

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

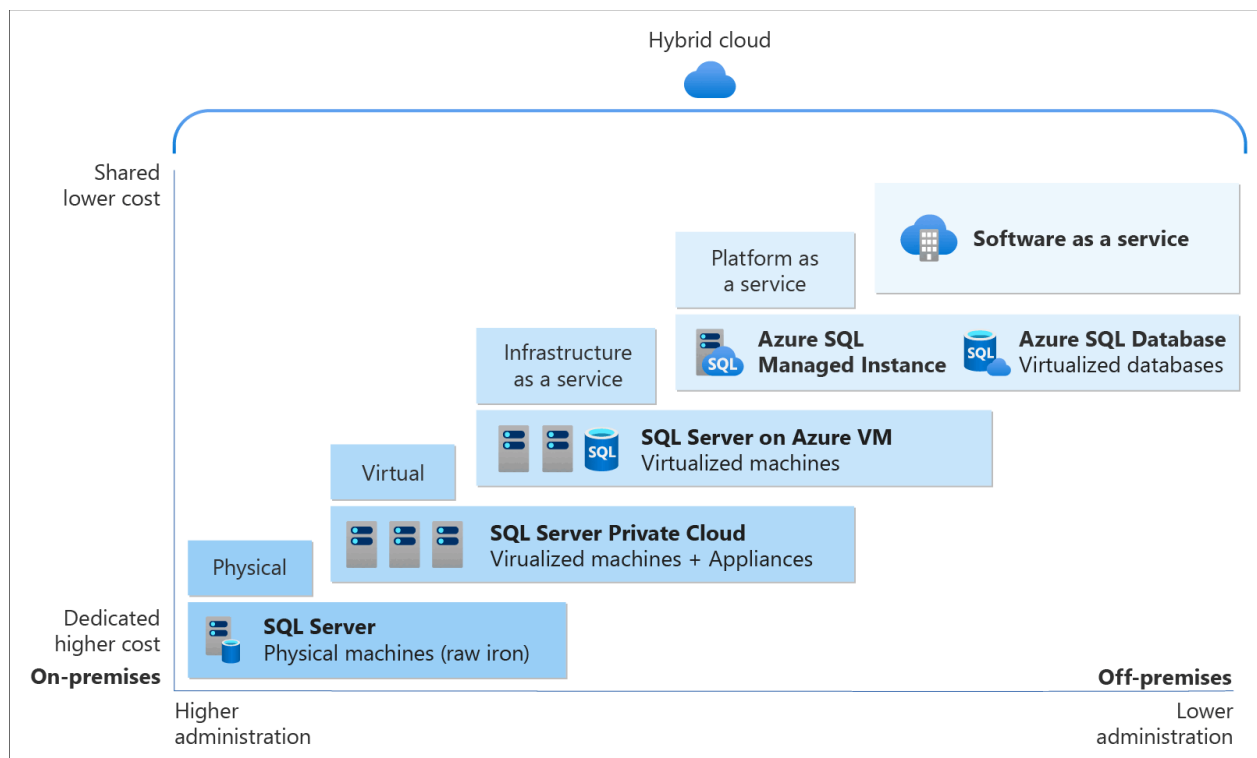
Server in Azure

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:

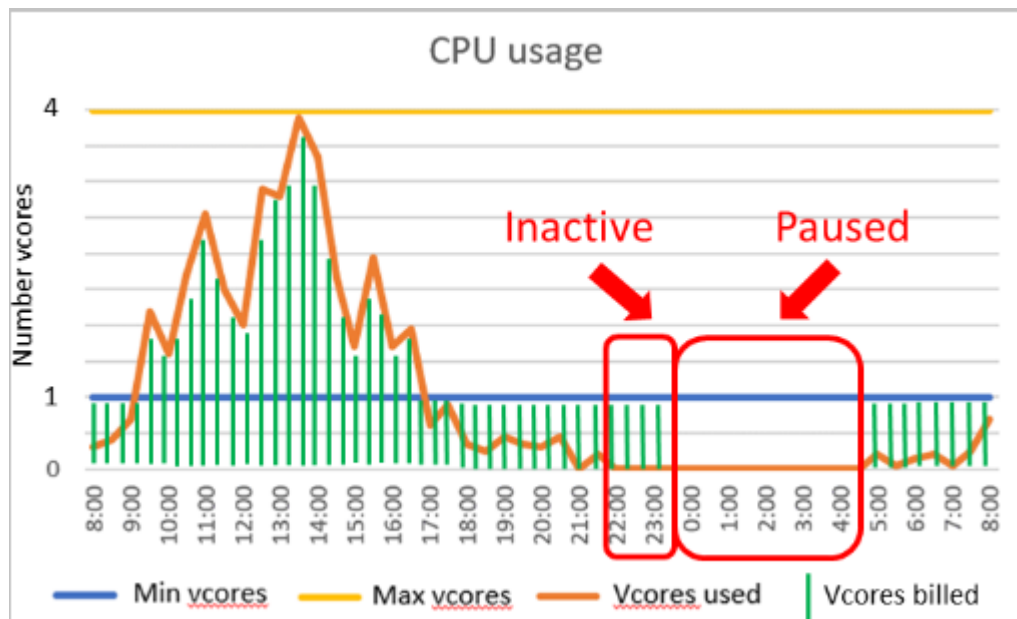
Service Tier	Best For	Storage Type	Latency	Compute Tiers	Resilience Features	Max Database Size
General Purpose	General-purpose	Azure premium	Higher than BC	Provisioned, Serverless	N/A	4 TB

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>

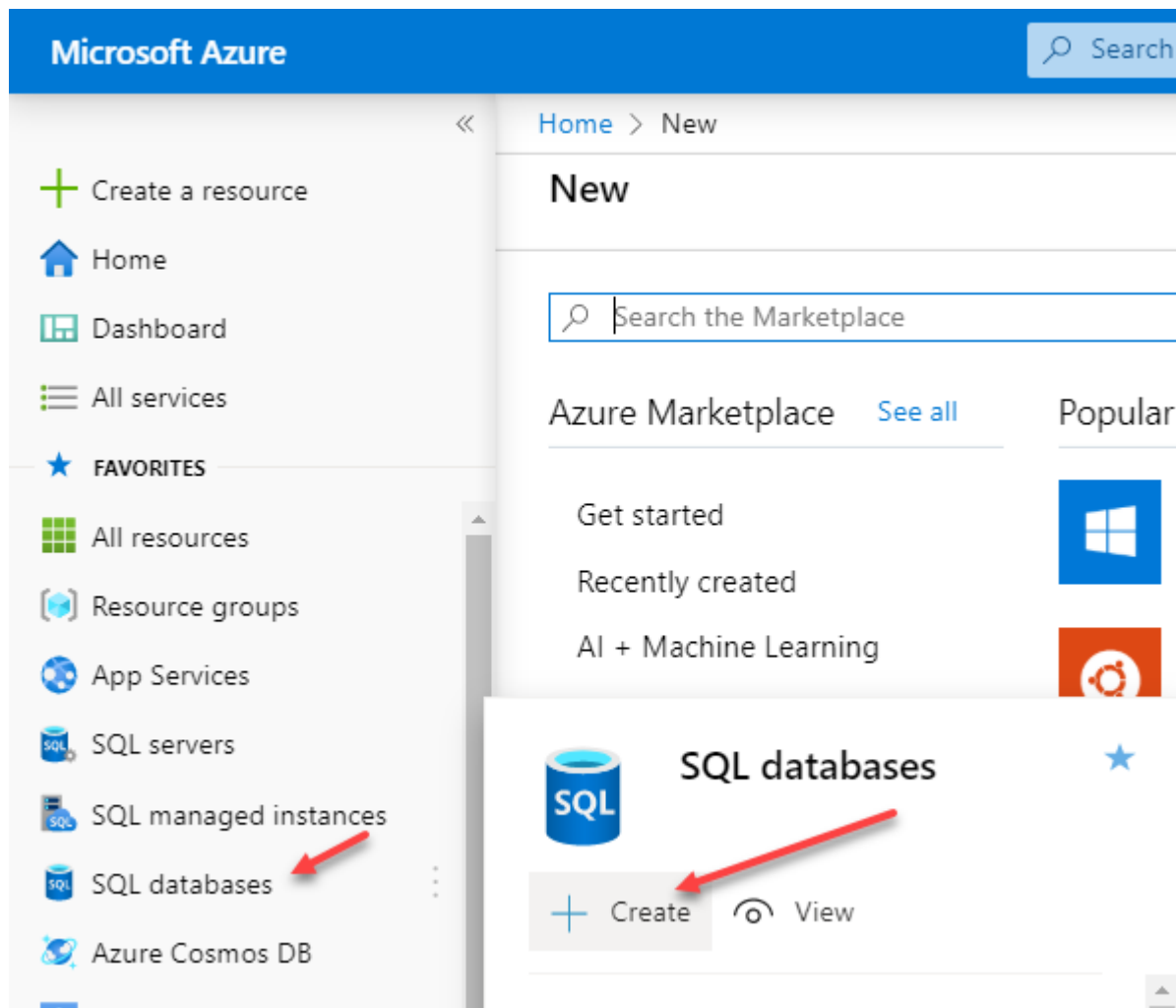
Explore single SQL database

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, (ConvertTo-SecureString "Password123!" -AsPlainText -Force))

# 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

Search to filter databases...

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

Search to filter databases...

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
 Search to filter databases...

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

Search to filter databases...

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

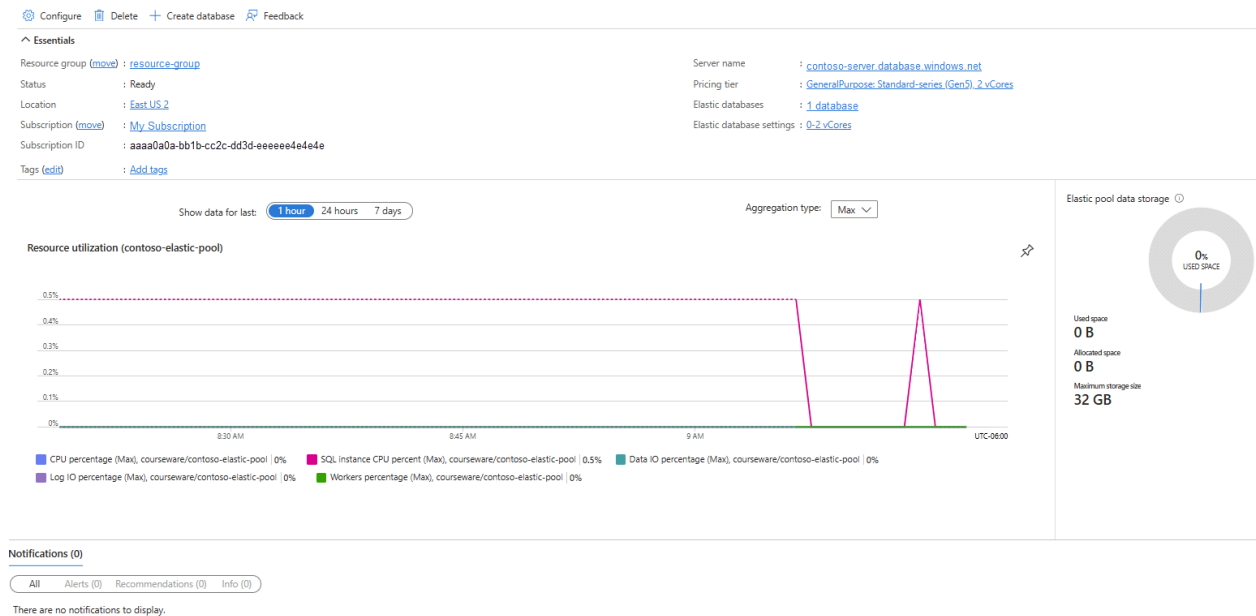
Showing 1 - 1 of 1 results.

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.

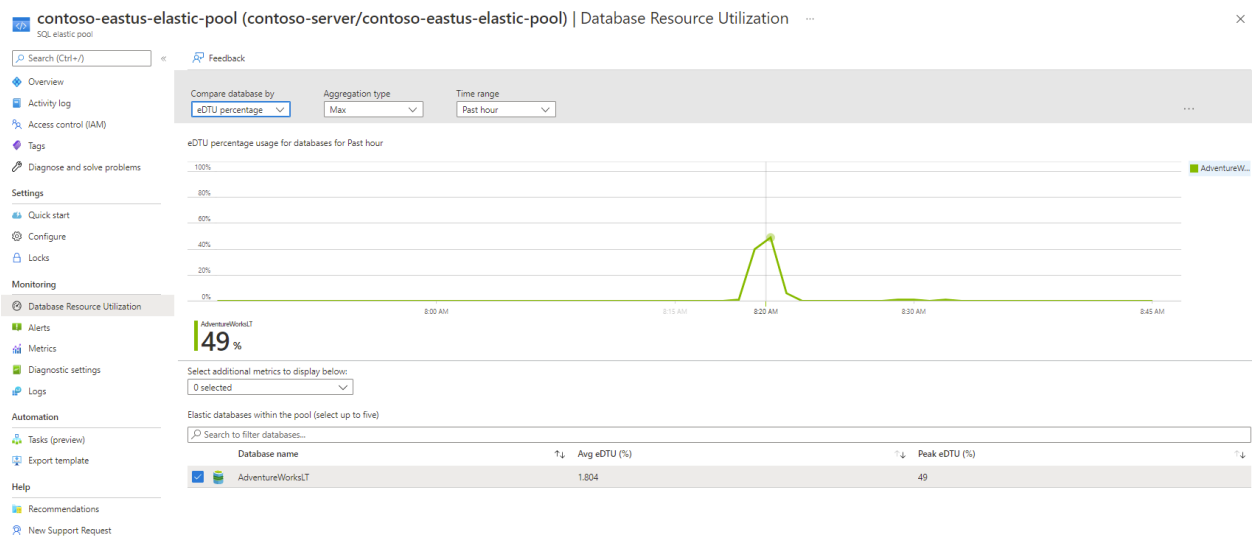


To adjust the resources allocated to your elastic pool, use the **Configure** option in the **Settings** section of the elastic pool management side menu. You can perform many changes after the creation of an elastic pool.

- 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

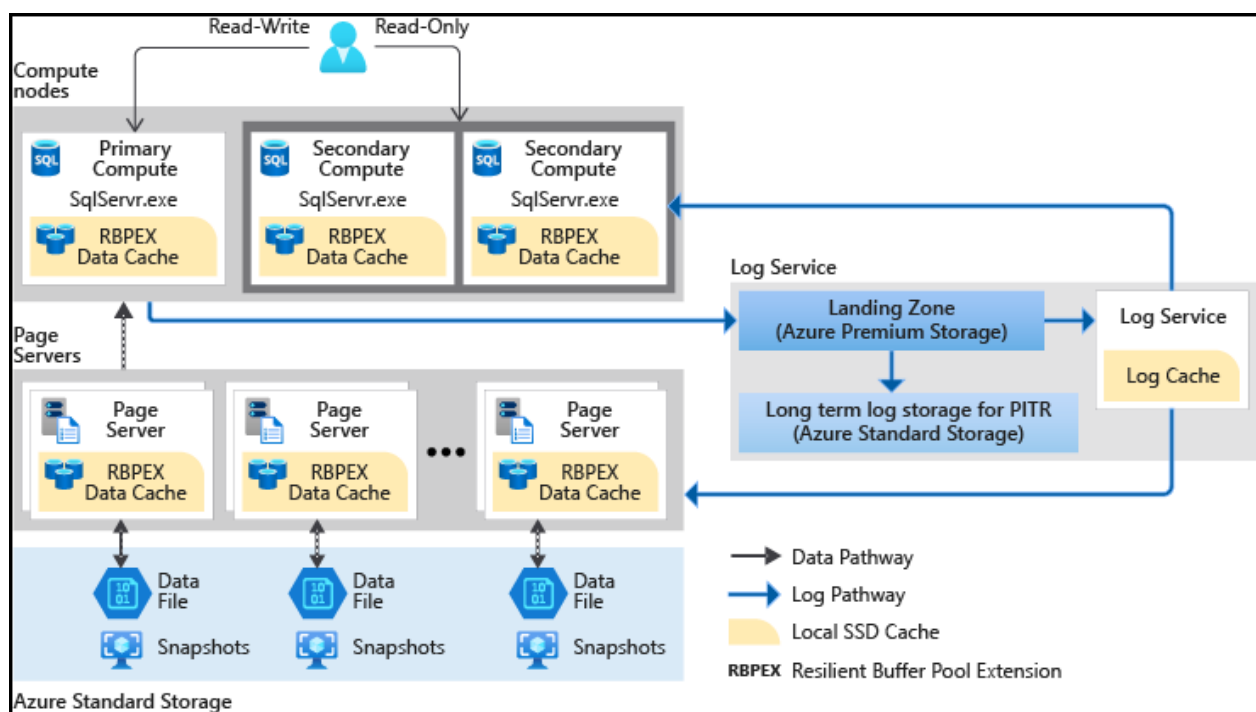
The Hyperscale service tier eliminates many of the practical limitations traditionally associated with cloud databases. The resources of a single node constrain most databases, but Hyperscale databases have no such restrictions. With its flexible storage architecture, storage expands as needed, and there's 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.

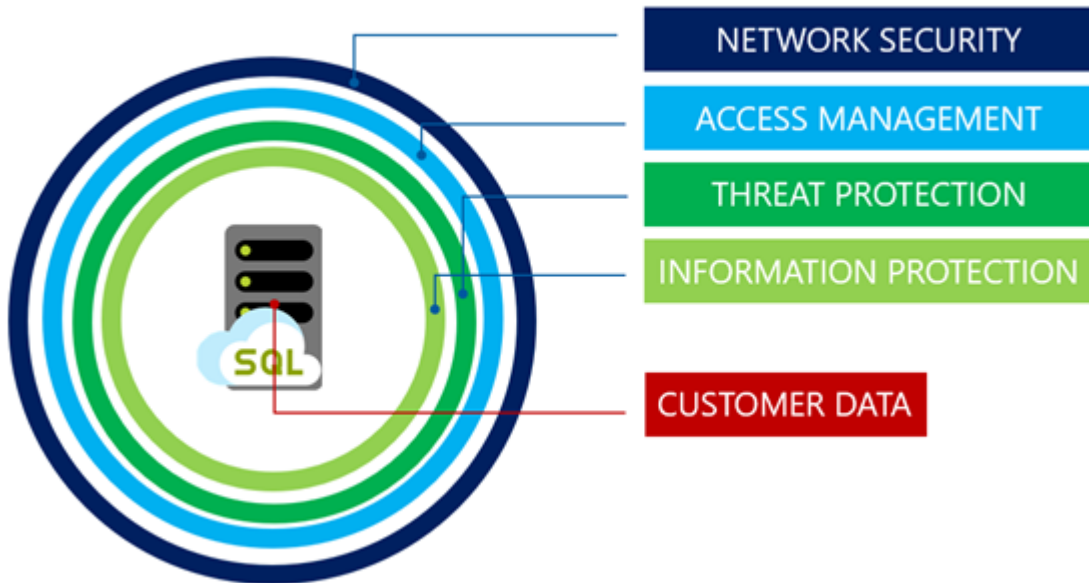
- **Scaling Up/Down** – You can increase or decrease the primary compute resources, such as CPU and memory, quickly and efficiently. Since the storage is shared, these scaling operations aren't dependent on the database's data volume.
- **Scaling In/Out** – You can create more compute replicas to handle 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