**Explore provisioning and deploying relational database services in Azure**

# Introduction

* 1 minute

Azure supports a number of database services, enabling you to run popular database management systems, such as SQL Server, PostgreSQL, and MySQL, in the cloud.

Azure database services are fully managed, freeing up valuable time you’d otherwise spend managing your database. Enterprise-grade performance with built-in high availability means you can scale quickly and reach global distribution without worrying about costly downtime. Developers can take advantage of industry-leading innovations such as built-in security with automatic monitoring and threat detection, automatic tuning for improved performance. On top of all of these features, you have guaranteed availability.

Suppose you're a data engineer at Contoso, and are responsible for creating and managing databases. You've been asked to set up three new relational data stores: Azure SQL database, PostgreSQL, and MySQL.

In this module, you'll explore the options available for creating and configuring Azure relational data services.

## Learning objectives

In this module, you will:

* Provision relational data services
* Configure relational data services
* Explore basic connectivity issues
* Explore data security

## Next unit: Describe provisioning relational data services

# Describe provisioning relational data services

* 4 minutes

In the sample scenario, Contoso has decided that the organization will require several different relational stores. As the data engineer, you've been asked to set up data stores using Azure SQL Database, PostgreSQL, and MySQL.

In this module, you'll learn how to provision these services.

## What is provisioning?

Provisioning is the act of running series of tasks that a service provider, such as Azure SQL Database, performs to create and configure a service. Behind the scenes, the service provider will set up the various resources (disks, memory, CPUs, networks, and so on) required to run the service. You'll be assigned these resources, and they remain allocated to you (and charged to you), until you delete the service.

How the service provider provisions resources is opaque, and you don't need to be concerned with how this process works. All you do is specify parameters that determine the size of the resources required (how much disk space, memory, computing power, and network bandwidth). These parameters are determined by estimating the size of the workload that you intend to run using the service. In many cases, you can modify these parameters after the service has been created, perhaps increasing the amount of storage space or memory if the workload is greater than you initially anticipated. The act of increasing (or decreasing) the resources used by a service is called scaling.

This video summarizes the process that Azure performs when you provision a service:

Azure provides several tools you can use to provision services:

* The Azure portal. This is the most convenient way to provision a service for most users. The Azure portal displays a series of service-specific pages that prompt you for the settings required, and validates these settings, before actually provisioning the service.
* The Azure command-line interface (CLI). The CLI provides a set of commands that you can run from the operating system command prompt or the Cloud Shell in the Azure portal. You can use these commands to create and manage Azure resources. The CLI is suitable if you need to automate service creation; you can store CLI commands in scripts, and you can run these scripts programmatically. The CLI can run on Windows, macOS, and Linux computers. For detailed information about the Azure CLI, read [What is Azure CLI](https://docs.microsoft.com/en-us/cli/azure/what-is-azure-cli).
* Azure PowerShell. Many administrators are familiar with using PowerShell commands to script and automate administrative tasks. Azure provides a series of commandlets (Azure-specific commands) that you can use in PowerShell to create and manage Azure resources. You can find further information about Azure PowerShell online, at [Azure PowerShell documentation](https://docs.microsoft.com/en-us/powershell/azure). Like the CLI, PowerShell is available for Windows, macOS, and Linux.
* Azure Resource Manager templates. An Azure Resource Manager template describes the service (or services) that you want to deploy in a text file, in a format known as JSON (JavaScript Object Notation). The example below shows a template that you can use to provision an instance of Azure SQL Database.

JSONCopy

"resources": [

{

"name": "sql-server-dev",

"type": "Microsoft.Sql/servers",

"apiVersion": "2014-04-01-preview",

"location": "[parameters('location')]",

"tags": {

"displayName": "SqlServer"

}

"properties": {}

}

]

You send the template to Azure using the az deployment group create command in the Azure CLI, or New-AzResourceGroupDeployment command in Azure PowerShell. For more information about creating and using Azure Resource Manager templates to provision Azure resources, see [What are Azure Resource Manager templates?](https://docs.microsoft.com/en-us/azure/azure-resource-manager/templates/overview)

## Next unit: Describe provisioning Azure SQL Database

# Describe provisioning Azure SQL Database

<https://www.microsoft.com/en-us/videoplayer/embed/RE4AkhG?pid=RE4AkhG-ax-86-id-oneplayer&postJsllMsg=true&autoplay=false&mute=false&loop=false&market=en-us&playFullScreen=false>

One of the most popular deployments within Azure relational data services is Azure SQL Database. This video demonstrates how to provision an Azure SQL Database instance, to create a database and server.

# Describe provisioning PostgreSQL and MySQL

* 3 minutes

Azure relational data services enable you to work with other leading relational database providers, such as PostgreSQL and MySQL. These services are called Azure Database for PostgreSQL and Azure Database for MySQL.

In this unit, you'll see how to provision these data stores in Azure.

## How to provision Azure Database for PostgreSQL and Azure Database for MySQL

As with Azure SQL Database, you can provision a PostgreSQL or MySQL database interactively using the Azure portal. You can find both of these services in the Azure Marketplace:

The processes for provisioning Azure Database for PostgreSQL and Azure Database for MySQL are very similar.

**Note**

PostgreSQL also gives you the hyperscale option, which supports ultra-high performance workloads.

The hyperscale deployment option supports:

* Horizontal scaling across multiple machines. This option enables the service to add and remove computers as workloads increase and diminish.
* Query parallelization across these servers. The service can split resource intensive queries into pieces which can be run in parallel on the different servers. The results from each server are aggregated back together to produce a final result. This mechanism can deliver faster responses on queries over large datasets.
* Excellent support for multi-tenant applications, real time operational analytics, and high throughput transactional workloads

The information below summarizes the fields and settings required when provisioning a PostgreSQL or a MySQL database service:

The **Basics** tab, prompts for the following details:

* **Subscription**. Select your Azure subscription.
* **Resource Group**. Either pick an existing resource group, or select **Create new** to build a new one.
* **Server Name**. Each MySQL or PostgreSQL database must have a unique name that hasn't already been used by someone else. The name must be between 3 and 31 characters long, and can only contain lower case letters, digits, and the "-" character.
* **Data Source**. Select **None** to create a new server from scratch. You can select **Backup** if you're creating a server from a geo-backup of an existing Azure Database for MySQL server.
* **Location**. Either select the region that is nearest to you, or the region nearest to your users.
* **Version**. The version of MySQL or PostgreSQL to deploy.
* **Compute + storage**. The compute, storage, and backup configurations for your new server. The **Configure server** link enables you to select the resources required to support you database workloads. These resources include the amount of computing power, memory, backups, and redundancy options (for high availability).

**Note**

The term compute refers to the amount of processor power available, but in terms of size and number of CPUs allocated to the service.

You can select between three pricing tiers, each of which is designed to support different workloads:

* + **Basic**. This tier is suitable for workloads that require light compute and I/O performance. Examples include servers used for development or testing or small-scale, infrequently used applications.
  + **General Purpose**. Use this pricing tier for business workloads that require balanced compute and memory with scalable I/O throughput. Examples include servers for hosting web and mobile apps and other enterprise applications.
  + **Memory Optimized** This tier supports high-performance database workloads that require in-memory performance for faster transaction processing and higher concurrency. Examples include servers for processing real-time data and high-performance transactional or analytical apps.

You can fine-tune the resources available for the selected tier. You can scale these resources up later, if necessary.

**Note**

The **Configure** page displays the performance that General Purpose and Memory Optimized configurations provide in terms of **IOPS**. IOPS is an acronym for Input/Output Operations per seconds, and is a measure of the read and write capacity available using the configured resources.

* **Admin username**. A sign-in account to use when you're connecting to the server. The admin sign-in name can't be **azure\_superuser**, **admin**, **administrator**, **root**, **guest**, or **public**.
* **Password**. Provide a new password for the server admin account. It must contain from 8 to 128 characters. Your password must contain characters from three of the following categories: English uppercase letters, English lowercase letters, numbers (0-9), and non-alphanumeric characters (!, $, #, %, and so on).

After you've specified the appropriate settings, select **Review + create** to provision the server.

## Next unit: Describe configuring relational data services

# Describe configuring relational data services

* 6 minutes

After you've provisioned a resource, you'll often need to configure it to meet the needs of your applications and environment. For example, you might need to set up network access, or open a firewall port to enable your applications to connect to the resource.

In this unit, you'll learn how to enable network access to your resources, and how you can prevent accidental exposure of your resources to third parties. You'll see how to use authentication and access control to protect the data managed by your resources.

## Configure connectivity and firewalls

The default connectivity for Azure relational data services is to disable access to the world.

### Configure connectivity to virtual networks and on-premises computers

To enable connectivity, use the **Firewalls and virtual networks** page for a service. To enable connectivity, choose **Selected networks**. Three further sections will appear, labeled **Virtual network**, **Firewall**, and **Exceptions**.

**Note**

An Azure Virtual Network is a representation of your own network in the cloud. A virtual network enables you to connect virtual machines and Azure services together, in much the same way that you might use a physical network on-premises. Azure ensures that each virtual network is isolated from other virtual networks created by other users, and from the Internet. Azure enables you to specify which machines (real and virtual), and services, are allowed to access resources on the virtual network, and which ports they can use.

In the **Virtual networks** section, you can specify which virtual networks are allowed to route traffic to the service. When you create items such as web applications and virtual machines, you can add them to a virtual network. If these applications and virtual machines require access to your resource, add the virtual network containing these items to the list of allowed networks.

If you need to connect to the service from an on-premises computer, in the **Firewall** section, add the IP address of the computer. This setting creates a firewall rule that allows traffic from that address to reach the service.

The **Exceptions** setting allows you to enable access to any other of your services created in your Azure subscription.

The image below shows the **Firewalls and virtual networks** page for an Azure SQL database. MySQL and PostgreSQL have a similar page.

**Note**

Azure SQL Database communicates over port 1433. If you're trying to connect from within a corporate network, outbound traffic over port 1433 might not be allowed by your network's firewall. If so, you can't connect to your Azure SQL Database server unless your IT department opens port 1433.

**Important**

A firewall rule of 0.0.0.0 enables all Azure services to pass through the server-level firewall rule and attempt to connect to a single or pooled database through the server.

### Configure connectivity from private endpoints.

**Azure Private Endpoint** is a network interface that connects you privately and securely to a service powered by Azure Private Link. Private Endpoint uses a private IP address from your virtual network, effectively bringing the service into your virtual network. The service could be an Azure service such as Azure App Service, or your own Private Link Service. For detailed information, read [What is Azure Private Endpoint?](https://docs.microsoft.com/en-us/azure/private-link/private-endpoint-overview).

The **Private endpoint connections** page for a service allows you to specify which private endpoints, if any, are permitted access to your service. You can use the settings on this page, together with the **Firewalls and virtual networks** page, to completely lock down users and applications from accessing public endpoints to connect to your Azure SQL Database account.

## Configure authentication

With Azure Active Directory (AD) authentication, you can centrally manage the identities of database users and other Microsoft services in one central location. Central ID management provides a single place to manage database users and simplifies permission management.

You can use these identities and configure access to your relational data services.

For detailed information on using Azure AD with Azure SQL database, visit the page [What is Azure Active Directory authentication for SQL database](https://docs.microsoft.com/en-us/azure/sql-database/sql-database-aad-authentication) on the Microsoft website. You can also authenticate users connecting to [Azure Database for PostgreSQL](https://docs.microsoft.com/en-us/azure/postgresql/concepts-aad-authentication) and [Azure Database for MySQL](https://docs.microsoft.com/en-us/azure/mysql/concepts-azure-ad-authentication) with AD.

## Configure access control

Azure AD enables you to specify who, or what, can access your resources. Access control defines what a user or application can do with your resources once they've been authenticated.

Access management for cloud resources is a critical function for any organization that is using the cloud. Azure role-based access control (Azure RBAC) helps you manage who has access to Azure resources, and what they can do with those resources. For example, using RBAC you could:

* Allow one user to manage virtual machines in a subscription and another user to manage virtual networks.
* Allow a database administrator group to manage SQL databases in a subscription.
* Allow a user to manage all resources in a resource group, such as virtual machines, websites, and subnets.
* Allow an application to access all resources in a resource group.

You control access to resources using Azure RBAC to create role assignments. A role assignment consists of three elements: a security principal, a role definition, and a scope.

* A **security principal** is an object that represents a user, group, service principal, or managed identity that is requesting access to Azure resources.
* A **role definition**, often abbreviated to role, is a collection of permissions. A role definition lists the operations that can be performed, such as read, write, and delete. Roles can be given high-level names, like owner, or specific names, like virtual machine reader. Azure includes several built-in roles that you can use, including:
  + **Owner** - Has full access to all resources including the right to delegate access to others.
  + **Contributor** - Can create and manage all types of Azure resources but can't grant access to others.
  + **Reader**- Can view existing Azure resources.
  + **User Access Administrator** - Lets you manage user access to Azure resources.

You can also create your own custom roles. For detailed information, see [Create or update Azure custom roles using the Azure portal](https://docs.microsoft.com/en-us/azure/role-based-access-control/custom-roles-portal) on the Microsoft website.

* A **scope** lists the set of resources that the access applies to. When you assign a role, you can further limit the actions allowed by defining a scope. This is helpful if, for example, you want to make someone a Website Contributor, but only for one resource group.

You add role assignments to a resource in the Azure portal using the **Access control (IAM)** page. The **Role assignments** tab enables you to associate a role with a security principal, defining the level of access the role has to the resource. For further information, read [Add or remove Azure role assignments using the Azure portal](https://docs.microsoft.com/en-us/azure/role-based-access-control/role-assignments-portal).

## Configure advanced data security

Apart from authentication and authorization, many services provide additional protection through advanced data security.

Advanced data security implements threat protection and assessment. Threat protection adds security intelligence to your service. This intelligence monitors the service and detects unusual patterns of activity that could be harmful, or compromise the data managed by the service. Assessment identifies potential security vulnerabilities and recommends actions to mitigate them.

The image below shows the **Advanced data security** page for SQL database. The corresponding pages for MySQL and PostgreSQL are similar.

## Next unit: Describe configuring Azure SQL Database, Azure Database for PostgreSQL, and Azure Database for MySQL

# Describe configuring Azure SQL Database, Azure Database for PostgreSQL, and Azure Database for MySQL

* 7 minutes

This unit explores the specific configuration options available to each type of data store within Azure relational data services.

## Configure Azure SQL Database

The overarching principle for network security of the Azure SQL Database offering is to allow only the connection and communication that is necessary to allow the service to operate. All other ports, protocols, and connections are blocked by default. Virtual local area networks (VLANs) and access control lists (ACLs) are used to restrict network communications by source and destination networks, protocols, and port numbers.

**Note**

An ACL contains a list of resources, and the objects (users, computers, and applications) that are allowed to access those resources. When an object attempts to use a resource that is protected by an ACL, if it's not in the list, it won't be given access.

Items that implement network-based ACLs include routers and load balancers. You control traffic flow through these items by defining firewall rules.

The following steps describe how a connection is established to an Azure SQL database:

* Clients connect to a gateway that has a public IP address and listens on port 1433.
* Depending on the effective connection policy, the gateway either redirects traffic to the database cluster, or acts as a proxy for the database cluster.

**Note**

Azure SQL Database uses a clustered topology to provide high availability. Each server and database is transparently replicated to ensure that a server is always accessible, even in the event of a database or server failure.

* Inside the database cluster, traffic is forwarded to the appropriate Azure SQL database.

### Connectivity from within Azure

If you're connecting from within another Azure service, such as a web application running under Azure App Service, your connections have a connection policy of Redirect by default. A policy of Redirect means that after your application establishes a connection to the Azure SQL database through the gateway, all following requests from your application will go directly to the database rather than through the gateway. If connectivity to the database subsequently fails, your application will have to reconnect through the gateway, when it might be directed to a different copy of the database running on another server in the cluster.

### Connectivity from outside of Azure

If you're connecting from outside Azure, such as an on-premises application, your connections have a connection policy of Proxy by default. A policy of Proxy means the connection is established via the gateway, and all subsequent requests flow through the gateway. Each request could (potentially) be serviced by a different database in the cluster.

### Configure DoSGuard

Denial of service (DoS) attacks are reduced by a SQL Database gateway service called DoSGuard. DoSGuard actively tracks failed logins from IP addresses. If there are multiple failed logins from a specific IP address within a period of time, the IP address is blocked from accessing any resources in the service for a short while.

In addition, the Azure SQL Database gateway performs the following tasks:

* It validates all connections to the database servers, to ensure that they are from genuine clients.
* It encrypts all communications between a client and the database servers.
* It inspects each network packet sent over a client connection. The gateway validates the connection information in the packet, and forwards it to the appropriate physical server based on the database name that's specified in the connection string.

## Configure Azure Database for PostgreSQL

When you create your Azure Database for PostgreSQL server, a default database named postgres is created. To connect to your database server, you need your full server name and admin sign-in credentials. You can easily find the server name and sign in information on the server **Overview** page in the portal. This page contains the Server name and the Server admin sign-in name.

**Note**

Connections to your Azure Database for PostgreSQL server communicate over port 5432. When you try to connect from within a corporate network, outbound traffic over port 5432 might not be allowed by your network's firewall. If so, you can't connect to your server unless your IT department opens port 5432.

### Configure server parameters and extensions

A PostgreSQL database server has a number of configuration parameters that you can set. These parameters support fine-tuning of the database, and debugging of code in the database. You can modify these parameters using the **Server parameters** page in the Azure portal.

If you're familiar with PostgreSQL, you'll find that not all parameters are supported in Azure. The [Server parameters](https://docs.microsoft.com/en-us/azure/postgresql/concepts-servers#server-parameters) page on the Microsoft website describes the PostgreSQL parameters that are available.

PostgreSQL also provides the ability to extend the functionality of your database using extensions. Extensions bundle multiple related SQL objects together in a single package that can be loaded or removed from your database with a single command. After being loaded in the database, extensions function like built-in features. You install an extension in your database before you can use it. To install a particular extension, run the CREATE EXTENSION command from psql tool to load the packaged objects into your database. Not all PostgreSQL extensions are supported in Azure. For a full list, read [PostgreSQL extensions in Azure Database for PostgreSQL - Single Server](https://docs.microsoft.com/en-us/azure/postgresql/concepts-extensions).

### Configure read replicas

You can replicate data from an Azure Database for PostgreSQL server to a read-only server. Azure Database for PostgreSQL supports replication from the master server to up to five replicas. Replicas are updated asynchronously with the PostgreSQL engine native replication technology.

Read replicas help to improve the performance and scale of read-intensive workloads. Read workloads can be isolated to the replicas, while write workloads can be directed to the master.

A common scenario is to have BI and analytical workloads use read replicas as the data source for reporting.

Because replicas are read-only, they don't directly reduce the burden of write operations on the master. This feature isn't targeted at write-intensive workloads.

Replicas are new servers that you manage similar to regular Azure Database for PostgreSQL servers. For each read replica, you're billed for the provisioned compute in vCores and storage in GB/month.

Use the **Replication** page for a PostgreSQL server in the Azure portal to add read replicas to your database:

## Configure Azure Database for MySQL

In order to connect to the MySQL database you've provisioned, you'll need to enter the connection information. This information includes fully qualified server name and sign-in credentials. You can find this information on the **Overview** page for your server:

**Note**

Connections to your Azure Database for MySQL server communicate over port 3306. When you try to connect from within a corporate network, outbound traffic over port 3306 might not be allowed by your network's firewall. If so, you can't connect to your server unless your IT department opens port 3306.

**Important**

By default, SSL connection security is required and enforced on your Azure Database for MySQL server.

### Configure server parameters

Like PostgreSQL, a MySQL database server has a number of configuration parameters that you can set. You can modify these parameters using the **Server parameters** page in the Azure portal.

You can find more information about the parameters available for MySQL in Azure on the [How to configure server parameters in Azure Database for MySQL by using the Azure portal](https://docs.microsoft.com/en-us/azure/mysql/howto-server-parameters) page on the Microsoft website.

### Configure read replicas

This feature is similar to that available for PostgreSQL. You can create up to five read replicas for a MySQL database. This feature enables you to geo-replicate data across regions and distribute the overhead associated with read-intensive workloads. Replication is asynchronous from the master server, so there may be some lag between records being written at the master and becoming available across all replicas.

Read replication isn't intended to support write-heavy workloads.

Use the **Replication** page for a MySQL server in the Azure portal to add read replicas to your database.

## Next unit: Exercise: Provision Azure relational database services

# Exercise: Provision Azure relational database services

* 5 minutes

This module requires a sandbox to complete. A [**sandbox**](https://docs.microsoft.com/en-us/learn/support/faq?pivots=sandbox) gives you access to Azure resources. Your Azure subscription will not be charged. The sandbox may only be used to complete training on Microsoft Learn. Use for any other reason is prohibited, and may result in permanent loss of access to the sandbox.

Due to the impact of the global health pandemic, Azure resources are being prioritized towards health and safety organizations. You may experience some issues when you deploy resources used in the exercises. Please try again or choose a different region. For more information, see Azure blog post - [**Update #3: Business continuity with Azure**](https://azure.microsoft.com/blog/update-3-business-continuity-azure/).

Activate sandbox

Top of Form

Choose your database

Azure SQL DatabaseAzure Database for PostgreSQLAzure Database for MySQL

Bottom of Form

As part of your role at Contoso as a data engineer, you've been asked to create and configure SQL Server, PostgreSQL, and MySQL servers for Azure.

The free sandbox allows you to create resources in a subset of the Azure global regions. Select a region from the following list when you create resources:

* West US 2
* South Central US
* Central US
* East US
* West Europe
* Southeast Asia
* Japan East
* Brazil South
* Australia Southeast
* Central India

## Create your Azure SQL Database service

In this exercise you'll set up your Azure SQL Database instance, which includes creating your server.

Over time if you realize you need additional compute power to keep up with demand, you can adjust performance options or even switch between the DTU and vCore performance models.

1. Sign into the [Azure portal](https://portal.azure.com/learn.docs.microsoft.com) using the same account you activated the sandbox with.
2. In the portal, select **Create a resource** from the upper left-hand corner. Select **Databases**, then select **SQL Database**.
3. Enter the following values into the form:

| **TABLE 1** | |
| --- | --- |
| **Setting** | **Value** |
| **Subscription** | Concierge Subscription |
| **Resource group** | [sandbox resource group name] |
| **Database name** | Contoso |
| **Want to use SQL elastic pool?** | No |

1. Under **Server**, select **Create new**, fill out the form with the following values, and then select **OK**:

| **TABLE 2** | |
| --- | --- |
| **Setting** | **Value** |
| **Server name** | Use your initials and the date in numeric format. For example, jpws01012020 |
| **Server admin login** | azureadmin |
| **Password** | Pa55w.rd |
| **Confirm password** | Pa55w.rd |
| **Location** | Select the default location |

1. Under **Compute + storage**, select **Configure database**.
2. On the **General Purpose** tab, leave **vCores** set to 2, change **Data max size** to **50 GB**, and then select **Apply**
3. Back on the **Create SQL Database** page, select **Additional settings**.
4. Use these values to fill out the form.

| **TABLE 3** | |
| --- | --- |
| **Setting** | **Value** |
| **Use existing data** | None |
| **Database Collation** | SQL\_Latin1\_General\_CP1\_CI\_AS |
| **Advanced Data Security** | Not now |

1. Select **Review + Create**, and then select **Create** to create your Azure SQL database.
2. On the toolbar, select **Notifications** to monitor the deployment process.

When the process completes, select **Pin to dashboard** to pin your database server to the dashboard so that you have quick access when you need it later.