance testing.
SQL Server 2022 Web on Windows Server 2022	This image contains the Web edition of SQL Server 2022 on Windows Server 2022. This provides a low-cost database solution for medium-size web applications. It includes the core database engine and Management Studio for integrated administration and development. Also includes Integration Services for moving and transforming data, and Analysis Services for data mining. we recommend that you use a virtual machine size of E16ds-v4 or higher.
SQL Server 2022 Enterprise on Windows Server 2022	This image contains the full version of SQL Server 2022 Enterprise edition on Windows Server 2022. SQL Server 2022 represents a major step towards making SQL Server a platform that gives you choices of development languages, data types, on-premises or cloud, and operating systems by bringing the power of SQL Server to Linux, Linux-based Docker containers, and Windows. It provides comprehensive capabilities for mission-critical transactional processing, data warehousing, and real-time business intelligence. It includes the database engine with support for In-Memory transactional processing and analytics, automatic database tuning and new graph capabilities for modeling many-to-many relationships. This edition also includes Always On for +99.99% high availability and read scale-out capabilities. Various layers of protection, including innovative features like Always Encrypted and Row-Level Security for the highest data protection. Includes Integration Services for moving and transforming data, Analysis Services for data mining and Master Data Services for data modeling. we recommend that you use a virtual machine size of E16ds-v4 or higher.
SQL Server 2022 Standard on Windows Server 2022	Breakthrough performance and faster insights delivered by a hybrid data platform. This image contains the Standard edition of SQL Server 2022 on Windows Server 2022. Standard edition provides core data management capabilities for medium-size transactional processing and data warehousing. It includes the database engine with a basic version of Always On high availability and Row-Level Security. Includes Management Studio for integrated administration and development. we recommend that you use a virtual machine size of E16as-v4 or higher.

The disadvantage of using the portal to create Azure resources is that it isn't an easily repeatable process. However, it's easy to get started with the portal, where you can quickly get up and running

with resources.

SQL Server configuration

When you provision an Azure virtual machine running SQL Server, you can also configure specific SQL Server settings, such as Security and Networking, SQL Authentication preferences, SQL instance settings, and a few other options. These options are located on the **SQL Server settings** tab.

Create a virtual machine ...

 [Help me create a low cost VM](#) [Help me create a VM optimized for high availability](#) [Help me choose the right VM size for my workload](#)

[Basics](#) [Disks](#) [Networking](#) [Management](#) [Monitoring](#) [Advanced](#) [SQL Server settings](#) [Tags](#) [Review + create](#)

Security & Networking

SQL connectivity *

Port *

SQL Authentication

SQL Authentication ⓘ ☒ Disable ☐ Enable

Azure Key Vault integration ⓘ ☒ Disable ☐ Enable

Storage configuration

Customize performance, size, and workload type to optimize storage for this virtual machine. For optimal performance, separate drives will be created for data and log storage by default. [Learn more about SQL Server best performance practices.](#)

Storage	SQL Data: 1024 GiB, 5000 IOPS, 200 MB/s, Premium SSD SQL Log: 1024 GiB, 5000 IOPS, 200 MB/s, Premium SSD SQL tempdb: Use local SSD drive Change configuration
---------	--

SQL instance settings

Customize additional SQL instance settings including collation, MAXDOP, server memory limit and optimize for ad-hoc workload.

Instance settings	Default configuration MAXDOP: 0 SQL Server memory limits: 0 - 2147483647 MB Collation: SQL_Latin1_General_CP1_CI_AS Change SQL instance settings
-------------------	--

SQL Server License

Save up to 43% with licenses you already own. Already have a SQL Server license? [Learn more](#)

SQL Server License ⓘ ☒ No ☐ Yes

Automated patching

Set a patching window during which all Windows and SQL patches will be applied.

Automated patching ⓘ	Disabled Change configuration
----------------------	---

Automated backup

Automated backup ⓘ ☒ Disable ☐ Enable

[< Previous](#) [Next : Tags >](#) [Review + create](#)

For more information about the SQL Server settings available when creating a virtual machine, see [Provision SQL Server on Azure VM \(Azure portal\)](#).

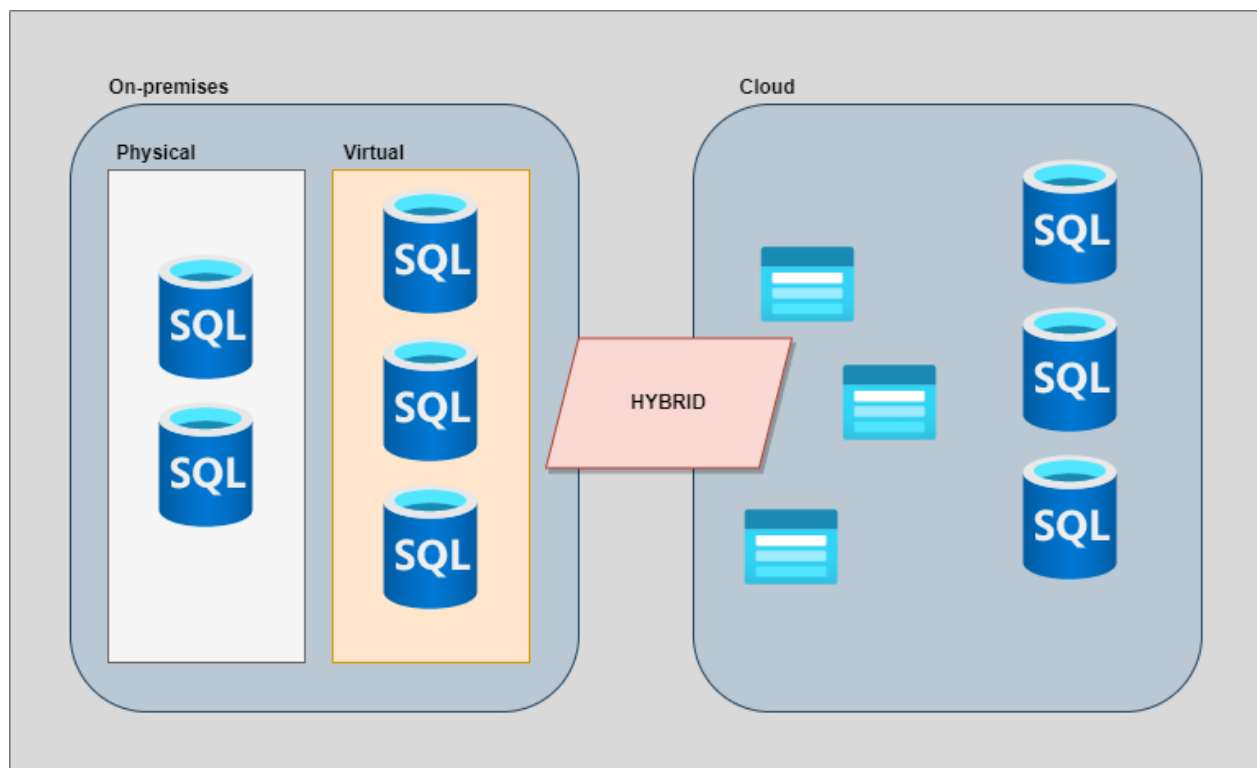
3. Understand hybrid scenarios

Understand hybrid scenarios

Completed

Suppose your organization has a significant investment in SQL Server infrastructure on-premises and in local data centers. In that case, it's crucial to be aware that using the cloud isn't an all-or-nothing proposition. There are ways to use existing on-premises infrastructure in a hybrid capacity with Azure to improve operational resiliency and reduce costs.

Adopting a hybrid infrastructure is an excellent initial step for organizations that rely on on-premises solutions and are cautious about transitioning to the cloud. Many modern organizations use a combination of physical and virtualized SQL Server deployments on-premises, which can be extended to the cloud as part of a hybrid solution. The hybrid SQL Server platform combines the advantages of both on-premises and cloud services, serving as a complementary middle ground. Typically, the cloud component uses IaaS services such as storage or SQL Server virtual machines.



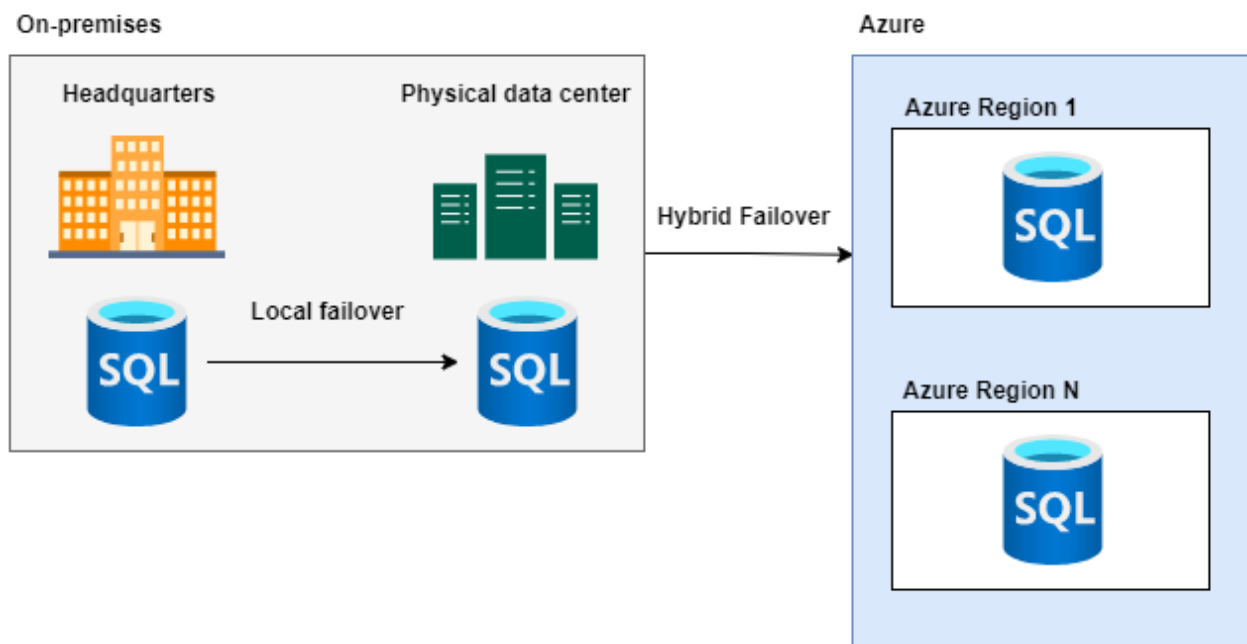
In addition to extending on-premises solutions, the same patterns apply to existing alternative cloud solutions, enabling cloud-to-cloud hybrid implementations. Let's review some of the most common hybrid scenarios for SQL Server.

Hybrid scenarios for SQL Server

Let's review a few strategies for when deploying a hybrid solution for SQL Server.

Disaster Recovery

Disaster Recovery is the most common scenario for a hybrid deployment of SQL Server. It ensures business continuity if catastrophic events occur. Organizations can distribute deployments across multiple data centers for failover in an on-premises approach. These data centers typically reside within the same geographic region as the organization, making them susceptible to significant regional disasters. Physical data centers are also costly to deploy, monitor, and maintain. Arguably, the cost of spinning up Azure virtual machines running SQL Server in various geographical regions is much less than establishing a new physical data center in another geography.



In this hybrid approach, Azure is used for [DR failover](#) (to one or more regions) while the regular day-to-day processing continues to use on-premises servers for local high availability.

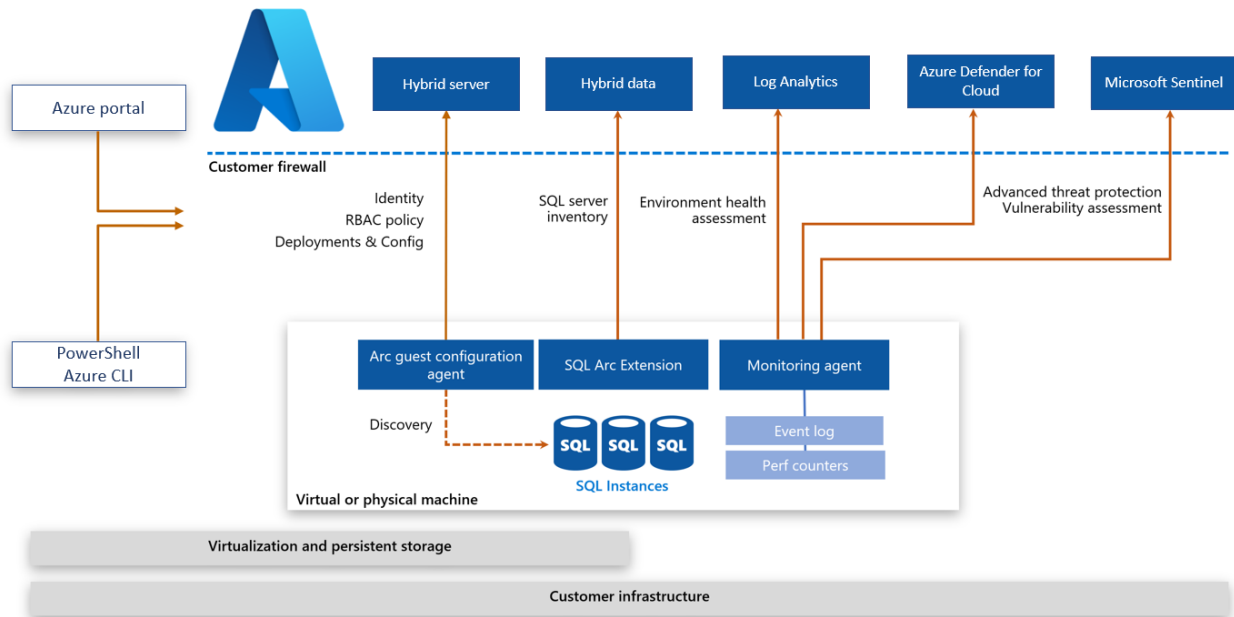
SQL Server Backups

SQL Server Backups is another common hybrid scenario. Backups may be done directly into Azure Storage via URL or Azure file share (SMB). This scenario protects against data loss when on-site backup storage fails. In addition, these backups may also be restored to virtual machines in Azure and tested as part of Disaster Recovery procedures.

Another scenario uses Azure Storage to store on-premises **SQL Server data files for user databases**. These are user files and not system databases. If the local storage fails, the user files are safely stored in the cloud, preventing data loss. In addition, by using Azure storage, there are [built-in reliability guarantees](#) so storing these files in the cloud is more resilient. For this hybrid scenario, it's essential to keep the network communication secure, evaluate the network latency of the solution, and ensure the storage account is locked down using ACLs and Microsoft Entra ID.

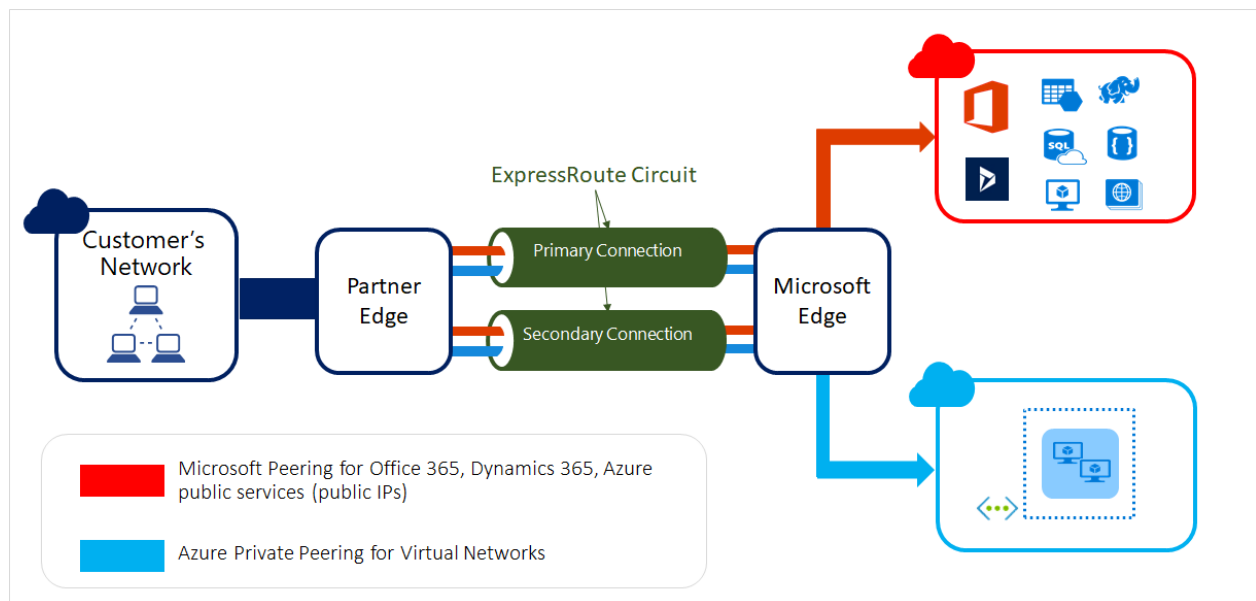
Azure Arc-enabled SQL Servers

Having **Azure Arc-enabled SQL Servers** extends and centralizes Azure management services to SQL Server instances hosted on-premises, in your data centers, on the edge, and in multicloud environments. In this hybrid scenario, Azure Arc enables the inventory of all registered SQL Server deployments and assesses their configurations, usage patterns, and security to provide actions and recommendations based on best practices. By using **SQL Server enabled by Azure Arc**, you gain the benefits of centralized server management. You also get real-time security alerts and vulnerability reporting on both on-premises SQL Servers and their host operating systems.



Security considerations

When deploying a hybrid SQL solution, all core infrastructure, such as Active Directory and DNS, must exist on-premises and in Azure. In addition, secure two-way communication must exist between the on-premises network and Azure. This secured communication can take the form of a **site-to-site (S2S) VPN** or a dedicated **ExpressRoute** tunnel. When evaluating different connectivity methods, it's vital to determine the amount of latency acceptable for your organization. Regardless of the solution chosen, network security must also be at the forefront of the implementation.



The ExpressRoute connections tend to be more costly, but they provide the best security and lowest latency as all communication flows over a direct secured channel independent of the public internet. However, common drawbacks for this solution include overall cost and the inability to apply ExpressRoute between cloud providers in a multicloud solution.

4. Explore performance and security

<https://learn.microsoft.com/en-us/training/modules/deploy-iaas-solutions-with-azure-sql/4-explore-performance-and-security>

Explore performance and security

Completed

The Azure ecosystem offers several performance and security options for SQL Server instances on Azure virtual machines. Each option provides various capabilities, such as different disk types that meet the capacity and performance requirements of your workload.

Storage considerations

SQL Server requires good storage performance to deliver robust application performance, whether it be an on-premises instance or installed in an Azure VM. Azure provides a wide variety of storage solutions to meet the needs of your workload. While Azure offers various types of storage (blob, file, queue, table), in most cases SQL Server workloads will use Azure managed disks. The exceptions are that a Failover Cluster Instance can be built on file storage and backups will use blob storage. Azure-

managed disks act as a block-level storage device that is presented to your Azure VM. Managed disks offer a number of benefits including 99.999% availability, scalable deployment (you can have up to 50,000 VM disks per subscription per region), and integration with availability sets and zones to offer higher levels of resiliency in case of failure.

Azure-managed disks all offer two types of encryption. Azure Server-side encryption is provided by the storage service and acts as encryption-at-rest provided by the storage service. Azure Disk Encryption uses BitLocker on Windows, and DM-Crypt on Linux to provide OS and Data disk encryption inside of the VM. Both technologies integrate with Azure Key Vault and allow you to bring your own encryption key.

Each VM will have at least two disks associated with it:

- **Operating System disk** – Each virtual machine will require an operating system disk that contains the boot volume. This disk would be the C: drive in the case of a Windows platform virtual machine, or /dev/sda1 on Linux. The operating system will be automatically installed on the operating system disk.
- **Temporary disk** – Each virtual machine will include one disk used for temporary storage. This storage is intended to be used for data that doesn't need to be durable, such as page files or swap files. Because the disk is temporary, you shouldn't use it for storing any critical information like database or transaction log files as they'll be lost during maintenance or a reboot of the virtual machine. This drive will be mounted as D:\ on Windows, and /dev/sdb1 on Linux.

Additionally, you can and should add additional data disks to your Azure VMs running SQL Server.

- **Data disks** – The term data disk is used in the Azure portal, but in practice these are just additional managed disks added to a VM. These disks can be pooled to increase the available IOPS and storage capacity, using Storage Spaces on Windows or Logical Volume Management on Linux.

Furthermore, each disk can be one of several types:

Feature	Ultra Disk	Premium SSD v2	Premium SSD	Standard SSD	Standard HDD
Disk type	SSD	SSD	SSD	SSD	HDD
Best for	IO-intensive workloads	Performance-sensitive workloads	Performance-sensitive workloads	Lightweight workloads	Backups, non-critical workloads
Max disk size	65,536 GiB	64,000 GiB	32,767 GiB	32,767 GiB	32,767 GiB
Max throughput	10,000 MB/s	1,200 MB/s	900 MB/s	750 MB/s	500 MB/s
Max IOPS	160,000	80,000	20,000	6,000	2,000

The best practices for SQL Server on Azure recommend using Premium Disks pooled for increased IOPs and storage capacity. Data files should be stored in their own pool with read-caching on the Azure disks.

Transaction log files won't benefit from this caching, so those files should go into their own pool without caching. TempDB can optionally go into its own pool, or using the VM's temporary disk, which offers low latency since it's physically attached to the physical server where the VMs are running. Properly configured Premium SSD will see latency in single digit milliseconds. For mission critical workloads that require latency lower than that, you should consider Ultra SSD.

Security considerations

There are several regulations and standards that Azure complies with that makes it possible to build a compliant solution with SQL Server running in a virtual machine.

Microsoft Defender for SQL

[Microsoft Defender for SQL](#) provides Microsoft Defender for Cloud security features such as vulnerability assessments and security alerts.

Azure Defender for SQL can be used to identify and mitigate potential vulnerabilities in your SQL Server instance and database. The vulnerability assessment feature can detect potential risks in your SQL Server environment and help you remediate them. It also provides insight into your security state and actionable steps to resolve security issues.

Microsoft Defender for Cloud

Microsoft Defender for Cloud is a unified security management system that evaluates and offers opportunities for improving several security aspects of your data environment. Microsoft Defender for Cloud is a security management tool that allows you to gain insight into your security state across hybrid cloud workloads, reduce your exposure to attacks, and respond to detected threats quickly.

Performance considerations

Most of the existing on-premises SQL Server performance features are also available on Azure virtual machines (VMs). Among the options offered is data compression, which can improve the performance of I/O-intensive workloads while decreasing the size of the database. Similarly, table and index partitioning can improve query performance of large tables, while improving performance and scalability.

Table partitioning

[Table partitioning](#) provides many benefits, but often this strategy is only considered when the table becomes large enough that starts compromising query performance. Identifying which tables are candidates for table partitioning is a good practice that could lead to fewer disruptions and

interventions. When you filter your data using your partition column, only a subset of the data is accessed, not the entire table. Similarly, maintenance operations on a partitioned table will reduce maintenance duration, for example, by compressing specific data in a particular partition or rebuilding specific partitions of an index.

There are four main steps required when defining a table partition:

- The filegroups creation, which defines the files involved when the partitions are created.
- The partition function creation, which defines the partition rules based on the specified column.
- The partition scheme creation, which defines the filegroup of each partition.
- The table to be partitioned.

The example below illustrates how to create a partition function for January 1, 2021 through December 1, 2021, and distribute the partitions across different filegroups.

```
-- Partition function
CREATE PARTITION FUNCTION PartitionByMonth (datetime2)
AS RANGE RIGHT
-- The boundary values defined is the first day of each month, where the table
FOR VALUES ('20210101', '20210201', '20210301',
            '20210401', '20210501', '20210601', '20210701',
            '20210801', '20210901', '20211001', '20211101',
            '20211201');

-- The partition scheme below will use the partition function created above, and as
CREATE PARTITION SCHEME PartitionByMonthSch
AS PARTITION PartitionByMonth
TO (FILEGROUP1, FILEGROUP2, FILEGROUP3, FILEGROUP4,
    FILEGROUP5, FILEGROUP6, FILEGROUP7, FILEGROUP8,
    FILEGROUP9, FILEGROUP10, FILEGROUP11, FILEGROUP12);

-- Creates a partitioned table called Order that applies PartitionByMonthSch partiti
CREATE TABLE Order ([Id] int PRIMARY KEY, OrderDate datetime2)
ON PartitionByMonthSch (OrderDate) ;
GO
```

Data compression

SQL Server offers different options of [data compression](#). While SQL Server still stores compressed data on 8 KB pages, when the data is compressed, more rows of data can be stored on a given page, which allows the query to read fewer pages. Reading fewer pages has a twofold benefit: it reduces the amount of physical IO performed and it allows more rows to be stored in the buffer pool, making more efficient use of memory. We recommend enabling database page compression where appropriate.

The tradeoffs to compression are that it does require a small amount of CPU overhead; however, in most cases, the storage IO benefits far outweigh any additional processor usage.

```
SQLQuery6.sql - dca...14 (jdantoni (107))* X SQLQuery4.sql - VM...VM1\jdantoni (78)) SQ

SELECT
    COUNT(*)
FROM    Production.TransactionHistory
WHERE   TransactionDate > '2008-01-01'

SELECT
    COUNT(*)
FROM    Production.TransactionHistory_Page
WHERE   TransactionDate > '2008-01-01'

100 %
Results Messages

(1 row affected)
Table 'TransactionHistory'. Scan count 1, logical reads 992, physical read

(1 row affected)
Table 'TransactionHistory_Page'. Scan count 1, logical reads 273, physical

Completion time: 2020-04-22T14:55:13.3744311+00:00
```

The image above shows this performance benefit. These tables have the same underlying indexes; the only difference is that the clustered and nonclustered indexes on the *Production.TransactionHistory_Page* table are page compressed. The query against the page compressed object performs 72% fewer logical reads than the query that uses the uncompressed objects.

Compression is implemented in SQL Server at the object level. Each index or table can be compressed individually, and you have the option of compressing partitions within a partitioned table or index. You can evaluate how much space you'll save by using the `sp_estimate_data_compression_savings` system stored procedure. Prior to SQL Server 2019, this procedure didn't support columnstore indexes or columnstore archival compression.

- **Row compression** - Row compression is fairly basic and doesn't incur much overhead; however, it doesn't offer the same amount of compression (measured by the percentage reduction in storage space required) that page compression may offer. Row compression basically stores each value in each column in a row in the minimum amount of space needed to store that value. It uses a variable-length storage format for numeric data types like integer, float, and decimal, and it stores fixed-length character strings using variable length format.
- **Page compression** - Page compression is a superset of row compression, as all pages will initially be row compressed prior to applying the page compression. Then a combination of techniques called prefix and dictionary compression are applied to the data. Prefix

compression eliminates redundant data in a single column, storing pointers back to the page header. After that step, dictionary compression searches for repeated values on a page and replaces them with pointers, further reducing storage. The more redundancy in your data, the greater the space savings when you compress your data.

- **Columnstore archival compression** - Columnstore objects are always compressed; however, they can be further compressed using archival compression, which uses the Microsoft *XPRESS* compression algorithm on the data. This type of compression is best used for data that is infrequently read but needs to be retained for regulatory or business reasons. While this data is further compressed, the CPU cost of decompression tends to outweigh any performance gains from IO reduction.

Additional options

Below is a list of additional SQL Server features and actions to consider for production workloads:

- Enable backup compression
- Enable instant file initialization for data files
- Limit autogrowth of the database
- Disable autoshrink/autoclose for the databases
- Move all databases to data disks, including system databases
- Move SQL Server error log and trace file directories to data disks
- Set max SQL Server memory limit
- Enable lock pages in memory
- Enable optimize for adhoc workloads for OLTP heavy environments
- Enable Query Store.
- Schedule SQL Server Agent jobs to run DBCC CHECKDB, index reorganize, index rebuild, and update statistics jobs
- Monitor and manage the health and size of the transaction log files

For more information about performance best practices, see [Best practices for SQL Server on Azure VMs](#).

5. Explain high availability and disaster recovery options

<https://learn.microsoft.com/en-us/training/modules/deploy-iaas-solutions-with-azure-sql/5-explain-high-availability-and-disaster-recovery-options>

Explain high availability and disaster recovery options

Completed

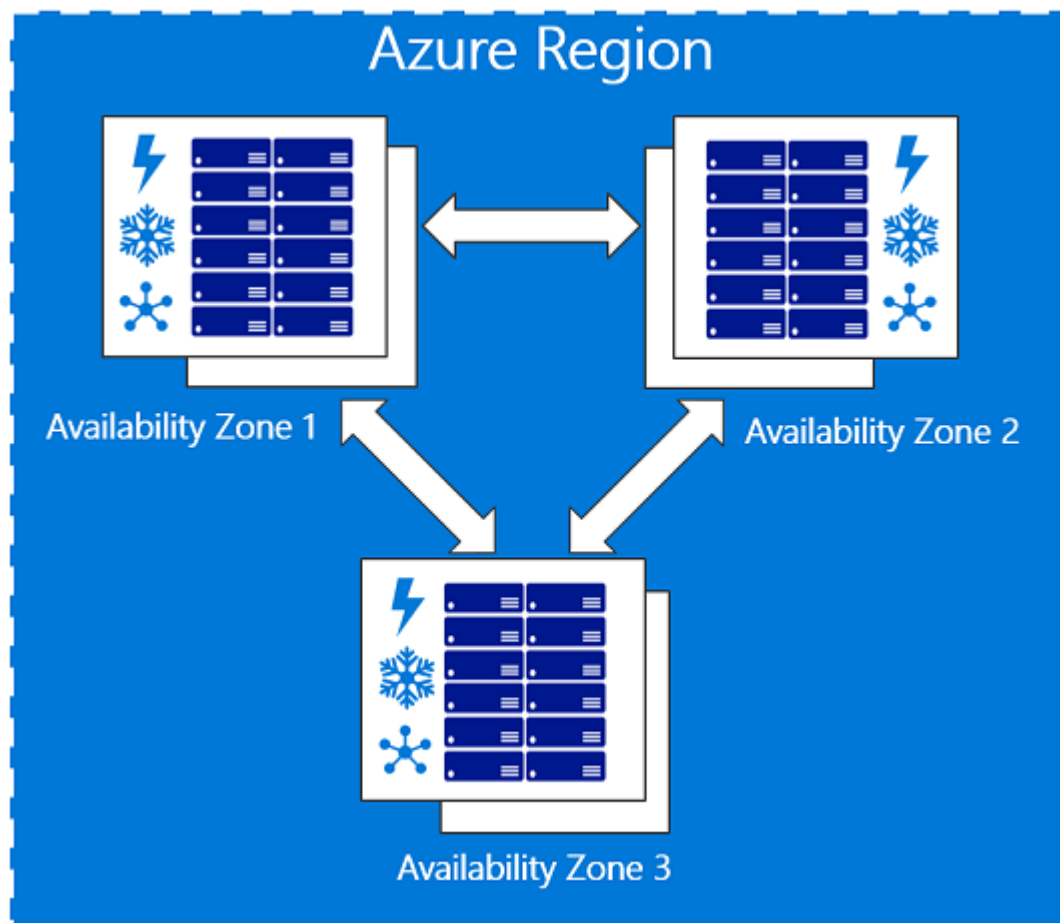
Beyond its built-in high availability, the Azure platform offers two options for providing higher levels of availability for VM and some PaaS workloads. Availability Zones and Availability Sets protect your workloads from planned maintenance activity and potential hardware failures.

High availability options

Most SQL Server high availability solutions are available on Azure virtual machines (VMs). In an Azure-only solution, the entire HADR system runs in Azure. In a hybrid configuration, part of the solution runs in Azure and the other part runs on-premises in your organization. The flexibility of the Azure environment enables you to move partially or completely to Azure to satisfy the budget and HADR requirements of your SQL Server database systems.

Availability Zones

Availability Zones are unique physical locations within a region. Each zone is made up of one or more data centers equipped with independent power, cooling, and networking. Within Azure regions that support Availability Zones, you can specify in which zone you want the virtual machine to reside when you choose to use Available Zones during VM creation. There are three Availability Zones within each supported Azure region. Availability Zones provide high availability against data center failures when you deploy multiple VMs into different zones. In addition, they also provide a means for Microsoft to perform maintenance (using a grouping called an update domain) within each region by only updating one zone at any given time. You can spread out your virtual machine ecosystem across three zones in the region. Utilizing Availability Zones in conjunction with your Azure virtual machines raises your uptime to four nines (99.99%) which equates to a maximum of 52.60 minutes of downtime per year. You can identify which Azure regions support Availability Zones at docs.microsoft.com. If Availability Zones are available in your region, and your application can support the minimal cross-zone latency, Availability Zones will provide the highest level of availability for your application.



When you deploy a VM into a region with an availability zone you have the option to deploy in Zone 1, 2, and 3. These zones are logical representations of physical data centers, which means a deployment to Zone 1 in one subscription, does not mean that Zone 1 represents the same data center in another subscription.

Availability Sets

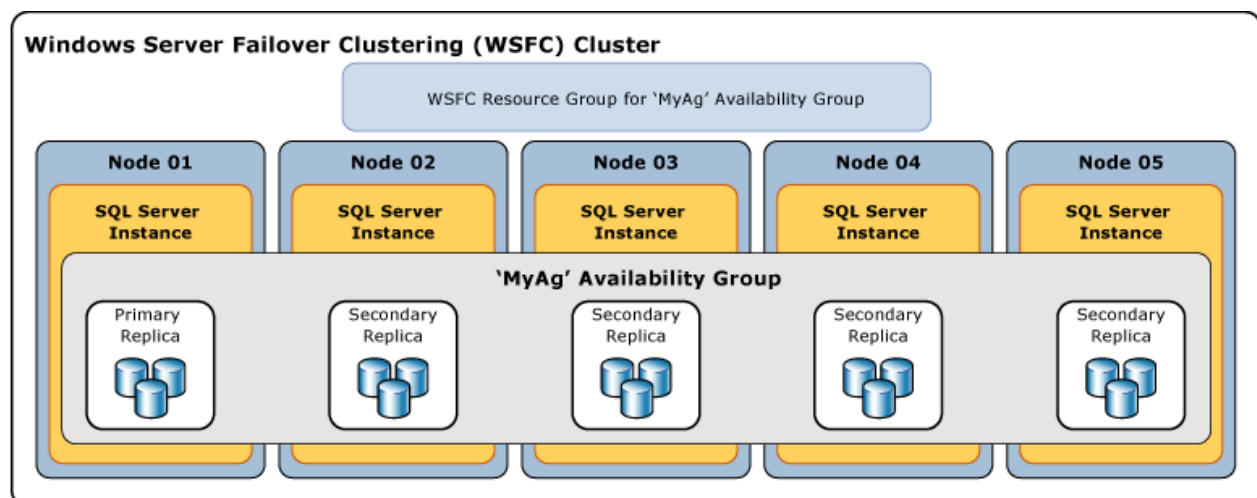
Availability sets are similar to Availability Zones, except instead of spreading workloads across data centers in a region, they spread workloads across servers and racks in a data center. Since nearly all workloads in Azure are virtual, you can use availability sets in order to guarantee that the two VMs containing your Always On Availability Group members are not running on the same physical host. Availability sets can provide up to 99.95% availability, and should be used when Availability Zones are unavailable in a region, or an application cannot tolerate intra-zone latency.

Always On availability groups (AG)

Always On availability groups can be implemented between two or more (up to a maximum of nine) SQL Server instances running on Azure virtual machines or across an on-premises data center and Azure. In an availability group, database transactions are committed to the primary replica, and then the transactions are sent either synchronously or asynchronously to all secondary replicas. The physical distance between the servers (that is, whether or not they are in the same Azure region) dictates which availability mode you should choose. Generally, if the workload requires the lowest

possible latency or the secondary replicas are geographically spread apart, asynchronous availability mode is recommended. If the replicas are within the same Azure region and the applications can withstand some level of latency, synchronous commit mode should be considered. Synchronous mode will help to ensure that each transaction is committed to one or more secondaries before allowing the application to continue. Always On availability groups provide both high availability and disaster recovery, because a single availability group can support both synchronous and asynchronous availability modes. The unit of failover for an availability group is a group of databases, and not the entire instance.

Always On Availability Groups can also be used for disaster recovery purposes. You can implement up to nine replicas of a database across Azure regions, and stretch this architecture even further using Distributed Availability Groups. Availability Groups ensure that a viable copy of your database(s) is in another location beyond the primary region. By doing so, you help to ensure that your data ecosystem is protected against natural disasters as well as some human made ones.



The image above shows a logical diagram of an Always On Availability Group, running on a Windows Server Failover Cluster. There are one primary and four secondary replicas. In this scenario all five replicas could be synchronous, or some combination of synchronous and asynchronous replicas. Keep in mind that the unit of failover is the group of databases and not the instance. While a failover cluster instance provides HA at an instance level, it doesn't provide disaster recovery.

SQL Server Failover Cluster instances

If you need to protect the entire instance, you could use a SQL Server Failover Cluster Instance (FCI), which provides high availability for an entire instance, in a single region. An FCI doesn't provide disaster recovery without being combined with another feature like availability groups or log shipping. FCIs also require shared storage that can be provided on Azure by using shared file storage or using Storage Spaces Direct on Windows Server.

For Azure workloads, availability groups are the preferred solution for newer deployments, because the shared storage required of FCIs increases the complexity of deployments. However, for migrations from on-premises solutions, an FCI may be required for application support.

Disaster Recovery options

While the Azure platform offers 99.9% up time by default, disasters can still occur and affect application uptime. It's important that you have a proper disaster recovery plan in place when you are performing any type of migration. Azure offers us several methods to ensure that your SQL Server on a virtual machine is protected in case of a disaster. There are two components to this protection. First, there are Azure platform options like geo-replicated storage for backups and Azure Site Recovery, which is an all-encompassing disaster recovery solution for all of your workloads. Second, there are SQL Server specific offerings like Availability Groups and backups.

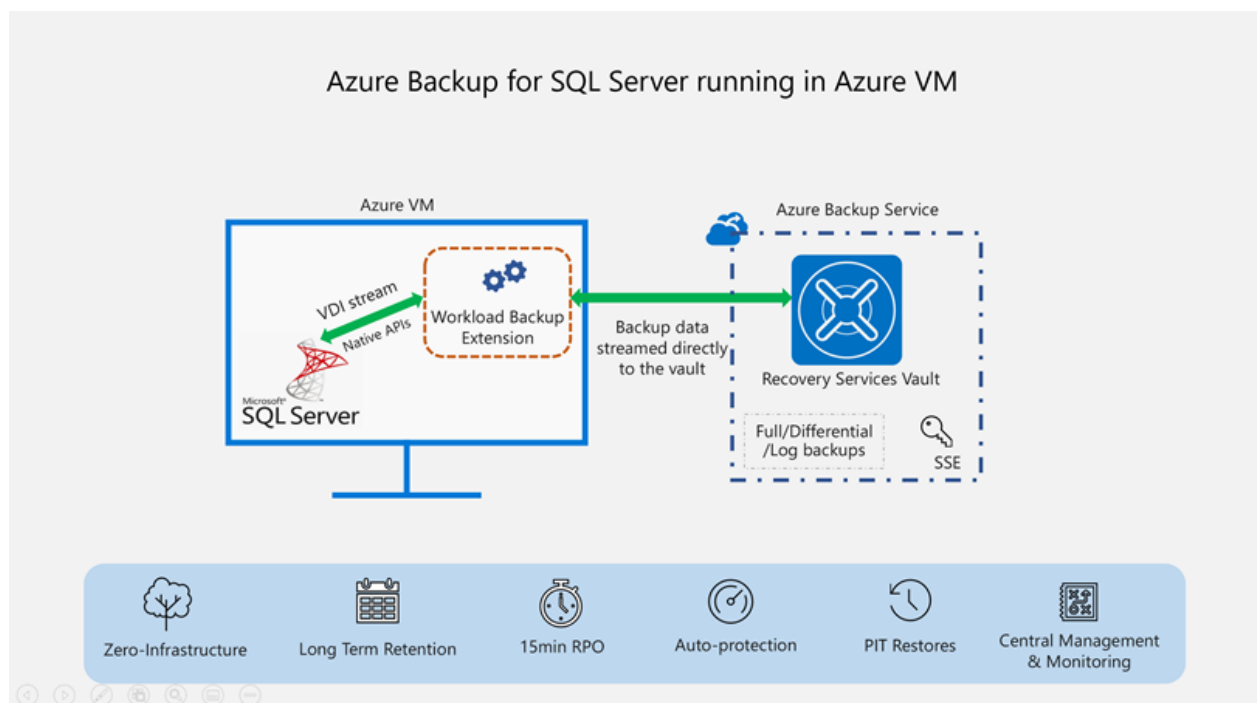
Native SQL Server backups

Backups are considered the life blood of any database administrator and it's not any different when working with a cloud solution. With SQL Server on an Azure virtual machine, you have granular control of when backups occur and where they're stored. You can use SQL agent jobs to back up directly to a URL linked to Azure blob storage. Azure provides the option to use geo-redundant storage (GRS) or read-access geo-redundant storage (RA-GRS) to ensure that your backup files are stored safely across the geographic landscape.

Additionally as part of the Azure SQL VM service provider, you can have your backups automatically managed by the platform.

Azure Backup for SQL Server

The Azure Backup solution requires an agent to be installed on the virtual machine. The agent then communicates with an Azure service that manages automatic backups of your SQL Server databases. Azure Backup also provides a central location that you can use to manage and monitor the backups to ensure meeting any specified RPO/RTO metrics.

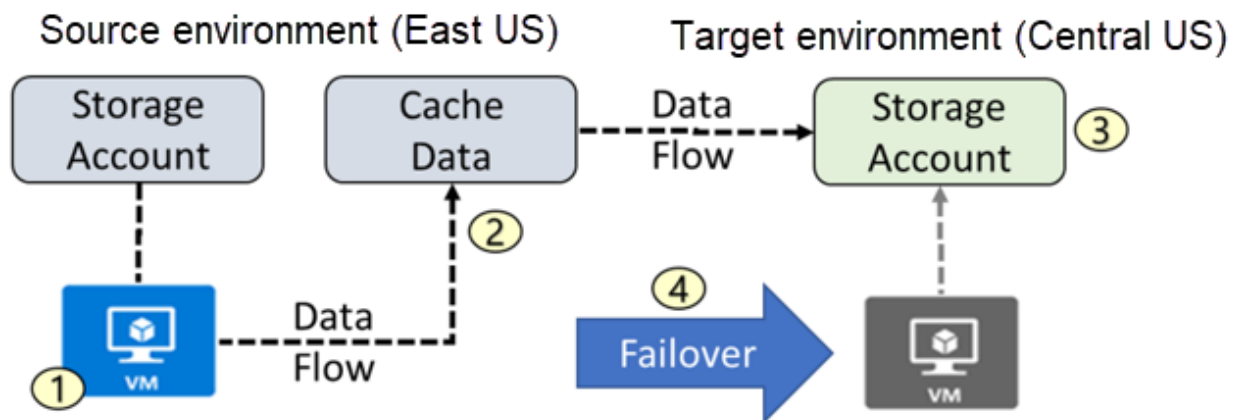


As shown above, the Azure Backup solution is a comprehensive, enterprise backup solution that provides long-term data retention, automated management, and additional data protection. This option costs more than simply performing your own backups, or using the Azure resource provider for SQL Server, but does offer a more complete backup feature set.

Azure Site Recovery

Azure Site Recovery is a low-cost solution that will perform block level replication of your Azure virtual machine. This service offers various options, including the ability to test and verify your disaster recovery strategy. This solution is best used for stateless environments (for example, web servers) versus transactional database virtual machines.

Azure Site Recovery is supported for use with SQL Server, but keep in mind that you will need to set a higher recovery point which means potential loss. In this case, your RTO will essentially be your RPO.



1. VM is registered with Azure Site Recovery
2. Data is continuously replicated to cache
3. Cache is replicated to the target storage account
4. During failover the virtual machine is added to the target environment

6. Exercise: Provision a SQL Server on an Azure Virtual Machine

<https://learn.microsoft.com/en-us/training/modules/deploy-iaas-solutions-with-azure-sql/6-exercise-provision-sql-server-azure-virtual-machine>

Exercise: Provision a SQL Server on an Azure Virtual Machine

Completed

Now it's your chance to deploy a SQL Server on an Azure Virtual Machine.

In this exercise, you'll explore the Azure portal and use it to create an Azure VM with SQL Server installed. Then they'll connect to the virtual machine through Remote Desktop Protocol.

Note

To complete this exercise, you'll need a Microsoft Azure subscription. If you don't already have one, you can sign up for a free trial at <https://azure.com/free>.

Launch the exercise and follow the instructions.

[Launch Exercise](#)

7. Module assessment

<https://learn.microsoft.com/en-us/training/modules/deploy-iaas-solutions-with-azure-sql/7-knowledge-check>

Module assessment

Completed

8. Summary

<https://learn.microsoft.com/en-us/training/modules/deploy-iaas-solutions-with-azure-sql/8-summary>

Summary

Completed

Azure provides a great deal of flexibility for deploying SQL Server to an Azure virtual machine. The hybrid licensing options reduce your cost. It also allows for additional protection for high availability and disaster recovery as part of your software assurance benefits. Azure Resource Manager offers many ways to deploy your resources in both a programmatic and graphical fashion.

Now that you've reviewed this module, you should be able to:

- Explore the basics of SQL Server in an Infrastructure as a Service (IaaS) offering
- Learn how hybrid scenario works
- Explore performance and security options available
- Understand the high availability and disaster recovery options

Deploy PaaS solutions with Azure SQL

1. Introduction

<https://learn.microsoft.com/en-us/training/modules/deploy-paas-solutions-with-azure-sql/1-introduction>

Introduction

Completed

The Platform as a Service (PaaS) offering for SQL Server can be a great solution for certain workloads. The PaaS offering provides less granular control over the infrastructure. It also relegates management of the underlying components (memory, CPU, storage, operating system, etc.) to Microsoft Azure.

This module focuses on ways to provision and deploy Azure SQL Database and Azure SQL managed instances, as well as provide guidance on the various options when performing a migration to these platforms.

Learning objectives

At the end of this module, you'll be able to:

- Understand PaaS provisioning and deployment options
- Understand elastic pools and hyperscale features
- Examine SQL Managed Instances

2. Explain PaaS options for deploying SQL Server in Azure

<https://learn.microsoft.com/en-us/training/modules/deploy-paas-solutions-with-azure-sql/2-explain-paas-options-deploy-sql-server-azure>

Explain PaaS options for deploying SQL

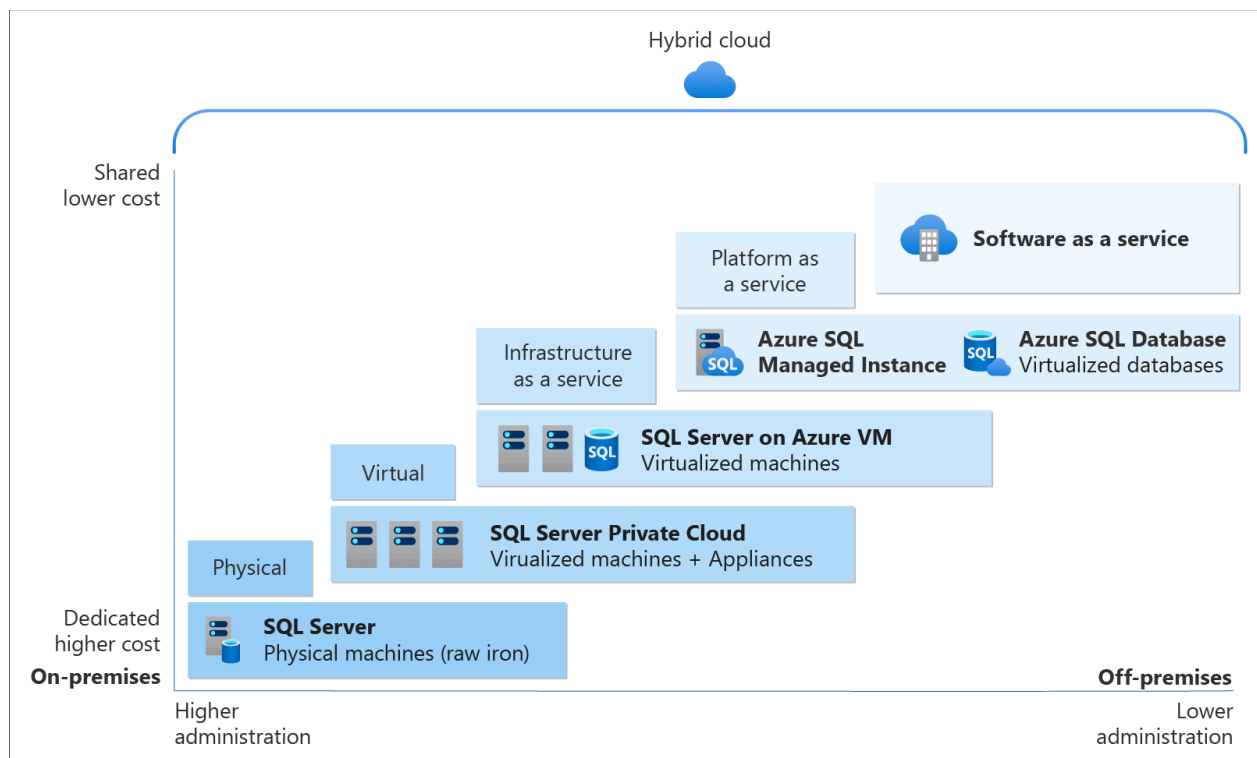
Server in Azure

Completed

Platform as a Service (PaaS) provides a complete development and deployment environment in the cloud, which can be used for simple cloud-based applications and for advanced enterprise applications.

Azure SQL Database and Azure SQL Managed Instance are part of PaaS offering for Azure SQL.

- **Azure SQL Database** – Part of a family of products built upon the SQL Server engine, in the cloud. It gives developers a great deal of flexibility in building new application services, and granular deployment options at scale. SQL Database offers a low maintenance solution that can be a great option for certain workloads.
- **Azure SQL Managed Instance**– It's best for most migration scenarios to the cloud as it provides fully managed services and capabilities.



Each offering provides a certain level of administration you have over the infrastructure, by the degree of cost efficiency.

Deployment models

Azure SQL Database is available in two different deployment models:

- **Single database** – A single database that is billed and managed on a per-database level. You manage each of your databases individually from scale and data size perspectives. Each database deployed in this model has its own dedicated resources, even if deployed to the same logical server.
- **Elastic Pools** – A group of databases that are managed together and share a common set of resources. Elastic pools provide a cost-effective solution for software as a service application model, as resources are shared between all databases. You can configure resources based either on the DTU-based purchasing model or the vCore-based purchasing model.

Purchasing model

In Azure, all services support physical hardware, and you have the flexibility to choose between two different purchasing models:

Database Transaction Unit (DTU)

[DTU-based purchasing model](#) is calculated based on a formula combining compute, storage, and I/O resources. It's a good choice for customers who want simple, preconfigured resource options.

The DTU purchasing model comes in several different service tiers, such as Basic, Standard, and Premium. Each tier has varying capabilities, providing a wide range of options when choosing this platform.

In terms of performance, the Basic tier is used for less demanding workloads, while the Premium tier is used for intensive workload requirements.

Compute and storage resources are dependent on the DTU level, and they provide a range of performance capabilities at a fixed storage limit, backup retention, and cost.

For more information about DTU purchasing model, see [DTU-based purchasing model overview](#).

vCore

The [vCore-based purchasing model](#) allows you to purchase a specified number of vCores based on your given workloads. vCore is the default purchasing model when purchasing Azure SQL Database resources. vCore databases have a specific relationship between the number of cores and the amount of memory and storage provided to the database. The vCore purchasing model is supported by both Azure SQL Database and Azure SQL Managed Instance.

You can purchase vCore databases in three different service tiers as well:

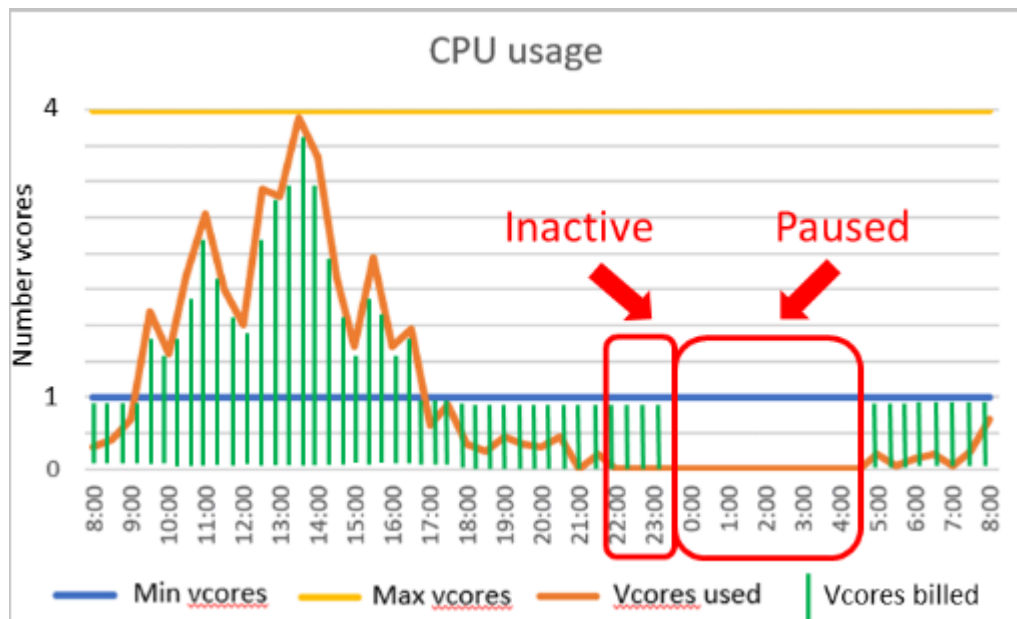
Service Tier	Best For	Storage Type	Latency	Compute Tiers	Resilience Features	Max Database Size
General Purpose	General-purpose	Azure premium	Higher than BC	Provisioned, Serverless	N/A	4 TB

Service Tier	Best For	Storage Type	Latency	Compute Tiers	Resilience Features	Max Database Size
	workloads	storage				
Business Critical	High-performing workloads	Local SSDs	Lowest	Provisioned	Built-in read-only replica, highest resilience to failure	4 TB
Hyperscale	Large-scale databases	Azure premium storage	Varies	Provisioned	Unique architecture for scaling	100 TB

The General Purpose service tier offers two compute options: **Provisioned** and **Serverless**. Provisioned resources are preallocated and billed hourly based on vCores configured, ideal for consistent workloads. Serverless resources autoscale based on demand and are billed per second, making it cost-efficient for variable workloads.

Serverless

The term "Serverless" can be misleading because you still deploy your Azure SQL Database to a logical server for connection. [Serverless](#) is a compute tier that automatically scales resources up or down based on workload demand. When the workload no longer requires compute resources, the database is paused, and only storage is billed during the inactive period. Upon a connection attempt, the database resumes and becomes available.



The setting to control pausing is called the autopause delay, with a minimum value of 60 minutes and a maximum value of seven days. If the database remains idle for that duration, it pauses.

Once the database is inactive for the specified time, it pauses until a subsequent connection attempt. Configuring a compute autoscaling range and an autopause delay affects database performance and compute costs.

Applications using serverless should be configured to handle connection errors and include retry logic, as connecting to a paused database generates a connection error.

Another difference between serverless and the standard vCore model of Azure SQL Database is that with serverless, you can specify a minimum and maximum number of vCores. Memory and I/O limits are proportional to the specified range.

Compute Hardware

Select the hardware configuration based on your workload requirements. Availability of compute optimized, memory optimized, and confidential computing hardware depends on the region, service tier, and compute tier.

Hardware Configuration

Standard-series (Gen5)
up to 80 vCores, up to 240 GB memory
[Change configuration](#)

Max vCores

Min vCores

2.02 GB MIN MEMORY 3 GB MAX MEMORY

The image shows the configuration screen for a serverless database in the Azure portal. You have the option to select a minimum as low as half of a vCore and a maximum as high as 16 vCores.

Serverless isn't fully compatible with all the features in Azure SQL Database since some of them require background processes to run constantly, such as:

- Geo-replication
- Long-term backup retention
- A job database in elastic jobs
- The sync database in SQL Data Sync (Data Sync is a service that replicates data between a group of databases)

Note

Serverless is currently only supported in the General Purpose tier in the vCore purchasing model.

Backups

One of the most important features of the Platform as a Service offering is backups. In this case, the system automatically performs backups without any intervention from you. Azure blob geo-

redundant storage stores these backups, and by default, retains them for between 7 and 35 days, depending on the service tier of the database. Basic and vCore databases default to seven days of retention, and administrators can adjust this value for vCore databases. You can extend the retention time by configuring long-term retention (LTR), allowing you to retain backups for up to 10 years.

In order to provide redundancy, you're also able to use read-accessible geo-redundant blob storage. This storage would replicate your database backups to a secondary region of your preference. It would also allow you to read from that secondary region if needed. Manual backups of databases aren't supported, and the platform will deny any request to do so.

Database Backups are taken on a given schedule:

- **Full** – Once a week
- **Differential** – Every 12 hours
- **Log** – Every 5-10 minutes depending on transaction log activity

This backup schedule should meet the needs of most recovery point/time objectives (RPO/RTO), however, each customer should evaluate whether they meet your business requirements.

There are several options available for restoring a database. Due to the nature of Platform as a Service, you can't manually restore a database using conventional methods, such as issuing the T-SQL command `RESTORE DATABASE`.

Regardless of which restore method is implemented, it isn't possible to restore over an existing database. If a database needs to be restored, the existing database must be dropped or renamed prior to initiating the restore process. Furthermore, keep in mind that depending on the platform service tier, restore times aren't guaranteed and could fluctuate. It's recommended that you test the restore process to obtain a baseline metric on how long a restore could potentially take.

- **Restore using the Azure portal** – Using the Azure portal you have the option of restoring a database to the same logical server for Azure SQL Database, or you can use the restore to create a new database on a new server in any Azure region.
- **Restore using scripting Languages** – Both PowerShell and Azure CLI can be utilized in order to restore a database.

Note

Copy-only backup to Azure blob storage is available for SQL Managed Instance. SQL Database doesn't support this feature.

For more information about automated backups, see [Automated backups - Azure SQL Database & Azure SQL Managed Instance](#).

Active geo-replication

Geo-replication is a business continuity feature that asynchronously replicates a database to up to four secondary replicas. As transactions are committed to the primary (and its replicas within the same region), the transactions are sent to the secondaries to be replayed. Since this communication is done asynchronously, the calling application doesn't have to wait for the secondary replica to commit the transaction before SQL Server returns control to the caller.

The secondary databases are readable and can be used to offload read-only workloads, thus freeing up resources for transactional workloads on the primary or placing data closer to your end users. Furthermore, the secondary databases can be in the same region as the primary or in another Azure region.

You can initiate a failover either manually by the user or programmatically by the application. If a failover occurs, you can update the application connection strings to reflect the new endpoint of what is now the primary database.

Failover groups

Failover groups are built on top of the technology used in geo-replication but provide a single endpoint for connection. The major reason for using failover groups is that they provide endpoints that can be utilized to route traffic to the appropriate replica. Your application can then connect after a failover without connection string changes.

3. Explore single SQL database

<https://learn.microsoft.com/en-us/training/modules/deploy-paas-solutions-with-azure-sql/3-explore-single>

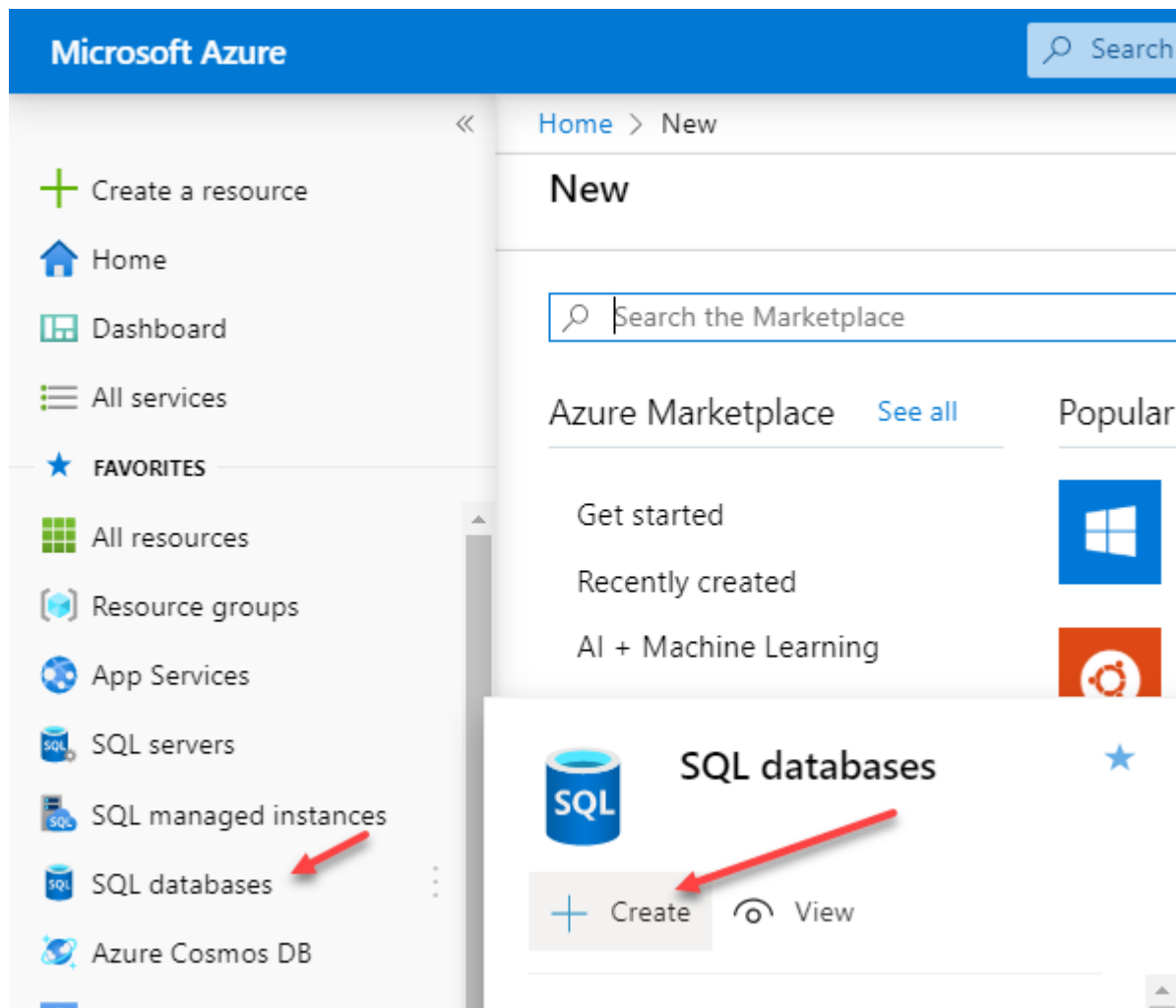
Explore single SQL database

Completed

Let's look at several methods for deploying a single Azure SQL Database.

Deploying via the portal

Creating an SQL database through the Azure portal is simple. First, navigate to the Azure portal and select "SQL Databases" from the left-hand menu. In the slide-out panel that appears, select "Create."



In the pane in the image below, you'll notice that the subscription should already be provided for you. You'll need to supply the following information:

- **Resource Group** – If you have an existing resource group you wish to use, select it from the drop-down list. To create a new resource group for this Azure SQL Database, choose the **Create new** option.
- **Database Name** – Provide a name for your database.
- **Server** – Each database must reside on a logical server. If you already have one in the appropriate region, you can select it. Otherwise, select the **Create new** link and follow the prompts to create a new logical server to host the database.
- **Want to use SQL elastic pool?** – Decide whether to use an elastic pool.
- **Compute + storage** – Determine the appropriate compute resources needed. By default, it's set to Gen5, 2vCore, with 32 GB of storage. Select **Configure database** to view and choose alternate configuration options.

[Basics](#) [Networking](#) [Security](#) [Additional settings](#) [Tags](#) [Review + create](#)

Create a SQL database with your preferred configurations. Complete the Basics tab then go to Review + Create to provision with smart defaults, or visit each tab to customize. [Learn more](#)

Want to try Azure SQL Database for free? Create a free serverless database with the first 100,000 vCore seconds, 32GB of data, and 32GB of backup storage free per month for the lifetime of the subscription. Limit ten free databases per subscription. [Learn more](#)

[Apply offer](#)

SQL Database Hyperscale: Low price, high scalability, and best feature set. [Learn more](#)

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * [?](#)

Resource group * [?](#)

[Create new](#)

Database details

Enter required settings for this database, including picking a logical server and configuring the compute and storage resources

Database name *

Server * [?](#)

[Create new](#)

Want to use SQL elastic pool? [?](#) ☐ Yes ☒ No

Workload environment ☒ Development ☐ Production

[?](#) Default settings provided for Development workloads. Configurations can be modified as needed.

Compute + storage * [?](#)

General Purpose - Serverless
Standard-series (Gen5), 1 vCore, 32 GB storage
[Configure database](#)

Here, you'll notice that the service tier is General Purpose and the compute tier is Serverless, as discussed previously. Serverless is billed per second based on the number of vCores in use. The alternative option is Provisioned, where compute resources are preallocated and billed per hour based on the number of configured vCores.

Service and compute tier

Select from the available tiers based on the needs of your workload. The vCore model provides a wide range of configuration controls and offers Hyperscale and Serverless to automatically scale your database based on your workload needs. Alternately, the DTU model provides set price/performance packages to choose from for easy configuration. [Learn more](#)

SQL Database Hyperscale: Low price, high scalability, and best feature set. [Learn more](#)

Service tier General Purpose (Most budget friendly)

[Compare service tiers](#)

Compute tier

☒ **Provisioned** - Compute resources are pre-allocated. Billed per hour based on vCores configured.

☐ **Serverless** - Compute resources are auto-scaled. Billed per second based on vCores used.

Compute Hardware

Select the hardware configuration based on your workload requirements. Availability of compute optimized, memory optimized, and confidential computing hardware depends on the region, service tier, and compute tier.

Hardware Configuration

Standard-series (Gen5)

up to 128 vCores, up to 625 GB memory

[Change configuration](#)

Save money

Already have a SQL Server License? Save with a license you already own with Azure Hybrid Benefit. Actual savings may vary based on region and performance tier. [Learn more](#)

☐ Yes ☒ No

vCores [Compare vCore options](#)

2

Data max size (GB)

32

9.6 GB LOG SPACE ALLOCATED

Deploying an Azure SQL Database via PowerShell/CLI

You can also deploy your database using Azure PowerShell or the Azure CLI. The image below shows the PowerShell example where you're creating a new resource group, and defining an admin called SqlAdmin and then creating a new server, database, and firewall rule.

```
# Connect-AzAccount

# The SubscriptionId in which to create these objects
$SubscriptionId = ''

# Set the resource group name and location for your server
$resourceGroupName = "myResourceGroup-$(Get-Random)"
$location = "westus2"

# Set an admin login and password for your server
$adminSqlLogin = "SqlAdmin"
$password = "ChangeYourAdminPassword1"
```

```
# Set server name - the logical server name has to be unique in the system
$serverName = "server-$(Get-Random)"

# The sample database name
$databaseName = "mySampleDatabase"

# The ip address range that you want to allow to access your server
$startIp = "0.0.0.0"
$endIp = "0.0.0.0"

# Set subscription
Set-AzContext -SubscriptionId $subscriptionId

# Create a resource group
$resourceGroup = New-AzResourceGroup -Name $resourceGroupName -Location $location

# Create a server with a system wide unique server name
$server = New-AzSqlServer -ResourceGroupName $resourceGroupName `
    -ServerName $serverName `
    -Location $location `
    -SqlAdministratorCredentials $(New-Object -TypeName System.Management.Automation.PSCredential -ArgumentList $serverName, (Get-Random -Bytes 128))

# Create a server firewall rule that allows access from the specified IP range
$serverFirewallRule = New-AzSqlServerFirewallRule -ResourceGroupName $resourceGroupName `
    -ServerName $serverName `
    -FirewallRuleName "AllowedIPs" -StartIpAddress $startIp -EndIpAddress $endIp

# Create a blank database with an S0 performance level
$database = New-AzSqlDatabase -ResourceGroupName $resourceGroupName `
    -ServerName $serverName `
    -DatabaseName $databaseName `
    -RequestedServiceObjectiveName "S0" `
    -SampleName "AdventureWorksLT"
```

The Azure CLI can also be used to deploy an Azure SQL Database as shown below:

```
#!/bin/bash

# set execution context (if necessary)
az account set --subscription <replace with your subscription name or id>

# Set the resource group name and location for your server
resourceGroupName=myResourceGroup-$(RANDOM)
location=westus2

# Set an admin login and password for your database
adminlogin=ServerAdmin
password=`openssl rand -base64 16`

# password=<EnterYourComplexPasswordHere1>

# The logical server name has to be unique in all of Azure
```

```

servername=server-$RANDOM

# The ip address range that you want to allow to access your DB
startip=0.0.0.0
endip=0.0.0.0

# Create a resource group
az group create \
  --name $resourceGroupName \
  --location $location

# Create a logical server in the resource group
az sql server create \
  --name $servername \
  --resource-group $resourceGroupName \
  --location $location \
  --admin-user $adminlogin \
  --admin-password $password

# Configure a firewall rule for the server

az sql server firewall-rule create \
  --resource-group $resourceGroupName \
  --server $servername \
  -n AllowYourIp \
  --start-ip-address $startip \
  --end-ip-address $endip

# Create a database in the server
az sql db create \
  --resource-group $resourceGroupName \
  --server $servername \
  --name mySampleDatabase \
  --sample-name AdventureWorksLT \
  --edition GeneralPurpose \
  --family Gen4 \
  --capacity 1 \

# Echo random password
echo $password

```

Deploying Azure SQL Database Using Azure Resource Manager templates

Another method for deploying resources is as mentioned earlier using an Azure Resource Manager template. A Resource Manager template gives you the most granular control over your resources, and Microsoft provides a GitHub repository called [Azure-Quickstart-Templates](#), which hosts Azure Resource Manager templates that you can reference in your deployments. A PowerShell example of deploying a GitHub based template is shown below:

```
#Define Variables for parameters to pass to template
$projectName = Read-Host -Prompt "Enter a project name"
$location = Read-Host -Prompt "Enter an Azure location (i.e. centralus)"
$adminUser = Read-Host -Prompt "Enter the SQL server administrator username"
$adminPassword = Read-Host -Prompt "Enter the SQL server administrator password" -/
$resourceGroupName = "${projectName}rg"

#Create Resource Group and Deploy Template to Resource Group
New-AzResourceGroup -Name $resourceGroupName -Location $location

New-AzResourceGroupDeployment -ResourceGroupName $resourceGroupName `
-TemplateUri "https://raw.githubusercontent.com/Azure/azure-quickstart-templates/r
-administratorLogin $adminUser -administratorLoginPassword $adminPassword

Read-Host -Prompt "Press [ENTER] to continue ..."
```

This script automates the creation of an Azure resource group and the deployment of a SQL logical server within it. It prompts the user for a project name, Azure location, and SQL server administrator credentials, then creates a resource group using the provided details. Finally, it deploys a SQL logical server to the resource group using a predefined ARM template from a specified URI, and pauses execution until the user presses `ENTER`.

4. Deploy SQL database elastic pool

<https://learn.microsoft.com/en-us/training/modules/deploy-paas-solutions-with-azure-sql/4-deploy-elastic-pool>

Deploy SQL database elastic pool

Completed

Elastic pools are a deployment option in which you purchase Azure compute resources (CPU, memory, and storage) that is then shared among multiple databases defined as belonging to the same pool. An easy comparison to an on-premises SQL Server is that an elastic pool is like a SQL Server instance that has multiple user databases. By using elastic pools, you can easily manage pool resources while at the same time potentially saving costs. Elastic pools also facilitate easy scalability up to the set limits such that if a single database within the pool needs resources due to an unpredictable workload, the resources are there. If the entire pool needs extra resources, a simple slider option within the Azure portal facilitates scaling the elastic pool up or down.

Creating new elastic pools

Using the Azure portal, search for “SQL elastic pools”. Then select **Create** to open the **Create SQL Elastic pool** page.

Create SQL Elastic pool ...

Microsoft

Basics Security Additional settings Tags Review + create

Create a SQL Elastic pool with your preferred configurations. Elastic pools provide a simple and cost effective solution for managing the performance of multiple databases within a fixed budget. Complete the Basic tab, then go to Review + Create to provision with smart defaults, or visit each tab to customize. [Learn more](#)

SQL Database Hyperscale: Low price, high scalability, and best feature set. [Learn more](#)

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *

Resource group *

[Create new](#)

Elastic pool details

Enter required settings for this pool, including picking a logical server and configuring the compute and storage resources.

Elastic Pool Name *

Server *

[Create new](#)

Compute + storage *

GeneralPurpose
Standard-series (Gen5), 2 vCores, 32 GB, 0 databases.
[Configure elastic pool](#)

[Review + create](#)

[Next : Security >](#)

Adding a database to an existing pool

1. Using the Azure portal, locate the pool to which you're adding a database.

Save
 Cancel
 Feedback

Pool settings
 Databases
 Per database settings
 Always Encrypted

+ Add databases
 ✕ Remove from pool
 ↶ Revert selected

✓ Databases to be removed from pool

Database name	↑↓ Pricing tier	↑↓
Currently, there are no databases selected to be removed from this pool. To remove databases, select databases then click 'Remov...		

✓ Ready to be added to this pool

Database name	↑↓ Pricing tier	↑↓ Data space used	↑↓
Currently, there are no databases selected to be added to the pool. To add databases, click 'Add databases' above.			

✓ Currently in this pool (--/100)

☰ Select all
 ☰ Columns

Database name	↑↓ Status	↑↓ Avg CPU (%)	↑↓ Peak CPU (%)	↑↓ Data space used	↑↓
Currently there are no databases in the pool. To add databases, click 'Add databases' above.					

2. Select + **Add databases** to add your database to the pool, then select **Apply**.

Add databases
✕

☰ Select all
 Selected/Total databases
1/1

Database name	↑↓ Pricing tier	↑↓ Database size	↑↓
☒ my-db	General Purpose - Serverless: Stand...	228.75 MB	

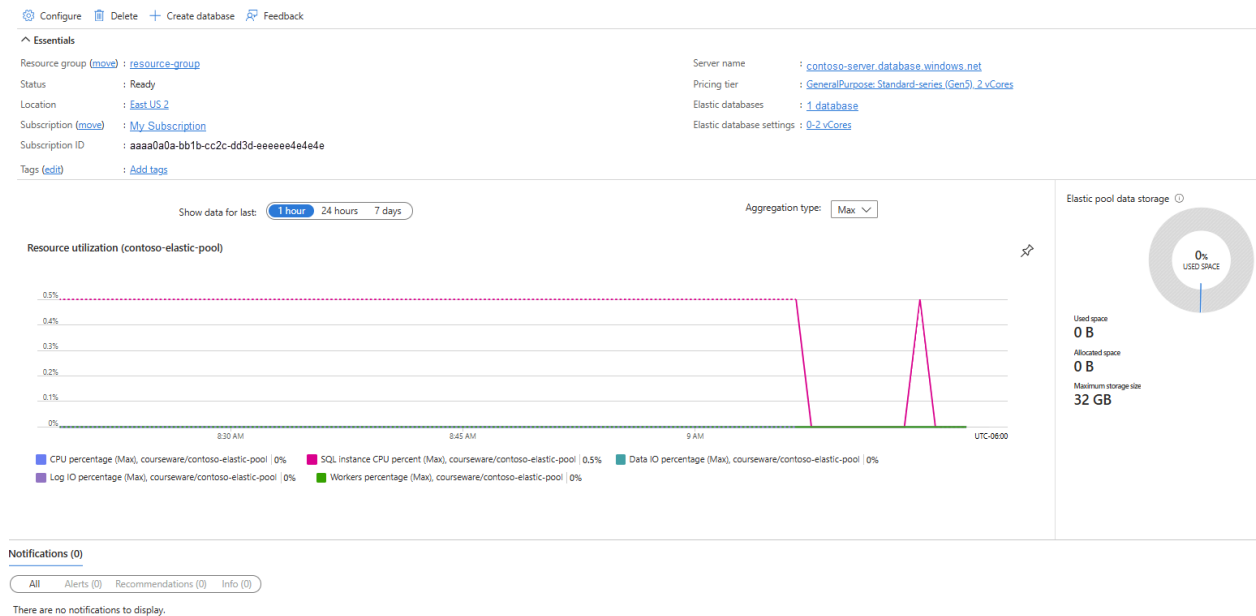
Showing 1 - 1 of 1 results.

Apply

Your database selection is then added to the **Ready to be added to this pool section**. Select **Save**.

Managing pool resources

The Azure portal provides comprehensive insights into the state and health of your elastic pool. You can monitor resource utilization and identify which database is consuming the most resources. This information is valuable for identifying performance issues or determining if a database isn't well-suited for the pool, especially if one database is using most of the resources.

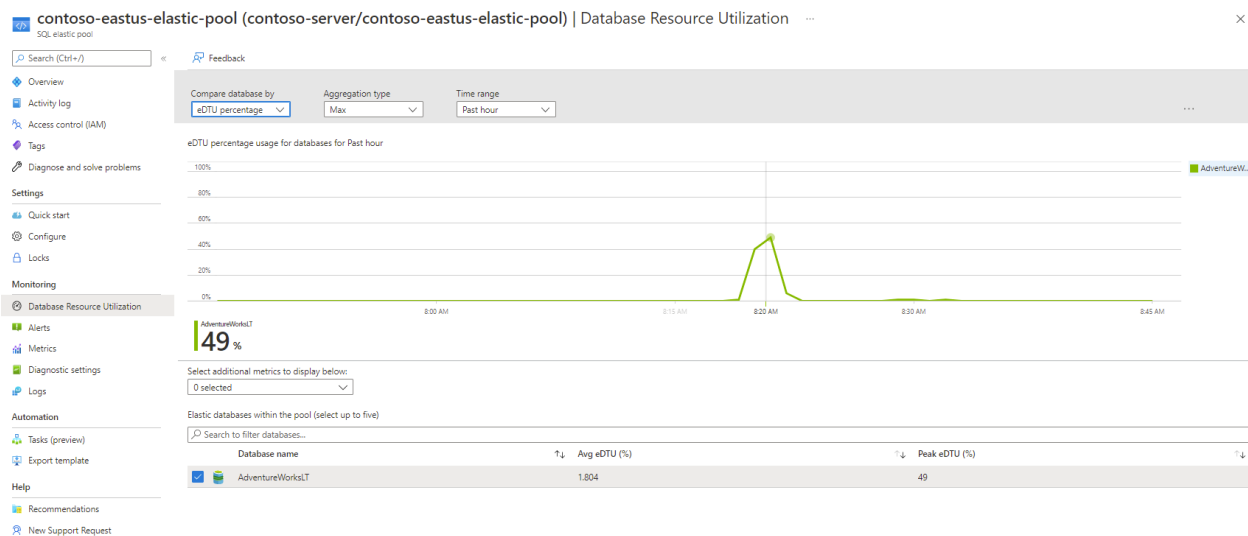


To adjust the resources allocated to your elastic pool, use the **Configure** option in the **Settings** section of the elastic pool management side menu. You can perform many changes after the creation of an elastic pool.

- Pool size, including DTUs, vCores, and storage size.
- Service tier.
- Resources per database.
- Databases included in the pool, by adding or removing them.

Some changes, such as the minimum and maximum DTUs or vCores per database are performed online. You can also change the total size of the pool or add and remove databases as needed. Active connections are ended as the resizing completes.

One of the most useful features is the ability to monitor database resource utilization. This feature provides an easy way to assess the performance of databases within the pool.



An elastic pool is a good fit for multitenant databases where each tenant has its own copy of the database. Balance the workload across databases so as not to allow one database to monopolize all the pool's resources.

5. Understand SQL database hyperscale

<https://learn.microsoft.com/en-us/training/modules/deploy-paas-solutions-with-azure-sql/5-understand-sql-database-hyperscale>

Understand SQL database hyperscale

Completed

Azure SQL Database was historically limited to 4 TB of storage per database due to physical infrastructure constraints. However, the Hyperscale service tier revolutionizes this by allowing databases to exceed 100 TB. Hyperscale uses horizontal scaling techniques to add compute nodes as data sizes grow. While the cost of Hyperscale is similar to Azure SQL Database, there's an extra per terabyte storage cost. It's important to note that once a database is converted to Hyperscale, it can't be reverted to a standard Azure SQL Database.

Hyperscale is ideal for most business workloads, offering flexibility, and high performance with independently scalable compute and storage resources. It separates the query processing engine from the components providing long-term storage and durability, allowing storage capacity to scale smoothly as needed.

The Hyperscale service tier, part of the vCore-based purchasing model, is the newest and most scalable option, significantly exceeding the limits of the General Purpose and Business Critical tiers.

Benefits

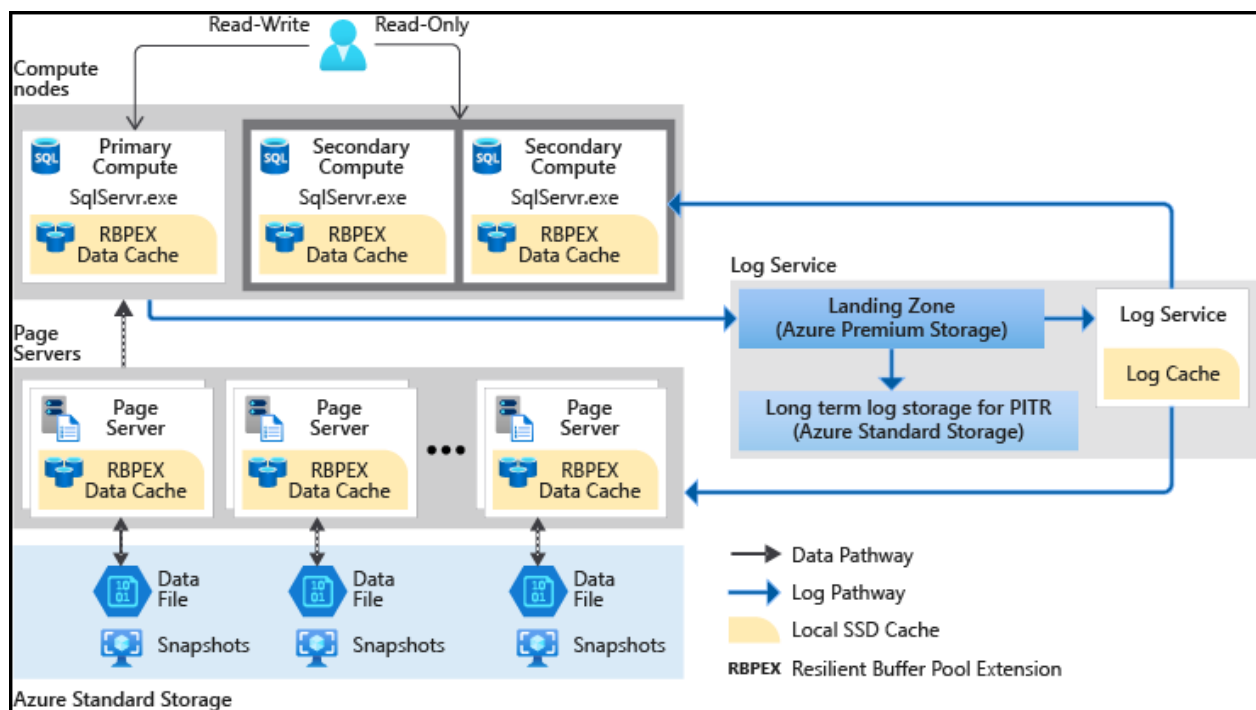
The Hyperscale service tier eliminates many of the practical limitations traditionally associated with cloud databases. The resources of a single node constrain most databases, but Hyperscale databases have no such restrictions. With its flexible storage architecture, storage expands as needed, and there's no predefined maximum size. You're billed only for the capacity you use. For read-intensive workloads, Hyperscale offers rapid scale-out by provisioning more replicas to handle read operations.

Moreover, the time required for database backups or scaling operations is no longer dependent on the data volume. Hyperscale databases can be backed up instantly, and you can scale a database with tens of terabytes up or down in minutes. This flexibility ensures that your initial configuration choices don't constrain you. Additionally, Hyperscale provides fast database restores, completing in minutes rather than hours or days.

Hyperscale provides rapid scalability based on your workload demand.

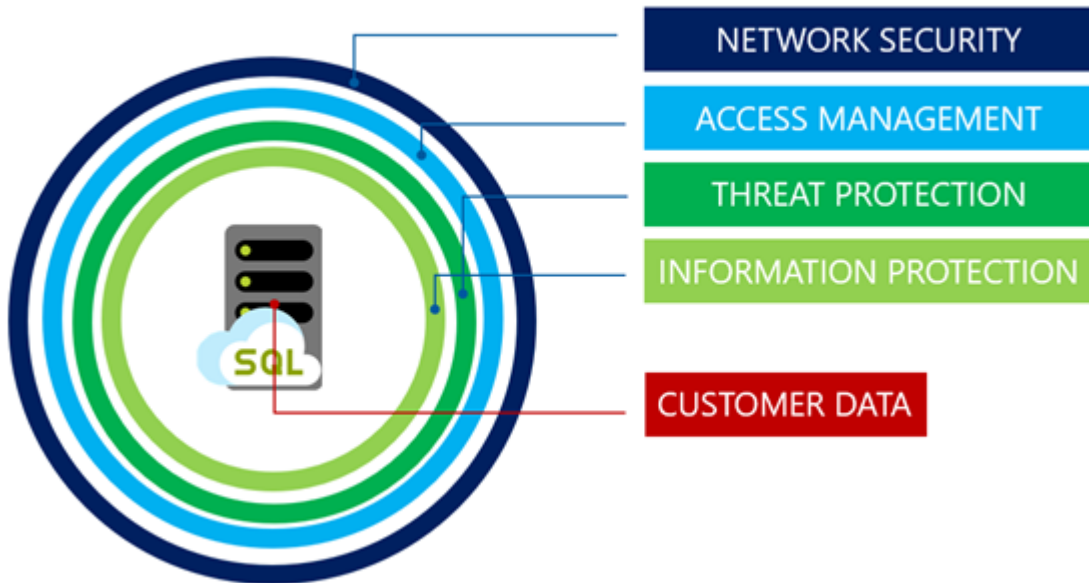
- **Scaling Up/Down** – You can increase or decrease the primary compute resources, such as CPU and memory, quickly and efficiently. Since the storage is shared, these scaling operations aren't dependent on the database's data volume.
- **Scaling In/Out** – You can create more compute replicas to handle read requests, effectively offloading the reading workload from the primary compute. These replicas also serve as hot standby, ready to take over if there's a primary compute failure.

Provisioning more compute replicas is a quick, online operation. To connect to these read-only replicas, set the *ApplicationIntent* argument in your connection string to **ReadOnly**. Connections with the **ReadOnly** application intent are automatically routed to one of the read-only compute replicas.



Security considerations

Security for the Hyperscale service tier offers the same robust capabilities as other Azure SQL Database tiers. It employs a layered defense-in-depth approach, providing comprehensive protection from the outermost layers inward.



- **Network Security** is the first layer of defense, utilizing IP firewall rules to control access based on the originating IP address. Additionally, Virtual Network firewall rules enable communication from selected subnets within a virtual network.
- **Access Management** is provided through the following authentication methods to verify user identity:
 - SQL Authentication
 - Microsoft Entra authentication
 - Windows Authentication for Microsoft Entra principals

Azure SQL Database Hyperscale also supports [Row-Level Security \(RLS\)](#), allowing customers to control access to specific rows in a database table based on user characteristics, such as group membership or execution context.

- **Threat Protection** includes robust auditing and threat detection capabilities. SQL Database and SQL Managed Instance auditing track database activities and help maintain compliance with security standards by recording events to an audit log in a customer-owned Azure storage account. Advanced Threat Protection analyzes your logs to detect unusual behavior and potential threats to your databases. It generates alerts for suspicious activities such as SQL injection, potential data infiltration, brute force attacks, and anomalies in access patterns that can indicate privilege escalations or the use of breached credentials.

- **Information Protection** is provided in the following ways:
 - Transport Layer Security (Encryption-in-transit)
 - Transparent Data Encryption (Encryption-at-rest)
 - Key management with Azure Key Vault
 - Always Encrypted (Encryption-in-use)
 - Dynamic data masking

Performance considerations

The Hyperscale service tier is designed for customers with large on-premises SQL Server databases who want to modernize by moving to the cloud, and for those already using Azure SQL Database who need to significantly expand their database capacity. It's also ideal for customers seeking high performance and scalability.

Key performance capabilities of Hyperscale include:

- Nearly instantaneous database backups using file snapshots stored in Azure Blob storage, without affecting compute resources.
- Fast database restores based on file snapshots, completing in minutes rather than hours or days, regardless of data size.
- Enhanced overall performance due to higher transaction log throughput and faster transaction commit times, irrespective of data volumes.
- Rapid scale-out by provisioning one or more read-only replicas to offload read workloads and serve as hot standby.
- Rapid scale-up, allowing you to quickly increase compute resources to handle heavy workloads and scale them back down when not needed.

Deploying Azure SQL Database Hyperscale

To deploy an Azure SQL Database with the Hyperscale tier, follow the same process as deploying a regular SQL database, with the following differences:

1. Under **Compute + storage**, select the **Configure database** link.

[Basics](#) [Networking](#) [Security](#) [Additional settings](#) [Tags](#) [Review + create](#)

Create a SQL database with your preferred configurations. Complete the Basics tab then go to Review + Create to provision with smart defaults, or visit each tab to customize. [Learn more](#)

Want to try Azure SQL Database for free? Create a free serverless database with the first 100,000 vCore seconds, 32GB of data, and 32GB of backup storage free per month for the lifetime of the subscription. Limit ten free databases per subscription. [Learn more](#)

[Apply offer](#)

SQL Database Hyperscale: Low price, high scalability, and best feature set. [Learn more](#)

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ

Resource group * ⓘ

[Create new](#)

Database details

Enter required settings for this database, including picking a logical server and configuring the compute and storage resources

Database name *

Server * ⓘ

[Create new](#)

Want to use SQL elastic pool? ⓘ ☐ Yes ☒ No

Workload environment ☒ Development ☐ Production

Default settings provided for Development workloads. Configurations can be modified as needed.

Compute + storage * ⓘ

General Purpose - Serverless
Standard-series (Gen5), 1 vCore, 32 GB storage
[Configure database](#)

2. For **Service tier**, select **Hyperscale**.

Configure ...

 Feedback

Service and compute tier

Select from the available tiers based on the needs of your workload. The vCore model provides a wide range of configuration controls and offers Hyperscale and Serverless to automatically scale your database based on your workload needs. Alternately, the DTU model provides set price/performance packages to choose from for easy configuration. [Learn more](#)

Service tier

 Databases originally created in Hyperscale

Compute Hardware

Select the hardware configuration based on confidential computing hardware depends on

Hardware Configuration

Hyperscale (On-demand scalable storage) 

vCore-based purchasing model

General Purpose (Scalable compute and storage options)

Hyperscale (On-demand scalable storage)

Business Critical (High transaction rate and high resiliency)

DTU-based purchasing model

Basic (For less demanding workloads)

Standard (For workloads with typical performance requirements)

Premium (For IO-intensive workloads)

[Change configuration](#)

3. Review the hardware configurations available and select the most appropriate configuration for your database.
4. Optionally, review the other tabs to make adjustments if needed.
5. On the **Review + create** tab, select **Create**.

6. Examine SQL managed instance

<https://learn.microsoft.com/en-us/training/modules/deploy-paas-solutions-with-azure-sql/6-examine-sql-managed-instance>

Examine SQL managed instance

Completed

While many organizations initially migrate to Azure using IaaS offerings, the platform as a service (PaaS) provides extra benefits. With PaaS, the service handles installing and patching SQL Server, so you no longer need to perform these tasks. Consistency checks, backups, security, and performance tools are also included as part of the managed service.

[Azure SQL Managed Instance](#) is a fully functional SQL Server instance that is nearly 100% compatible with your on-premises ecosystem. It includes features like SQL Agent, access to tempdb, cross-database queries, and common language runtime (CLR). This service uses the same infrastructure as Azure SQL Database and offers all the benefits of PaaS, such as automatic backups, automatic patching, and built-in high availability.

Azure SQL Managed Instance capabilities

Azure SQL Managed Instance offers a seamless migration path for existing applications by enabling restores from on-premises backups. Unlike Azure SQL Database, which is designed for single database structures, SQL Managed Instance provides a full SQL Server instance, supporting up to 100 databases and granting access to system databases. It also includes features not available in Azure SQL Database, such as cross-database queries, common language runtime (CLR), and the use of SQL Agent through the msdb system database.

When creating an Azure SQL Managed Instance, you can choose between two service tiers: Business Critical and General Purpose. These tiers align with the Azure SQL Database vCore model, as SQL Managed Instances are purchased using this model. The primary differences between the two tiers are that Business Critical includes In-Memory OLTP and provides a readable secondary, features not available in the General Purpose tier. Both tiers offer high availability, with the Business Critical tier using Always On Availability Groups for higher resilience. Additionally, both tiers allow for independent configuration of storage and compute resources.

Managed Instance link

The Link feature provides hybrid capabilities by replicating databases from SQL Server instances to Azure SQL Managed Instance. It uses distributed availability groups, part of the Always On availability group technology, to replicate data. Transaction log records are replicated as part of these distributed availability groups.

Transaction log records on the primary instance can't be truncated until they're replicated to the secondary instance. Regular transaction log backups help reduce the risk of running out of space on your primary instance.

The Link feature can also be used as a hybrid disaster recovery solution, allowing you to fail over your SQL Server databases hosted anywhere to a database running on SQL Managed Instance. Additionally, you can use the Link feature to provide a read-only secondary database in SQL Managed Instance, offloading intensive read-only operations.

For more information about how to configure link feature for Azure SQL Managed Instance, see [Prepare environment for link feature - Azure SQL Managed Instance](#).

Instance pool

Instance pools provide a cost-efficient way to migrate smaller SQL Server instances to the cloud. Instead of consolidating smaller databases into a larger managed instance, which requires extra

governance and security planning, instance pools allow you to pre-provision resources based on your total migration needs and requirements.

The instance pool feature offers a fast deployment time of up to 10 minutes, making it ideal for scenarios where deployment duration is crucial. Additionally, all instances in a pool share the same virtual machine, and the total IP allocation is independent of the number of instances deployed.

To learn how to deploy an instance pool for SQL Managed Instance, see [Deploy Azure SQL Managed Instance to an instance pool](#).

High availability and Disaster recovery

Azure SQL Managed Instance, as a PaaS service, inherently provides high availability. A standalone SQL Managed Instance offers a 99.99% Service Level Agreement (SLA), ensuring a maximum of 52.60 minutes of downtime per year. The architecture mirrors that of Azure SQL Database: the General Purpose tier uses storage replication for availability, while the Business Critical tier employs multiple replicas for enhanced resilience.

Azure SQL Managed Instance offers auto-failover groups for disaster recovery, protecting the entire managed instance and all its databases. This feature asynchronously replicates data from the primary Azure SQL Managed Instance to a secondary instance, but it's currently limited to the paired Azure region of the primary instance, with only one replica allowed.

Similar to Azure SQL Database, auto-failover groups provide read-write and read-only listener endpoints, simplifying connection string management. If there's a failover, the application connection strings are automatically routed to the appropriate instance. However, these endpoints follow a slightly different format: `<fog-name>.zone_id.database.windows.net` for SQL Managed Instance, compared to `<fog-name>.zone_id.database.windows.net` whereas Azure SQL Database is in the `<fog-name>.secondary.database.windows.net` for Azure SQL Database.

Both the primary and secondary managed instances must be within the same DNS zone. This ensures that the same multi-domain certificate can be used for client connection authentication between the instances in the failover group. You can specify a "DNS Zone Partner" through the Azure portal, PowerShell, or Azure CLI.

Backups

Automatic backups are configured by default for Azure SQL Managed Instance. A key difference between Azure SQL Managed Instance and Azure SQL Database is that with SQL Managed Instance, you can manually create a copy-only backup of a database. These backups must be stored to a URL, as local storage access isn't permitted. Additionally, you can configure long-term retention (LTR) to retain automatic backups for up to 10 years in geo-redundant Azure Blob storage.

Database backups follow the same schedule as Azure SQL Database, and these schedules can't be adjusted.

- **Full** – Once a week
- **Differential** – Every 12 hours

- **Transaction Log** – Every 5-10 minutes depending on transaction log usage

Restoring a database to an Azure SQL Managed Instance is similar to the process with Azure SQL Database. You can use Azure portal, PowerShell, or Azure CLI. To restore from one instance to another, both instances must reside within the same Azure subscription and region. Additionally, you can't restore the entire managed instance, only individual databases within the SQL Managed Instance.

As with Azure SQL Database, you can't restore over an existing database. You need to drop or rename the existing database before restoring it from backup. Since SQL Managed Instance is a fully functional SQL Server instance, you can execute a **RESTORE** command, which isn't possible with Azure SQL Database. However, as a PaaS service, there are some limitations:

- You must restore from a URL endpoint, as local drives aren't accessible.
- You can use the following options (in addition to specifying the database):
 - FILELISTONLY
 - HEADERONLY
 - LABELONLY
 - VERIFYONLY
- Backup files containing multiple log files can't be restored
- Backup files containing multiple backup sets can't be restored
- Backups containing In-Memory/FILESTREAM can't be restored

By default, databases in a managed instance are encrypted using [Transparent Data Encryption \(TDE\)](#) with a Microsoft-managed key. To take a user-initiated copy-only backup, you must disable TDE for the specific database. If a database is encrypted, you can restore it, but you need access to either the certificate or asymmetric key used for encryption. Without these, you can't restore the database to a SQL Managed Instance.

To learn the new features for Azure SQL Managed Instance, see [What's new in Azure SQL Managed Instance?](#).

7. Exercise: Deploy an Azure SQL Database

<https://learn.microsoft.com/en-us/training/modules/deploy-paas-solutions-with-azure-sql/7-exercise-deploy-azure-sql-database>

Exercise: Deploy an Azure SQL Database

Completed

Now it's your chance to deploy an Azure SQL Database.

In this exercise, you'll learn how to configure basic resources needed to deploy an Azure SQL Database with a Virtual Network Endpoint.

Note

To complete this exercise, you'll need an [Azure subscription](#).

Launch the exercise and follow the instructions.

[Launch Exercise](#)

8. Module assessment

<https://learn.microsoft.com/en-us/training/modules/deploy-paas-solutions-with-azure-sql/8-knowledge-check>

Module assessment

Completed

9. Summary

<https://learn.microsoft.com/en-us/training/modules/deploy-paas-solutions-with-azure-sql/9-summary>

Summary

Completed

In this module, you learned about the platform as a service options for SQL Server on Azure. Azure SQL Database is a flexible platform, well suited for software as service applications and has many options for large databases, multiple databases, or development environments that don't need to be online 24x7. Azure SQL Managed Instance is a PaaS version of SQL Server that allows you to easily move your on-premises environments into a managed service. SQL Managed Instance also supports many server-level capabilities that aren't available with Azure SQL Database.

Now that you've reviewed this module, you should be able to:

- Understand PaaS provisioning and deployment options
- Understand elastic pools and hyperscale features
- Examine SQL Managed Instances

Migrate SQL Server workloads to Azure SQL Database

1. Introduction

<https://learn.microsoft.com/en-us/training/modules/migrate-sql-workloads-azure-sql-databases/1-introduction>

Introduction

Completed

Azure SQL Database is a Platform as a Service (PaaS) database engine that provides a complete development and deployment environment in the cloud, which can be used for simple cloud-based applications and for advanced enterprise applications.

Migrating to SQL Database allows you to modernize your application by taking advantage of its PaaS capabilities. This enables you to eliminate dependencies on technical components that are scoped at the instance level, such as SQL Agent jobs. Azure SQL Database offers a low-maintenance solution that can be an excellent choice for certain workloads.

You may have specific requirements that are better suited for Azure SQL Database rather than Azure SQL Managed Instance in the following scenarios:

- You need to simplify deployment for databases with intermittent, unpredictable usage.
- New databases without usage history where compute sizing is difficult or not possible to estimate prior to deployment.
- The complexity of deployment and development is a concern.
- Your storage requirements are higher than what Azure SQL Managed Instance offers, and database consolidation isn't an option.

The process of migrating a SQL Server database running on an Azure virtual machine to Azure SQL Database is similar to the steps we'll learn in this module.

Note

Before proceeding, it is important to ensure that you have reviewed the [Assess SQL Server databases for migration to Azure SQL](#). This module will introduce you to the assessment tools

and help you discover new features in the target SQL Server platform that your database can benefit from after an upgrade.

Use case scenario

Throughout this module, we're using an example scenario to explain key data migration concepts.

Suppose you work for a company that builds bikes and bicycle parts. You have several legacy database servers that you want to upgrade, including a product database, a parts stock database, and a human resources database. You also want to move from a capital expenditure model to an operational expenditure model, and benefit from the scalability and availability of Azure services. You plan to migrate your SQL Server databases to Azure SQL Database. Your board of directors has asked you to plan the migration project and has made you responsible for the execution of the migration tasks.

You'll learn how to migrate SQL Server databases to Azure SQL Database. You'll begin by exploring the pre-migration considerations you need to take into account before a migration, and how to create an Azure SQL database. You'll then explore the different methods for offline and online migrations, and look at ways to move data to Azure SQL Database.

2. Choose the right Azure SQL Database feature

<https://learn.microsoft.com/en-us/training/modules/migrate-sql-workloads-azure-sql-databases/2-choose-right-sql-database-option-azure>

Choose the right Azure SQL Database feature

Completed

In our bicycle manufacturing scenario, you've already identified and profiled the databases that you want to migrate to Azure SQL Database. Now, you want to plan the migration, considering data recoverability, disaster recovery, security, and other implementation details.

You'd like to know the tools and features available to support with the migration process to Azure SQL Database.

Benefits of Azure SQL Database

The following summarize the benefits of deploying single and elastic pool databases:

Category	Feature
Backup and recovery	Automatic backup
	Point-in-time restore
	Backup retention 7 days+
	Long-term backup retention stores backups for up to 10 years
High availability	99.99% availability guarantee
	Built-in availability with three secondary replicas
	Zone redundancy via Azure availability zones
Disaster recovery	Geo-restore of database backups
	Active-geo replication between Azure regions
Service scalability	Dynamic scale-up and scale-down
	Scale out with multiple shards
	Share compute resources between databases using elastic pools
Security	Support for Microsoft Entra authentication
	Cloud-only security features such as Advanced Threat Protection
	Transparent data encryption (TDE) enabled by default
	Support for dynamic and static data masking, row-level security, and Always Encrypted
	Firewall allowlist
Licensing	DTU purchasing model for predictive costing
	vCore purchasing model, enabling storage to be scaled independently of compute
	Combine the vCore purchasing model with Azure Hybrid Benefit for SQL Server to realize cost savings of up to 30 percent

Tip

To review the benefits of migrating to Azure SQL Database and the features available, please refer to [Deploy PaaS solutions with Azure SQL](#) module.

Exclusive features of Azure SQL Database

Some features are supported in Azure SQL Database that aren't available in other Azure SQL offerings:

Feature	Definition
Hyperscale	Cloud-native architecture that allows for independently scalable compute and storage, providing greater flexibility and resources than other tiers.
Auto-scale	With serverless compute tier
Automatic tuning (indexes)	This built-in feature automatically identifies and creates indexes that can improve the performance of your workload. It also verifies that query performance has improved and removes unused or duplicate indexes.
Elastic query	Allows you to run T-SQL queries that bridge multiple databases in SQL Database. This feature is useful for applications using three- and four-part names that can't be changed.
Elastic jobs	The elastic job feature is the SQL Server Agent replacement for Azure SQL Database. To some extent, elastic job is equivalent to the Multi Server Administration feature available on SQL Server instance.
Query Performance Insights (QPI)	This tool helps find the queries to optimize to improve overall workload performance and efficiently use the resource that you're paying for.

Important

To understand additional feature differences between SQL Database, SQL Server, and Azure SQL Managed Instance, as well as the differences among different Azure SQL Database options, see [SQL Database features](#).

Migration options supported

There are two modes of migration to Azure SQL Database: **Online** and **Offline**. The online mode has minimal or no downtime, while the offline mode experiences downtime during the migration process.

Tool	Migration mode
Azure Database Migration Service	Offline
Transactional replication	Online
Azure Migrate	Offline
Import Export Wizard/BACPAC	Offline
Bulk copy (bcp utility)	Offline
Azure Data Factory	Offline

* Can have a higher performance impact, depending on the workload.

Note

We recommend that you use the [Azure Database Migration Service](#) for large migrations and enhanced overall experience.

Migration performance

Consider the following recommendations when migrating to Azure SQL Database:

- Monitor data file I/O and latency on the source, and mitigate any bottlenecks.
- Scale up the target Azure SQL database to Business Critical Gen5 8 vCore or use the Hyperscale service tier to minimize latency for log files.
- Ensure that your network bandwidth can accommodate the maximum log ingestion rate.
- Choose the highest service tier and compute size for maximum transfer performance, and scale down after migration.
- Minimize the distance between BACPAC files and the destination data center.
- Disable auto update and auto create statistics during migration.
- Partition tables and indexes, drop indexed views, and recreate them after migration.
- Consider migrating rarely queried historical data to a separate database in Azure SQL Database, and query it using elastic queries.

Retry application connections

When migrating to Azure SQL Database, it's important to anticipate occasional transient failures when connecting to the database resource, and implement a proper retry logic method. Setting a maximum number of retries before the program terminates is also important.

We recommend waiting for 5 seconds at a minimum on your first retry. Each subsequential retry should increase the delay exponentially, up to a maximum of 60 seconds.

Note

If a SELECT statement fails with a transient error for SQL Database, don't directly retry it. Instead, retry the SELECT statement in a new connection.

To learn more about the connection retry principals, see [Troubleshoot transient connection errors in SQL Database and SQL Managed Instance](#).

3. Use Azure Database Migration Service to migrate to Azure SQL Database

Use Azure Database Migration Service to migrate to Azure SQL Database

Completed

If you can afford to take the database offline while you migrate to Azure, you have a few tools you can use.

In our bicycle manufacturing scenario, the HR database is considered business-critical but is rarely used at weekends. You've planned to execute an offline migration between Friday evening and Monday morning, but you want to assess the best migration method.

It's assumed that all pre-migration checks have been done with [Azure Migrate](#). This process ensures that feature and compatibility issues are addressed.

Migrate using Azure Database Migration Service with Azure CLI

[Azure Database Migration Service](#) is a fully managed service designed to enable seamless migrations from multiple database sources to Azure data platforms with minimal downtime. You can use Azure CLI or PowerShell to automate database migrations, making it ideal for migrating databases at scale.

The Azure CLI `az datamigration` extension provides commands to create and manage database migrations to Azure SQL Database. This approach is particularly useful for:

- Automating migrations as part of CI/CD pipelines
- Migrating multiple databases at scale
- Scripting repeatable migration processes

Prerequisites

Before starting the migration, ensure you have:

1. **Azure CLI installed** with the `datamigration` extension
2. **Azure Database Migration Service** created in your subscription
3. **Target Azure SQL Database** provisioned with the schema already deployed
4. **Self-hosted integration runtime** configured for connectivity to your source SQL Server

To install the Azure CLI `datamigration` extension, run:

```
az extension add --name datamigration
```

Create the Database Migration Service

First, create an Azure Database Migration Service to orchestrate your migration activities:

```
# Create the Azure Database Migration Service
az datamigration sql-service create \
  --resource-group "<YourResourceGroup>" \           # Name of your Azure resource group
  --sql-migration-service-name "<YourMigrationService>" \ # Name for the migration service
  --location "<YourLocation>"                        # Azure region (e.g., eastus)
```

Migrate the database schema

Before migrating data, you need to migrate the database schema from the source to the target. Use the `az datamigration sql-server-schema` command:

```
# Migrate schema from source to target database
az datamigration sql-server-schema \
  --action "MigrateSchema" \
  --src-sql-connection-str "Server=<YourSourceServer>;Initial Catalog=<YourSourceDatabase>" \
  --tgt-sql-connection-str "Server=<YourTargetServer>.database.windows.net;Initial Catalog=<YourTargetDatabase>"
```

Start the database migration

Create a new database migration to copy data from the source to target:

```
# Create a database migration to Azure SQL Database
az datamigration sql-db create \
  --resource-group "<YourResourceGroup>" \           # Name of your Azure resource group
  --sqldb-instance-name "<YourTargetServer>" \        # Name of the target Azure SQL Database
  --target-db-name "<YourTargetDB>" \                 # Name of the target database
  --source-database-name "<YourSourceDB>" \           # Name of the source database
  --source-sql-connection authentication="SqlAuthentication" \
    data-source="<YourSourceServer>" \                # Source SQL Server hostname
    user-name="<YourSourceUser>" \                    # Source SQL Server username
    password="<YourSourcePassword>" \                 # Source SQL Server password
    encrypt-connection=true \
    trust-server-certificate=true \
  --target-sql-connection authentication="SqlAuthentication" \
    data-source="<YourTargetServer>.database.windows.net" \ # Target Azure SQL Database
    user-name="<YourTargetUser>" \                    # Target database username
    password="<YourTargetPassword>" \                 # Target database password
    encrypt-connection=true \
    trust-server-certificate=true \
  --scope "/subscriptions/<YourSubscription>/resourceGroups/<YourResourceGroup>/providers/Microsoft.DataMigration/migrationServices/<YourMigrationService>" \
  --migration-service "/subscriptions/<YourSubscription>/resourceGroups/<YourResourceGroup>/providers/Microsoft.DataMigration/migrationServices/<YourMigrationService>"
```

Migrate specific tables

To migrate only specific tables, use the `--table-list` parameter:

```
# Create a database migration for specific tables
az datamigration sql-db create \
  --resource-group "<YourResourceGroup>" \
  --sqldb-instance-name "<YourTargetServer>" \
  --target-db-name "<YourTargetDB>" \
  --source-database-name "<YourSourceDB>" \
  --source-sql-connection authentication="SqlAuthentication" \
    data-source="<YourSourceServer>" \
    user-name="<YourSourceUser>" \
    password="<YourSourcePassword>" \
    encrypt-connection=true \
    trust-server-certificate=true \
  --target-sql-connection authentication="SqlAuthentication" \
    data-source="<YourTargetServer>.database.windows.net" \
    user-name="<YourTargetUser>" \
    password="<YourTargetPassword>" \
    encrypt-connection=true \
    trust-server-certificate=true \
  --table-list "[Person].[Person]" "[Person].[EmailAddress]" "[Sales].[Customer]" \
  --scope "/subscriptions/<YourSubscription>/resourceGroups/<YourResourceGroup>/providers/Microsoft.DataMigration/dmigrations/<YourMigration>" \
  --migration-service "/subscriptions/<YourSubscription>/resourceGroups/<YourResourceGroup>/providers/Microsoft.DataMigration/dmigrations/<YourMigration>"
```

The Database Migration Service optimizes the migration process by skipping empty tables, even if you select them.

Migration status

There are a few statuses that keep you updated on the progress of the migration.

- **Preparing for copy:** The service is in the process of disabling autostats, triggers, and indexes in the target table.
- **Copying:** The data copy from the source database to the target database is in progress.
- **Copy finished:** Data copy is finished, and the service is waiting on other tables to finish copying.
- **Rebuilding indexes:** The service is rebuilding indexes on target tables.
- **Succeeded:** All data is copied and the indexes are rebuilt.

Monitor migration using Azure CLI

You can check the status of your migration using the `az datamigration sql-db show` command:

```
# Check the status of the database migration
az datamigration sql-db show \
  --resource-group "<YourResourceGroup>" \
  --sqldb-instance-name "<YourTargetServer>" \
```

```
--target-db-name "<YourTargetDB>" \
--expand "MigrationStatusDetails" # Include detailed migration
```

This command returns detailed information about the migration, including the current status and any errors encountered.

Wait for migration completion

You can use the `wait` command to pause script execution until the migration completes:

```
# Wait for the migration to complete before continuing
az datamigration sql-db wait \
  --resource-group "<YourResourceGroup>" \
  --sqldb-instance-name "<YourTargetServer>" \
  --target-db-name "<YourTargetDB>" \
  --created # Wait until migration is c
```

Cancel a migration

If you need to stop an in-progress migration:

```
# Cancel an in-progress migration
az datamigration sql-db cancel \
  --resource-group "<YourResourceGroup>" \
  --sqldb-instance-name "<YourTargetServer>" \
  --target-db-name "<YourTargetDB>" \
  --migration-operation-id "<YourMigrationOperationId>" # ID from the migration
```

Monitor migration from the Azure portal

You can also monitor the migration activity using Azure Database Migration Service in the Azure portal.

To monitor your database migration, go to the Azure portal and find your instance of the Database Migration Service. Once you've located the service, you can view its instance overview. Select **Monitor migrations** to access detailed information about your ongoing database migration.

AdventureWorks ...

[Cancel migration](#)
[Delete migration](#)
[Retry](#)
[Copy migration details](#)
[Refresh](#)

Essentials

Source database name : AdventureWorks

Source server : localhost

Migration status : ○ In progress

Target database name : my-db

Target server : courseware

Target version : Azure SQL Database

Table name ↑↓	Status ↑↓	Data read ↑↓	Data written ↑↓	Rows read ↑↓	Rows copied ↑↓	Copy throughput ↑↓	Copy duration ↑↓	Parallel copy type ↑↓	Used parallel copies ↑↓	Copy start ↑↓
[Sales].[SalesReason]	i Copying	0.00 MB	0.00 MB	0	0	0.00 MB	00:00:07	None	1	09/01/2023, 10:39:46 AM
[Sales].[SalesTaxRate]	i Copying	0.00 MB	0.00 MB	0	0	0.00 MB	00:00:11	None	1	09/01/2023, 10:39:42 AM
[Sales].[SpecialOfferProduct]	✓ Copy finished	0.02 MB	0.02 MB	538	538	0.00 MB	00:00:14	None	1	09/01/2023, 10:39:34 AM
[Sales].[SalesPersonQuotaHistory]	✓ Copy finished	0.01 MB	0.01 MB	163	163	0.00 MB	00:00:16	None	1	09/01/2023, 10:39:24 AM
[Sales].[SpecialOffer]	✓ Copy finished	0.00 MB	0.00 MB	16	16	0.00 MB	00:00:15	None	1	09/01/2023, 10:39:23 AM
[Sales].[SalesTerritoryHistory]	✓ Copy finished	0.00 MB	0.00 MB	17	17	0.00 MB	00:00:28	None	1	09/01/2023, 10:39:11 AM
[Sales].[Store]	✓ Copy finished	0.61 MB	0.61 MB	701	701	0.03 MB	00:00:26	None	1	09/01/2023, 10:39:06 AM
[Sales].[SalesTerritory]	✓ Copy finished	0.00 MB	0.00 MB	10	10	0.00 MB	00:00:30	None	1	09/01/2023, 10:39:04 AM
[Sales].[ShoppingCartItem]	✓ Copy finished	0.00 MB	0.00 MB	3	3	0.00 MB	00:00:28	None	1	09/01/2023, 10:38:44 AM
[Sales].[SalesPerson]	✓ Copy finished	0.00 MB	0.00 MB	17	17	0.00 MB	00:00:30	None	1	09/01/2023, 10:38:33 AM

Showing 1 - 10 of 70 results.

< 1 2 3 4 5 >

After the migration status is **Succeeded**, navigate to the target server, and validate the target database. Check the database schema and data.

Performance considerations

Migration speed heavily depends on the target Azure SQL Database SKU and the self-hosted Integration Runtime host. We strongly recommend that you scale up your Azure SQL Database compute resources before initiating the migration process for an optimal migration experience.

When deciding on the server to install the self-hosted integration runtime, make sure this machine can handle the CPU and memory load of the data copy operation.

Azure SQL Database migration can be slow with a large volume of tables due to the time Azure Data Factory (ADF) takes to start activities, even for small tables.

Tables with large blob columns may fail to migrate due to timeout.

We recommend up to 10 concurrent database migrations per self-hosted integration runtime on a single computer. Scale out the self-hosted runtime or create separate instances on different computers to increase the concurrent database migrations.

Migrate at scale using PowerShell

You can also perform an offline migration of the database from SQL Server on-premises to an Azure SQL Database by using PowerShell.

The following example migrates the *AdventureWorks* database to Azure SQL Database.

```
# Set up secure credentials for source and target connections
$sourcePass = ConvertTo-SecureString "<YourSourcePassword>" -AsPlainText -Force
$targetPass = ConvertTo-SecureString "<YourTargetPassword>" -AsPlainText -Force

# Start the database migration to Azure SQL Database
```

```
New-AzDataMigrationToSqlDb `
  -ResourceGroupName "<YourResourceGroup>" `           # Name of your Azure resource group
  -SqlDbInstanceName "<YourTargetServer>" `             # Name of the target Azure SQL Database
  -Kind "SqlDb" `
  -TargetDbName "<YourTargetDB>" `                     # Name of the target database
  -SourceDatabaseName "<YourSourceDB>" `               # Name of the source database
  -SourceSqlConnectionAuthentication SQLAuthentication `
  -SourceSqlConnectionDataSource "<YourSourceServer>" ` # Source SQL Server host name or IP address
  -SourceSqlConnectionUserName "<YourSourceUser>" `    # Source SQL Server user name
  -SourceSqlConnectionPassword $sourcePass `
  -Scope "/subscriptions/<YourSubscription>/resourceGroups/<YourResourceGroup>/providers/Microsoft.DataMigration/migrations/<YourMigration>" `
  -TargetSqlConnectionAuthentication SQLAuthentication `
  -TargetSqlConnectionDataSource "<YourTargetServer>.database.windows.net" `
  -TargetSqlConnectionUserName "<YourTargetUser>" `    # Target database user name
  -TargetSqlConnectionPassword $targetPass `
  -MigrationService "/subscriptions/<YourSubscription>/resourceGroups/<YourResourceGroup>/providers/Microsoft.DataMigration/migrations/<YourMigration>"
```

The following example migrates a subset of tables from the *AdventureWorks* database.

```
# Migrate specific tables from source to target database
New-AzDataMigrationToSqlDb `
  -ResourceGroupName "<YourResourceGroup>" `
  -SqlDbInstanceName "<YourTargetServer>" `
  -Kind "SqlDb" `
  -TargetDbName "<YourTargetDB>" `
  -SourceDatabaseName "<YourSourceDB>" `
  -SourceSqlConnectionAuthentication SQLAuthentication `
  -SourceSqlConnectionDataSource "<YourSourceServer>" `
  -SourceSqlConnectionUserName "<YourSourceUser>" `
  -SourceSqlConnectionPassword $sourcePass `
  -Scope "/subscriptions/<YourSubscription>/resourceGroups/<YourResourceGroup>/providers/Microsoft.DataMigration/migrations/<YourMigration>" `
  -TargetSqlConnectionAuthentication SQLAuthentication `
  -TargetSqlConnectionDataSource "<YourTargetServer>.database.windows.net" `
  -TargetSqlConnectionUserName "<YourTargetUser>" `
  -TargetSqlConnectionPassword $targetPass `
  -TableList "[Person].[Person]", "[Person].[EmailAddress]" ` # Specify tables to migrate
  -MigrationService "/subscriptions/<YourSubscription>/resourceGroups/<YourResourceGroup>/providers/Microsoft.DataMigration/migrations/<YourMigration>"
```

To learn more about database migration commands, refer to the following links: [PowerShell module for data migration](#) and [Azure CLI for data migration](#).

4. Migrate to Azure SQL Database using BACPAC

Migrate to Azure SQL Database using BACPAC

Completed

A SQL Server database can be imported into an Azure SQL database using a **.bacpac** file.

A **.bacpac** file is a compressed file containing the metadata and data from the database. The data can be imported from the Azure Blob Storage or from a local storage in an on-premises environment.

For optimal scale and performance in production environments, it's recommended to use the **SQLPackage** utility. Running multiple **SqlPackage** commands in parallel for subsets of tables can significantly accelerate import/export operations.

Import from a BACPAC file in the Azure portal

You can follow these steps to import a **.bacpac** file in Azure SQL Database.

1. To import from a BACPAC file into a new single database using the Azure portal, open the appropriate database server page and then, on the toolbar, select **Import database**.
2. Select the storage account and container for the BACPAC file, and then select the BACPAC file from which to import.
3. Specify the new database size (usually the same as the origin) and provide the destination SQL Server credentials, and then select **OK**.
4. To monitor an import's progress, open the database server page and, under **Settings**, select **Import/Export history**. When successful, the import has a **Completed** status.

Additionally, you can also use **SqlPackage** to import a BACPAC file as it's faster than using the Azure portal. The following command imports the **AdventureWorks2019** database from local storage to an Azure SQL Database server called `<server-name>`. It creates a new database called **myMigratedDatabase** with a **Premium** service tier and a **P6** service objective.

Change these values as appropriate for your environment.

```
SqlPackage.exe /a:import /tcs:"Data Source=<server-name>.database.windows.net;Init:
```

Tip

To increase the speed of the import process, you can scale your database to a higher service tier and compute size, providing more and faster resources. Once the import is complete, you can scale down to your desired service tier and compute size.

5. Use an online method to migrate to Azure SQL Database

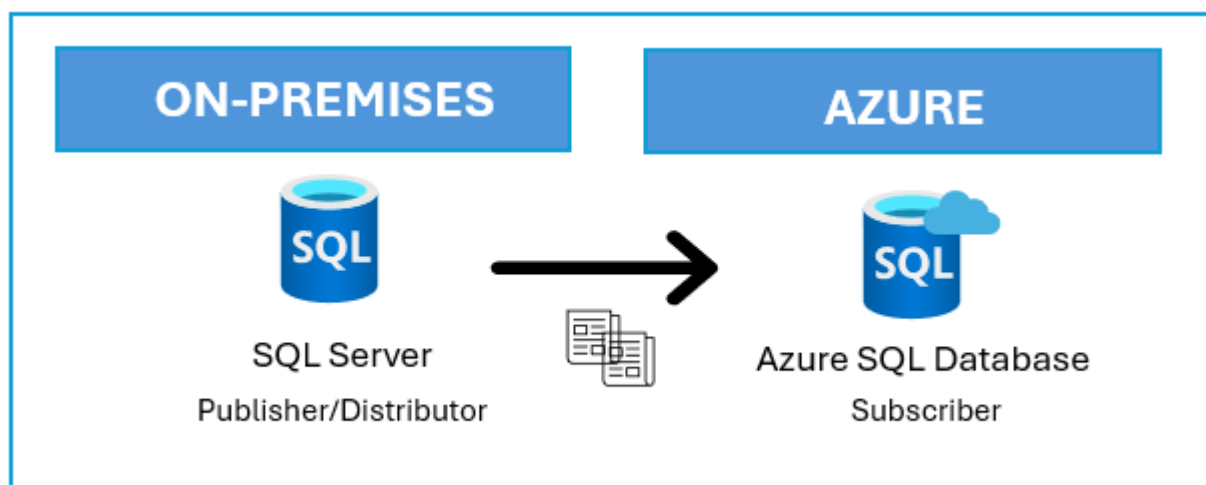
<https://learn.microsoft.com/en-us/training/modules/migrate-sql-workloads-azure-sql-databases/6-migrate-sql-server-azure-sql-database-online>

Use an online method to migrate to Azure SQL Database

Completed

If you need a database to remain online to users throughout the migration process, you can use transactional replication to move the data. Transactional replication is the only online method available for migrating to Azure SQL Database.

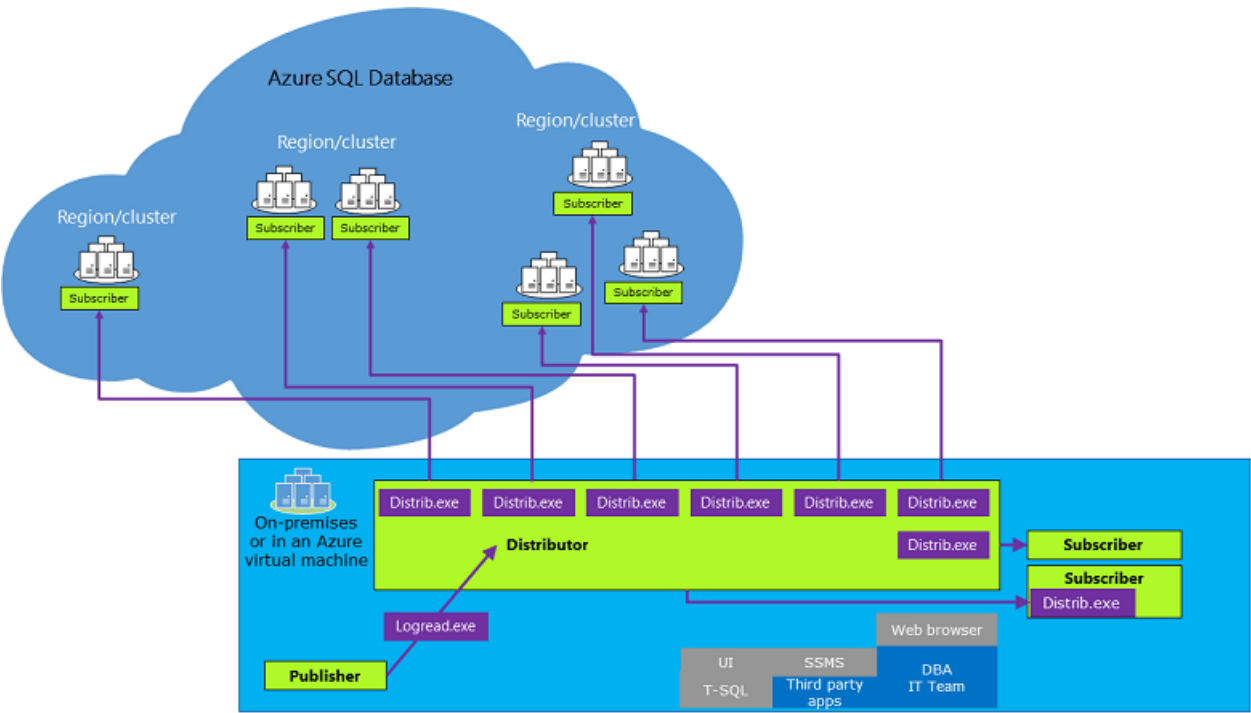
In our bicycle manufacturer scenario, warehouses run on a 24 hour, 7 days a week basis, and there are no periods of inactivity. Your board of directors wants to be sure that the inventory database is constantly available, even during the migration to Azure SQL Database.



What is transactional replication?

Transactional replication is a way to move data between continuously connected database servers.

The process begins with a snapshot of the publication database objects and data. Once the initial snapshot is taken, any subsequent changes to the data or schema at the Publisher are typically delivered to Azure SQL Database in near real-time as they occur.



Azure SQL Database supports both transactional and snapshot replication as a push subscriber. That means Azure SQL Database can receive and apply changes from a publisher using either a transactional or snapshot replication.

The publisher and/or distributor can be an instance of SQL Server that is either running on-premises, on an Azure virtual machine in the cloud, or as an Azure SQL Managed Instance.

You can configure transactional replication through SQL Server Management Studio, or by executing Transact-SQL statements on the publisher. Transactional replication can't be configured from the Azure portal.

Transactional replication requires the following components:

Role	Definition
Publisher	A database instance that hosts the data to be replicated (source).
Subscriber	Receives the data being replicated by the <i>Publisher</i> (target).
Distributor	Collects changes in the articles from a <i>Publisher</i> and distributes them to the <i>Subscribers</i> .
Article	A database object; for example, a table that's included in the <i>Publication</i> .
Publication	A collection of one or more articles from the database being replicated.
Subscription	A request from a <i>Subscriber</i> for a <i>Publication</i> .

Set up a transactional replication

Follow the steps below to migrate the table `[Person].[Person]` from *AdventureWorks* database to Azure SQL Database with no downtime. Transactional replication can only use SQL Server authentication logins to connect to Azure SQL Database.

Parameter	Definition
<code>@distributor</code>	Source instance name.
<code>@publisher</code>	Source instance name.
<code>@subscriber</code>	Azure SQL Database in the format: <code><server>.database.windows.net</code> . The Azure SQL Database must exist before running the script.
<code>@dbname</code>	Database name at the source.
<code>@publisher_login</code>	SQL user with required permissions at the source.
<code>@publisher_password</code>	Password for the SQL user.
<code>@destination_db</code>	Database name at the destination.
<code>@subscriber_login</code>	SQL user with required permissions at the destination.
<code>@subscriber_password</code>	Password for the SQL user.
<code>@working_directory</code>	Replication working directory, change this location as appropriate.

Adjust the parameters above according to your own environment when running the script.

Create the distributor

The following script creates the distributor database, distributor publishers, and the agents.

```
USE [master]
GO

EXEC sp_adddistributor @distributor = N'CONTOSO-SRV', @password = N''
GO

EXEC sp_adddistributiondb
    @database = N'distribution',
    @data_folder = N'C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQL
    @data_file = N'distribution.MDF',
    @data_file_size = 13,
    @log_folder = N'C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLS
    @log_file = N'distribution.LDF',
    @log_file_size = 9,
    @min_distretention = 0,
    @max_distretention = 72,
    @history_retention = 48,
    @deletebatchsize_xact = 5000,
    @deletebatchsize_cmd = 2000,
```

```

        @security_mode = 1
GO

-- Adding the distribution publishers
exec sp_adddistpublisher
    @publisher = N'CONTOSO-SRV',
    @distribution_db = N'distribution',
    @security_mode = 1,
    @working_directory = N'C:\REPL',
    @trusted = N'false',
    @thirdparty_flag = 0,
    @publisher_type = N'MSSQLSERVER'
GO

exec sp_addsubscriber
    @subscriber = N'contoso.database.windows.net',
    @type = 0,
    @description = N'Azure SQL Database (target)'
GO

-- Enabling the replication database
use master
exec sp_replicationdboption
    @dbname = N'AdventureWorks',
    @optname = N'publish',
    @value = N'true'
GO

--Adds a Log Reader agent for the AdventureWorks database.
exec [AdventureWorks].sys.sp_addlogreader_agent
    @publisher_security_mode = 1
GO

--Adds a Queue Reader agent for the distributor.
exec [AdventureWorks].sys.sp_addqreader_agent
    @frompublisher = 1
GO

```

Create the transactional publication

The following script creates the transactional publication of the `AdventureWorks` database from the publisher.

```

USE [AdventureWorks]
GO

EXEC sp_addpublication
    @publication = N'REPL-AdventureWorks',
    @description = N'Transactional publication of database ''AdventureWorks'' f:
    @sync_method = N'concurrent',
    @retention = 0,
    @allow_push = N'true',
    @allow_pull = N'true',

```

```

        @allow_anonymous = N'true',
        @enabled_for_internet = N'false',
        @snapshot_in_defaultfolder = N'false',
        @alt_snapshot_folder = N'C:\REPL',
        @compress_snapshot = N'true',
        @ftp_port = 21,
        @ftp_login = N'anonymous',
        @allow_subscription_copy = N'false',
        @add_to_active_directory = N'false',
        @repl_freq = N'continuous',
        @status = N'active',
        @independent_agent = N'true',
        @immediate_sync = N'true',
        @allow_sync_tran = N'false',
        @autogen_sync_procs = N'false',
        @allow_queued_tran = N'false',
        @allow_dts = N'false',
        @replicate_ddl = 1,
        @allow_initialize_from_backup = N'false',
        @enabled_for_p2p = N'false',
        @enabled_for_het_sub = N'false'
GO

exec sp_addpublication_snapshot
        @publication = N'REPL-AdventureWorks',
        @frequency_type = 1,
        @frequency_interval = 0,
        @frequency_relative_interval = 0,
        @frequency_recurrence_factor = 0,
        @frequency_subday = 0,
        @frequency_subday_interval = 0,
        @active_start_time_of_day = 0,
        @active_end_time_of_day = 235959,
        @active_start_date = 0,
        @active_end_date = 0,
        @publisher_security_mode = 0,
        @publisher_login = N'sqladmin',
        @publisher_password = N'<pwd>'

```

Create the article for the publication

The following script creates the article for the `[Person].[Person]` table.

```

USE [AdventureWorks]
GO

EXEC sp_addarticle
        @publication = N'REPL-AdventureWorks',
        @article = N'Person',
        @source_owner = N'Person',
        @source_object = N'Person',
        @type = N'logbased',
        @description = N'',

```

```

        @creation_script = N'',
        @pre_creation_cmd = N'drop',
        @schema_option = 0x000000000803509F,
        @identityrangemanagementoption = N'none',
        @destination_table = N'Person',
        @destination_owner = N'Person',
        @status = 24,
        @vertical_partition = N'false',
        @ins_cmd = N'CALL [sp_MSins_PersonPerson]',
        @del_cmd = N'CALL [sp_MSdel_PersonPerson]',
        @upd_cmd = N'SCALL [sp_MSupd_PersonPerson]'
GO

```

Create the subscription and the subscription agent

The following script creates the push subscription to the Azure SQL Database subscriber.

```

USE [AdventureWorks]
GO

EXEC sp_addsubscription
    @publication = N'REPL-AdventureWorks',
    @subscriber = N'contoso.database.windows.net',
    @destination_db = N'my-db',
    @subscription_type = N'Push',
    @sync_type = N'automatic',
    @article = N'all',
    @update_mode = N'read only',
    @subscriber_type = 0

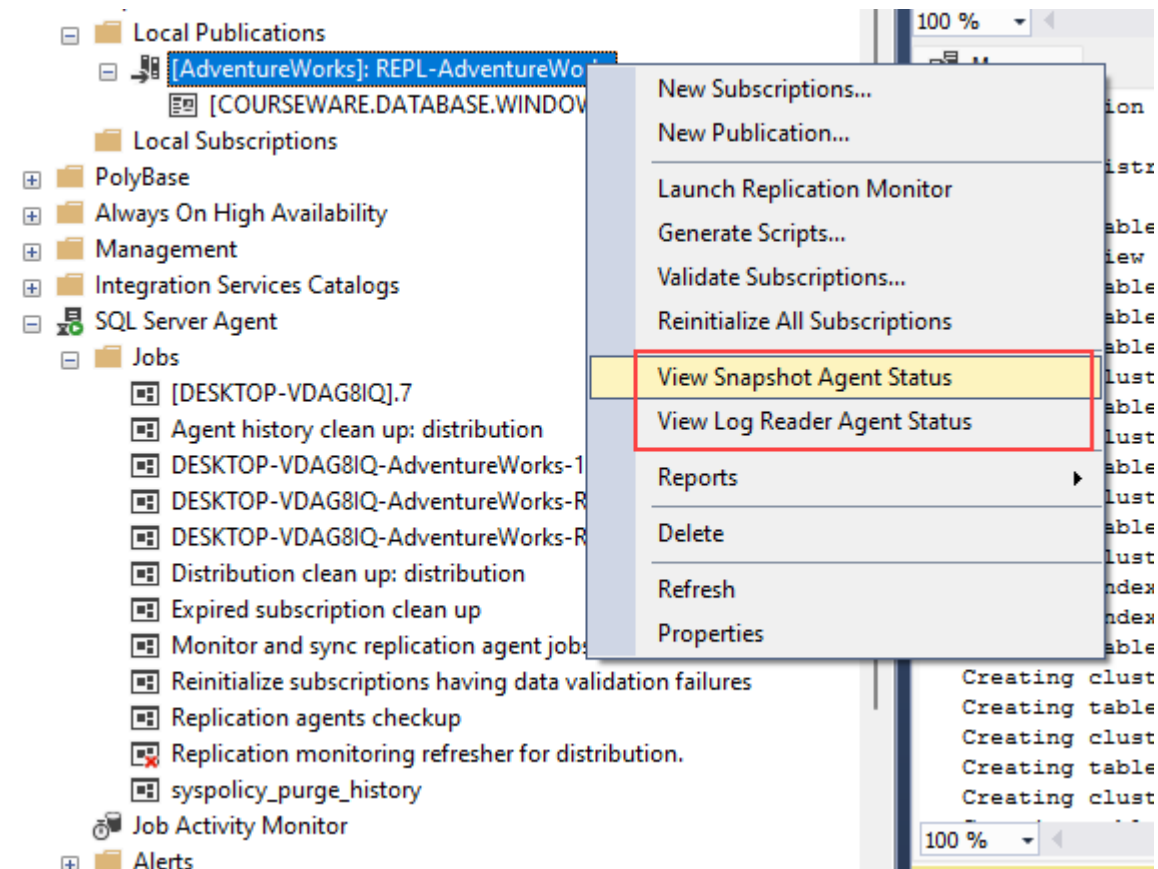
exec sp_addpushsubscription_agent
    @publication = N'REPL-AdventureWorks',
    @subscriber = N'contoso.database.windows.net',
    @subscriber_db = N'my-db',
    @job_login = null,
    @job_password = null,
    @subscriber_security_mode = 0,
    @subscriber_login = N'sqladmin',
    @subscriber_password = '<pwd>',
    @frequency_type = 64,
    @frequency_interval = 1,
    @frequency_relative_interval = 1,
    @frequency_recurrence_factor = 0,
    @frequency_subday = 4,
    @frequency_subday_interval = 5,
    @active_start_time_of_day = 0,
    @active_end_time_of_day = 235959,
    @active_start_date = 0,
    @active_end_date = 0,
    @dts_package_location = N'Distributor'
GO

```

Initiate and monitor replication

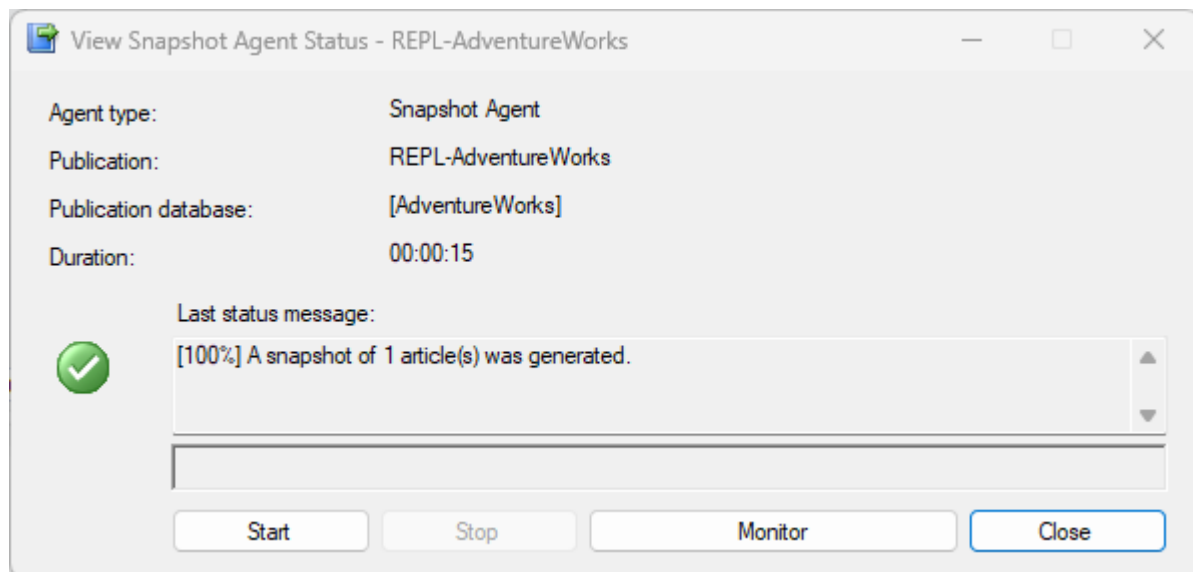
Replication management and monitoring are not supported from Azure SQL Database. Instead, perform these activities from SQL Server. To initiate replication, start the snapshot job, log reader job, and distributor job.

You can monitor the **Snapshot Agent** and **Log Reader Agent** by right-clicking on the publication and selecting the appropriate option. If the agents are not running, start them.

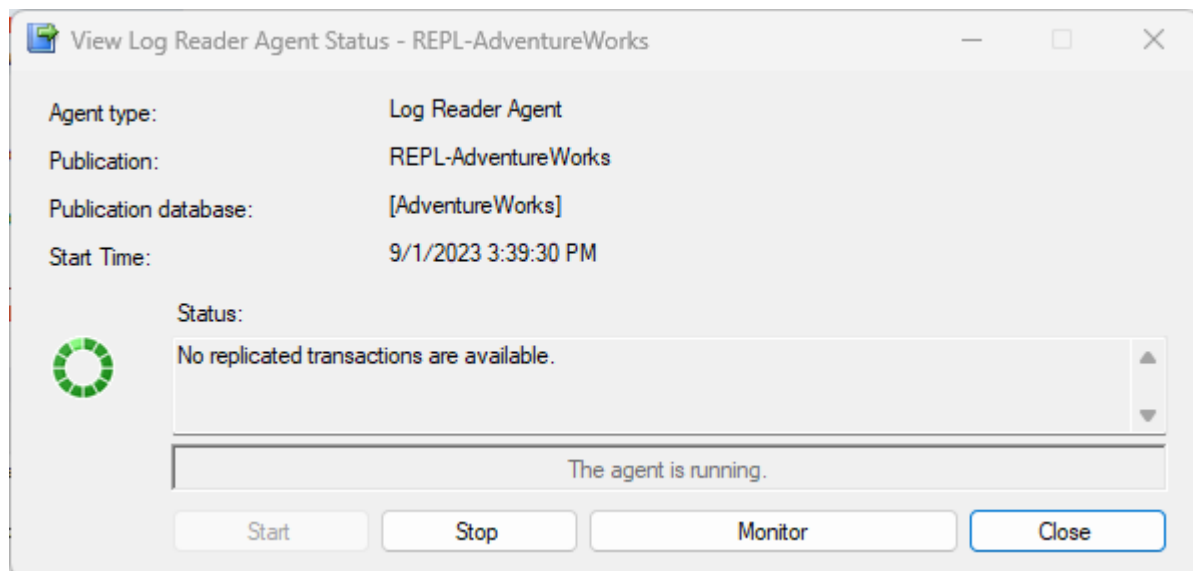


To view the synchronization status, right-click on the subscription, select **View Synchronization Status**, and then start the agent. If you encounter any error messages, check the agent jobs history on the SQL Server Agent. If the agents are running as expected, you should see the following results.

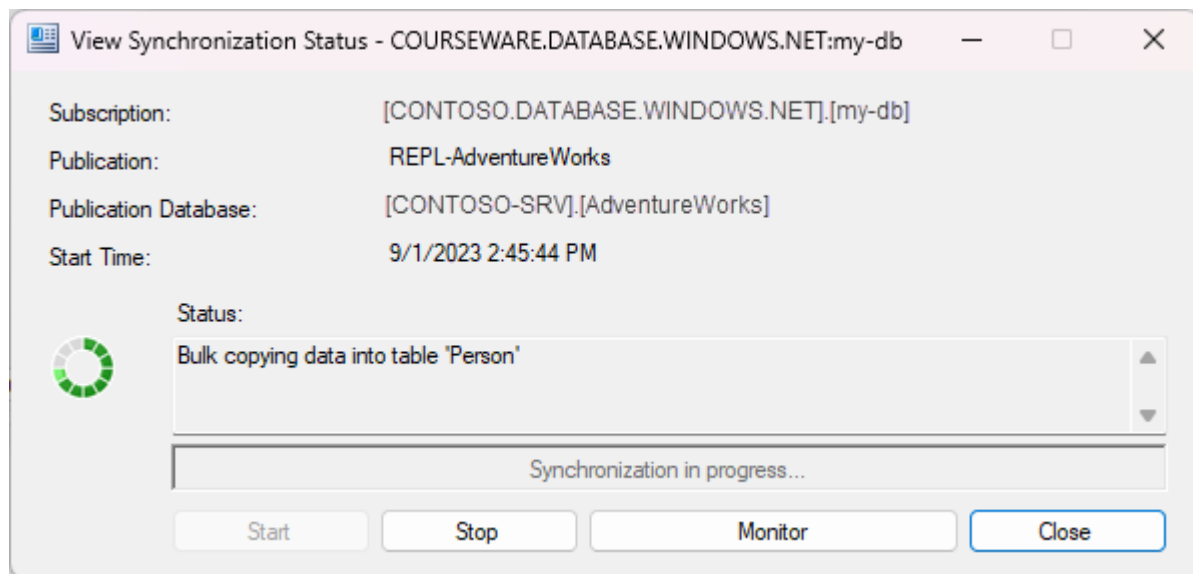
Snapshot Agent:



Log Reader Agent:



Synchronization Status:



After the data is fully replicated to Azure SQL Database, you can direct the connections to the subscriber database, and then stop and remove the replication.

To learn more about supported configurations, see [Replication to Azure SQL Database](#).

6. Move data to Azure SQL Database

<https://learn.microsoft.com/en-us/training/modules/migrate-sql-workloads-azure-sql-databases/7-move-data-sql-database>

Move data to Azure SQL Database

Completed

While there are methods available for migrating an entire schema and its data, there are also cases where only a subset of the database is needed. Fortunately, many of the methods we've seen support partial data migration, and we'll learn about a few others.

In our bicycle manufacturer scenario, suppose the company has an on-premises SQL Server database that contains several years' worth of sales, customer, and product data. The company wants to migrate to an Azure SQL Database to take advantage of the scalability and flexibility of the cloud. However, they only need to migrate the customer and product tables, as they want to keep their sales data on-premises for security reasons.

Bulk copy

The [bcp utility](#) allows bulk exporting of data from a SQL Server table into a data file and vice versa. The utility is versatile, enabling data transfer between SQL Server and other programs or databases.

Understanding the schema and data types of the table is essential for using the bcp command effectively, unless a preexisting format file is available.

Azure Data Factory

You can use [Azure Data Factory](#) for data migration rather than entire database migration. Azure Data Factory can migrate and transform data from source SQL Server databases, and it's commonly used for business intelligence workloads (BI).

7. Exercise - Migrate a SQL Server database to Azure SQL Database

<https://learn.microsoft.com/en-us/training/modules/migrate-sql-workloads-azure-sql-databases/8-exercise-migrate-sql-database>

Exercise - Migrate a SQL Server database to Azure SQL Database

Completed

Now it's your chance to migrate a SQL Server database using a few of the tools and features available.

In this exercise, you learn how to migrate a SQL Server database to Azure SQL Database.

Note

To complete this exercise, you need an [Azure subscription](#).

Launch the exercise and follow the instructions.

[Launch Exercise](#)

8. Module assessment

<https://learn.microsoft.com/en-us/training/modules/migrate-sql-workloads-azure-sql-databases/9-knowledge-check>

Module assessment

Completed

9. Summary

<https://learn.microsoft.com/en-us/training/modules/migrate-sql-workloads-azure-sql-databases/10-summary>

Summary

Completed

In our bike manufacturing scenario, you created and executed a plan to migrate your servers to Azure SQL Database using both offline and online methods.

Now, with the migration complete, your users have the best availability and scalability for their data and you're confident that you can respond to future changes in demand quickly. The method chosen to migrate the database is typically dependent on how much time the SQL Server databases can be offline.

For additional reading, you can refer to the following resources:

- [Migration overview: SQL Server to Azure SQL Database](#)
- [Azure Database Migration Service](#)
- [Transactional Replication with Azure SQL Database](#)

Migrate SQL Server workloads to Azure SQL Managed Instance

1. Introduction

<https://learn.microsoft.com/en-us/training/modules/migrate-sql-workloads-azure-managed-instances/1-introduction>

Introduction

Completed

Azure SQL managed instances make it easy and quick to lift and shift a database from on-premises into the cloud.

Suppose you work for a sports clothing retail company. Your organization has been moving many of its systems into Azure to realize improved cost-of-ownership and better availability. So far, you haven't considered migrating your business-critical databases to Azure because you've heard that databases often need modification to work in Azure SQL.

You want to evaluate Azure's database hosting solutions, choose the best one to host your database, and do the migration. You'd prefer to migrate with as little database administrator and developer time as possible to keep the project budget low.

Easy migration: nearly 100% like SQL Server

Data migration

- Native backup/restore
- Configurable DB file layout
- DMS (migrations at scale)

Security

- Integrated Auth (Azure AD)
- Encryption (TDE, AE)
- SQL Audit
- Row-Level Security
- Dynamic Data Masking

Programmability

- Global temp tables
- Cross-database queries and transactions
- Linked servers
- CLR modules

Operational

- DMVs & XEvents
- Query Store
- SQL Agent
- DB Mail (external SMTP)

Scenario enablers

- Service Broker
- Change Data Capture
- Transactional Replication

In this module, you'll learn about Azure SQL Managed Instance, how it differs from other database solutions available in Azure, and how to choose the best cloud database solution for your project.

2. Evaluate migration scenarios

<https://learn.microsoft.com/en-us/training/modules/migrate-sql-workloads-azure-managed-instances/2-evaluate-migration-scenarios>

Evaluate migration scenarios

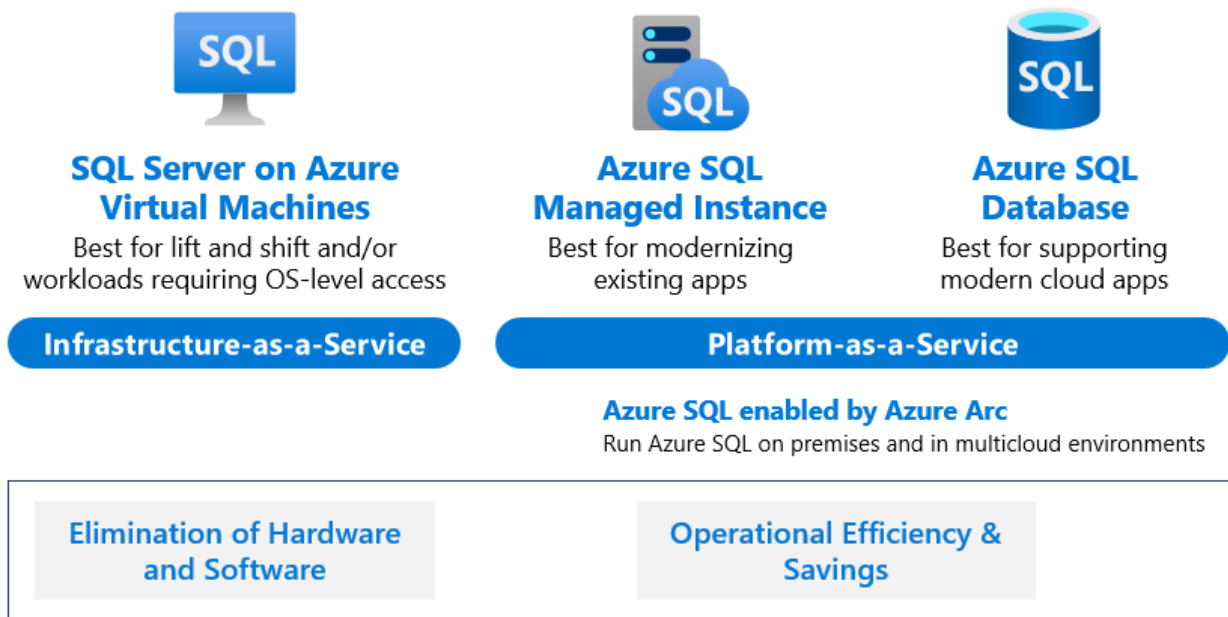
Completed

Azure SQL managed instance is designed to make it easy to host existing databases in the cloud by providing almost 100 percent compatibility with on-premises versions of SQL Server.

In your sports clothing company, you have a database that stores the product details for your entire catalog. The website uses the database to display product details to customers, by the sales representatives' smartphone apps to keep them informed about the catalog, and by a data analysis solution to populate product dimensions in a data cube. The database is considered business-critical by the board of directors. You've been asked to migrate this database to the cloud so the systems that depend on it need as little modification as possible. You want to evaluate Azure SQL Managed Instance for this project.

What is Azure SQL Managed Instance?

The Azure SQL platform as a service (PaaS) family includes the Azure SQL Database and Azure SQL Managed Instance. The goal of Azure SQL Managed Instance is to provide SQL Server applications with a fully managed PaaS experience in the Azure cloud.



Azure SQL Managed Instance is designed to enable a *lift and shift* solution for customers. The managed instance looks to bring applications, databases, and supporting technologies to Azure PaaS. Previously, without SQL Managed Instance, migration scenarios where an organization's application required access to any technology outside of the database (for example SQL Agent jobs, cross database joins, and SQL Server Integration Services) would be prevented from moving to the cloud. The only way a DBA or developer could migrate an on-premises application would be to employ one of the following approaches:

- Move the database and supporting technologies to an infrastructure as a service (IaaS) model.
- Rewrite the application with a fully PaaS model on Azure SQL Database, with extra development to address migration blockers.

The decision to migrate applications to Azure often hinges on whether an organization has the resources to adapt their application to Azure's PaaS model and manage the application code, as vendor support for modifications is typically limited. So, many opt for SQL Server on IaaS to use the full SQL Server experience without the need to overhaul existing applications. Despite Azure SQL Database's capabilities, the heavy dependence of many applications on technologies outside its scope presents challenges. However, SQL Managed Instance, code-named "*cloud lifter*," aims to overcome these hurdles, facilitating the migration to a SQL-based PaaS solution in Azure without needing application redesign.

Review key features

The most important features of SQL Managed Instance include:

Key Features	Description
Backwards compatibility	Managed instance provides backward compatibility to SQL Server 2008 databases. Direct migration from SQL Server 2005 database servers is also supported, with the compatibility level for migrated SQL Server 2005 databases being updated to SQL Server 2008.
Easy lift and shift	Managed instance has close to 100 percent compatibility to SQL Server. This compatibility includes core SQL Server components, programmability enhancements, instance-scoped features, such as cross database joins, and management tools that most existing SQL-based applications need to function correctly.
Fully managed PaaS	PaaS benefits include removing the need for managing hardware and all the overhead that comes from doing physical maintenance on SQL Server servers. You also have the benefits of quickly scaling up and scaling down, and provisioning resources in the cloud. SQL Managed Instance is built on the SQL Server engine, so it's always up to date with the latest SQL features and functionality.
Security features	You can enable security features at the SQL Managed Instance level just as you do at the database level. These features include the Vulnerability Assessment and the Advance Threat Protection settings. Finally, at the managed instance level, you can configure Transparent Data Encryption (TDE) and whether you want to bring your own key (BYOK) for encryption.
Secure network isolation	One of the unique aspects of managed instance, network security isolation is where managed instance has complete security isolation from any other tenant in the Azure cloud. In a typical default deployment SQL endpoint, managed instance is solely exposed through a private IP address that only allows connectivity from private Azure networks or hybrid networks. For on-premises applications to connect to managed instance, you'd need an Azure ExpressRoute configuration or a VPN gateway.
Instance failover groups	An instance failover group is a set of databases managed by a single database server, or within a single managed instance, that can fail over as a unit to another region. You use instance failover groups when all or some of the primary databases have gone offline due to an outage in the primary region.

Migration options supported

There are two modes of migration to Azure SQL Managed Instance: **online** and **offline**. The online mode has minimal or no downtime, while the offline mode experiences downtime during the migration process.

- **Log Replay Service.** It's an online migration option, and used when you need more control of your database migration project.
- **Managed Instance link..** The Managed Instance link, using distributed availability groups, securely extends your data estate by replicating data almost instantly (online) between any hosted SQL Server and Azure SQL Managed Instance, and vice versa.

- **SQL Server migration in Azure Arc.** When your SQL Server instance is enabled by Azure Arc, you can migrate databases to SQL Managed Instance directly from the Azure portal using either the Managed Instance link or Log Replay Service methods.
- **Native backup and restore.** Backup and restore are a simple migration method favored by many SQL Server professionals. It's the easiest migration option for customers who can provide full database backups to Azure Storage.
- **Transactional replication.** Transactional replication is a way to move data between continuously connected database servers. It's best to be used for online or offline migration of large and complex databases.

While most of the tools facilitate [migration to Azure SQL Database](#) as well, there are some that are exclusively supported by SQL Managed Instance. In the next units, we'll learn about a few of them in more detail.

Tip

Learn more about how to [design a SQL Server migration strategy](#).

3. Migrate using Azure Arc

<https://learn.microsoft.com/en-us/training/modules/migrate-sql-workloads-azure-managed-instances/3-azure-arc>

Migrate using Azure Arc

Completed

Azure Arc provides a streamlined pathway to migrate SQL Server databases to Azure SQL Managed Instance directly from the Azure portal. When your SQL Server instance is enabled by Azure Arc, you gain access to integrated assessment, target selection, and migration capabilities—all managed through a unified experience.

In our bicycle manufacturing company scenario, the distribution team runs business-critical databases that require minimal downtime during migration. The team wants to use the Azure portal for a simplified migration experience while maintaining control over the migration method.

Understand how to migrate SQL Server with Azure Arc

When your SQL Server instance is enabled by Azure Arc, the Azure portal becomes your central hub for database migration to SQL Managed Instance. The migration process guides you through four key stages:

1. **Assess source instance** - Evaluate your SQL Server instance to determine migration readiness
2. **Select target** - Choose or create your SQL Managed Instance destination
3. **Migrate data** - Execute the migration using your chosen method
4. **Monitor and cutover** - Track progress and complete the migration

Discovery of SQL Server instances and generation of readiness reports happen automatically every weekend, but you can also trigger assessments manually at any time. The assessment evaluates feature compatibility, identifies potential migration blockers, and provides right-sized recommendations for your target SQL Managed Instance configuration.

Database migration through Azure Arc is available for SQL Server 2012 and later versions, with the specific migration method determining exact version requirements.

Benefits of Azure Arc for migration

Azure Arc simplifies the migration journey by consolidating the entire process within the Azure portal. Rather than switching between multiple tools and interfaces, you can assess readiness, select or create your target instance, execute the migration, and monitor progress from a single unified experience. The automated assessment capabilities evaluate feature compatibility and identify potential migration blockers, while also providing right-sized recommendations for your target SQL Managed Instance configuration based on your workload characteristics.

Microsoft Copilot is integrated into the Azure portal migration experience to assist you throughout the process. You can interactively chat with Copilot to get help comparing migration methods, understanding assessments, starting migrations, and monitoring progress—making it easier to choose and execute the right approach for your scenario.

Additionally, Azure Arc provides visibility into your overall migration status, showing the number of databases migrated, migrations in progress, and your recommended target configuration. This centralized view helps you track progress across your data estate and ensures nothing is overlooked during large-scale migration projects.

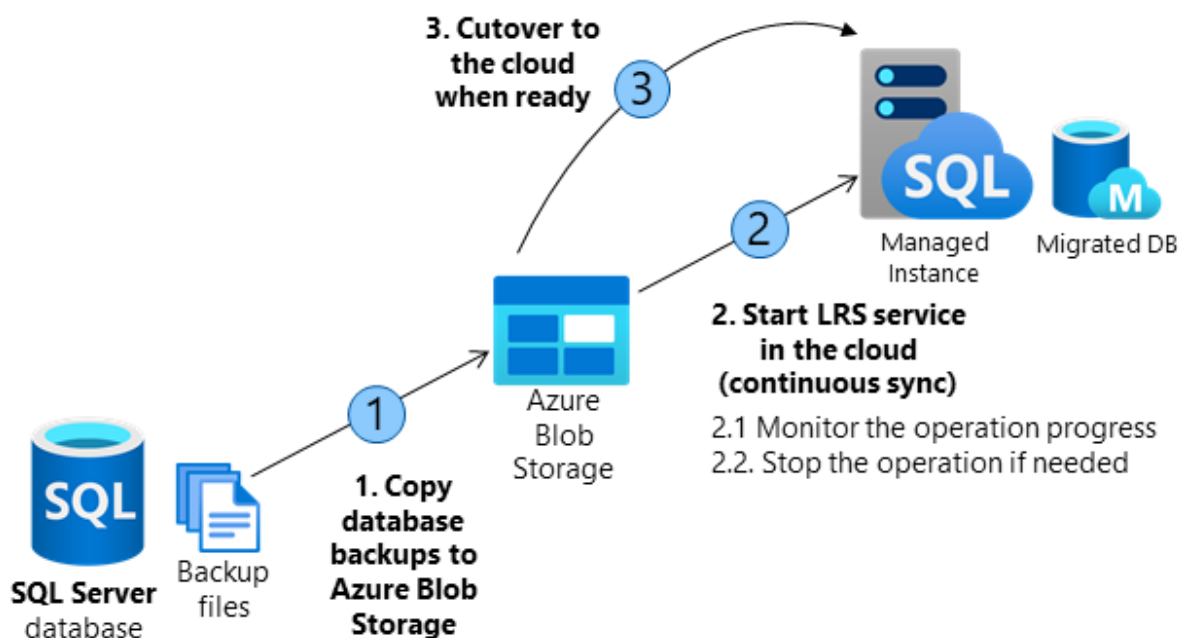
For detailed guidance on migrating SQL Server databases to Azure SQL Managed Instance using Azure Arc, see [Migration to Azure SQL Managed Instance](#).

4. Use Log Replay Service (LRS) to migrate

Use Log Replay Service (LRS) to migrate

Completed

Log Replay Service (LRS) is a tool that enables custom migrations of databases from on-premises SQL Servers to SQL Managed Instance in the cloud. It uses log shipping technology and is useful in cases where more control is needed, when there's little tolerance for downtime, or when Azure Data Migration Service can't be used.



LRS can be used directly with PowerShell, CLI cmdlets, or API, to manually build and orchestrate database migrations to SQL Managed Instance. Some of the reasons to consider using LRS include:

- More control over the database migration project
- Little tolerance for downtime on migration cutover
- Inability to install DMS executable in the environment
- Lack of file access to database backups
- Inability to open networking ports from the environment to Azure

Understand migration types

There are two migration modes available for LRS.

Mode	Description	Recommended for	Backup Chain Availability
Autocomplete	Automatically finishes the migration when the last backup file is restored	Passive workloads	Entire backup chain must be available in advance
Continuous	Continuously scans for new backup files and restores them, allowing for data catch-up	Active workloads	Backup chain can be added during migration

Regardless of the mode, plan to complete the migration within 30 days, as the LRS job will be automatically canceled after this time.

Secure the migration process

To run LRS, you must have one of the following Azure role-based access control (RBAC) role: Subscription Owner, SQL Managed Instance Contributor, or a custom role with the permission `Microsoft.Sql/managedInstances/databases/*`.

An Azure Blob Storage account is required and works as an intermediary storage for backup files between your SQL Server instance and your SQL Managed Instance. To use Azure Blob storage with a firewall, another configuration is required. You must add the SQL Managed Instance subnet to the storage account's virtual network firewall rules using MI subnet delegation and the Storage service endpoint. Also, you can use either a SAS token or a managed identity to access your Azure Blob Storage account, but not both.

Improve backup and restore performance

You can split full and differential backups into multiple files, instead of using a single file, to improve backup and restore performance. This is because multiple files can be read or written to in parallel, reducing the time it takes to complete the backup or restore operation.

Also, enabling backup compression can help improve network transfer speeds. Compressed backups are smaller in size, which means they take less time to transfer over the network. This can be especially useful when transferring large backups to or from Azure.

We strongly recommend enabling `CHECKSUM` for backups, even though it isn't required. SQL Managed Instance performs an integrity check on backups without `CHECKSUM`, which can increase the time it takes to restore the database. By enabling `CHECKSUM`, you can speed up the restore operations.

5. Migrate using Managed Instance link

Migrate using Managed Instance link

Completed

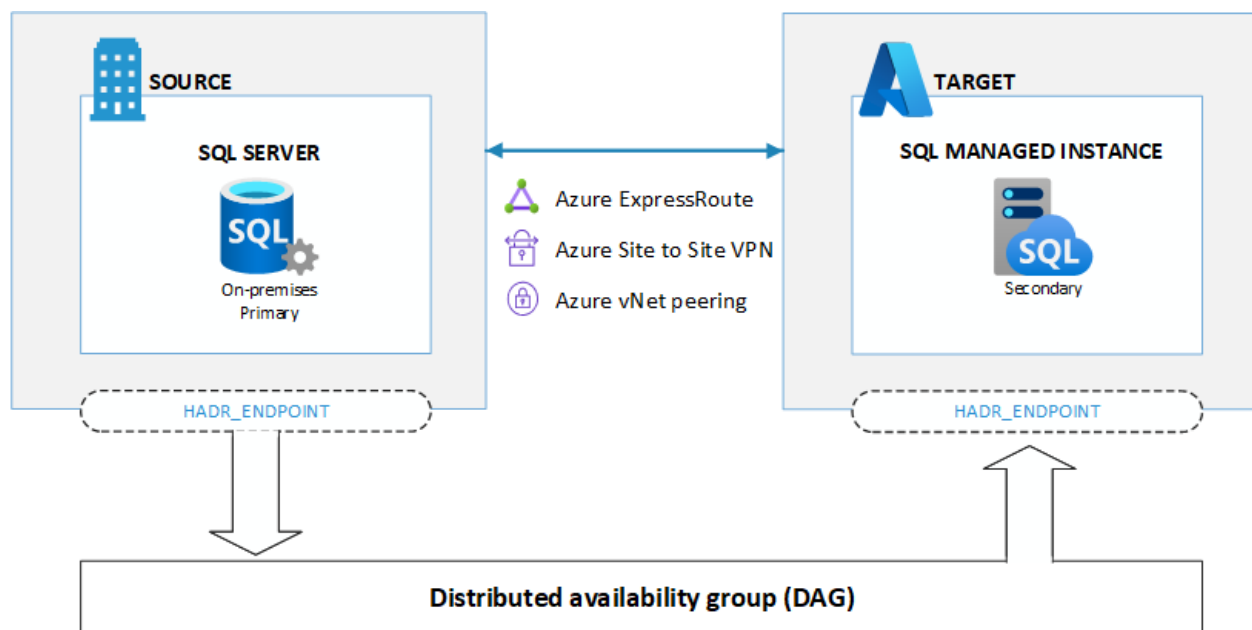
Azure SQL Managed Instance link feature offers a true online migration option compared to all other services and tools available. In addition, by partially running workloads on Azure, organizations can adopt a phased approach to cloud migration while still taking advantage of the benefits that Azure provides.

While the link is designed to replicate one database per link, it can be configured to replicate multiple databases from a single SQL Server instance to one or more SQL managed instances, or even replicate the same database to multiple SQL managed instances. This is achieved by setting up multiple links, each corresponding to a database-managed instance pair.

Hybrid flexibility with Azure SQL Managed Instance link feature

Azure SQL Managed Instance link feature allows you to replicate your SQL Server databases hosted anywhere to Azure, and fail over to the cloud in the event of a disaster or major business disruption. Azure SQL Managed Instance link also ensures seamless failover between the primary and secondary databases.

One of the advantages of using Azure SQL Managed Instance is that it's a platform as a service (PaaS), which means that the latest hardware maintenance, patching, and updates are applied and managed automatically by Azure. This ensures that your database environment is always up-to-date and secure, while also reducing the risk of downtime due to hardware failures or software vulnerabilities.



As we can see above, the link feature uses distributed availability group (DAG) and it's scoped per database (one link per one database). This allows you to consolidate multiple parallel SQL Server databases into an Azure SQL Managed Instance or scale them out across multiple instances and regions worldwide.

The link feature provides two types of replication:

- **One-way replication.** One-way replication is available for SQL Server versions 2016 and 2019 and allows you to replicate data one way from a SQL Server instance to your managed instance.
- **Two-way replication.** SQL Server 2022 provides a two-way replication feature, where you can replicate data between your managed instance and SQL Server instances, manually fail over during a disaster, and manually fail back after the disaster is mitigated. It supports an online failover but an offline failback. A preview of the online failback is available to sign up for.

Extended capabilities to the cloud

In addition to migrate workloads, there are several ways to use the link feature and use Azure services and resources, which include:

Feature	Description
Offload read-only workloads	You may want to configure secondary replicas on your SQL Server to Azure to offload reporting needs. The link feature is database scoped, allowing for consolidation of read-only workloads in Azure, which can be used to bring data closer to customers in any supported region worldwide with minimum effort.
Automated backups	Secondary replicas running on Azure SQL Managed Instance are automatically backed up to your Azure Blob Storage account, which significantly reduces administrative efforts and improve reliability.

Feature	Description
Business continuity	As a disaster recovery solution, the link feature allows you to fail over to Azure SQL Managed Instance and fail back after the disaster is mitigated.

Enable the link feature

To configure the link feature, you must follow the same steps regardless of whether you're migrating to Azure SQL Managed Instance, configuring disaster recovery on the cloud, offloading workloads to Azure, or aiming to reduce backup operations and management costs.

You can use either a wizard in SQL Server Management Studio (SSMS) or scripts. The main advantage of using scripts is that they can be automated, which can improve your deployment process, saving time and effort.

- Replicate a database by using [Azure SQL Managed Instance link wizard available in SSMS](#).
- Replicate a database by using [T-SQL and PowerShell scripts](#).

There are a few SQL Server features that aren't supported by Azure SQL Managed Instance link. For example, you can't enable the link feature if the functionality that's used on the primary SQL Server database isn't supported on Azure SQL Managed Instance, such as file tables and file streams.

For the full list of supported features, see [Limitations of Azure SQL Managed Instance link](#).

As we've seen, Azure SQL Managed Instance link feature enables organizations to confidently extend their SQL Server environments to Azure while also benefiting from the scalability, performance, and security features that Azure SQL Managed Instance offers.

Choosing between migration methods

The choice between Managed Instance link and LRS depends on your specific requirements:

Consideration	Managed Instance link	Log Replay Service
Minimum downtime required	Best choice - cutover in seconds	Longer cutover, especially for Business Critical tier
Need to read data during migration	Supported	Not available - database is in restoring state
SQL Server version	2016 and later	2012 and later
SQL Server edition	Enterprise, Standard, Developer	All editions
Network configuration	Requires VPN or private endpoint setup	Works with public endpoint by default
Migration duration	Unlimited	Maximum 30 days

Consideration	Managed Instance link	Log Replay Service
Reverse migration needed	Supported (depends on SQL MI update policy)	Not supported

For most business-critical workloads targeting either General Purpose or Business Critical SQL Managed Instance, the Managed Instance link provides the best migration experience with minimal downtime.

LRS is well-suited for general purpose workloads where some planned downtime is acceptable, or when migrating from older SQL Server versions or editions not supported by the Managed Instance link.

6. Move data to SQL Managed Instance

<https://learn.microsoft.com/en-us/training/modules/migrate-sql-workloads-azure-managed-instances/6-synchronize-data>

Move data to SQL Managed Instance

Completed

Many migrations involve a period when the on-premises and the cloud database must be kept synchronized. For example, there might be a time when clients make changes to both databases.

You've migrated the sports retail products database into Azure SQL Managed Instance. The website is already using the cloud database. You're starting to reconfigure clients to use the new database. You've decided to transition users to the new system in batches. For each team, you'll take time to resolve any problems before migrating the next users. This approach allows for troubleshooting and problem resolution without disrupting all users at once. Next, you'll reconfigure the data analysis system to use the new database in Azure. During this time, you want to ensure that the cloud and on-premises databases are synchronized every hour.

You'll explore various methods for implementing data synchronization. These methods can also be employed to selectively migrate data, should you require only a subset of the tables to be transferred. This flexibility allows for a more tailored approach to data migration.

Connectivity options with on-premises servers

Often, you want to keep data in on-premises databases synchronized with Azure SQL Managed Instance. You might want to stage the migration of client applications to the new database, for example, which means there's a period when clients connect to both databases.

Before you choose a data synchronization method, it's important to ensure you have connectivity that's secure. There are three different connectivity options available to establish communication between computers on-premises and resources in Azure.

- **Point-to-Site.** A Point-to-Site (P2S) VPN gateway connection lets you create a secure connection to your virtual network from an individual client computer.
- **Site-to-Site.** A Site-to-Site VPN gateway is used to connect an entire on-premises site to the Azure network.
- **ExpressRoute.** Azure ExpressRoute enables you to create private connections between Azure datacenters and on-premises infrastructure, or infrastructure in a colocation environment. ExpressRoute connections don't go over the public internet, and offer more reliability, faster speeds, lower latencies, and higher security than typical internet connections.

Public endpoint

Public endpoint for SQL Managed Instance helps you connect to the database from the internet without using a VPN, and is designed for data communication only. Public endpoint for data can simultaneously coexist with the private endpoint. For security reasons, the implementation allows for Separation of Duties (SoD) between a database administrator and a network administrator when enabling the public endpoint.

To enable public endpoint for managed instance, two steps are required. For SoD, you'll need two separate roles, with the following database and network permissions, to complete these steps:

1. A database administrator who has role-based access control permissions in scope `Microsoft.Sql/managedInstances/*` must run a PowerShell script to enable public endpoint for managed instance.
2. A network administrator who has role-based access control permissions in scope `Microsoft.Network/*` must open the port 3342 used by the public endpoint on the network security group (NSG), and provide a UDR route to avoid asymmetric routing.

Choose a synchronization method

You can use many methods to synchronize data from a SQL Database managed instance to an on-premises server and back.

Native backup and restore

You can restore a database in Azure SQL Managed Instance from an Azure Blob Storage file using Shared Access Signature (SAS).

This involves creating a credential with access to Azure Blob Storage, and then using the `BACKUP DATABASE` command with the `COPY_ONLY` option. If your database is larger than 200 GB, you can use a striped backup by providing several URL locations.

```
BACKUP DATABASE YourDatabase TO URL = 'https://youraccount.blob.core.windows.net/yc
```

To restore the database in SQL Managed Instance:

```
RESTORE DATABASE YourDatabase FROM URL = 'https://youraccount.blob.core.windows.net'
```

BACPAC file using SqlPackage

A BACPAC file is essentially a zipped version of both your database's metadata and data. While this deployment method is compatible with SQL Database, SQL Managed Instance doesn't support migration via BACPAC within the Azure portal. As an alternative, the [SQLPackage utility](#) should be used with the BACPAC file.

Bulk Copy Program (BCP)

The [BCP utility](#) is a command-line tool that exports tables to files so you can import them. Use this approach to migrate from a single SQL Database to SQL Managed Instance and back.

Azure Data Factory (ADF)

[Azure Data Factory](#) is built for data movement and orchestration, with the focus on ingestion. ADF has the integration runtime support to run SSIS packages, and the public internet support for SQL Managed Instance.

Transactional replication

[Transactional replication](#) is a way to move data between continuously connected database servers.

The process begins with a snapshot of the publication database objects and data. Once the initial snapshot is taken, any subsequent changes to the data or schema at the Publisher are typically delivered to Azure SQL Managed Instance in near real-time as they occur.

SQL Managed Instance is flexible because it can be a publisher, distributor, and subscriber.

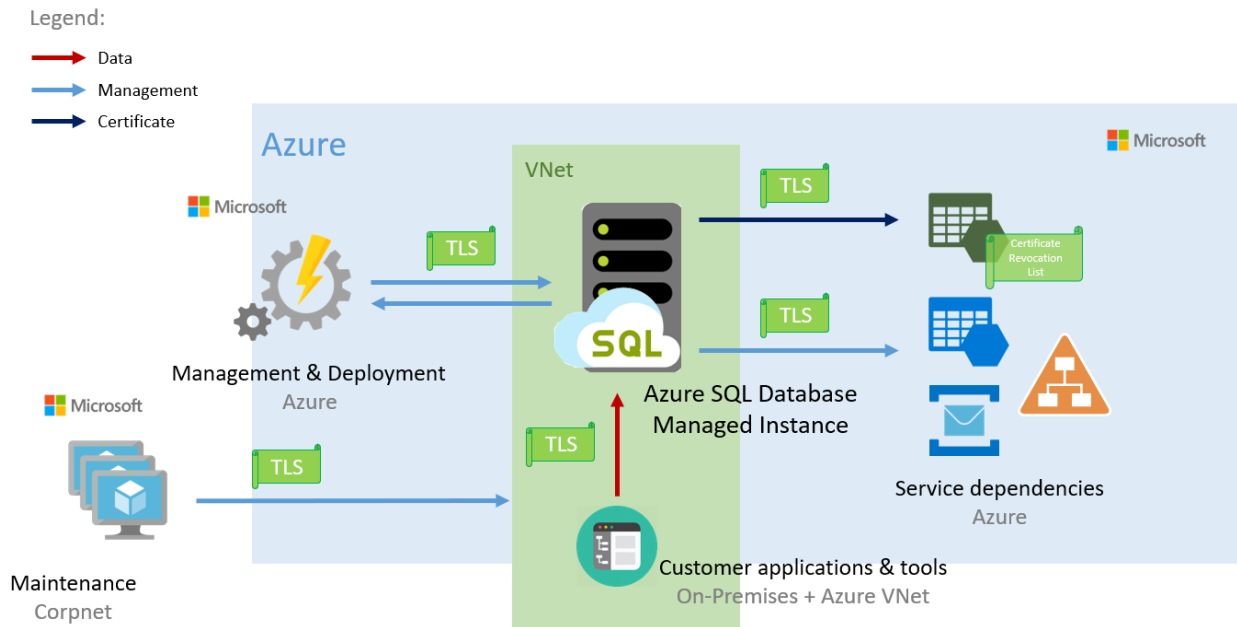
Replication is one of the few technologies that allows you to replicate parts of a table. We refer to these table parts as *articles*. This data is then sent to a distributor, which is a supplier of the data to any number of subscribers.

Requirements

- Connectivity uses SQL Authentication between replication participants.
- An Azure Storage Account share for the working directory used by replication.
- Open port 445 (TCP outbound) in the security rules of the managed instance subnet to access the Azure file share.
- Open port 1433 (TCP outbound) if the publisher or distributor is on a managed instance and the subscriber is on-premises.

Connecting applications to a SQL managed instance

A SQL managed instance must be placed inside an Azure virtual network subnet that's dedicated to managed instances. This deployment gives you a secure private IP address and the ability to connect to on-premises networks.



Users and client applications can connect to the managed instance database through the Azure portal, PowerShell, Azure CLI, and the REST API.

All communications are encrypted and signed using certificates. To check the trustworthiness of communicating parties, managed instances constantly verify these certificates through certificate revocation lists. If the certificates are revoked, the SQL managed instance closes the connections to protect the data.

7. Exercise - Migrate a SQL Server database to Azure SQL Managed Instance

<https://learn.microsoft.com/en-us/training/modules/migrate-sql-workloads-azure-managed-instances/7-exercise-migrate-sql-managed-instance>

Exercise - Migrate a SQL Server database to Azure SQL Managed Instance

Completed

Now it's your chance to migrate a SQL Server database using Log Replay Service.

In this exercise, you'll learn how to migrate a SQL Server database to Azure SQL Managed Instance using Log Replay Service.

Note

To complete this exercise, you will need an [Azure subscription](#).

Launch the exercise and follow the instructions.

Launch Exercise

8. Module assessment

<https://learn.microsoft.com/en-us/training/modules/migrate-sql-workloads-azure-managed-instances/8-knowledge-check>

Module assessment

Completed

9. Summary

<https://learn.microsoft.com/en-us/training/modules/migrate-sql-workloads-azure-managed-instances/9-summary>

Summary

Completed

In your sports clothing retail company, you needed an easy and quick way to migrate your products database into Azure. You chose Azure SQL Managed Instance because it provides a platform as a service (PaaS) implementation of SQL Server in the cloud. Azure SQL Database also has almost 100 percent compatibility with on-premises versions of SQL Server. You migrated the data, and ensured it was synchronized with the on-premises database so you could reconfigure client applications and the data analysis system.

Before SQL Managed Instance was available, system architects who wanted to migrate a database to Azure had two options:

- You could use Azure SQL Database, which provided a full PaaS solution with the consequent reduction in administrative time. However, this often required significant modification of the database, because Azure SQL Database doesn't support all the features of SQL Server.
- You could install SQL Server on a virtual machine in Azure. However, this infrastructure as a service (IaaS) solution requires you to run, administer, and update the virtual machine, its operating system, and SQL Server yourself. You could expect to spend more time on administration.

SQL Managed Instance provides a solution that's a full PaaS but supports almost all the features of SQL Server. This solution gives you the low administrative load you'd expect from Azure SQL Database but without the need to rewrite your database and change your client apps.

For additional reading, you can refer to the following URLs:

- [What is Azure SQL Managed Instance?](#)
- [Migrate SQL Server workloads to Azure SQL Database](#)
- [Tutorial: Migrate SQL Server to an Azure SQL Managed Instance online using DMS \(classic\)](#)