Table of Contents

# Azure MySQL Developer Guide

Welcome to **THE** comprehensive guide to developing MySQL based applications on Microsoft Azure! The sections in this guide will help you navigate through the typical journey a MySQL developer building cloud-based applications will encounter. The guide will start with basics such as how to get started with tools and common helpful resources. With a base development environment setup and configured, you will then continue the journey by understanding Azure landing zones and how they can be configured in multiple ways to support MySQL applications. Finally, the ultimate goal is for you to successfully deploy a stable, performant MySQL application running securely in Microsoft Azure with cloud best practices such as security and high-availability first and foremost. Let’s get started. # Introduction to MySQL

Choosing a relational database is an important consideration for any application. MySQL database is one of the most popular open source database choices, suitable for heavily-utilized enterprise applications.

## Common Use Cases for MySQL

MySQL supports a rich set of SQL query capabilities and offers excellent performance through storage engines optimized for transactional and non-transactional workloads, in-memory processing, and robust server configuration through modules. Consult the [MySQL Documentation](https://dev.mysql.com/doc/refman/8.0/en/features.html) for a more in-depth review of MySQL’s features.

One of the most common uses cases for MySQL databases is web applications. Due to MySQL’s scalability, popular content management systems, such as [WordPress](https://wordpress.org/) and [Drupal](https://www.drupal.org/) utilize it for persistence. More broadly, LAMP apps, which integrate Linux, Apache, MySQL, and PHP, leverage scalable web servers, languages, and database engines to serve a large fraction of global web services.

### Comparison with Other RDBMS Offerings

Though MySQL has a distinct set of advantages, it competes with a vast number of other relational database offerings. Though the emphasis of this guide is operating MySQL on Azure to architect scalable applications, it is important to be aware of other offerings, notably MariaDB.

While MariaDB is compatible with the MySQL protocol, the project is not managed by Oracle, and its maintainers claim that this allows them to better compete with proprietary databases. However, MySQL has over twenty years of development experience, and businesses appreciate the platform’s maturity.

Another popular open source MySQL competitor is PostgreSQL. MySQL supports many of the advanced features of PostgreSQL, such as JSON storage, replication and failover, and partitioning, in an easy to use manner.

## Deployment Models

MySQL has a plethora of deployment options for development and production environments alike.

### On-premises

MySQL is a cross-platform offering, and corporations can utilize their on-premises hardware to deploy highly-available MySQL configurations. On-premises deployments of MySQL require significant capital expenditure.

### Docker

MySQL offers a [Docker image](https://hub.docker.com/_/mysql) to operate MySQL in containerized applications. Containerized MySQL can persist data to the machine with the Docker runtime, and it can even operate from an existing MySQL data directory.

### Other Clouds

MySQL databases can be deployed on other public cloud platforms by utilizing VMs, such as Amazon EC2; this is known as an *IaaS* deployment. However, these platforms offer their own managed MySQL products, such as [Amazon RDS for MySQL](https://aws.amazon.com/rds/mysql/) and [Google Cloud SQL for MySQL](https://cloud.google.com/sql/docs/mysql#docs). # Introduction to hosting MySQL on Azure

Now that you understand the benefits of a MySQL deployment and a few common models to operate MySQL, this section explains approaches to host MySQL on Azure and the advantages of the Azure platform.

## Azure Platform

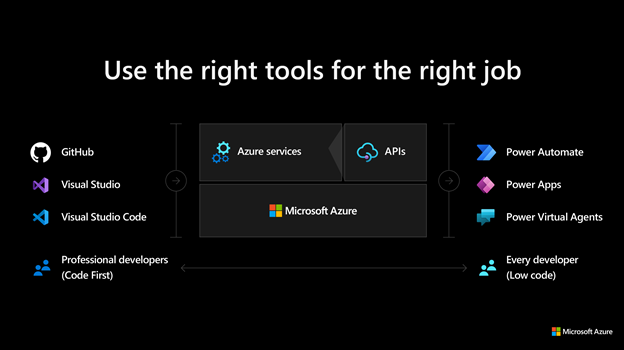
Microsoft announced Azure, formally known as Windows Azure, in 2008. At its inception, the only database offering that Windows Azure had was *SQL Azure*, based on Microsoft SQL Server. Ever since Windows Azure became generally available in 2010, Microsoft has made its cloud platform far more flexible, including support for open source databases such as MySQL and VM (IaaS) offerings.

### Advantages

The Azure platform is trusted by millions of customers and over 90,000 Cloud Solution Providers partnered with Microsoft. Azure allows firms to easily modernize their applications, expedite application development, and tailor their applications for their industries. In fact, according to internal Microsoft research, 85% of Fortune 100 companies have employed Azure AI solutions to innovate and greatly reduce their time to market for new features. Moreover, a [July 2020 Forrester study](https://azure.microsoft.com/resources/the-revenue-and-growth-opportunities-for-microsoft-azure-partners/) found that the studied group of Microsoft partners increased their gross margin by 12% due to customer demand for Azure solutions.

By offering solutions on Azure, ISVs can access the largest B2B market. In addition, through the [Azure Partner Builder’s Program](https://partner.microsoft.com/marketing/azure-isv-technology-partners), Azure assists ISVs with offering their solutions for customers to evaluate, purchase, and deploy.

Azure’s development tools, such as Visual Studio and low-code Power Apps, are part of the platform’s meteoric success. Companies that adopt capable, modern tools are 65% more innovative, according to a [2020 McKinsey & Company report.](https://azure.microsoft.com/mediahandler/files/resourcefiles/developer-velocity-how-software-excellence-fuels-business-performance/Developer-Velocity-How-software-excellence-fuels-business-performance-v4.pdf)



This image demonstrates common development tools on the Microsoft cloud platform to expedite application development.

### Free Subscription Offering

To facilitate developers’ adoption of Azure, Microsoft offers a [free subscription](https://azure.microsoft.com/free/search/) with $200 credit, applicable for thirty days; yearlong access to free quotas for popular services, including Azure Database for MySQL; and access to always free Azure service tiers.

## MySQL on Azure Hosting Options

The concepts IaaS (Infrastructure as a Service) and PaaS (Platform as a Service) describe the responsibilities of the public cloud provider and the enterprise customer to manage their data. Both approaches are common to host MySQL on Azure.

### IaaS

In the IaaS model, developers deploy MySQL on Azure Virtual Machines. This provides the customer with the flexibility to choose when to patch the VM OS and MySQL engine, and the option to install other software on the database server, such as antivirus utilities. Microsoft is responsible for the underlying VM hardware that constitutes the Azure infrastructure.

Because IaaS MySQL hosting gives greater control over the MySQL database engine and the OS, many organizations choose it to migrate on-premises solutions while minimizing capital expenditure.

### PaaS (DBaaS)

In the PaaS model, developers deploy a managed MySQL environment on Azure. Unlike IaaS, they cede control over patching the MySQL engine and OS to the Azure platform, and Azure automates many administrative tasks, like providing high availability, backups, and protecting data.

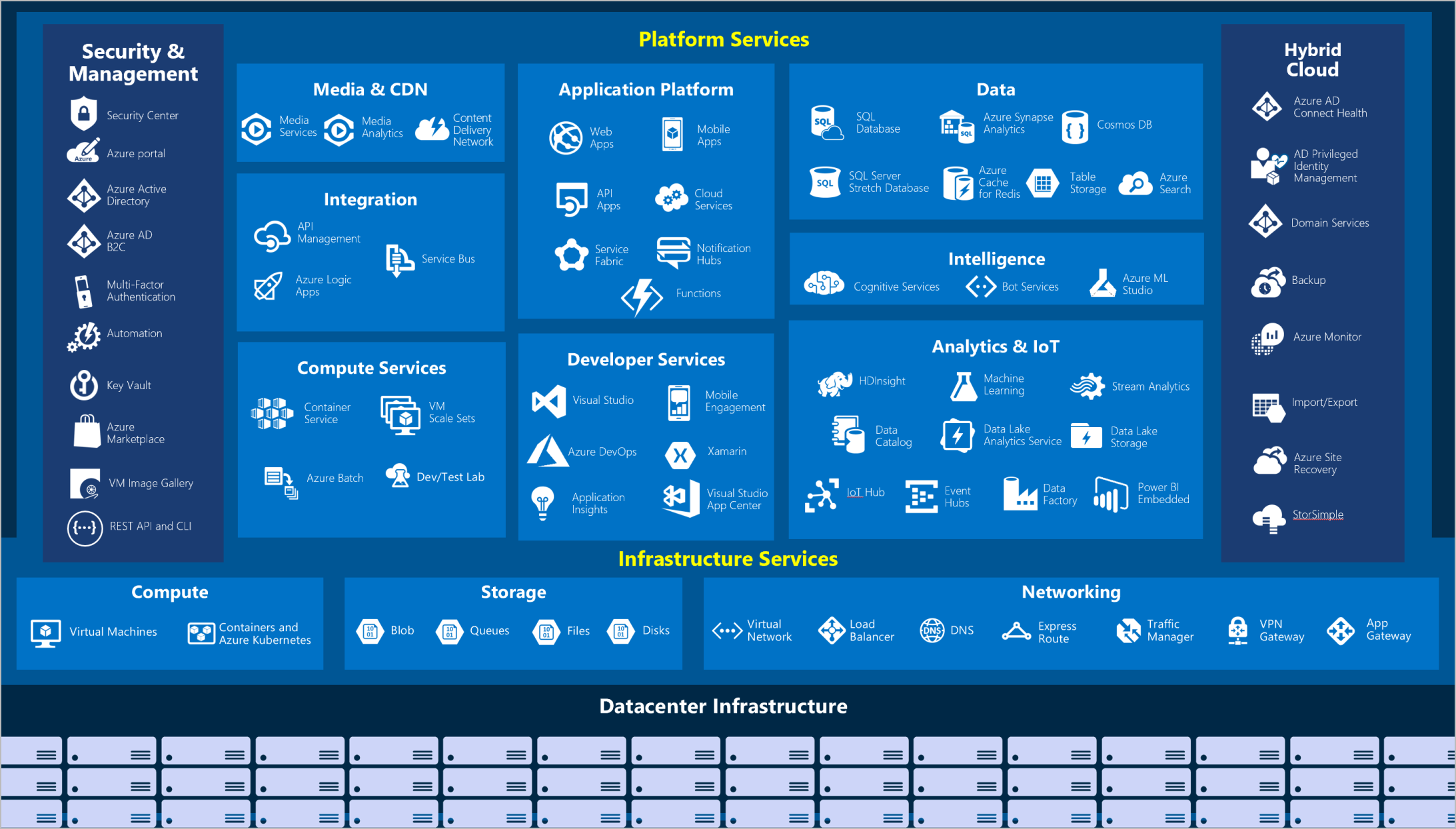
Like IaaS, customers are still responsible for managing query performance, database access, and database objects, such as indexes. It is suitable for applications where the MySQL configuration exposed by Azure is sufficient and access to the OS is unnecessary.

The Azure DBaaS MySQL offering is [Azure Database for MySQL](https://azure.microsoft.com/services/mysql/#features), which is based on MySQL community edition and supports common administration tools and programming languages. As addressed in further depth, Microsoft provides multiple deployment modes to alleviate the weaknesses of PaaS databases.

# Introduction to Azure

Now that you understand why millions of users choose Azure, and the basic database deployment models on the cloud (IaaS vs. PaaS), this document will provide more theory about how developers interact with Azure.

The image below from the [Azure Fundamentals Microsoft Learn Module](https://docs.microsoft.com/learn/modules/intro-to-azure-fundamentals/) demonstrates how IaaS and PaaS can classify Azure services. Moreover, Azure empowers flexible *hybrid cloud* deployments and supports a variety of common tools, such as Visual Studio and the Azure CLI, to manage Azure environments.



This image shows the classification of Azure services into IaaS and PaaS categories.

Here is a summary of the Azure services used in the whitepaper scenario besides the managed MySQL offerings described previously.

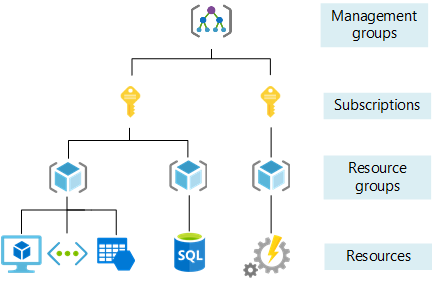
* **Virtual Machines (IaaS)**: You will begin by running a PHP sample application on an Azure Windows Server Virtual Machine.
* **Azure App Service (PaaS)**: You will deploy the PHP application to Azure App Service, a flexible, simple-to-use application hosting service.
* **Azure Container Instances (PaaS)**: You will *containerize* your app on the VM to operate in an environment isolated from other development tools installed on the system. Azure Container Instances provides a managed environment to operate containers.
* **Azure Kubernetes Service (PaaS)**: AKS also hosts containerized apps, but it is optimized for more advanced orchestration scenarios, such as high availability.

This list is not exhaustive; for a more comprehensive view, consult the [Azure Fundamentals Microsoft Learn](https://docs.microsoft.com/learn/modules/intro-to-azure-fundamentals/tour-of-azure-services) module.

## The Azure Management Hierarchy

Azure provides a flexible resource hierarchy to simplify cost management and security. This hierarchy consists of four levels:

* **Management groups**: Management groups consolidate multiple Azure subscriptions for compliance and security purposes.
* **Subscriptions**: Subscriptions govern cost control and access management. Azure users cannot provision Azure resources without a subscription.
* **Resource groups**: Resource groups consolidate the individual Azure resources for a given deployment. All provisioned Azure resources belong to one resource group. In this whitepaper, you will provision a *resource group* in your *subscription* to hold the required resources.
  + Resource groups have a geographic location that determines where metadata about that resource group is stored
* **Resources**: An Azure resource is an instance of a service. An Azure resource belongs to one resource group located in one subscription.
  + Most Azure resources are provisioned in a particular region



This image shows Azure resource scopes.

The *Azure Resource Manager* governs interaction with Azure resources. All Azure management tools, including the CLI, PowerShell module, REST API, and browser-based Portal, interact with the Azure Resource Manager. Observe that security is built into Azure Resource Manager.

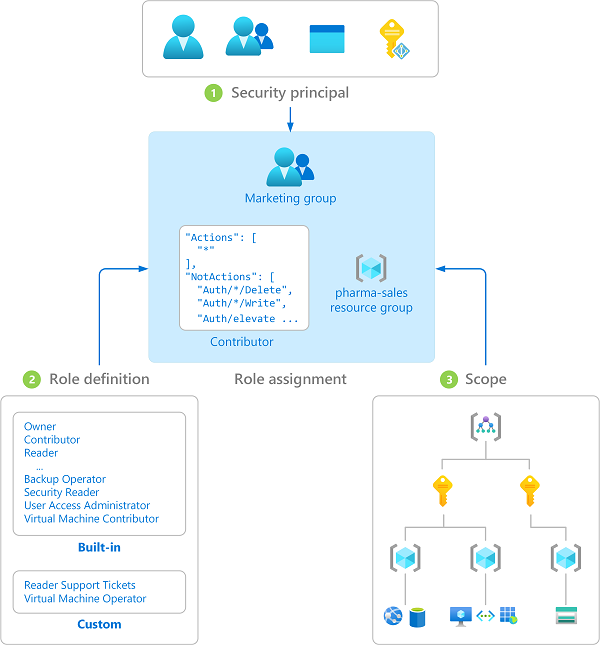


This image demonstrates how the Azure Resource Manager provides a robust, secure interface to Azure resources.

After users and services authenticate with Azure, *role-based access control* (RBAC), built on the Azure Resource Manager, dictates whether they are authorized to access a resource, modify the resource, or give other users and services access to that resource. RBAC consists of three parts:

* A *role definition* describes a set of actions that can be performed. It can be broad or granular.
* A *security principal* represents a user, a group of users, or a service.
* The *scope* dictates at what level a role assignment to a security principal applies.

An example is assigning the *Contributor* role over a resource group to a developer in your organization. In this case, the *Contributor* role allows the developer to manage all resources contained within the resource group but not manage other users’ access to those resources. The scope is the resource group, and the security principal is the developer’s account in Azure Active Directory.



This image demonstrates the three components of Azure RBAC.

### Azure Management Tools

The flexibility and variety of Azure’s management tools make it intuitive for all users, irrespective of their skill level with certain technologies.

The **Azure portal** gives developers a quick view of the state of their Azure resources. It supports extensive user configuration and simplifies custom reporting. The **Azure mobile app** provides similar features for mobile users.

**Azure PowerShell** and the **Azure CLI** (for Bash shell users) are useful for automating tasks that cannot be performed in the Azure portal. Note that both of these tools follow an *imperative* approach, meaning that users must explicitly script the creation of resources in the correct order. These tools do not support creating resources in parallel.

Lastly, **ARM templates** are optimal for deploying resources in a *declarative* manner. Azure Resource Manager can potentially create the resources in an ARM template in parallel. ARM templates are useful to create multiple identical environments, such as development, staging, and production environments.

### Other Tips

To develop an effective hierarchy, administrators must consult with cloud architects and financial and security personnel. Here are a few other best practices to follow for Azure deployments.

* **Adopt a naming convention:** Names in Azure should include business details, such as the organization department, and operational details for IT personnel, like the workload.
* **Adopt other Azure governance tools:** Azure provides mechanisms such as [resource tags](https://docs.microsoft.com/azure/azure-resource-manager/management/tag-resources?tabs=json) and [resource locks](https://docs.microsoft.com/azure/azure-resource-manager/management/lock-resources?tabs=json) to facilitate compliance, cost management, and security.

## Resources to guide an Azure Deployment

### Support

Azure provides [multiple support plans for businesses](https://azure.microsoft.com/support/plans/), depending on their business continuity requirements. There is also a large user community:

* [StackOverflow Azure Tag](https://stackoverflow.com/questions/tagged/azure)
* [@Azure on Twitter](https://twitter.com/azure)

### Training

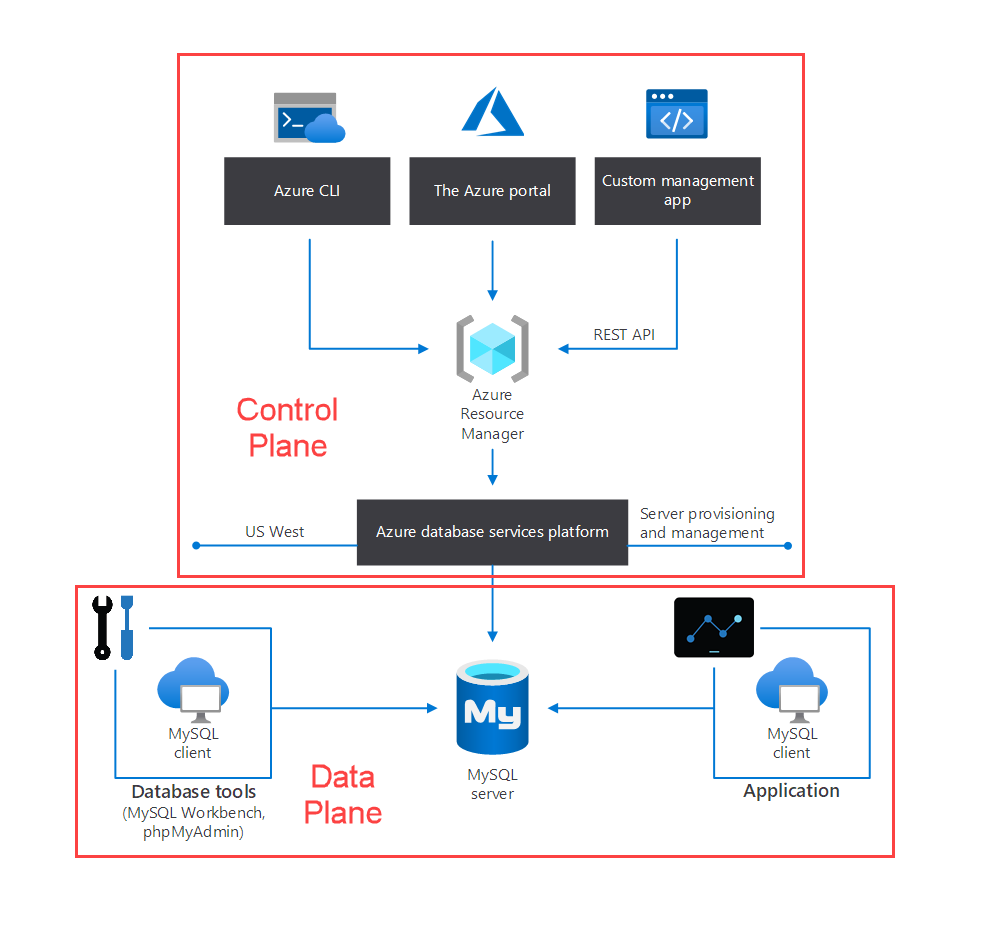
* [Azure Certifications & Exams](https://docs.microsoft.com/learn/certifications/browse/?products=azure)
* [Microsoft Learn](https://docs.microsoft.com/learn/)
  + [Azure Fundamentals (AZ-900) Learning Path](https://docs.microsoft.com/learn/paths/az-900-describe-cloud-concepts/)

# Introduction to Azure Database for MySQL

As mentioned previously, developers can deploy MySQL on Azure through Virtual Machines (IaaS) or Azure Database for MySQL (PaaS). Though PaaS offerings do not support direct management of the OS and the database engine, they have built-in support for high availability, automating backups, and meeting compliance requirements. Moreover, Azure Database for MySQL supports MySQL Community Editions 5.6, 5.7, and 8.0, making it flexible for most migrations. For most use cases, Azure PaaS MySQL allows developers to focus on application development and deployment, instead of OS and RDBMS management, patching, and security.

As the image below demonstrates, Azure Resource Manager handles resource configuration, meaning that standard Azure management tools, such as the CLI, PowerShell, and ARM templates, are still applicable. This is termed the *control plane*.

For managing database objects and access controls on those objects, standard MySQL management tools, such as [MySQL Workbench](https://www.mysql.com/products/workbench/), still apply. This is known as the *data plane*.



This image demonstrates the control plane for Azure PaaS MySQL.

## Azure Database for MySQL Deployment Modes

Azure provides both *Single Server* and *Flexible Deployment* modes. Below is a summary of these offerings. For a more comprehensive comparison table, please consult [this document.](https://docs.microsoft.com/azure/mysql/select-right-deployment-type)

### Single Server

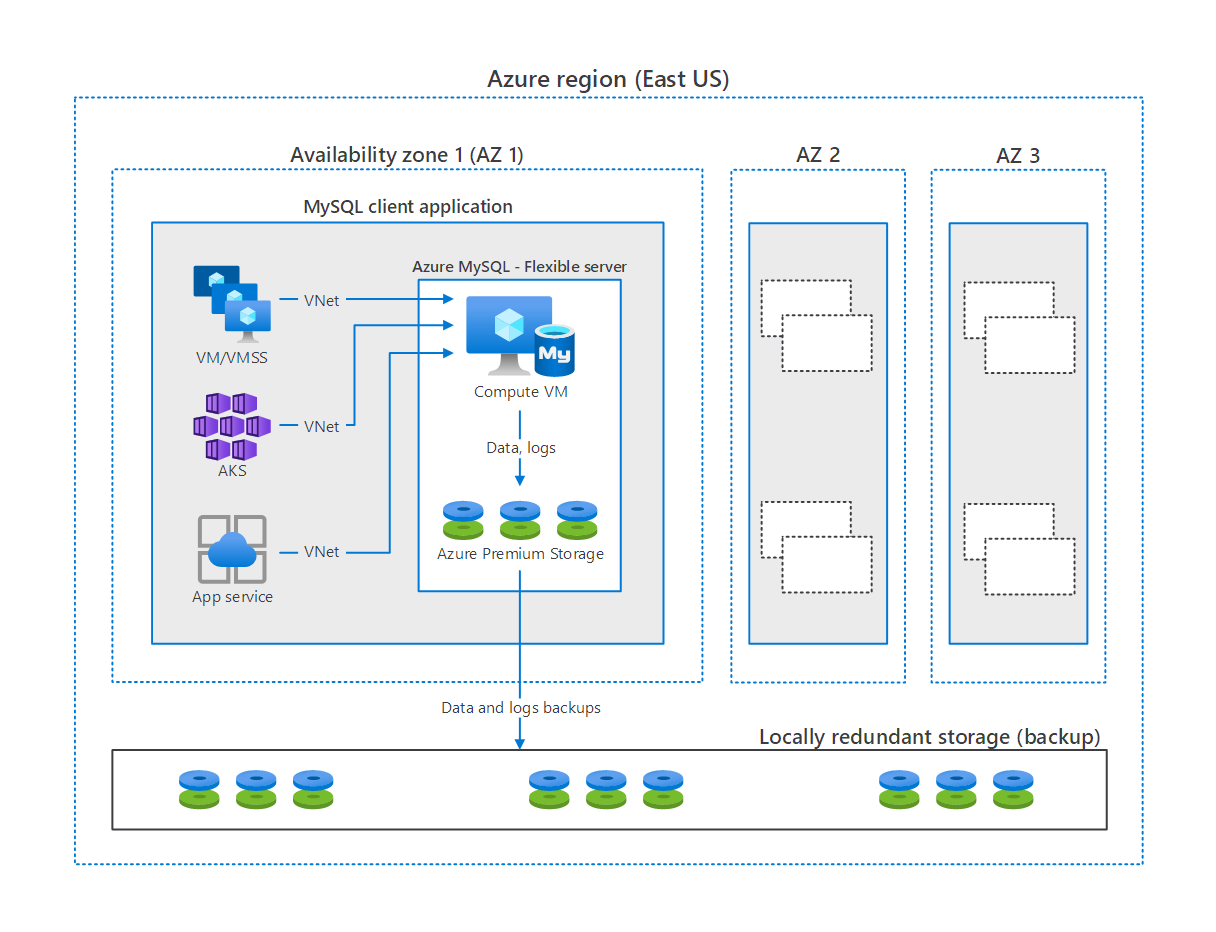
Single Server is suitable when apps do not need extensive database customization. Single Server will manage patching, offer high availability, and manage backups on a predetermined schedule (though developers can set the backup retention times between a week and 35 days). To reduce compute costs, developers can [pause the Single Server offering.](https://docs.microsoft.com/azure/mysql/how-to-stop-start-server) It offers an [SLA of 99.99%](https://azure.microsoft.com/updates/azure-database-for-mysql-general-availability/).

For a refresher on how the SLAs of individual Azure services affect the SLA of the total deployment, review the associated [Microsoft Learn Module.](https://docs.microsoft.com/learn/modules/choose-azure-services-sla-lifecycle/)

### Flexible Server

Flexible Server is also managed by the Azure platform, but it exposes more control to the user. Cost management is one of the major advantages of Flexible Server: it supports a *burstable* tier, which is based on the B1MS Azure VM tier and is optimized for workloads that do not continually use the CPU. Just like Single Server, [Flexible Server can also be paused.](https://docs.microsoft.com/azure/mysql/flexible-server/how-to-restart-stop-start-server-cli) The image below shows how Flexible Server works for a non-high availability arrangement.

*Locally-redundant storage* replicates data within a single *availability zone*. *Availability zones* are present within a single Azure region (such as East US) and are geographically isolated. All Azure regions that support availability zones have at least three.

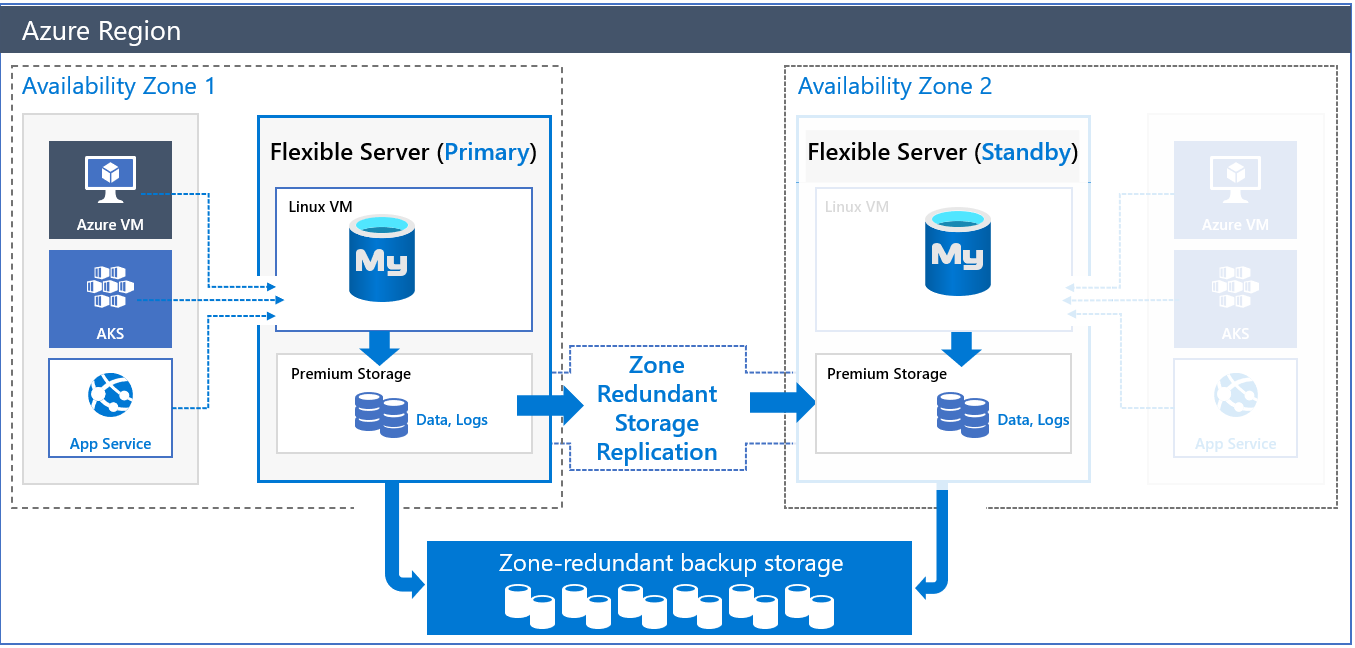


This image demonstrates how MySQL Flexible Server works, with compute, storage, and backup storage.

#### High Availability

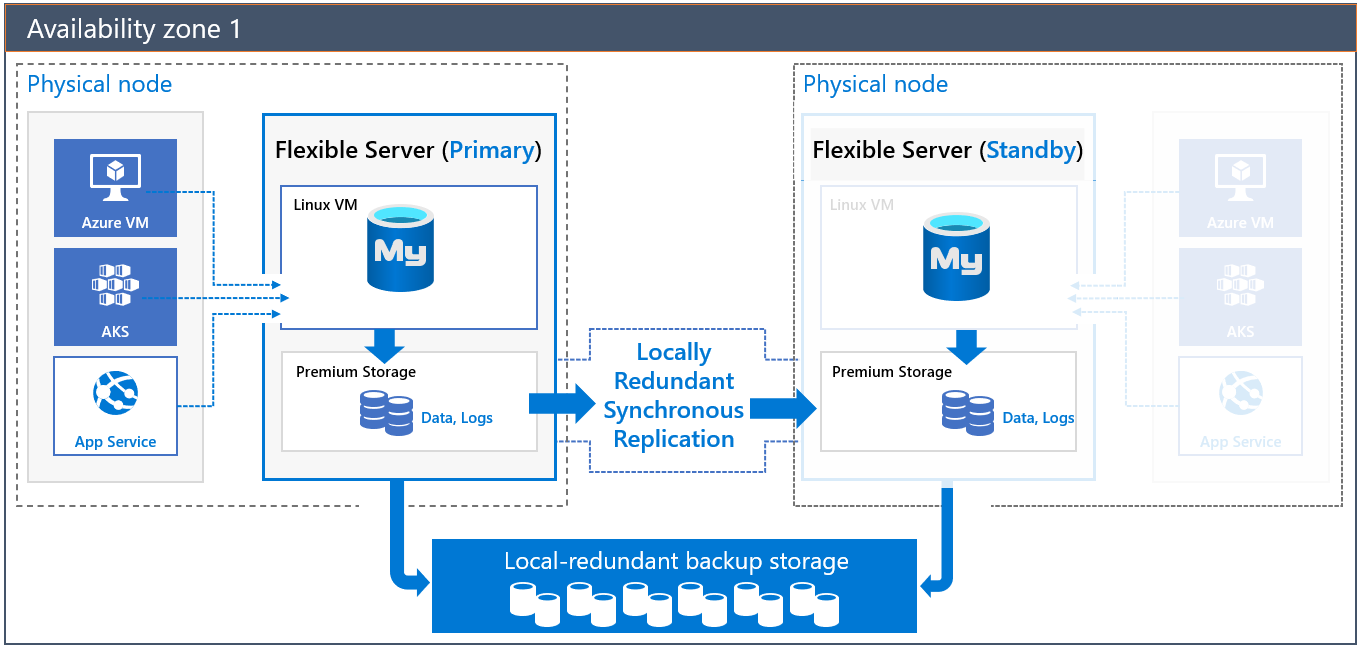
The image above does not feature high availability. Flexible Server implements HA by provisioning another VM to serve as a standby.

It is possible to provision this secondary Flexible Server VM in another availability zone, as shown below. As mentioned previously, this HA option is only supported for Azure regions with availability zones. While this option does provide redundancy against zonal failure, there is more latency between the zones that affects replication.



This image demonstrates Zone-Redundant HA for MySQL Flexible Server.

To compensate for the latency challenges, Azure provides HA within a single zone. In this configuration, both the primary node and the standby node are in the same zone. All Azure regions support this mode. Of course, it does not insulate against zonal failure.



This image demonstrates HA for MySQL Flexible Server in a single zone.

To learn more about HA with MySQL Flexible Server, consult the [documentation.](https://docs.microsoft.com/azure/mysql/flexible-server/concepts-high-availability)

#### Flexible Server Pricing & TCO

All three MySQL Flexible Server tiers offer a storage range between 20 GiB and 16 TiB and the same backup retention period range of 1-35 days. However, they differ in core count and memory per vCore. Choosing a compute tier affects the database IOPS and pricing.

* Burstable: This tier corresponds to a B-series Azure VM. Instances provisioned in this tier have 1-2 vCores. It is ideal for applications that do not utilize the CPU consistently.
* General Purpose: This tier corresponds to a Ddsv4-series Azure VM. Instances provisioned in this tier have 2-64 vCores and 4 GiB memory per vCore. It is ideal for most enterprise applications requiring a strong balance between memory and vCore count.
* Memory Optimized: This tier corresponds to an Edsv4-series Azure VM. Instances provisioned in this tier have 2-64 vCores and 8 GiB memory per vCore. It is ideal for high-performance or real-time workloads that depend on in-memory processing.

To estimate the TCO for Azure Database for MySQL, use the [Azure Pricing Calculator](https://azure.microsoft.com/pricing/calculator/). Note that you can also use the [Azure TCO Calculator](https://azure.microsoft.com/pricing/tco/calculator/) to estimate the cost savings of deploying PaaS Azure MySQL over the same deployment in an on-premises data center. Simply indicate your on-premises hardware and the Azure landing zone, adjust calculation parameters, like the cost of electricity, and observe the potential savings.

#### Flexible Server Unsupported Features

Azure provides a [detailed list of the limitations of Flexible Server](https://docs.microsoft.com/azure/mysql/flexible-server/concepts-limitations). Here are a few notable ones.

* Support for only the InnoDB and MEMORY storage engines; MyISAM is unsupported
* The DBA role and the SUPER privilege are unsupported
* SELECT ... INTO OUTFILE statements to write query results to files are unsupported, as the filesystem is not directly exposed by the service

# Setup and Tools

## Azure Free Account

As described in the [Why Move to Azure document](../02_IntroToMySQL/02_01_Why_Move_To_Azure.md) document, Azure offers a $200 free credit for developers to trial Azure. Enroll today to explore MySQL offerings on Azure.

## Azure Subscriptions and Limits - TODO

## Azure Authentication - TODO

## Creating Landing Zones

The term *landing zone* refers to an Azure environment that supports application migration and modernization by facilitating scalability, security, governance, and more. Resources can be deployed to an Azure environment through the following tools.

## Azure CLI Tools

The Azure CLI is geared towards Bash shell users and is useful for automating tasks that cannot easily be performed in the Azure portal. Note that the CLI follows an *imperative* approach, meaning that users must explicitly script the creation of resources in the correct order, handle errors, and more.

It is possible to run the Azure CLI from the [Azure Cloud Shell](shell.azure.com) or to [download the CLI tools locally from Microsoft.](https://docs.microsoft.com/cli/azure/install-azure-cli)

## Azure PowerShell

Like the Azure CLI, Azure PowerShell is a useful automation tool that falls into the imperative infrastructure management category. It is geared towards Windows administrators, though PowerShell is cross-platform.

Again, developers can run Azure PowerShell directly from the [Azure Cloud Shell](shell.azure.com) or install the Az module from the PowerShell Gallery, as described in the [installation document.](https://docs.microsoft.com/powershell/azure/install-az-ps?view=azps-6.6.0)

## Visual Studio Code

Visual Studio Code is an open-source, cross-platform text editor. It offers useful utilities for various languages through extensions. Download Visual Studio Code from the [Microsoft download page.](https://code.visualstudio.com/download)

There is a [MySQL](https://marketplace.visualstudio.com/items?itemName=formulahendry.vscode-mysql) extension that allows developers to organize their database connections, administer databases, and query databases. Consider adding it to your Visual Studio Code workflow for MySQL.

# Environment Setup

## Azure

## Windows

## Linux

## MacOS

## SDKs

# Provision Flexible Server and Database

This document illustrates how to deploy MySQL Flexible Server using various Azure management tools.

## Azure Portal

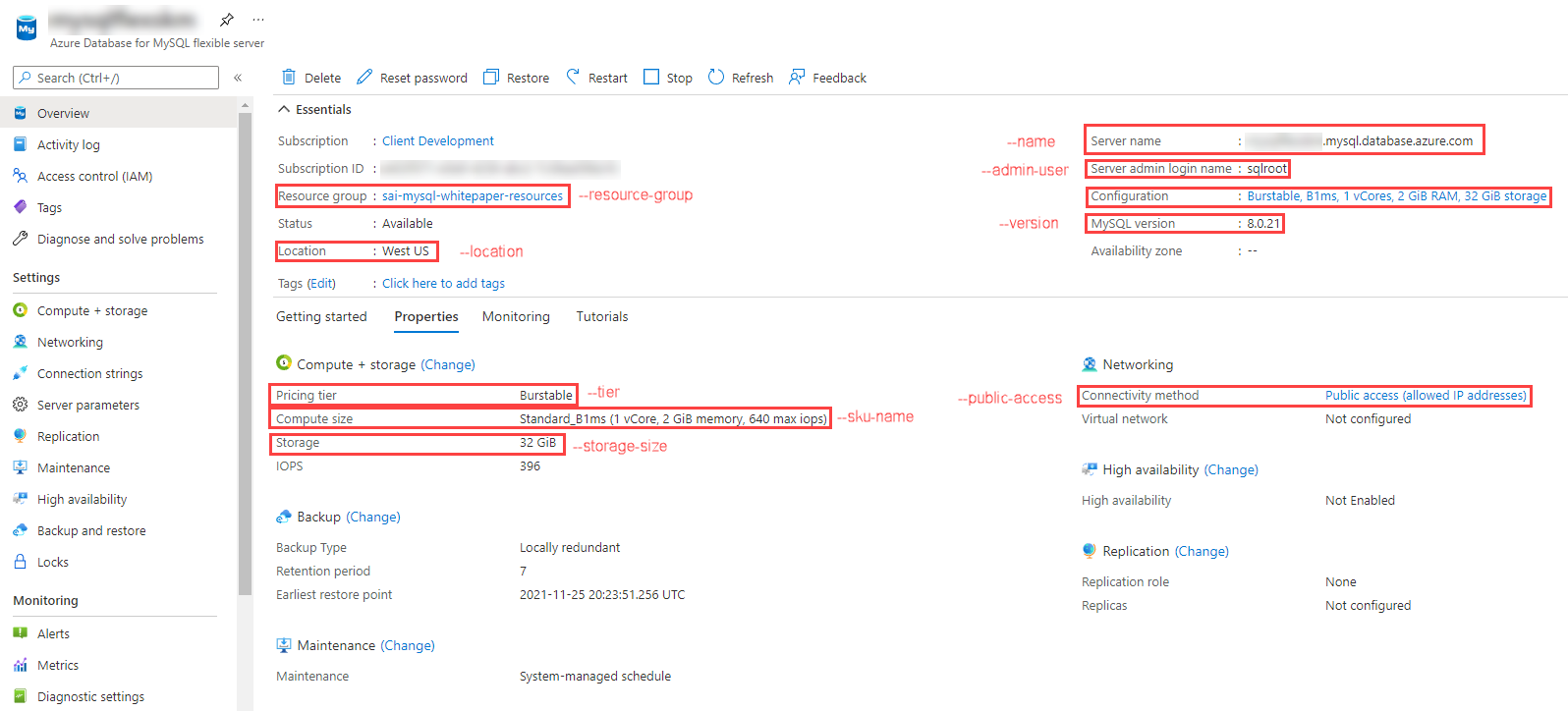
Azure provides a [quickstart document](https://docs.microsoft.com/azure/mysql/flexible-server/quickstart-create-server-portal) for users who would like to use the Azure portal to provision Flexible Server. While this is a great opportunity to explore the configuration parameters of Flexible Server, IaC approaches, like the imperative Azure CLI or the declarative ARM template, are preferable to create deployments that can easily be replicated in other environments.

## Azure CLI

The Azure CLI az mysql flexible-server set of commands is very robust. [Azure’s quickstart guide](https://docs.microsoft.com/azure/mysql/flexible-server/quickstart-create-server-cli) demonstrates how the az mysql flexible-server create and az mysql flexible-server db create commands can automatically populate server parameters. Note that it is possible to exercise greater control over these commands by reviewing the documentation for the [flexible-server create](https://docs.microsoft.com/cli/azure/mysql/flexible-server?view=azure-cli-latest#az_mysql_flexible_server_create) and [flexible-server db create](https://docs.microsoft.com/cli/azure/mysql/flexible-server/db?view=azure-cli-latest#az_mysql_flexible_server_db_create) commands.

Running the CLI commands from [Azure Cloud Shell](shell.azure.com) is preferable, as the context is already authenticated with Azure.

The image below, from a successful CLI provisioning attempt for Flexible Server, maps CLI flags to various Flexible Server parameters.



This image demonstrates the MySQL Flexible Server provisioned through Bash CLI commands.

## ARM Template

Azure provides a [quickstart document](https://docs.microsoft.com/azure/mysql/flexible-server/quickstart-create-arm-template#review-the-template) with a comprehensive ARM template for a Flexible Server deployment. We have also provided a [sample ARM template](mysql-flexible-server-template.json) that just requires the serverName, administratorLogin, and administratorLoginPassword parameters to deploy: the Azure sample template requires additional parameters to run. It can be deployed with the New-AzResourceGroupDeployment PowerShell command in the quickstart or the az deployment group create CLI command.

# Query Azure Database for MySQL Using the Azure CLI

This guide explains how to perform queries against Azure Database for MySQL Flexible Server using the Azure CLI and the az mysql flexible-server utilities.

## Setup

While the Azure sample demonstrates how to provision a Flexible Server instance using the CLI, you can follow one of the provisioning methods in the [Provision MySQL Flexible Server](./03_05_Provision_MySQL_Flexible_Server.md) document.

## Instructions

This guide is based on a [Microsoft document.](https://docs.microsoft.com/azure/mysql/flexible-server/connect-azure-cli#create-a-database)

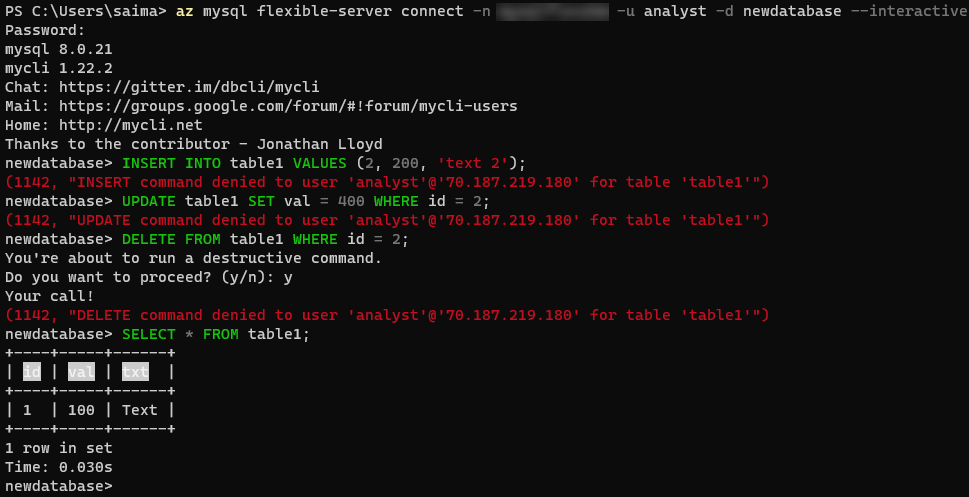
The Azure CLI supports running queries interactively, via the az mysql flexible-server connect command, which is similar to running queries interactively against a MySQL instance through the MySQL CLI. It is also possible to run an individual SQL query or a SQL file using the az mysql flexible-server execute command.

Note that these commands require the rdbms-connect CLI extension, which is automatically installed if it is not present. If you encounter permissions errors from the Azure Cloud Shell, execute the commands from a local installation of the Azure CLI.

In addition to the queries in the document, you can run basic admin queries. The statements below create a new user analyst that can read data from all tables in newdatabase.

USE newdatabase;  
CREATE USER 'analyst'@'%' IDENTIFIED BY '[SECURE PASSWORD]';  
GRANT SELECT ON newdatabase.\* TO 'analyst'@'%';  
FLUSH PRIVILEGES;

The new analyst user can also connect to newdatabase in the Flexible Server instance. The new user can only query tables in newdatabase.



This image demonstrates running queries against the Flexible Server instance using the Azure CLI.

For more details on creating databases and users in Single Server and Flexible Server, consult [this document.](https://docs.microsoft.com/azure/mysql/howto-create-users?tabs=flexible-server) Note that it uses the mysql CLI.

# Java (Spring Boot) Language Support

This guide will demonstrate how to operate a Spring Framework application that queries Azure Database for MySQL through the Spring Data JPA. We will also present Azure extensions for popular Java development tools.

## Setup

### Prerequisites

Please complete the instructions for [working with Flexible Server in MySQL Workbench.](03_06_Query_MySQL_Workbench.md) Utilize version 8.0.26 as you complete the guide to ensure compatibility with Single Server.

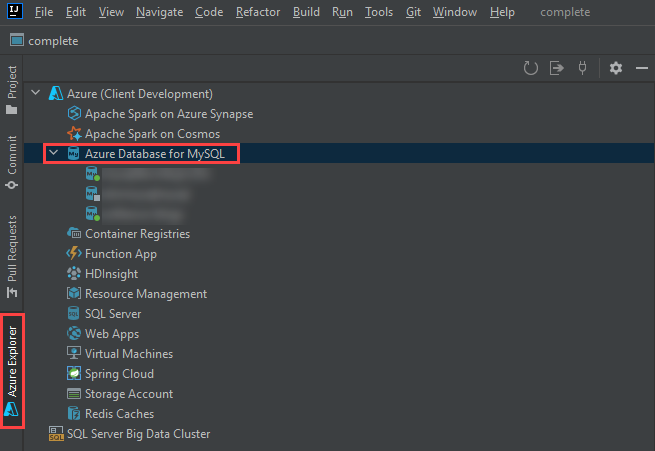
Moreover, download Postman, a popular REST client. If you are more comfortable with another utility, such as curl, feel free to use it instead.

### IntelliJ Setup

Download the [IntelliJ IDEA](https://www.jetbrains.com/idea/download) IDE. Community edition will suffice. It comes with a custom JDK, so it is not necessary to install the JDK separately.

After installing IntelliJ, install the [Azure Toolkit for IntelliJ](https://plugins.jetbrains.com/plugin/8053-azure-toolkit-for-intellij/) plugin. Then, authenticate with Azure, as described in [this](https://docs.microsoft.com/azure/developer/java/toolkit-for-intellij/sign-in-instructions) document.

Once everything is equipped, you will see an **Azure Explorer** tab on the left side of the screen. Note that it is possible to manage Azure Database for MySQL Single Server instances from the Azure Explorer.

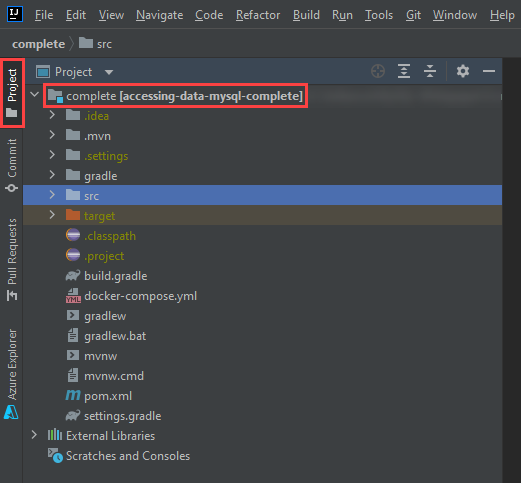


This image demonstrates the Azure Toolkit for IntelliJ plugin, with the Azure Database for MySQL node expanded.

### App Setup

Clone the [gs-accessing-data-mysql](https://github.com/spring-guides/gs-accessing-data-mysql) repository to your local machine. This is an example app from the Spring documentation.

Open the complete directory in the repository root in IntelliJ. If you are prompted to choose between using the Maven configuration or the Gradle configuration, choose the Maven one.

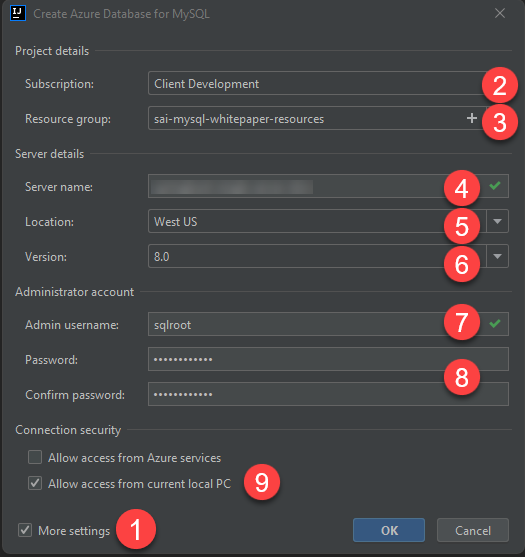


This image shows the complete project opened in IntelliJ in the Project tab.

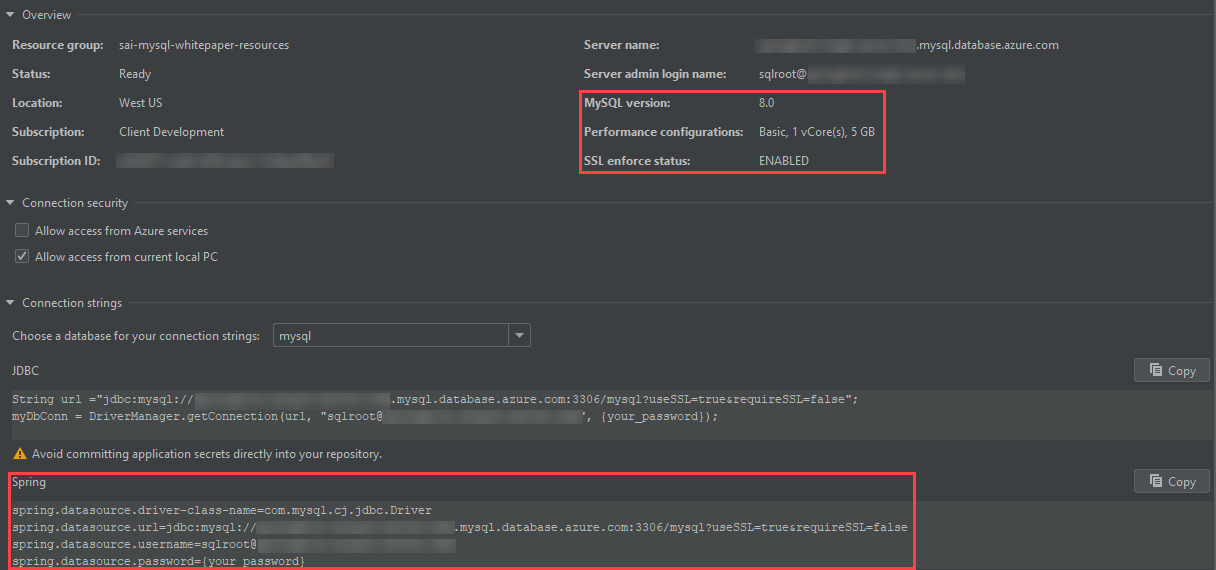
### Database Setup

The IntelliJ Azure explorer supports Azure Database for MySQL Single Server, but not Flexible Server. Luckily, you can provision a Single Server instance directly within the Azure Explorer.

1. Navigate to the **Azure Explorer** tab, right-click on **Azure Database for MySQL**, and select **+ Create**.
2. The **Create Azure Database for MySQL** dialog box will open. Select **+ More settings** (1) and populate the following parameters:
   * **Project details**
     + **Subscription** (2)
     + **Resource group** (3): choose an existing resource group from the dropdown or create a new one by pressing **+**
   * **Server details**
     + **Server name** (4): provide a unique value, like springboot-single-server-SUFFIX
     + **Location** (5): choose an Azure location near you
     + **Version** (6): choose 8.0
   * **Administrator account**
     + **Admin username** (7): enter sqlroot
     + **Password/confirm password** (8): choose a secure password
   * **Connection security**
     + Select **Allow access from current local PC** (9)

* 
* This image demonstrates how to create a new MySQL Single Server instance from IntelliJ and populate it with the parameters above.

1. Select **OK**. Allow the task to continue in the background.
2. Once provisioning completes (it should only take a few minutes), observe the new MySQL Single Server instance appear in the Azure explorer. Right-click the instance and select **Show properties**. A panel will open with basic information about the instance, including Spring connection information for the application.properties file.

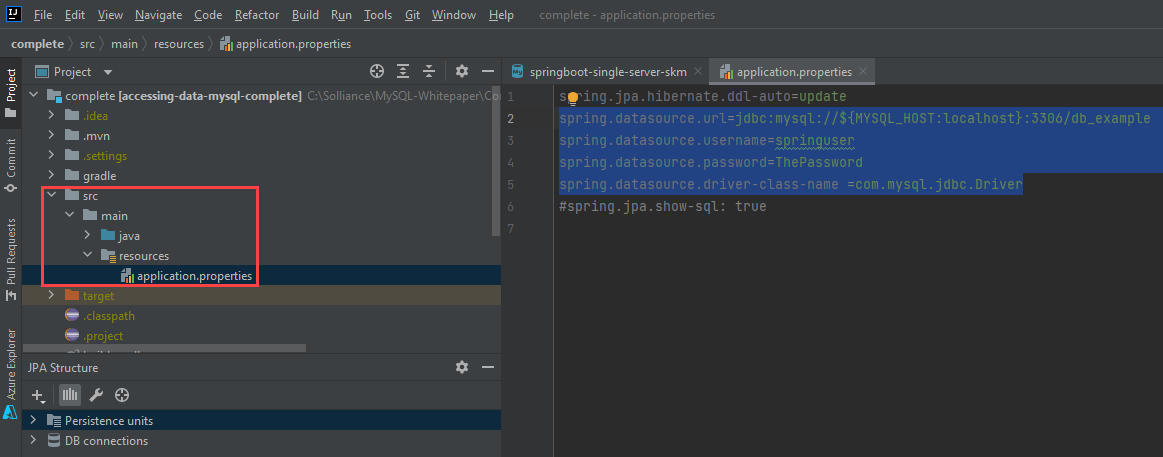
* 
* This image demonstrates Single Server MySQL connection information from the IntelliJ Azure explorer.

1. Create a new connection to your Azure Database for MySQL Single Server instance from MySQL Workbench. Use the following SQL statement to create a new database called newdatabase. This application will not function with the provided mysql system database.

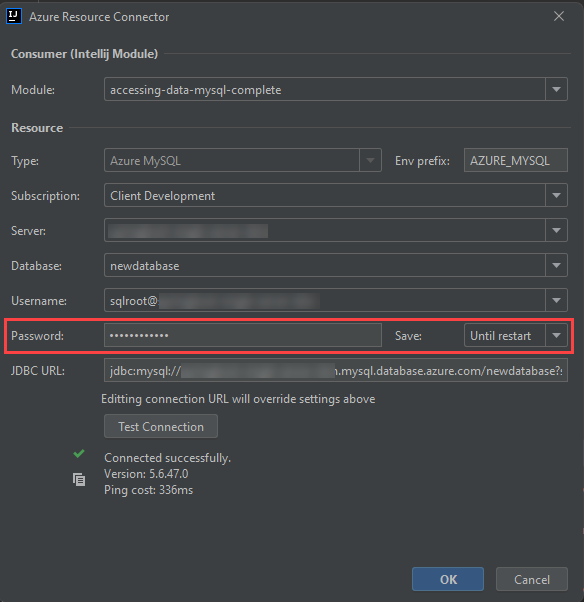
* CREATE DATABASE newdatabase;

## Run the App

1. Open application.properties from the project hierarchy: src > main > resources. Delete all the spring.datasource.\* entries.

* 
* This image demonstrates how to edit the application.properties file.

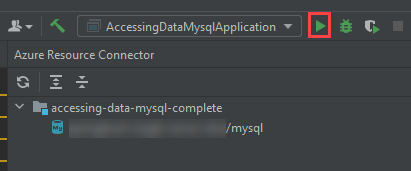
1. Navigate to the **Azure Explorer**, right-click the Single Server instance you provisioned, and select **Connect to Project (Preview)**.
2. In the **Azure Resource Connector** window, keep all parameters the same. Simply populate the **Password**. Then, select **OK**.

* 
* This image demonstrates the Azure Resource Connector dialog box.

1. Replace the contents you removed from the application.properties file with the following. Notice how the connection information is encapsulated in environment variables.

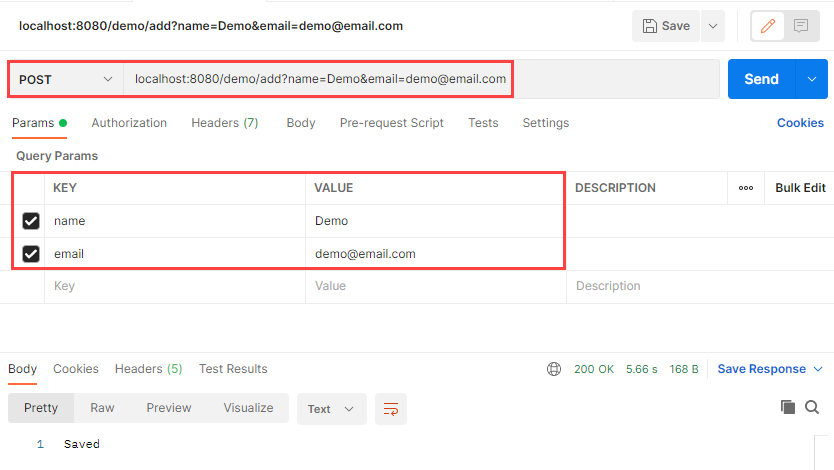
* spring.datasource.url=${AZURE\_MYSQL\_URL}  
  spring.datasource.username=${AZURE\_MYSQL\_USERNAME}  
  spring.datasource.password=${AZURE\_MYSQL\_PASSWORD}

1. Start the application from the upper right-hand corner of the screen.

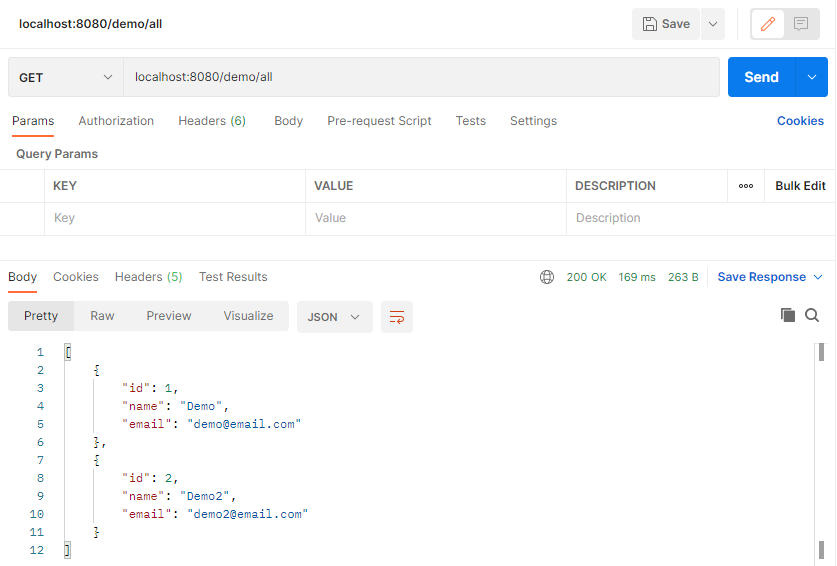
* 
* This image shows how to start the Spring Boot app from IntelliJ.

## Testing the App

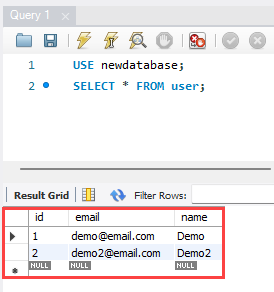
1. Open Postman, or the REST client of your choice. Make a POST request to http://localhost:8080/demo/add with the URL parameters name and email.

* 
* This image shows how to make a POST request to the Java app endpoint.

1. Make a GET request to http://localhost:8080/demo/all. The entries that you added through the POST request will be displayed.

* 
* This image shows how to make a GET request to the Java app endpoint.

1. As expected, the data is persisted to the MySQL Single Server instance.

* 
* This image shows the user data persisted to the MySQL Single Server instance with a query in MySQL Workbench.

## Stop the App

1. Stop the app in IntelliJ.
2. In the **Azure Explorer**, right-click the MySQL Single Server instance you created and select **Stop**.

Congratulations. You have successfully installed IntelliJ, the Azure Explorer extension, created a MySQL Single Server instance, and securely operated an app using the Single Server.

# PHP Language Support

This document demonstrates how to manipulate data in an Azure Database for MySQL Flexible Server instance and query it using PHP and the *MySQLi* library, which is provided with PHP.

## Setup

Follow one of the methods in the [Provision MySQL Flexible Server](03_05_Provision_MySQL_Flexible_Server.md) document to create a Flexible Server instance with a database.

Moreover, install PHP on your system from the [downloads page.](https://windows.php.net/download/) These instructions were tested with Non Thread Safe PHP 8.0.13.

Your php.ini file needs to uncomment the extension=mysqli and extension=openssl lines for these steps to work.

A text editor such as Visual Studio Code may also be useful.

Lastly, download the [connection certificate](https://dl.cacerts.digicert.com/DigiCertGlobalRootCA.crt.pem) that is used for SSL connections with the MySQL Flexible Server instance. In these snippets, the certificate was saved to C:\Tools on Windows. Adjust this if necessary.

## Instructions

Microsoft’s [quickstart guide](https://docs.microsoft.com/azure/mysql/flexible-server/connect-php) performs standard CRUD operations against the MySQL instance from a console app. This document modifies the code segments from the guide to provide an encrypted connection to the Flexible Server instance.

The first code snippet creates a table called Products with four columns, including a primary key. Adjust the host, username (most likely sqlroot), password, and db\_name (most likely newdatabase) parameters to the values you used during provisioning. Moreover, adjust the certificate path in the mysqli\_ssl\_set() method.

<?php  
$host = '[SERVER NAME].mysql.database.azure.com';  
$username = 'sqlroot';  
$password = '[PASSWORD]';  
$db\_name = 'newdatabase';  
  
//Establishes the connection  
$conn = mysqli\_init();  
mysqli\_ssl\_set($conn, NULL, NULL, "C:\Tools\DigiCertGlobalRootCA.crt.pem", NULL, NULL);  
mysqli\_real\_connect($conn, $host, $username, $password, $db\_name, 3306, MYSQLI\_CLIENT\_SSL);  
if (mysqli\_connect\_errno()) {  
die('Failed to connect to MySQL: '.mysqli\_connect\_error());  
}  
  
// Run the create table query  
if (mysqli\_query($conn, '  
CREATE TABLE Products (  
`Id` INT NOT NULL AUTO\_INCREMENT ,  
`ProductName` VARCHAR(200) NOT NULL ,  
`Color` VARCHAR(50) NOT NULL ,  
`Price` DOUBLE NOT NULL ,  
PRIMARY KEY (`Id`)  
);  
')) {  
printf("Table created\n");  
}  
  
//Close the connection  
mysqli\_close($conn);  
?>

You should see a console output with the message Table created.

The second code snippet uses the same logic to start an SSL-secured connection and to close the connection. This time, it leverages a prepared insert statement with bound parameters.

<?php  
$host = '[SERVER NAME].mysql.database.azure.com';  
$username = 'sqlroot';  
$password = '[PASSWORD]';  
$db\_name = 'newdatabase';  
  
//Establishes the connection  
$conn = mysqli\_init();  
mysqli\_ssl\_set($conn, NULL, NULL, "C:\Tools\DigiCertGlobalRootCA.crt.pem", NULL, NULL);  
mysqli\_real\_connect($conn, $host, $username, $password, $db\_name, 3306, MYSQLI\_CLIENT\_SSL);  
if (mysqli\_connect\_errno()) {  
die('Failed to connect to MySQL: '.mysqli\_connect\_error());  
}  
  
//Create an Insert prepared statement and run it  
$product\_name = 'BrandNewProduct';  
$product\_color = 'Blue';  
$product\_price = 15.5;  
if ($stmt = mysqli\_prepare($conn, "INSERT INTO Products (ProductName, Color, Price) VALUES (?, ?, ?)")) {  
mysqli\_stmt\_bind\_param($stmt, 'ssd', $product\_name, $product\_color, $product\_price);  
mysqli\_stmt\_execute($stmt);  
printf("Insert: Affected %d rows\n", mysqli\_stmt\_affected\_rows($stmt));  
mysqli\_stmt\_close($stmt);  
}  
  
//Close the connection  
mysqli\_close($conn);  
?>

You should see the console output message Insert: Affected 1 rows.

The third code snippet utilizes the mysqli\_query() method, just like the first code snippet. However, it also utilizes the mysqli\_fetch\_assoc() method to parse the result set.

<?php  
$host = '[SERVER NAME].mysql.database.azure.com';  
$username = 'sqlroot';  
$password = '[PASSWORD]';  
$db\_name = 'newdatabase';  
  
//Establishes the connection  
$conn = mysqli\_init();  
mysqli\_ssl\_set($conn, NULL, NULL, "C:\Tools\DigiCertGlobalRootCA.crt.pem", NULL, NULL);  
mysqli\_real\_connect($conn, $host, $username, $password, $db\_name, 3306, MYSQLI\_CLIENT\_SSL);  
if (mysqli\_connect\_errno()) {  
die('Failed to connect to MySQL: '.mysqli\_connect\_error());  
}  
  
//Run the Select query  
printf("Reading data from table: \n");  
$res = mysqli\_query($conn, 'SELECT \* FROM Products');  
while ($row = mysqli\_fetch\_assoc($res)) {  
var\_dump($row);  
}  
  
//Close the connection  
mysqli\_close($conn);  
?>

PHP returns an array with the column values for the row inserted in the previous snippet.

The next snippet uses a prepared update statement with bound parameters. It modifies the Price column of the record.

<?php  
$host = '[SERVER NAME].mysql.database.azure.com';  
$username = 'sqlroot';  
$password = '[PASSWORD]';  
$db\_name = 'newdatabase';  
  
//Establishes the connection  
$conn = mysqli\_init();  
mysqli\_ssl\_set($conn, NULL, NULL, "C:\Tools\DigiCertGlobalRootCA.crt.pem", NULL, NULL);  
mysqli\_real\_connect($conn, $host, $username, $password, $db\_name, 3306, MYSQLI\_CLIENT\_SSL);  
if (mysqli\_connect\_errno()) {  
die('Failed to connect to MySQL: '.mysqli\_connect\_error());  
}  
  
//Run the Update statement  
$product\_name = 'BrandNewProduct';  
$new\_product\_price = 15.1;  
if ($stmt = mysqli\_prepare($conn, "UPDATE Products SET Price = ? WHERE ProductName = ?")) {  
mysqli\_stmt\_bind\_param($stmt, 'ds', $new\_product\_price, $product\_name);  
mysqli\_stmt\_execute($stmt);  
printf("Update: Affected %d rows\n", mysqli\_stmt\_affected\_rows($stmt));  
  
//Close the connection  
mysqli\_stmt\_close($stmt);  
}  
  
//Close the connection  
mysqli\_close($conn);  
?>

After executing these commands, you should receive the message Update: Affected 1 rows.

The final code snippet deletes a row from the table using the ProductName column value. It again uses a prepared statement with bound parameters.

<?php  
$host = '[SERVER NAME].mysql.database.azure.com';  
$username = 'sqlroot';  
$password = '[PASSWORD]';  
$db\_name = 'newdatabase';  
  
//Establishes the connection  
$conn = mysqli\_init();  
mysqli\_ssl\_set($conn, NULL, NULL, "C:\Tools\DigiCertGlobalRootCA.crt.pem", NULL, NULL);  
mysqli\_real\_connect($conn, $host, $username, $password, $db\_name, 3306, MYSQLI\_CLIENT\_SSL);  
if (mysqli\_connect\_errno()) {  
die('Failed to connect to MySQL: '.mysqli\_connect\_error());  
}  
  
//Run the Delete statement  
$product\_name = 'BrandNewProduct';  
if ($stmt = mysqli\_prepare($conn, "DELETE FROM Products WHERE ProductName = ?")) {  
mysqli\_stmt\_bind\_param($stmt, 's', $product\_name);  
mysqli\_stmt\_execute($stmt);  
printf("Delete: Affected %d rows\n", mysqli\_stmt\_affected\_rows($stmt));  
mysqli\_stmt\_close($stmt);  
}  
  
//Close the connection  
mysqli\_close($conn);  
?>

Congratulations. You successfully created an SSL-secured connection with Flexible Server, created a table (DDL), and performed CRUD operations against that table (DML).

# Python Language Support

This guide will demonstrate how to query Azure Database for MySQL Flexible Server using the mysql-connector-python library on Python 3.

## Setup

Follow one of the methods in the [Provision MySQL Flexible Server](03_05_Provision_MySQL_Flexible_Server.md) document to create a Flexible Server instance with a database.

Moreover, install Python 3.7 or above from the [Downloads page](https://www.python.org/downloads/). This sample was tested using Python 3.8.

A text editor like Visual Studio Code will greatly help.

Though a Python Virtual Environment is not necessary for the sample to run, using one will avoid conflicts with packages installed globally on your system. The commands below will create a Virtual Environment called venv and activate it on Windows. Instructions will differ for other OS.

python -m venv venv  
.\venv\Scripts\activate

## Instructions

This document is based on [Microsoft’s sample](https://docs.microsoft.com/azure/mysql/flexible-server/connect-python).

The first code snippet creates a table, inventory, with three columns. It uses raw queries to create the inventory table and insert three rows. If the snippet succeeds, you should see an output like the one below.

Connection established  
Finished dropping table (if existed).  
Finished creating table.  
Inserted 1 row(s) of data.  
Inserted 1 row(s) of data.  
Inserted 1 row(s) of data.  
Done.

Note that the sample establishes an SSL connection with the MySQL instance. You can use the statement below (placed before cursor and conn are closed) to validate the use of SSL.

cursor.execute("SHOW STATUS LIKE 'Ssl\_cipher'")  
print(cursor.fetchone())

If you want to bind the [SSL public certificate](https://dl.cacerts.digicert.com/DigiCertGlobalRootCA.crt.pem) with your connections to Flexible Server, which is recommended by Azure, download the public certificate to a location on your machine (such as C:\Tools). Then, edit the config dictionary to add the ssl\_ca key and the file path of your certificate as the value.

config = {  
 'host':'[SERVER].mysql.database.azure.com',  
 'user':'sqlroot',  
 'password':'[PASSWORD]',  
 'database':'newdatabase',  
 'ssl\_ca': 'C:\Tools\DigiCertGlobalRootCA.crt.pem'  
}

The second code snippet connects to the MySQL instance and executes a raw query to SELECT all rows from the inventory table. This time, it uses the fetchall() method to parse the result set into a Python iterable. You should see an output like the one below.

Connection established  
Read 3 row(s) of data.  
Data row = (1, banana, 150)  
Data row = (2, orange, 154)  
Data row = (3, apple, 100)  
Done.

The third code snippet executes an UPDATE statement to change the quantity value of the record identified by name. You should see an output like the one below.

Connection established  
Updated 1 row(s) of data.  
Done.

The final snippet executes a raw DELETE statement against the inventory table targeting records identified by name. You should see an output like the one below.

Connection established  
Deleted 1 row(s) of data.  
Done.

At this point, you have successfully opened a connection to Flexible Server, created a table (DDL), and performed CRUD operations (DML) against data in the table.

If you created a Python Virtual Environment for this document, simply enter deactive into the console.

# Java Language Support

This document describes tools to interact with Azure Database for MySQL (Single Server and Flexible Server) through Java.

## Example Code

Refer to the [Connect and Query sample for Java](./03_Connect_Query_Java_IntelliJ.md), which uses IntelliJ, Spring Boot, and Spring Data JPA.

## Application Connectors

*MySQL Connector/J* is a JDBC-compatible API which natively implements the MySQL protocol in Java, rather than utilizing client libraries. The Connect and Query sample does not directly utilize *MySQL Connector/J*, but Microsoft provides a sample that uses this technology.

To allow developers to focus on implementing business logic, applications commonly use persistence frameworks like Spring Data JPA. Spring Data JPA extends the JPA specification, which governs *object-relational mapping* (ORM) technologies in Java. It functions on top of JPA implementations, like the Hibernate ORM. The Connect and Query sample leverages Spring Data JPA and *MySQL Connector/J* to access the Azure MySQL instance and expose data through a web API.

Flexible Server is compatible with all Java client utilities for MySQL Community Edition. However, Microsoft has only validated *MySQL Connector/J* for use with Single Server due to its network connectivity setup. Refer to [this](https://docs.microsoft.com/azure/mysql/concepts-compatibility) document for more information about drivers compatible with Single Server.

### Resources

1. [MySQL Connector/J Introduction](https://dev.mysql.com/doc/connector-j/8.0/en/connector-j-overview.html)
2. MySQL Connector/J Microsoft Samples
   * [Single Server](https://docs.microsoft.com/azure/mysql/connect-java)
   * [Flexible Server](https://docs.microsoft.com/azure/mysql/flexible-server/connect-java)
3. [Introduction to Spring Data JPA](https://www.baeldung.com/the-persistence-layer-with-spring-data-jpa)
4. [Hibernate ORM](https://hibernate.org/orm/)

## Tooling

The Connect and Query sample leverages IntelliJ, which is one of the most widely used Java IDEs. This section provides resources for other common tools.

### Eclipse

Eclipse is another popular IDE for Java development. It supports extensions for enterprise Java development, including powerful utilities for Spring applications. Moreover, through the Azure Toolkit for Eclipse, developers can quickly deploy their applications to Azure directly from Eclipse.

**Tool-Specific Resources**

1. [Installing the Azure Toolkit for Eclipse](https://docs.microsoft.com/azure/developer/java/toolkit-for-eclipse/installation)
2. [Create a Hello World web app for Azure App Service using Eclipse](https://docs.microsoft.com/azure/developer/java/toolkit-for-eclipse/create-hello-world-web-app)

### Maven

Maven improves the productivity of Java developers by managing builds, dependencies, releases, documentation, and more. Maven projects are created from archetypes. Microsoft provides the Maven Plugins for Azure to help Java developers work with Azure Functions, Azure App Service, and Azure Spring Cloud from their Maven workflows.

**Note**: Application patterns with Azure Functions, Azure App Service, and Azure Spring Cloud are addressed in the [End-to-End development story.](../04_EndToEndDev/04_End_To_End_Development.md)

**Tool-Specific Resources**

1. [Maven Introduction](https://maven.apache.org/guides/getting-started/index.html)
2. [Develop Java web app on Azure using Maven (App Service)](https://docs.microsoft.com/learn/modules/publish-web-app-with-maven-plugin-for-azure-app-service/)
3. [Deploy Spring microservices to Azure (Spring Cloud)](https://docs.microsoft.com/learn/modules/azure-spring-cloud-workshop/)
4. [Develop Java serverless Functions on Azure using Maven](https://docs.microsoft.com/learn/modules/develop-azure-functions-app-with-maven-plugin/)

# Security and Compliance in Azure Database for MySQL

Azure Database for MySQL provides extensive platform management and simple integration with new or existing applications. However, a critical factor for many sensitive industries is being compliant with regulations. Azure has addressed these customer concerns.

## Data Encryption

Both Azure Database for MySQL offerings, Single Server and Flexible Server, offer data encryption at-rest. Data, backups, and temporary files created during query execution are all encrypted.

While Azure can manage encryption keys, Single Server supports bring your own key (BYOK), providing organizations full key lifecycle control. This feature is only supported in the General Purpose and Memory Optimized tiers.

**Configuring Data Encryption At-Rest Guides**

* [Single Server BYOK](https://docs.microsoft.com/azure/mysql/concepts-data-encryption-mysql)

Moreover, data in-transit is protected using SSL/TLS, which is enforced by default. However, it is possible to allow insecure connections for legacy applications or enforce a minimum TLS version for connections. Consult the guides below, as Flexible Server’s TLS enforcement status can be set through the require\_secure\_transport MySQL server parameter.

**Configuring Data Encryption In-Motion Guides**

* [Single Server](https://docs.microsoft.com/azure/mysql/concepts-ssl-connection-security)
* [Flexible Server](https://docs.microsoft.com/azure/mysql/flexible-server/how-to-connect-tls-ssl)

## Security Best Practices Overview

Organizations must take proactive security measures to protect their workloads. Azure simplifies following best practices.

### Implement Network Security

Both MySQL PaaS offerings support public connectivity, which permits certain hosts to access the instance over the public internet, and private connectivity, which limits access to an Azure virtual network deployment. The difference between public and private access is addressed in the [network security document.](./03_Network_Security.md)

### Access Management

When provisioning PaaS MySQL, Azure requires administrator user credentials. It is best practice to create non-administrator users for each database in the MySQL instance. Follow [this](https://docs.microsoft.com/azure/mysql/howto-create-users) guide for more information on how to create new databases and manage access.

### Monitoring and Threat Protection

[Microsoft Defender for open-source relational databases](https://docs.microsoft.com/azure/defender-for-cloud/defender-for-databases-introduction) tracks unusual database activity, like brute force login attempts. It notifies administrators of anomalies and helps them patch vulnerabilities. Currently, it is supported in the General Purpose and Memory Optimized tiers of Single Server. Enable it by following [this](https://docs.microsoft.com/azure/defender-for-cloud/defender-for-databases-usage) guide.

Single Server and Flexible Server also support audit logging. Note that excessive audit logging degrades server performance, so be mindful of the events and users configured for logging.

**Configuring Audit Logging Guides**

* [Single Server](https://docs.microsoft.com/azure/mysql/concepts-audit-logs)
* [Flexible Server](https://docs.microsoft.com/azure/mysql/flexible-server/concepts-audit-logs)

# Network Security

As mentioned previously, network configuration affects security, application performance (latency), and compliance. This guide explains the fundamentals of PaaS MySQL networking.

## Public vs. Private Access

### Public Access

Public access allows hosts, including Azure services, to access the PaaS MySQL instance via the public internet. Firewall ACLs limit access to hosts that fall within the allowed IP address ranges. They are set at the server-level, meaning that they govern network access to all databases on the instance. While it is best practice to create rules that allow specific IP addresses or ranges to access the instance, developers can enable network access from all Azure public IP addresses. This is useful for Azure services without fixed public IP addresses, such as [Azure Functions](https://docs.microsoft.com/azure/azure-functions/functions-overview) that use public access.

Restricting access to Azure public IP addresses still provides network access to the instance to public IPs owned by other Azure customers.

**Configuring Public Access Guides**

* Single Server
  + [Azure portal](https://docs.microsoft.com/azure/mysql/howto-manage-firewall-using-portal)
  + [Azure CLI](https://docs.microsoft.com/azure/mysql/howto-manage-firewall-using-cli)
  + [ARM Reference for Firewall Rules](https://docs.microsoft.com/azure/templates/microsoft.dbformysql/servers/firewallrules?tabs=json)
* Flexible Server
  + [Azure portal](https://docs.microsoft.com/azure/mysql/flexible-server/how-to-manage-firewall-portal)
  + [Azure CLI](https://docs.microsoft.com/azure/mysql/flexible-server/how-to-manage-firewall-cli)
  + [ARM Reference for Firewall Rules](https://docs.microsoft.com/azure/templates/microsoft.dbformysql/flexibleservers/firewallrules?tabs=json)

### Private Access

#### Virtual Network Hierarchy

An Azure virtual network is similar to a network deployed on-premises: it provides network isolation for workloads. Each virtual network has a private IP allocation block. Choosing an allocation block is an important consideration, especially if your environment requires multiple virtual networks to be joined: the allocation blocks of the virtual networks cannot overlap. It is best practice to choose allocation blocks from [RFC 1918.](https://datatracker.ietf.org/doc/html/rfc1918)

**Note**: When deploying a resource such as a VM into a virtual network, the virtual network must be located in the same region and Azure subscription as the Azure resource. Review the [Introduction to Azure](../02_IntroToMySQL/02_02_Introduction_to_Azure.md) document for more information about regions and subscriptions.

Each virtual network is further segmented into subnets. Subnets improve virtual network organization and security, just as they do on-premises.

Virtual networks are joined through *peering*. The peered virtual networks can reside in the same or different Azure regions.

Lastly, note that it is possible to access resources in a virtual network from on-premises. Some organizations opt to use VPN connections through [Azure VPN Gateway](https://docs.microsoft.com/azure/vpn-gateway/vpn-gateway-about-vpngateways), which sends encrypted traffic over the Internet. Others opt for [Azure ExpressRoute](https://docs.microsoft.com/azure/expressroute/expressroute-introduction), which establishes a private connection to Azure through a service provider.

**More Information on Virtual Networks**

* [Introduction to Azure Virtual Networks](https://docs.microsoft.com/learn/modules/introduction-to-azure-virtual-networks/)
* Creating virtual networks
  + [Portal](https://docs.microsoft.com/azure/virtual-network/quick-create-portal)
  + [PowerShell](https://docs.microsoft.com/azure/virtual-network/quick-create-powershell)
  + [CLI](https://docs.microsoft.com/azure/virtual-network/quick-create-cli)
  + [ARM Template](https://docs.microsoft.com/azure/virtual-network/quick-create-template)

#### Flexible Server

Flexible Server supports deployment into a virtual network for secure access. Specifically, the target subnet must be *delegated*, meaning that it can only contain Flexible Server instances. Because Flexible Server is deployed in the virtual network, it has a private IP address. Virtual networks can be integrated with a private DNS zone to support name resolution for the Flexible Server instance.

**Note**: If the Flexible Server client, such as a VM, is located in a peered virtual network, then the private DNS zone created for the Flexible Server must also be integrated with the peered virtual network.

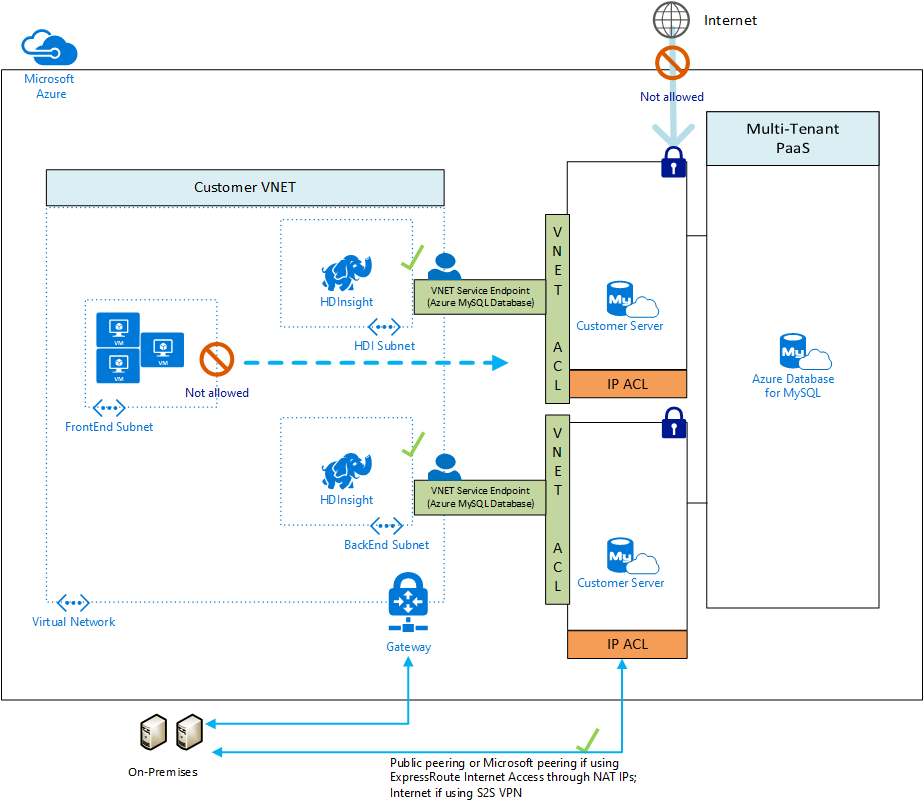
**Configuring Private Access for Flexible Server**

* [Azure portal](https://docs.microsoft.com/azure/mysql/flexible-server/how-to-manage-virtual-network-portal)
* [Azure CLI](https://docs.microsoft.com/azure/mysql/flexible-server/how-to-manage-virtual-network-cli)

#### Single Server

Private Access from Single Server can be accomplished through (1) *Service Endpoints* or (2) *Private Link*; Single Server does not natively support virtual networks like Flexible Server. Both of these methods require the General Purpose or Memory Optimized tier.

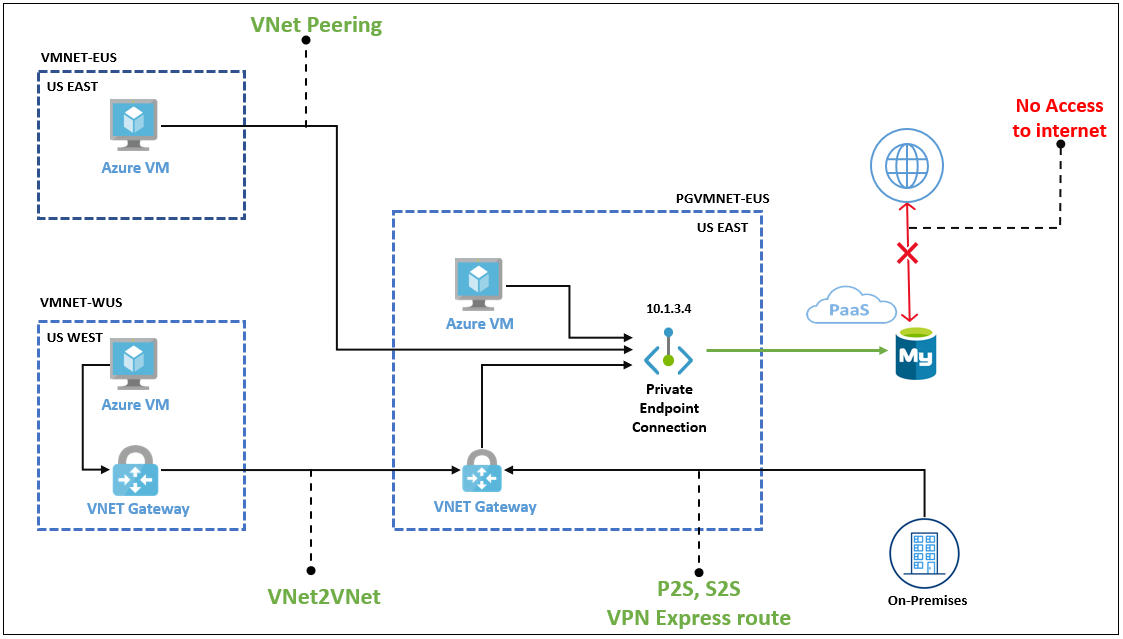
Service Endpoints only allow traffic from a given virtual network to access MySQL Single Server. Service endpoints are intended for Azure resources without public IPs, like VMs deployed in a virtual network, to access PaaS services securely. However, traffic leaves the virtual network, as shown in the image below, and access still occurs through the service public endpoint. In this image, HDISubnet and BackEndSubnet have been configured for access by ACLs in the Single Server instances, but FrontEndSubnet has not.



This image demonstrates how VNet service endpoints allow access to Single Server, but data leaves the virtual network.

Private Link uses *Private Endpoints* to replace public resource endpoints with private network interfaces accessible through private IP addresses. Unlike Service Endpoints, all network traffic is contained within the virtual network.

In the image below, since public access is disabled, access can only occur through the private endpoint in the PGVMNET-EUS virtual network. Other Azure virtual networks, including those in other regions, like VMNET-WUS, can be peered to the virtual network with the private endpoint. On-premises network can also be joined to Azure virtual networks, as explained previously.



This image explains how private endpoints work to bring PaaS services into virtual networks.

**Configuring Private Access for Single Server**

* Service Endpoints
  + [Portal](https://docs.microsoft.com/azure/mysql/howto-manage-vnet-using-portal)
  + [CLI](https://docs.microsoft.com/azure/mysql/howto-manage-vnet-using-cli)
* Private Link
  + [Portal](https://docs.microsoft.com/azure/mysql/howto-configure-privatelink-portal)
  + [CLI](https://docs.microsoft.com/azure/mysql/howto-configure-privatelink-cli)

# PHP Language Support

This document describes tools to interact with Azure Database for MySQL (Single Server and Flexible Server) through PHP.

## Example Code

Refer to the [Connect and Query sample for PHP.](./03_Connect_Query_PHP.md)

## Application Connectors

There are two major APIs to interact with MySQL in PHP: *MySQLi*, which is used in the Connect and Query sample, and *PDO*, which is used in the Laravel sample food ordering site. *MySQLi* and *PDO* are wrappers over the *mysqlnd* or *libmysqlclient* C libraries: it is highly recommended to use *mysqlnd* as the default backend library due to its more advanced features. *mysqlnd* is the default backend provided with PHP.

*MySQLi* is an improvement over the earlier *MySQL* API, which does not meet the security needs of modern applications.

*PDO*, or *PHP Data Objects*, allows applications to access databases in PHP through abstractions, standardizing data access for different databases. PDO works with a database-specific driver, like *PDO\_MYSQL*.

Flexible Server and Single Server are compatible with all PHP client utilities for MySQL Community Edition.

## Resources

1. [Backend libraries for mysqli and PDO\_MySQL](https://www.php.net/manual/en/mysqlinfo.library.choosing.php)
2. [Introduction to PDO](https://www.php.net/manual/en/intro.pdo.php)
3. [PDO\_MYSQL Reference](https://www.php.net/manual/en/ref.pdo-mysql.php) # Python Language Support

This document describes tools to interact with Azure Database for MySQL (Single Server and Flexible Server) through Python.

## Example Code

Refer to the [Connect and Query sample for Python.](./03_Connect_Query_Python.md)

## Application Connectors

*MySQL Connector/Python* offers a Python Database API specification-compatible driver for MySQL database access (PEP 249). It does not depend on a MySQL client library. The Python Connect and Query sample utilizes *MySQL Connector/Python*.

An alternative connector is *PyMySQL*. It is also PEP 249-compliant.

Django is a popular web application framework for Python. The Django ORM officially supports MySQL through (1) the *mysqlclient* Python wrapper for the native MySQL driver or (2) the *MySQL Connector/Python* API. *mysqlclient* is recommended for use with the Django ORM.

Flexible Server is compatible with all Python client utilities for MySQL Community Edition. However, Microsoft has only validated *MySQL Connector/Python* and *PyMySQL* for use with Single Server due to its network connectivity setup. Refer to [this](https://docs.microsoft.com/azure/mysql/concepts-compatibility) document for more information about drivers compatible with Single Server.

## Resources

1. [Introduction to MySQL Connector/Python](https://dev.mysql.com/doc/connector-python/en/connector-python-introduction.html)
2. [PyMySQL Samples](https://pymysql.readthedocs.io/en/latest/user/examples.html)
3. [MySQLdb (mysqlclient) User’s Guide](https://mysqlclient.readthedocs.io/user_guide.html#mysqldb)
4. [Django ORM Support for MySQL](https://docs.djangoproject.com/en/3.2/ref/databases/#mysql-notes)

# Query Azure Database for MySQL using MySQL Workbench

This guide explains how to perform queries against Azure Database for MySQL Flexible Server using MySQL Workbench, a UI-based management tool.

## Setup

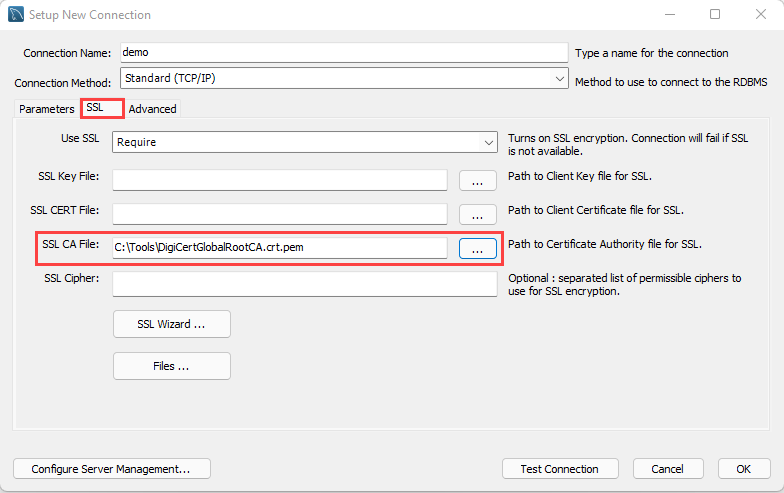
Follow one of the methods in the [Provision MySQL Flexible Server](03_05_Provision_MySQL_Flexible_Server.md) document to create a Flexible Server instance with a database.

Download MySQL Workbench from the [MySQL Downloads.](https://dev.mysql.com/downloads/workbench/) This document was written using version 8.0.26: we recommend this version because Single Server is not compatible with 8.0.27, so 8.0.26 has the greatest flexibility.

## Instructions

This guide is based on a [Microsoft document.](https://docs.microsoft.com/azure/mysql/flexible-server/connect-workbench) Follow the guide to create a new database in the Flexible Server instance, create a new table (inventory), query the table, update data in the table, and delete records from the table.

Note that MySQL Workbench can automatically initiate an SSL-secured connection to Azure Database for MySQL. However, it is recommended to use the [SSL public certificate](https://dl.cacerts.digicert.com/DigiCertGlobalRootCA.crt.pem) in your connections. To bind the SSL public certificate to MySQL Workbench, choose the downloaded certificate file as the **SSL CA File** on the **SSL** tab.



Add the SSL CA file on the SSL tab of the Setup New Connection dialog box.

# Monitoring and Alerts

Once the migration has been successfully completed, the next phase it to manage the new cloud-based data workload resources. Management operations include both control plane and data plane activities. Control plane activities are those related to the Azure resources versus data plane which is **inside** the Azure resource (in this case MySQL).

Azure Database for MySQL provides for the ability to monitor both of these types of operatonal activities using Azure-based tools such as [Azure Monitor](https://docs.microsoft.com/en-us/azure/azure-monitor/overview), [Log Analytics](https://docs.microsoft.com/en-us/azure/azure-monitor/platform/design-logs-deployment) and [Azure Sentinel](https://docs.microsoft.com/en-us/azure/sentinel/overview). In addition to the Azure-based tools, security information and event management (SIEM) systems can be configured to consume these logs as well.

Whichever tool is used to monitor the new cloud-based workloads, alerts will need to be created to warn Azure and database administrators of any suspicious activity. If a particular alert event has a well-defined remediation path, alerts can fire automated [Azure run books](https://docs.microsoft.com/en-us/azure/automation/automation-quickstart-create-runbook) to address the event.

The first step to creating a fully monitored environment is to enable MySQL log data to flow into Azure Monitor. Reference [Configure and access audit logs for Azure Database for MySQL in the Azure portal](https://docs.microsoft.com/en-us/azure/mysql/howto-configure-audit-logs-portal) for more information.

Once log data is flowing, use the [Kusto Query Language (KQL)](https://docs.microsoft.com/en-us/azure/data-explorer/kusto/query/) query language to query the various log information. Administrators unfamiliar with KQL can find a SQL to KQL cheat sheet [here](https://docs.microsoft.com/en-us/azure/data-explorer/kusto/query/sqlcheatsheet) or the [Get started with log queries in Azure Monitor](https://docs.microsoft.com/en-us/azure/azure-monitor/log-query/get-started-queries) page.

For example, to get the memory usage of the Azure Database for MySQL:

AzureMetrics  
| where TimeGenerated > ago(15m)  
| limit 10  
| where ResourceProvider == "MICROSOFT.DBFORMYSQL"  
| where MetricName == "memory\_percent"  
| project TimeGenerated, Total, Maximum, Minimum, TimeGrain, UnitName  
| top 1 by TimeGenerated

To get the CPU usage:

AzureMetrics  
| where TimeGenerated > ago(15m)  
| limit 10  
| where ResourceProvider == "MICROSOFT.DBFORMYSQL"  
| where MetricName == "cpu\_percent"  
| project TimeGenerated, Total, Maximum, Minimum, TimeGrain, UnitName  
| top 1 by TimeGenerated

Once you have created the KQL query, you will then create [log alerts](https://docs.microsoft.com/en-us/azure/azure-monitor/platform/alerts-unified-log) based of these queries.

## Server Parameters

As part of the migration, it is likely the on-premises [server parameters](https://docs.microsoft.com/en-us/azure/mysql/concepts-server-parameters) were modified to support a fast egress. Also, modifications were made to the Azure Database for MySQL parameters to support a fast ingress. The Azure server parameters should be set back to their original on-premises workload optimized values after the migration.

However, be sure to review and make server parameters changes that are appropriate for the workload and the environment. Some values that were great for an on-premises environment, may not be optimal for a cloud-based environment. Additionally, when planning to migrate the current on-premises parameters to Azure, verify that they can in fact be set.

Some parameters are not allowed to be modified in Azure Database for MySQL.

## PowerShell Module

The Azure Portal and Windows PowerShell can be used for managing the Azure Database for MySQL. To get started with PowerShell, install the Azure PowerShell cmdlets for MySQL with the following PowerShell command:

Install-Module -Name Az.MySql

After the modules are installed, reference tutorials like the following to learn ways you can take advantage of scripting your management activities:

* [Tutorial: Design an Azure Database for MySQL using PowerShell](https://docs.microsoft.com/en-us/azure/mysql/tutorial-design-database-using-powershell)
* [How to back up and restore an Azure Database for MySQL server using PowerShell](https://docs.microsoft.com/en-us/azure/mysql/howto-restore-server-powershell)
* [Configure server parameters in Azure Database for MySQL using PowerShell](https://docs.microsoft.com/en-us/azure/mysql/howto-configure-server-parameters-using-powershell)
* [Auto grow storage in Azure Database for MySQL server using PowerShell](https://docs.microsoft.com/en-us/azure/mysql/howto-auto-grow-storage-powershell)
* [How to create and manage read replicas in Azure Database for MySQL using PowerShell](https://docs.microsoft.com/en-us/azure/mysql/howto-read-replicas-powershell)
* [Restart Azure Database for MySQL server using PowerShell](https://docs.microsoft.com/en-us/azure/mysql/howto-restart-server-powershell)

## Azure Database for MySQL Upgrade Process

Since Azure Database for MySQL is a PaaS offering, administrators are not responsible for the management of the updates on the operating system or the MySQL software. However, it is important to be aware the upgrade process can be random and when being deployed, will stop the MySQL server workloads. Plan for these downtimes by rerouting the workloads to a read replica in the event the particular instance goes into maintenance mode.

**Note:** This style of failover architecture may require changes to the applications data layer to support this type of failover scenario. If the read replica is maintained as a read replica and is not promoted, the application will only be able to read data and it may fail when any operation attempts to write information to the database.

The [Planned maintenance notification](https://docs.microsoft.com/en-us/azure/mysql/concepts-monitoring#planned-maintenance-notification) feature will inform resource owners up to 72 hours in advance of installation of an update or critical security patch. Database administrators may need to notify application users of planned and unplanned maintenance.

**Note:** Azure Database for MySQL maintenance notifications are incredibly important. The database maintenance can take your database and connected applications down for a period of time. # Security

Moving to a cloud-based service doesn’t mean the entire internet will have access to it at all times. Azure provides best in class security that ensures data workloads are continually protected from bad actors and rouge programs.

## Authentication

Azure Database for MySQL supports the basic authentication mechanisms for MySQL user connectivity, but also supports [integration with Azure Active Directory](https://docs.microsoft.com/en-us/azure/mysql/concepts-azure-ad-authentication). This security integration works by issuing tokens that act like passwords during the MySQL login process. [Configuring Active Directory integration](https://docs.microsoft.com/en-us/azure/mysql/howto-configure-sign-in-azure-ad-authentication) is incredibly simple to do and supports not only users, but AAD groups as well.

This tight integration allows administrators and applications to take advantage of the enhanced security features of [Azure Identity Protection](https://docs.microsoft.com/en-us/azure/active-directory/identity-protection/overview-identity-protection) to further surface any identity issues.

**Note:** This security feature is supported by MySQL 5.7 and later. Most [application drivers](https://docs.microsoft.com/en-us/azure/mysql/howto-configure-sign-in-azure-ad-authentication) are supported as long as the clear-text option is provided.

## Threat Protection

In the event that user or application credentials are compromised, logs are not likely to reflect any failed login attempts. Compromised credentials can allow bad actors to access and download the data. [Azure Threat Protection](https://docs.microsoft.com/en-us/azure/mysql/concepts-data-access-and-security-threat-protection) can watch for anomalies in logins (such as unusual locations, rare users or brute force attacks) and other suspicious activities. Administrators can be notified in the event something does not look right.

## Audit Logging

MySQL has a robust built-in audit log feature. By default, this [audit log feature is disabled](https://docs.microsoft.com/en-us/azure/mysql/concepts-audit-logs) in Azure Database for MySQL. Server level logging can be enabled by changing the audit\_log\_enabled server parameter. Once enabled, logs can be accessed through [Azure Monitor](https://docs.microsoft.com/en-us/azure/azure-monitor/overview) and [Log Analytics](https://docs.microsoft.com/en-us/azure/azure-monitor/platform/design-logs-deployment) by turning on [diagnostic logging](https://docs.microsoft.com/en-us/azure/mysql/howto-configure-audit-logs-portal#set-up-diagnostic-logs).

To query for user connection related events, run the following KQL query:

AzureDiagnostics  
| where ResourceProvider =="MICROSOFT.DBFORMYSQL"  
| where Category == 'MySqlAuditLogs' and event\_class\_s == "connection\_log"  
| project TimeGenerated, LogicalServerName\_s, event\_class\_s, event\_subclass\_s, event\_time\_t, user\_s , ip\_s , sql\_text\_s  
| order by TimeGenerated asc

## Encryption

Data in the MySQL instance is encrypted at rest by default. Any automated backups are also encrypted to prevent potential leakage of data to unauthorized parties. This encryption is typically performed with a key that is created when the instance is created. In addition to this default encryption key, administrators have the option to [bring your own key (BYOK)](https://docs.microsoft.com/en-us/azure/mysql/concepts-data-encryption-mysql).

When using a customer-managed key strategy, it is vital to understand responsibilities around key lifecycle management. Customer keys are stored in an [Azure Key Vault](https://docs.microsoft.com/en-us/azure/key-vault/general/basic-concepts) and then accessed via policies. It is vital to follow all recommendations for key management, the loss of the encryption key equates to the loss of data access.

In addition to a customer-managed keys, use service-level keys to [add double encryption](https://docs.microsoft.com/en-us/azure/mysql/concepts-infrastructure-double-encryption). Implementing this feature will provide highly encrypted data at rest, but it does come with encryption performance penalties. Testing should be performed.

Data can be encrypted during transit using SSL/TLS. As previously discussed, it may be necessary to [modify your applications](https://docs.microsoft.com/en-us/azure/mysql/howto-configure-ssl) to support this change and also configure the appropriate TLS validation settings.

## Firewall

Once users are set up and the data is encrypted at rest, the migration team should review the network data flows. Azure Database for MySQL provides several mechanisms to secure the networking layers by limiting access to only authorized users, applications and devices.

The first line of defense for protecting the MySQL instance is to implement [firewall rules](https://docs.microsoft.com/en-us/azure/mysql/concepts-firewall-rules). IP addresses can be limited to only valid locations when accessing the instance via internal or external IPs. If the MySQL instance is destined to only serve internal applications, then [restrict public access](https://docs.microsoft.com/en-us/azure/mysql/howto-deny-public-network-access).

When moving an application to Azure along with the MySQL workload, it is likely there will be multiple virtual networks setup in a hub and spoke pattern that will require [Virtual Network Peering](https://docs.microsoft.com/en-us/azure/virtual-network/virtual-network-peering-overview) to be configured.

## Private Link

To limit access to the Azure Database for MySQL to internal Azure resources, enable [Private Link](https://docs.microsoft.com/en-us/azure/mysql/concepts-data-access-security-private-link). Private Link will ensure that the MySQL instance will be assigned a private IP rather than a public IP address.

**Note:** There are many other [basic Azure Networking considerations](https://docs.microsoft.com/en-us/azure/mysql/concepts-data-access-and-security-vnet) that must be taken into account that are not the focus of this guide.

Review a set of potential [security baseline](https://docs.microsoft.com/en-us/azure/mysql/security-baseline) tasks that can be implemented across all Azure resources. Not all of the items described on the reference link will apply to the specific data workloads or Azure resources.

## Security Checklist

* Use Azure AD authentication where possible.
* Enable Advanced Thread Protection.
* Enable all auditing features.
* Consider a Bring-Your-Own-Key (BYOK) strategy.
* Implement firewall rules.
* Utilize private endpoints for workloads that do not travel over the Internet.

# Optimization

## Monitoring Hardware and Query Performance

In addition to the audit and activity logs, server performance can also be monitored with [Azure Metrics](https://docs.microsoft.com/en-us/azure/azure-monitor/platform/data-platform-metrics). Azure metrics are provided in a one-minute frequency and alerts can be configured from them. For more information, reference [Monitoring in Azure Database for MySQL](https://docs.microsoft.com/en-us/azure/mysql/concepts-monitoring) for specifics on what kind of metrics that can be monitored.

As previously mentioned, monitoring metrics such as the cpu\_percent or memory\_percent can be important when deciding to upgrade the database tier. Consistently high values could indicate a tier upgrade is necessary.

Additionally, if cpu and memory do not seem to be the issue, administrators can explore database-based options such as indexing and query modifications for poor performing queries.

To find poor performing queries, run the following:

AzureDiagnostics  
| where ResourceProvider == "MICROSOFT.DBFORMYSQL"  
| where Category == 'MySqlSlowLogs'  
| project TimeGenerated, LogicalServerName\_s, event\_class\_s, start\_time\_t , query\_time\_d, sql\_text\_s  
| top 5 by query\_time\_d desc

## Query Performance Insight

In addition to the basic server monitoring aspects, Azure provides tools to monitor application query performance. Correcting or improving queries can lead to significant increases in the query throughput. Use the [Query Performance Insight tool](https://docs.microsoft.com/en-us/azure/mysql/concepts-query-performance-insight) to analyze the longest running queries and determine if it is possible to cache those items if they are deterministic within a set period, or modify the queries to increase their performance.

The slow\_query\_log can be set to show slow queries in the MySQL log files (default is OFF). The long\_query\_time server parameter can alert users for long query times (default is 10 sec).

## Upgrading the Tier

The Azure Portal can be used to scale between from General Purpose and Memory Optimized. If a Basic tier is chosen, there will be no option to upgrade the tier to General Purpose or Memory Optimized later. However, it is possible to utilize other techniques to perform a migration/upgrade to a new Azure Database for MySQL instance.

For an example of a script that will migrate from Basic to another server tier, reference [Upgrade from Basic to General Purpose or Memory Optimized tiers in Azure Database for MySQL](https://techcommunity.microsoft.com/t5/azure-database-for-mysql/upgrade-from-basic-to-general-purpose-or-memory-optimized-tiers/ba-p/830404).

## Scale the Server

Within the tier, it is possible to scale cores and memory to the minimum and maximum limits allowed in that tier. If monitoring shows a continual maxing out of CPU or memory, follow the steps to [scale-up to meet your demand](https://techcommunity.microsoft.com/t5/azure-database-for-mysql/upgrade-from-basic-to-general-purpose-or-memory-optimized-tiers/ba-p/830404).

## Moving Regions

Moving a database to a different Azure region depends on the approach and architecture. Depending on the approach, it could cause system downtime.

The recommended process is the same as utilizing read replicas for maintenance failover. However, compared to the planned maintenance method mentioned above, the speed to failover is much faster when a failover layer has been implemented in the application. The application should only be down for a few moments during the read replica failover process. More details are covered in the [Business Continuity and Disaster Recovery](03_BCDR.md) section.

## Optimization Checklist

* Monitor for slow queries.
* Periodically review the Performance Insight dashboard.
* Utilize monitoring to drive tier upgrades and scale decisions.
* Consider moving regions if the users or application needs change.

TODO - https://wemakewaves.medium.com/migrating-our-php-applications-to-docker-without-sacrificing-performance-1a69d81dcafb

# Business Continuity and Disaster Recovery (BCDR)

## Backup and Restore

As with any mission critical system, having a backup and restore as well as a disaster recovery (BCDR) strategy is an important part of your overall system design. If an unforseen event occurs, you should have the ability to restore your data to a point in time (Recovery Point Objective) in a reasonable amount of time (Recovery Time Objective).

### Backup

Azure Database for MySQL supports automatic backups for 7 days by default. It may be appropriate to modify this to the current maximum of 35 days. It is important to be aware that if the value is changed to 35 days, there will be charges for any extra backup storage over 1x of the storage allocated.

There are several current limitations to the database backup feature as described in the [Backup and restore in Azure Database for MySQL](https://docs.microsoft.com/en-us/azure/mysql/concepts-backup) docs article. It is important to understand them when deciding what additional strategies that should be implemented.

Some items to be aware of include:

* No direct access to the backups
* Tiers that allow up to 4TB have a full backup once per week, differential twice a day, and logs every five minutes
* Tiers that allow up to 16TB have backups that are snapshot based

**Note:** [Some regions](https://docs.microsoft.com/en-us/azure/mysql/concepts-pricing-tiers#storage) do not yet support storage up to 16TB.

### Restore

Redundancy (local or geo) must be configured during server creation. However, a geo-restore can be performed and allows the modification of these options during the restore process. Performing a restore operation will temporarily stop connectivity and any applications will be down during the restore process.

During a database restore, any supporting items outside of the database will also need to be restored. Review the migration process. See [Perform post-restore tasks](https://docs.microsoft.com/en-us/azure/mysql/concepts-backup#perform-post-restore-tasks) for more information.

## Read Replicas

[Read replicas](https://docs.microsoft.com/en-us/azure/mysql/concepts-read-replicas) can be used to increase the MySQL read throughput, improve performance for regional users and to implement disaster recovery. When creating one or more read replicas, be aware that additional charges will apply for the same compute and storage as the primary server.

## Deleted Servers

If an administrator or bad actor deletes the server in the Azure Portal or via automated methods, all backups and read replicas will also be deleted. It is important that [resource locks](https://docs.microsoft.com/en-us/azure/azure-resource-manager/management/lock-resources) are created on the Azure Database for MySQL resource group to add an extra layer of deletion prevention to the instances.

## Regional Failure

Although rare, if a regional failure occurs geo-redundant backups or a read replica can be used to get the data workloads running again. It is best to have both geo-replication and a read replica available for the best protection against unexpected regional failures.

**Note** Changing the database server region also means the endpoint will change and application configurations will need to be updated accordingly.

### Load Balancers

If the application is made up of many different instances around the world, it may not be feasible to update all of the clients. Utilize an [Azure Load Balancer](https://docs.microsoft.com/en-us/azure/load-balancer/load-balancer-overview) or [Application Gateway](https://docs.microsoft.com/en-us/azure/application-gateway/overview) to implement a seamless failover functionality. Although helpful and time-saving, these tools are not required for regional failover capability.

## WWI Case Study

WWI wanted to test the failover capabilities of read replicas so they performed the steps outlined below.

### Creating a read replica

* Open the Azure Portal.
* Browse to the Azure Database for MySQL instance.
* Under **Settings**, select **Replication**.
* Select **Add Replica**.
* Type a server name.
* Select the region.
* Select **OK**, wait for the instance to deploy. Depending on the size of the main instance, it could take some time to replicate.

**Note:** Each replica will incur additional charges equal to the main instance.

### Failover to read replica

Once a read replica has been created and has completed the replication process, it can be used for failed over. Replication will stop during a failover and make the read replica its own main instance.

Failover Steps:

* Open the Azure Portal.
* Browse to the Azure Database for MySQL instance.
* Under **Settings**, select **Replication**.
* Select one of the read replicas.
* Select **Stop Replication**. This will break the read replica.
* Modify all applications connection strings to point to the new main instance.

## BCDR Checklist

* Modify backup frequency to meet requirements.
* Setup read replicas for read intensive workloads and regional failover.
* Create resource locks on resource groups.
* Implement a load balancing strategy for applications for quick failover.

TODO - https://semaphoreci.com/blog/7-continuous-integration-tools-for-php-laravel