Microsoft

Introduction to

Microsoft Azure Database for MySQL

An overview of its features and capabilities

Abstract

Azure Database for MySQL makes it easy to build new applications or migrate existing ones to the cloud using the programming languages and frameworks of your choice. As a fully managed database service, it increases your productivity by freeing you from database configuration and administration, operating system management, and all the other things you need to deal with when running MySQL in a VM.

Azure Database for MySQL is based on the latest community versions of MySQL and lets you continue using familiar languages and tools like PHP, Python, Java, Node.js, .NET, and MySQL Workbench. You can provision new databases in minutes and independently scale compute and storage, knowing you'll get built-in high availability, automatic backups and updates, and a service level agreement of 99.99 percent.

With AI-powered performance optimization, up to 10 read replicas, and support for up to 16TB of data and 20,000 IOPS, Azure Database for MySQL can handle even the most demanding workloads. You'll also get advanced data security and compliance, including encryption at rest, encryption in transit, and built-in compliance with HIPAA, PCI DSS, FedRAMP, and more. Best of all, with Azure Database for MySQL, you can pay as you go under a simple yet flexible pricing model, knowing that your database service is built on a world-class infrastructure with unmatched global reach.

Intended audience

This paper is for architects, developers, and database administrators who are thinking of building and deploying new cloud apps on MySQL or moving existing MySQL apps and databases to the cloud. It provides an overview of Azure Database for MySQL and the value that it provides as a fully managed, open-source database service, including key use cases and an introduction to common application hosting environments. This guide also covers the various deployment options for Azure Database for MySQL and introduces the key concepts you'll need to understand to best put each deployment option to use.

Contents

.

Introduction: MySQL databases in the cloud

In the early days of computing, developers shared the code they wrote to learn from one another and evolve their software. This spirit of collaboration remains alive today with open-source software, which gives users rights to access, modify, and distribute source code—relying on a community of global contributors to achieve more than any one organization could on its own.

To fully appreciate the value of open-source software, it's worth examining the many benefits it provides, including open standards that are accessible to all, full visibility into the code base, community efforts to improve and support the code, a lack of licensing fees, and lower development costs. Openly available source code also speeds the delivery of new offerings, helps prevent vendor lock-in, facilitates choice across platforms, and can help attract more and better talent. Best practices for maximizing performance and availability are shared freely, as are those for deployment and operations, and reliability is maximized through community vetting.

It's for all these reasons that open-source databases such as MySQL have grown to be so popular. MySQL Community Edition is supported by a large and vibrant community of open-source developers. As the database component of the popular LAMP (Linux, Apache, MySQL, and PHP/Perl/Python) web application stack, MySQL remains one of the most popular database management systems among developers today.

In turn, the pervasiveness of MySQL is fueling explosive growth in MySQL database *services*, which let you reap the benefits of using MySQL while avoiding the many pitfalls of on-premises solutions or hosted, self-managed VMs. For example, with platform-as-a-service (PaaS) cloud services, you won't need to deal with deployment efforts, continual patching, painstaking performance optimization, complex high-availability mechanisms, intricate security and compliance concerns, and continual worries about scalability—all of which can lead to unpredictable DBA workloads, delays, and costs.

To make MySQL work in the enterprise, however, you'll need a lot more than just a hosted MySQL service. Rarely does any database exist in isolation, which is why you'll want to make sure your chosen cloud platform can deliver everything else you'll need. Specific services will depend on your app, such as whether you'll be building a containerized application, implementing microservices, ingesting massive data volumes, implementing real-time streaming analytics, or deploying a prepackaged app like Drupal, Magento, Moodle, or WordPress. And don't forget about development tools, languages, software frameworks, and integration; you won't want unnecessary complications in those areas either. After all, even if a platform provides all the services you need, it won't do you much good if you need to learn an entirely new skill set to best put those services to use.

In the rest of this paper, we'll provide an overview of Microsoft Azure Database for MySQL, a fully managed MySQL database service, and the value that it brings to you as an architect, developer, or database administrator. We'll also cover the two deployment options for the service and the key concepts you'll need to understand to best put each deployment option to use. Finally, we'll examine some popular use cases for Azure Database for MySQL and some application hosting environments that complement it.

Azure Database for MySQL

Azure Database for MySQL brings together MySQL innovations with all the benefits of the Microsoft Azure cloud. As a fully managed service, it frees you from the burden of infrastructure and database management to help you easily build new applications or migrate existing ones to the cloud using the programming languages and frameworks of your choice. You can provision new databases in minutes and independently scale compute and storage, knowing that you won't need to pay extra for things like automatic database patching and backups, monitoring tools, AI-powered performance optimization, or essential security features.

Azure Database for MySQL gives you:

- A highly available and flexible MySQL service that automatically handles all maintenance, patching, and updates—so that you can focus on application innovation.
- Support for the latest community versions of MySQL, including the ability to continue using familiar tools and languages like MySQL Workbench, PHP, Python, Java, Node.js, and .NET.
- A simplified developer experience, including streamlined deployment via tight integration with Azure App Service, Azure Kubernetes Service, and GitHub Actions.
- Support for up to 16TB of data, 20,000 IOPS, and 10 read replicas (for scaling out read-intensive workloads).
- A 99.99-percent, financially-backed service level agreement.
- Advanced data security and compliance features, including double encryption at rest, encryption in transit, Azure Defender[1], and built-in compliance with HIPAA, PCI DSS, FedRAMP, and more.
- Multiple pricing options—including the ability to pay-as-you-go or to save money by paying for compute capacity in advance.

Our new Flexible Server deployment option for Azure Database for MySQL delivers additional innovations, including greater flexibility and more granular control over database configuration and management. Additional capabilities in Flexible Server include custom maintenance windows, multiple configuration parameters for fine-grained tuning, zone-redundant high availability, and additional cost optimization controls—including burstable compute and the ability to start and stop servers to save money.

The capabilities provided by Azure Database for MySQL make it ideal for hosting enterprise-scale production workloads. You can get started immediately and build a broad range of solutions, knowing that you won't be limited by infrastructure issues, scalability, availability concerns, geographic presence, or excessive upfront costs. The possibilities are endless, which is one reason why open-source database services are becoming so popular.

As an Azure data service, Azure Database for MySQL is backed by Azure compute and Azure Storage. And it's built to work with the extensive range of other Azure services—including those for data ingestion, integration, application hosting, AI and machine learning, analytics, visualization, data governance, DevOps, and more.

Like all Azure services, Azure Database for MySQL is built on a world-class infrastructure with unmatched global reach; Azure has more global regions than any other cloud provider, enabling you to bring your apps closer to your users around the world, preserve data residency, and satisfy compliance demands. Finally,

[1] Azure Defender is a separate Azure service and is available at an additional cost.

when you choose Azure Database for MySQL, you'll get comprehensive compliance and Azure IP Advantage, a program for Azure customers that provides protection against intellectual property (IP) risks.

In the rest of this paper, we'll cover:

- The two deployment options for Azure Database for MySQL and when you might want to consider using each one.
- The key concepts you'll need to understand for each deployment option: Single Server and Flexible Server.
- Popular use cases for Azure Database for MySQL.
- Commonly used application hosting environments and other services for building apps with Azure Database for MySQL.

Deployment options

With Azure, you can run MySQL in two ways: within a hosted VM infrastructure-as-a-service (IaaS) environment, or within a fully managed platform-as-a-service (PaaS) environment. Although this paper focuses on the range of PaaS options provided on Azure, to understand the value provided by those options, it's worth first taking a quick look at you get—and what you don't get—when running a self-managed instance of MySQL within a VM.

With IaaS, you have full control over the MySQL database engine running within a VM, along with the underlying operating system, tools, and utilities—including the ability to run whichever versions you choose. However, this comes at a cost, in that you'll need to handle all database and operating system management and administration tasks on your own: maintaining and patching your servers, high availability design, disaster recovery, security design, performance tuning, and more. These tasks can require substantial time and effort, contributing significantly to your overall total cost of ownership (TCO).

Azure Database for MySQL, on the other hand, is a *fully managed* PaaS database service. This means that Microsoft automatically configures and manages the underlying infrastructure, operating system, and database software so that you don't have to—including the application of patches and security fixes. You'll also have access to a wealth of features that aren't typically included out-of-the-box with an IaaS environment, such as automated backups, built-in security and performance features, on-demand scalability, and guaranteed high availability.

However, even within the PaaS domain, one size rarely fits all. That's why Azure Database for MySQL supports two different deployment options:

- **Azure Database for MySQL - Single Server**, the first deployment option we introduced, automatically handles most database management functions, including patching, backups, and security—with minimal user configuration and control.
- **Azure Database for MySQL - Flexible Server** provides new innovations that go beyond what's in Single Server, including greater flexibility and more granular control over database configuration and management. Flexible Server also provides additional cost optimization controls, including a burstable compute pricing tier and the ability to start and stop the server for development or test scenarios—and only pay for storage while the database is stopped. Flexible Server is recommended for most new Azure Database for MySQL deployments, including migrations from on-premises or VM-based (IaaS) MySQL environments and new cloud-native apps.

Choose the right MySQL Server option in Azure provides additional information on the different ways you can deploy MySQL on Azure, including the main differences between each option and how specific business motivations might influence whether to choose IaaS or PaaS.

The following chapters in this guide cover the above deployment options in greater detail, including the key concepts you'll need to understand to best put each deployment option to use.

Preview deployment options and features

Throughout this guide, we've included features that are in preview—meaning that they're available for early access but have yet to reach general availability. In general, preview features are provided without a service level agreement, might have constrained capabilities, and should be tested thoroughly before using them to support production workloads.

Preview features change regularly, when our engineering and product teams are confident that they're ready to be promoted to general availability. Our online documentation for Azure Database for MySQL is

the definitive source of what's currently in preview. Azure updates is another good source to monitor—you'll typically see an entry when a preview feature is introduced and another entry when it reaches general availability. Finally, the Azure Database for MySQL blog often has posts from our engineering team about new features and capabilities in Azure Database for MySQL—including guidance on how to put them to use.

Key concepts: Single Server

Following are some of the key concepts you'll need to understand to get the most out of Azure Database for MySQL - Single Server. This deployment option is designed to handle most database management functions for you automatically, with minimal user configuration or control.

Azure Database for MySQL Single Server provides a deeper overview of the Single Server deployment option.

Supported versions

Azure Database for MySQL - Single Server is based on MySQL Community Edition, using InnoDB as the default storage engine. It supports all current major versions that are supported by the MySQL community. As of June 2021, the Single Server deployment option supports the following versions:

- MySQL Version 5.6[2]
- MySQL Version 5.7
- MySQL Version 8.0

Specific minor versions and bug fix releases will change over time, so it's best to check the Azure Database for MySQL version support policy for the latest details.

Also, it's worth noting that, with the Single Server deployment option, a gateway is used to redirect database connections to server instances—meaning you may need to specify a different port in your server connection string to connect to a specific version of MySQL. Connect to a gateway node that is running a specific MySQL version covers all the details.

Connection libraries, drivers, and tools compatibility

Azure Database for MySQL - Single Server is compatible with a wide variety of programming languages and drivers, including PHP, .NET, Node.js, Go, Python, and Java. It's best to check the online documentation for the latest information on supported drivers and version compatibility.

Your existing tools should continue to work with Azure Database for MySQL, as long as database manipulation operates within the confines of user permissions. Again, the online documentation is the best source of information about which management tools have been tested and found to be compatible with Azure Database for MySQL.

Regardless of which tools you choose, remember that you'll need to think about connection resiliency and how to handle transient errors

Databases and servers

Azure Database for MySQL - Single Server exposes access and features at the *server* level, which is created within an Azure subscription and is the parent resource for *one or more individual databases*. It collocates the computing resources for those databases in a specified Azure region, provides a namespace for the databases, and serves as a connection endpoint for server and database access. A server also defines the scope for management policies that apply to those databases—including sign-in, firewall, users, roles, and

[2] MySQL v5.6 is retired on Single Server as of February 2021. Starting September 1, 2021, you will not be able to create new v5.6 servers for the Azure Database for MySQL - Single Server deployment option. You will still be able to perform point-in-time recoveries and create read replicas for existing servers.

configurations. The online documentation for Azure Database for MySQL - Single Server provides more information on the concepts of databases and servers.

Server pricing tiers and resources

Pricing for a server under the Single Server deployment model is determined by the configuration of resources that you specify for that server, including the pricing tier, number of vCores, and amount of storage. Create a single database under the server, and that database has exclusive access to those resources; create multiple databases under the server, and those resources are shared.

Pricing tiers

You can create an Azure Database for MySQL server in one of three pricing tiers, which are differentiated by the amount of compute (vCores) that can be provisioned, the amount of memory per vCore, and the technology used for data storage. In general:

- **Basic** is good for workloads that require only light compute and I/O—such as test servers and small-scale, infrequently used applications.
- **General purpose** is good for workloads that require balanced compute and memory, with scalable I/O—including web apps, mobile apps, and many other enterprise apps. This is our most popular option, and it works well for most scenarios.
- **Memory optimized** is good for workloads that require in-memory performance for faster transaction processing and greater concurrency—including real-time data processing and high-performance transactional or analytical apps.

After you create a server, you can adjust the number of vCores, the hardware generation, and the pricing tier up or down, within seconds. Similarly, you can increase the amount of storage or adjust your backup retention period up or down without application downtime.

Note: Changing to and from the basic tier is not supported, so if you're planning to scale, we recommend you start with the general-purpose or memory-optimized tiers. Also, you can't change the backup storage type after a server is created.

Azure Database for MySQL pricing tiers covers the multiple compute options that are provided within each pricing tier.

Reserved Capacity

Azure Database for MySQL - Single Server supports reserved capacity pricing, which lets you reserve compute power for a period of one year or three years. By taking advantage of it, you can save up to 60 percent compared to the regular, pay-as-you-go payment options.

Prepay for Azure Database for MySQL compute resources with reserved capacity provides more information on reserved capacity, including how to determine the right server size and how to purchase reserved capacity. A pricing calculator can be found on the Azure Database for MySQL pricing page.

Storage

When you provision a server, you'll also need to provision the amount of storage capacity available to it. This storage is used for the database files, temporary files, transaction logs, and MySQL server logs. The total storage you provision also defines the I/O capacity available to your server in terms of I/O operations per second (IOPS). Azure Database for MySQL – Single Server currently supports up to 16 TB of storage and up to 20,000 IOPS.

You can add additional storage capacity during and after the creation of the server. You can also enable storage auto-grow, which allows the system to grow storage automatically based on the storage consumption of your workload.

Azure Database for MySQL - Single Server provides up to 100 percent of your provisioned server storage as backup storage, at no additional cost. Any backup storage you use in excess of this amount is charged in GB per month, according to the service's pricing model.

Server management

You can create and manage servers by using the Azure portal, the Azure CLI, Azure PowerShell, or Azure Resource Manager templates. Which to use depends on your skill-set and preferences. In general:

- Linux admins and traditional open-source developers often prefer a command-line interface, with the Azure CLI providing just that. The Azure CLI is also a good choice for developers and Linux admins who rely heavily on Ruby scripts, Unix shell environments, Groovy, and so on, and want to reuse such scripts when managing MySQL in a PaaS environment. Manage an Azure Database for MySQL - Single Server using the Azure CLI covers some basic server management tasks.
- Azure PowerShell provides similar scripting capabilities for developers and admins who are familiar with PowerShell, with virtually no additional learning curve required.
- Azure portal is often a preferred option for developers or admins from all backgrounds who want the power and productivity of a GUI, including ready access to built-in monitoring capabilities and performance-tuning aids like Query Store, Query Performance Insights, and Performance Recommendations. Azure portal also provides access to Azure Cloud Shell, an interactive, authenticated, browser-accessible shell for managing Azure resources. You can use either Linux Bash or PowerShell with Azure Cloud Shell, enabling you to choose the shell experience that best suits the way you work. Manage an Azure Database for MySQL server using the Azure portal walks you through some basic server management tasks.
- Azure Resource Manager (ARM) templates enable you to define your infrastructure as code, using JavaScript Object Notation (JSON) to specify the resources to deploy—such as an Azure Database for MySQL instance—and the properties for those resources. The infrastructure code becomes part of your project and can be stored and versioned in your source code repository, enabling you to automate the deployment of your infrastructure as well as the code it runs.

Start/stop a server

Azure Database for MySQL lets you stop a running server, then start it again later. When you stop the server, it remains in that state for the next 7 days. If you do not manually start it during this time, the server will automatically be started at the end of 7 days, after which you can choose to stop it again.

With this functionality, you can stop the database server when not in use (such as during non-work hours) and start it when it needs to be back online. Stop/start could also be helpful when using Azure Database for MySQL for development or test scenarios, enabling you to save on costs by turning off the server compute when not in use.

Restart a server

There are times when you may need to restart a running server, which causes a short outage as the service performs the operation. The server won't restart if the Azure Database for MySQL service is busy, such as when processing a previously requested operation like scaling-up the number of vCores.

The time required to complete a restart depends on the time the MySQL database engine requires to bring the databases back online in a consistent state. To decrease the restart time, you should minimize the amount of activity occurring on the server prior to the restart.

Server limitations

The database engines that power Azure Database for MySQL are the MySQL Community Editions, including any underlying limitations they may have, with additional limitations that we've implemented to ensure the supportability and manageability of the database service. Limitations in Azure Database for MySQL describes these additional limitations, including supported storage engines, privilege and data manipulation statement support, functional limits, and current known issues.

Server parameters

The MySQL engine provides many different server variables/parameters that can be used to configure and tune engine behavior. Some parameters can be set dynamically during runtime while others are "static", requiring a server restart in order to apply. Server parameters in Azure Database for MySQL provides additional detail on server parameters, configuration defaults, and other limits based on which pricing tier and compute options you've selected.

Move servers between Azure regions

There are times when you might want to move an existing Azure Database for MySQL server from one region to another—for example, as a part of your disaster recovery planning. Move an Azure Database for MySQL server to another region by using the Azure portal describes how to do this by using a cross-region read replica.

Networking

Connections to Azure Database for MySQL - Single Server are established through a gateway that routes incoming connections to the physical location of your server in an Azure server cluster. As a client connects to the database, the connection string to the server resolves to the gateway IP address. Inside the database cluster, traffic is forwarded to the appropriate Azure Database for MySQL server.

Under this architecture, in order to connect to Azure Database for MySQL - Single Server from outside of Azure, you may need to open up your client-side firewall to allow outbound traffic initiated from the client to reach our gateways. Connectivity architecture in Azure Database for MySQL provides additional information on this topic, including a complete, per-region list of the IP addresses used by our gateways and the answers to some frequently asked questions.

Firewall rules

By default, all access to an Azure Database for MySQL server is blocked by its firewall. You can use firewall rules to specify which computers can access a server based on the originating IP address for each request. These firewall rules apply to all databases on the logical server and are configured as ranges of IP addresses. You can also configure the firewall to allow access from all Azure services and all Azure subnets.

The online documentation for Azure Database for MySQL provides more information about firewall rules, including connecting from the internet, connecting from Azure, programmatically managing firewall rules, and troubleshooting firewall issues.

VNet endpoints

Virtual network (VNet) rules, a firewall security feature, control whether your Azure Database for MySQL server accepts communications sent from particular subnets on Azure virtual networks. To create a virtual network rule, you must first have a VNet and a VNet service endpoint to reference. If you configure your

database firewall to allow access from all Azure services and all Azure subnets, you can use virtual network rules to achieve much more granular control.

The online documentation for Azure Database for MySQL provides more information about VNet service endpoints and virtual network rules, including their benefits, characteristics, limitations, and use with ExpressRoute.

Private link

Azure Private Link allows you to create private endpoints for Azure Database for MySQL - Single Server to bring it inside your VNet. The private endpoint exposes a private IP address within a subnet, which you can use to connect to your database server just like any other resource in the VNet. You can use Private Link to establish cross-premises access to a private endpoint using ExpressRoute, private peering, or VPN tunneling—or choose to disable all access via a public endpoint. Private Link for Azure Database for MySQL discusses relevant use cases and how to use Private Link with Azure Database for MySQL.

Security

With Azure Database for MySQL, you have several capabilities to help ensure safe and secure access. To help protect your data, it's encrypted both in-transit and at-rest. Additional security mechanisms are built into the networking and access management mechanisms for Azure Database for MySQL, and you can opt into Azure Defender for an even greater level of security. Security in Azure Database for MySQL provides a deeper introduction to these security features.

Configuring SSL/TLS

Enforcing Secure Sockets Layer (SSL)/Transport Level Security (TLS) connections between your client applications and an Azure Database for MySQL server encrypts all traffic between the two, which can help protect against "man-in-the-middle" attacks. Enforcement of SSL/TLS connections is enabled by default when a new Azure Database for MySQL server is provisioned. It's worth noting, however, that some application frameworks don't enable SSL/TLS by default during installation, so you'll want to check this if your app is failing to connect to your database server.

SSL/TLS connectivity in Azure Database for MySQL provides more information on SSL/TLS connections. Other relevant articles cover how to configure SSL and how to enforce a minimum TLS version at the server level. Azure Database for MySQL currently supports TLS versions 1.0, 1.1, and 1.2

Azure Active Directory authentication

You can connect to Azure Database for MySQL using Azure Active Directory (Azure AD) authentication, using identities defined in Azure AD. This lets you manage database user identities in a central location, alongside user identities for other Azure services, thus simplifying and centralizing permission management. Azure AD supports federation with your on-premises Active Directory infrastructure to enable single sign-on (SSO) to cloud applications and data services.

Use Azure Active Directory for authenticating with MySQL covers the benefits of this approach and how it works, including administrator types, permissions, supported connection methods, and additional considerations. Additional articles cover how to configure Azure AD access and access using a Managed Service Identity for an Azure VM (which lets you authenticate to services that support Azure AD authentication, without needing to insert credentials into your code).

Threat Protection with Azure Defender

Azure Defender for open-source relational databases can help detect anomalous activities that may indicate unusual or potentially harmful attempts to access or exploit your databases. It's part of the integrated cloud

workload protection platform within Azure Security Center, a unified package of advanced security capabilities, which is available at an additional cost.

When Azure Defender detects a threat in any area of your environment, it generates a security alert that describes details of the affected resources, suggested remediation steps, and, in some cases, an option to trigger a logic app in response. Alerts are triggered upon anomalous database access and query patterns, suspicious database activities, and brute-force attacks. The alerts reference page provides a full list of security alerts for Azure Database for MySQL and other open-source relational databases.

Introduction to Azure Defender for open-source relational databases provides more information on its capabilities and benefits. Our quickstart on enabling Azure Defender walks you through how to set it up.

Data encryption with customer managed keys

By default, Azure Database for MySQL leverages Azure Storage encryption to encrypt data at-rest using Microsoft-managed cryptographic keys. If you'd prefer, you can use your own keys, enabling your organization to implement separation of duties in the management of keys and data. When you choose to use your own keys, you'll be responsible for—and in full control of—the key's lifecycle, key usage permissions, and auditing of operations on keys. Customer managed keys require the use of Azure Key Vault, which is free for up to 12 months and 10,000 transactions with a new Azure free account.

Data encryption with a customer-managed key provides more information on the benefits of using your own keys, requirements for doing so, how the process works, and more.

Infrastructure double encryption

With Azure Database for MySQL, data (including backups) are always encrypted on disk using a FIPS 140-2 validated cryptographic module and an AES 256-bit cipher. Infrastructure double encryption provides additional data protection through a second layer of encryption, which uses a different FIPS 140-2 validated cryptographic module with a different encryption algorithm and a second, service-managed cryptographic key. Infrastructure double encryption is not enabled by default because the additional layer of encryption may have a performance impact.

Azure Database for MySQL infrastructure double encryption provides more information on the benefits of infrastructure double encryption, supported scenarios, and limitations.

Auditing

The Azure Database for MySQL audit log can be used to track database-level activity—as is often required for compliance purposes. Audit logs are integrated with Azure Monitor diagnostic logs, enabling you to retain logs in Azure Storage, perform analysis with Azure Monitor Logs, and integrate with third-party tools via Event Hubs.

By default, audit logging is disabled. Audit logs in Azure Database for MySQL covers how to configure audit logging, how to access audit logs, diagnostic log schemas, and how to analyze audit logs by using Azure Monitor.

Compliance

Regulatory Compliance in Azure Policy provides Microsoft created and managed initiative definitions for compliance domains and security controls related to different standards. These definitions and security controls are known as built-ins, which you can assign individually to help make your Azure resources compliant with specific standards. Azure Policy Regulatory Compliance controls for Azure Database for MySQL lists the compliance domains and security controls for Azure Database for MySQL.

Security baseline

The Azure Security Benchmark contains recommendations to help you improve the security of your applications and data on Azure. It focuses cloud-centric control areas. These controls are consistent with well-known security benchmarks, such as those described by the Center for Internet Security (CIS) Controls Version 7.1.

Azure security baseline for Azure Database for MySQL applies guidance from the Azure Security Benchmark version 1.0 to Azure Database for MySQL. Topics are grouped by the security controls defined by the Azure Security Benchmark, including network security, logging and monitoring, identity and access control, data protection, vulnerability management, inventory and asset management, secure configuration, malware defense, data recovery, incident response, penetration tests, and red team exercises—with links to the Azure Security Benchmark documentation for more information on each topic.

Service maintenance

Azure Database for MySQL performs automated patching of the underlying hardware, OS, and database engine—including new service features as well as security and software updates. No user action or configuration settings are required for patching, and patches are tested extensively before they're rolled out using safe deployment practices. Minor version upgrades for the MySQL engine are included as part of this patching cycle.

A planned maintenance event is a maintenance window when such service updates are deployed to servers in a given Azure region. At a minimum, they're 30 days apart. You'll receive a notification event when a service update is deployed in the region hosting your servers, including a notification of the next maintenance window 72 hours in advance.

Planned maintenance notification in Azure Database for MySQL - Single Server provides more information on this topic, including duration and customer impact, how to receive and view planned maintenance notifications, and other useful information. If you hard-code the gateway IP addresses for Azure Database for MySQL in your application, which isn't recommended, you'll also need to consider what happens when Microsoft decommissions older gateway hardware.

Business continuity

Azure Database for MySQL provides several features that you can use to help ensure business continuity and prevent data loss. Overview of business continuity with Azure Database for MySQL - Single Server introduces these business continuity features and how you can put them to use, including an overview of recovery scenarios and how they can affect your Recovery Time Objective and Recovery Point Objective.

High availability

Azure Database for MySQL provides guaranteed high availability to help you avoid application downtime—including a financially backed service level agreement of 99.99 percent for all services that are in general availability. This high-availability model is based on built-in failover mechanisms that are triggered when a node-level interruption occurs, such as in the event of a hardware failure.

The entire failover process typically takes tens of seconds. Here's how it works: At all times, changes to an Azure Database for MySQL server occur in the context of a transaction, with changes recorded synchronously in Azure storage when the transaction is committed. If a node-level interruption occurs, the database server automatically creates a new node and attaches data storage to the new node. Any active connections are dropped, any in-flight transactions aren't committed, and an internal gateway transparently redirects any new connections to the new instance.

Because of how the failover process works, it's important that you build your database applications to detect and retry dropped connections and failed transactions. When your application retries, its connection is transparently redirected to the newly created instance. And because the redirect is handled transparently by the internal Azure gateway, your client application's connection string remains the same.

The same failover model makes it easy to scale compute resources up or down. When you do so, a new compute instance with the specified resources is created, and existing data storage is detached from the original instance and attached to the new instance. Client applications are disconnected, open, uncommitted transactions are canceled, and, after the client application retries the connection or makes a new connection, the gateway directs the connection to the newly sized instance.

The online documentation for Azure Database for MySQL provides more information on high availability.

Backup and restore

Azure Database for MySQL supports locally redundant backups in the basic tier, and both locally redundant and geographically redundant backups in the general-purpose and memory-optimized tiers. You can take advantage of these features to provide business continuity, such as recovering a server after a user or application error, or in the unlikely event of an Azure regional datacenter outage. The online documentation provides a good overview of recovery scenarios and how they can affect your Estimated Recovery Time and Recovery Point Objective.

To understand how you can use backup and restore as part of a comprehensive business continuity strategy, it's worth taking a closer look at how these features work: Azure Database for MySQL automatically creates full, differential, and transaction-log backups, which you can use to restore a server to any point in time within your configured backup retention period. Generally, full backups occur weekly, differential backups occur twice a day, and transaction-log backups occur every five minutes.

In the subset of Azure regions that support up to 16TB or storage, backups are snapshot-based. The first full snapshot backup is scheduled immediately after a server is created and is retained as the server's base backup. Subsequent snapshot backups are differential backups only and, while they do not occur on a fixed schedule, are performed three times per day. Transaction log backups occur every five minutes.

The default backup retention period is 7 days and can be configured for up to 35 days. All backups are secured using AES 256-bit encryption.

The basic pricing tier offers only locally redundant backup storage, whereas the general-purpose and memory-optimized pricing tiers provide the option to augment locally redundant backups with geographically redundant (geo-redundant) backups. When you choose geo-redundant backup storage, your backups aren't only stored within the region in which your server is hosted; they're also replicated to a paired datacenter in another region within the same geography, allowing you to restore your server in any Azure region in the event of a disaster. There's a delay between when backups are taken and when they're replicated to another region, so, if a disaster occurs, you may face some data loss.

When you provision a server, you need to decide whether to augment locally redundant backups with geo-redundant backups because you can't change this option afterward. Whichever option you choose, Azure Database for MySQL provides up to 100 percent of your provisioned server storage as backup storage at no additional cost.

Performing a restore creates a new server from the original server's backups. Recovery time depends on several factors, including the size of the database or databases, transaction log size, network bandwidth, and the total number of databases being recovered in the same region at the same time. The estimated recovery time is usually less than 12 hours.

Point-in-time restore is available with either backup redundancy option. It creates a new server in the same region as your original server. Geo-restore, which is available only if you configured your server for geo-redundant storage, allows you to restore your server to a different region.

The online documentation for Azure Database for MySQL provides additional detail on how backup and restore work.

Long-term backup retention

Native backup features within Azure Database for MySQL are well-suited for operational recoveries. For longer-term retention, you can use mysqldump to implement automated backups for long-term retention in Azure Storage.

Monitoring and tuning

Azure Database for MySQL provides several metrics that give you insight into server behavior, which you can use for both workload optimization and troubleshooting. All metrics have a one-minute frequency, and each metric provides 30 days of history. You can also enable slow query logging on your server and can use the intelligent performance optimization features in Azure Database for MySQL to track and help optimize query performance.

Monitoring in Azure Database for MySQL provides an overview of these features and capabilities, including a list of available metrics.

Slow query logs

In Azure Database for MySQL, the slow query log can be used to identify performance bottlenecks for troubleshooting. By default, the slow query log is disabled. Slow query logs in Azure Database for MySQL covers how to configure and access slow query logs, log retention periods, and integration of slow query logs with Azure Monitor Diagnostic Logs for further analysis.

Intelligent performance optimization

Azure Database for MySQL provides intelligent performance optimization, a set of features that make it easy to optimize query performance.

- Query Store tracks query performance over time, including query runtime statistics and wait events. You can control data collection and storage via various configuration settings.
- Query Performance Insight works with Query Store to provide visualizations within the Azure portal, which you can use to identify key queries that impact performance.
- Performance Recommendations identifies opportunities for creating new indexes that may improve workload performance. It takes into consideration various database characteristics, including the database's schema and the data collected by Query Store.

NOTE: Intelligent performance optimization features are available for Azure Database for MySQL running MySQL versions 5.7 and 8.0. They are not available with MySQL version 5.6.

Azure Advisor recommendations

The Azure Advisor system uses telemetry to issue performance and reliability recommendations for Azure Database for MySQL, which are made available in the Azure portal. In general, there are three types of recommendations:

- Performance recommendations – based on CPU usage, memory pressure connection pooling, disk utilization, and other server parameters.
- Reliability recommendations – based on storage limits and connection limits.

- Cost optimization – including recommendations on server right-sizing.

When recommendations are available, a preview will appear as a banner notification in the Azure portal, under Overview. Details can be viewed in the Notifications section, just below the resource usage graphs. The online documentation provides more information on Azure Advisor for Azure Database for MySQL.

Replication

Azure Database for MySQL supports two types of replication:

- **Data-in replication**, which you can use to synchronize data into Azure Database for MySQL from an existing MySQL server running on-premises, running in a virtual machine, or hosted by another cloud provider.
- **Read replicas**, which you can use to replicate data from an Azure Database for MySQL server (master) to up to five read-only servers (replicas).

Both data-in replication and read replicas in Azure Database for MySQL use the replication technology that's based on the binary log (binlog) file position native to MySQL.

Data-in replication

Data-in replication allows you to synchronize data from an external MySQL server into the Azure Database for MySQL service. You can use it to keep data synchronized between your on-premises servers and Azure Database for MySQL, as may be appealing when you have an existing local database server but want to move a copy of the data to an Azure region closer to end users. You can also use data-in replication to synchronize data between Azure Database for MySQL and different cloud providers, including VMs and database services hosted in those clouds. Replicate data into Azure Database for MySQL provides more information on data-in replication, including common usage scenarios, requirements and limitations, and other considerations.

You set up data-in replication by using stored procedures, which can be run in the MySQL Shell, MySQL Workbench, or any other tool that can be used to run queries against MySQL. How to configure Azure Database for MySQL Data-in Replication covers how to create a server instance to use as a replica, configure the source MySQL server, dump and restore the source server, link the source and replica servers to start data-in replication, and other useful stored procedures.

NOTE: Data-in replication is only available with the General Purpose and Memory Optimized pricing tiers.

Read replicas

Read replicas can help improve the performance and scale of read-intensive workloads, which can be isolated to the replicas while write workloads are directed to the master. A common scenario is to have business intelligence and analytical workloads use the read replica as the data source for reporting. Azure Database for MySQL - Single Server deployments support up to five read replicas. (Azure Database for MySQL - Flexible Server supports up to 10 read replicas.)

Like with data-in replication, read replicas are updated asynchronously via the native replication technology in the MySQL engine. From a provisioning perspective, read replicas are considered new servers, and you manage them similarly to how you manage regular Azure Database for MySQL servers. For each replica, you're billed for the provisioned compute and storage.

Read replicas in Azure Database for MySQL covers when you might want to use a read replica, how it works, and how to implement and monitor it.

NOTE: Like data-in replication, read replication is only available with the General Purpose and Memory Optimized pricing tiers.

Best practices

Where applicable, throughout the documentation for Azure Database for MySQL - Single Server, we highlight best practices that can help you get the most of it. In addition, there are dedicated articles on best practices for monitoring, performance optimization, application development, and operations.

Quickstarts

Our quickstarts can help you get started with Azure Database for MySQL, such as creating a server then connecting to it and querying it. You'll find the first one here, with the rest listed immediately below it in the navigation pane.

To use the quickstarts you'll need an Azure subscription. If you don't have an Azure subscription, you can create an Azure free account before you begin.

Key concepts: Flexible Server

Azure Database for MySQL - Flexible Server, a fully managed database service, is designed to provide more granular control and flexibility over database management functions and configuration settings than the Single Server deployment option. It also enables greater architectural flexibility, such as collocating your database engine with the client-tier of your app to reduce latency or configuring high availability across multiple availability zones.

The Flexible Server deployment option also provides better cost optimization controls, including the ability to stop and start your server (and pay for only the time it's running) and a burstable compute tier that's ideal for workloads that don't need full, continuous compute capacity. The Flexible Server deployment option also lets you specify custom patching schedules to minimize the impact of periodic maintenance windows.

Azure Database for MySQL - Flexible Server provides a deeper overview of the Flexible Server deployment option.

Supported versions

Azure Database for MySQL - Flexible Server is based on MySQL Community Edition, using the InnoDB storage engine as the default. It supports all current major versions that are supported by the MySQL community. As of May 2021, these include:

- MySQL Version 5.7
- MySQL Version 8.0

Specific minor versions and bug fix releases will change over time, so it's best to check the Azure Database for MySQL version support policy for the latest details.

Scheduled maintenance

As a fully managed service, Azure Database for MySQL - Flexible Server periodically performs scheduled maintenance on your database servers. You can let the service automatically choose a day and a time window for scheduled maintenance, in which case it will select a one-hour window between 11pm and 7am in the Azure region in which your server resides. Alternately, you can specify a maintenance window by choosing a day of the week and a one-hour period within that day. Different maintenance windows can be configured for each Flexible Server deployment in your Azure subscription.

Either way, whether you specify a maintenance window or let the service choose one for you, the service will normally alert you five days before performing any scheduled maintenance, and then let you know when scheduled maintenance is started and completed. Normally, there are at least 30 days between successful scheduled maintenance events. However, if an emergency update is needed to patch a severe vulnerability, you may not receive five days' notice, and the critical update may be applied to your server even if a successful scheduled maintenance was performed within the past 30 days.

Scheduled maintenance in Azure Database for MySQL - Flexible Server provides more information on maintenance windows, including how to specify one and some things you may want to consider when doing so.

Connection libraries, drivers, and tools

Azure Database for MySQL - Flexible Server is compatible with all the drivers and tools that are supported and compatible with MySQL community edition.

Databases and servers

Azure Database for MySQL - Flexible Server exposes access and features at the *server* level, which is created within an Azure subscription and is the parent resource for *one or more individual databases*. It collocates the computing resources for those databases in a specified Azure region, provides a namespace for the databases, and serves as a connection endpoint for server and database access. A server also defines the scope for management policies that apply to those databases—including sign-in, firewall, users, roles, and configurations.

The online documentation for Azure Database for MySQL - Flexible Server provides more information on servers, including how to start/stop a Flexible Server instance and how to manage a server.

Server pricing tiers and resources

Under the Flexible Server deployment option, pricing for an Azure Database for MySQL server is determined by the configuration of resources that you specify for that server, including the pricing tier, number of vCores, and amount of storage. Create a single database under the server, and that database has exclusive access to those resources; create multiple databases under the server, and those resources are shared.

We recently introduced a free, 12-month offer for the Flexible Server deployment option, which you can use to develop and test your applications and run small production workloads without additional costs. Available with an Azure free account, this offer provides up to 750 hours of compute time and 32GB of storage per month for the first 12 months.

Pricing tiers

You can create a server in one of three pricing tiers, which are differentiated by the amount of compute (vCores) that can be provisioned, the amount of memory per vCore, and the technology used for data storage. In general:

- **Burstable** is good for workloads that don't continuously need full compute capacity.
- **General purpose** is good for workloads that require balanced compute and memory, with scalable I/O—including web apps, mobile apps, and many other enterprise apps. This is our most popular option and works well for most scenarios.
- **Memory optimized** is good for workloads that require in-memory performance for faster transaction processing and greater concurrency—including real-time data processing and high-performance transactional or analytical apps.

After you create a server, you can adjust the number of vCores, the hardware generation, and the pricing tier up or down, within seconds. Similarly, you can increase the amount of storage or adjust your backup retention period up or down, without application downtime.

Compute and storage options in Azure Database for MySQL - Flexible Server provides more information on pricing tiers, including detailed specifications for each SKU within each pricing tier. Limits in Azure Database for MySQL - Flexible Server covers functional limitations for the service and associated limits for each SKU, such as the maximum number of concurrent connections.

For current pricing information, see the Azure Database for MySQL pricing page. When you configure a server, the Azure portal shows the monthly cost based on the options you've selected. If you don't have an Azure subscription, you can use the Azure pricing calculator to get an estimated cost—just select Add items, expand the Databases category, choose Azure Database for MySQL, then customize the options.

Reserved Capacity

Azure Database for MySQL - Flexible Server supports reserved capacity pricing, which lets you reserve compute power for a period of one year or three years. By taking advantage of it, you can save up to 60 percent compared to the regular, pay-as-you-go payment options.

Prepay for Azure Database for MySQL compute resources with reserved capacity provides more information on reserved capacity, including how to determine the right server size and how to purchase reserved capacity. (This article also applies to Flexible Server.) A pricing calculator can be found on the Azure Database for MySQL pricing page.

Storage

When you provision a server, you'll also need to provision the amount of storage capacity available to it. This storage is used for the database files, temporary files, transaction logs, and MySQL server logs. You can increase storage capacity after you create a server, but provisioned storage can't be reduced.

The total storage you provision also defines the I/O capacity available to your server in terms of I/O operations per second (IOPS). Azure Database for MySQL - Flexible Server currently supports up to 16 TB of storage and up to 20,000 IOPS.

It's worth noting that IOPS are also constrained by your compute tier, including underlying VM type and number of vCores. Even though you can select any storage size independent of the server type, you may not be able to use all the IOPS that the storage can provide, especially when you choose a server with a small number of vCores. You can monitor your I/O consumption in the Azure portal or by using Azure CLI commands.

The online documentation for Azure Database for MySQL - Flexible Server provides more information on provisioning storage.

Networking

When you create a server, you need to choose from one of two networking options. Your selection can't be changed after the server is created. Options include:

- **Private access (VNet integration)**, which lets you deploy your database server into an Azure Virtual Network that can be used to provide private and secure network communications.
- **Public access (allowed IP addresses)**, which lets you access your database server through a public endpoint—a publicly resolvable DNS address. With this networking option, you'll use firewall rules to define the range of IP addresses that are permitted to access your server.

The online documentation for Azure Database for MySQL - Flexible Server provides more information on each networking option, including guidance on choosing one and additional detail on how each option works.

Server management

After you create a new server, you'll be ready to manage it. Manage Azure Database for MySQL - Flexible Server using the Azure portal walks you through the basics on signing into the Azure portal, creating a server, scaling compute and storage, updating your admin password, and deleting a server. Manage an Azure Database for MySQL - Flexible Server using the Azure CLI covers how to perform similar tasks by using the Azure CLI.

Stop/start/restart a server

Azure Database for MySQL - Flexible Server lets you stop a running server, then start it again later. When you stop the server, it remains in that state for the next 7 days. If you do not manually start it during this time, the server will automatically be started at the end of 7 days, after which you can choose to stop it again.

With this functionality, you can stop the database server when not in use (such as during non-work hours) and start it when it needs to be back online. Stop/start could also be helpful when using Azure Database for MySQL for development or test scenarios, enabling you to save on costs by turning off the server compute when not in use.

Server parameters

The MySQL engine provides many server variables/parameters that can be used to configure and tune engine behavior. Some parameters can be set dynamically during runtime while others are "static", requiring a server restart in order to apply. Server parameters in Azure Database for MySQL - Flexible Server provides additional detail on server variables and configurable server parameters.

Limitations

Limitations in Azure Database for MySQL - Flexible Server describes associated limits for Azure Database for MySQL - Flexible Server, including supported storage engines, privilege and data manipulation statement support, functional limits, and current known related issues.

Business continuity

Azure Database for MySQL - Flexible Server provides several features that you can use to help ensure business continuity and prevent data loss. Overview of business continuity with Azure Database for MySQL - Flexible Server introduces these business continuity features and how you can put them to use, including an overview of recovery scenarios and how they can affect your Recovery Time Objective and Recovery Point Objective.

Backup and restore

Backups are an essential part of any business continuity strategy, providing a means to help protect data from accidental corruption or deletion. Azure Database for MySQL - Flexible Server automatically backs up your server and retains those backups for a default of 7 days, which you can optionally configure for anywhere from 1 to 35 days. When you restore a backup, within the retention period, you can specify the date and time to which you want to restore your data.

All backups in Azure Database for MySQL - Flexible Server are encrypted using AES 256-bit encryption and are stored in local redundant storage within an Azure region. The first snapshot is scheduled immediately after the server is created, after which daily differential snapshots are performed. Transaction log backups occur every five minutes, with the combination of data backups and transaction log backups enabling you to restore a server to any point-in-time within your configured backup retention period—a capability that's often-called point-in-time-restore.

Backup and restore in Azure Database for MySQL - Flexible Server provides more information on how backup and restore work for Flexible Server deployments, including additional detail on backup processes, backup retention, backup storage costs, and point-in-time restore.

Zone-redundant high availability

Azure Database for MySQL - Flexible Server is designed to support high availability across multiple availability zones—called zone-redundant high availability. High availability concepts in Azure Database for MySQL - Flexible Server provides more information on the high availability features for Flexible Server

deployments, including additional detail on steady-state operations, failover processes for planned and unplanned downtime, point-in-time restore, and zone-redundant high availability features and limitations.

Security

Many of the security features for Azure Database for MySQL - Flexible Server are currently in development. Our goal is to at least achieve feature-parity with the security features in the Single Server deployment option. As new security features are added to Flexible Server, we'll update the online documentation for Azure Database for MySQL, post about it in Azure updates, and may write about it on the Azure Database for MySQL blog.

Monitoring and logging

Monitoring your servers not only helps you know when troubleshooting might be needed, but it also gives you the data needed for performance tuning. Azure Database for MySQL - Flexible Server provides several built-in tools for monitoring and tuning, including resource usage metrics that are captured at one-minute intervals and stored for 30 days. You can also enable logging on your server, and then send those logs to Azure Monitor.

The online documentation for Azure Database for MySQL - Flexible Server provides more information on its monitoring and logging features.

Slow query logs

In Azure Database for MySQL - Flexible Server, the slow query log can be used to identify performance bottlenecks for troubleshooting. By default, the slow query log is disabled. Slow query logs in Azure Database for MySQL - Flexible Server covers how to configure slow query logs, how to access them, and how to analyze them by using Azure Monitor.

Audit logs

The Azure Database for MySQL - Flexible Server audit log can be used to track database-level activity—as is often required for compliance purposes. Audit logs are integrated with Azure Monitor diagnostic logs, enabling you to retain logs in Azure Storage, perform analysis with Azure Monitor Logs, and integrate with third-party tools via Event Hubs. By default, audit logging is disabled.

Audit logs in Azure Database for MySQL - Flexible Server covers how to configure audit logging, how to access audit logs, and how to analyze audit logs by using Azure Monitor.

Read replication

Read replicas can help improve the performance and scale of read-intensive workloads, which can be isolated to the replicas while write workloads are directed to the master. A common scenario is to have business intelligence and analytical workloads use the read replica as the data source for reporting. Azure Database for MySQL - Flexible Server supports up to 10 read replicas.

Like with data-in replication, read replicas are updated asynchronously via the native replication technology in the MySQL engine. From a provisioning perspective, read replicas are considered new servers, and you manage them similarly to how you manage regular Azure Database for MySQL servers. For each replica, you're billed for the provisioned compute and storage.

Read replicas in Azure Database for MySQL - Flexible Server covers when you might want to use a read replica, how it works, and how to implement and monitor it.

Quickstarts

Our quickstarts can help you get started with Azure Database for MySQL, such as creating a server then connecting to it and querying it. You'll find the first one here, with the rest listed immediately below it in the navigation pane.

To use the quickstarts you'll need an Azure subscription. If you don't have an Azure subscription, you can create an Azure free account before you begin.

Use cases

At a high level, the usage scenarios for Azure Database for MySQL can be divided into three types: migrations from another MySQL database, deployment of prepackaged apps that use MySQL, and new cloud-native apps.

Migrations

Azure Database for MySQL is a frequent target for migrations from an existing MySQL database, which might be running on-premises, in an IaaS environment on a hosted VM, or on another cloud provider's MySQL service. Regardless of where your MySQL database runs today, by migrating to Azure Database for MySQL, you'll be able to quickly modernize your apps and shift your focus from database management to app development. GeekWire and Minecraft Realms are two examples of migrations from MySQL to Azure Database for MySQL.

Migrations from MySQL to Azure Database for MySQL are often done using the Azure Database Migration Service, making the process fast and seamless with minimal downtime. Other ways to migrate from MySQL to Azure Database for MySQL include using dump and restore, dbForge Studio for MySQL, or MySQL Workbench. Finally, if you're using Amazon RDS for MySQL, the MySQL Workbench Migration Wizard provides an easy and convenient way to move your databases to Azure Database for MySQL.

Azure Database Migration Guides provide prescriptive guidance on performing migrations, case studies on Microsoft customers who have completed migrations, and a list of Microsoft partners for assistance with migrations. There's also a MySQL on-premises to Azure Database for MySQL Migration Guide.

Prepackaged app deployments

MySQL is a popular database for prepackaged apps like Alfresco, Drupal, Magento, Moodle, Shopify, and WordPress. On Azure, such deployments often include the use of Azure App Service or Azure Kubernetes Service (AKS) for application hosting, with Azure Database for MySQL used as the data tier.

Deployments of prepackaged apps on Azure using such architectures offer several advantages, beginning with the ease of management and administration provided by fully managed PaaS services. Deployment, monitoring, scaling, and other essential tasks are all made better through the tight integration between Azure application hosting environments and Azure Database for MySQL—including easy integration with other Azure services to extend your app's functionality and the ability to manage all tiers of your app in a single place, using a common set of tools.

Cloud-native apps

Azure Database for MySQL is often chosen as the data tier for new cloud-native apps. Built from the ground up to take advantage of cloud scale and performance, such apps are typically based on a microservices architecture, make extensive use of fully managed services, and take advantage of continuous delivery to improve reliability and accelerate time to market. Regardless of what the app does, building it on Azure Database for MySQL means you'll benefit from cloud scale, built-in intelligence, advanced security and compliance features, built-in high availability, and integration with the broader Azure ecosystem.

HSBC is an example of a Microsoft customer who built a cloud-native app using Azure Database for MySQL together and other Azure services to facilitate fast and secure payment processing.

Building applications on Azure

In the preceding pages, we've covered why Azure Database for MySQL is a great database service. However, rarely does a database exist in isolation. More likely than not, it's purpose is to function as a data store for one or more apps, which means those apps need to work well with it, efficiently storing and retrieving data. Similarly, those apps need to be developed, deployed, monitored, and scaled alongside your database. If all the parts of your app don't work well together throughout the application lifecycle, you're leaving a lot of the benefits of fully managed PaaS services off-the-table.

So what does it take to achieve this? At a minimum, it requires a fully managed application hosting environment that's as rich and robust as your database—albeit in slightly different ways. For development, you'll want to use the same familiar tools, with support for your desired development languages and frameworks. For deployment, you'll want to integrate changes to your database into your CI/CD pipelines, test and staging environments, and more.

In production, you'll want your app collocated with your database to optimize performance. You'll want to monitor your app hosting and development environments using an integrated set of tools, knowing which tier needs to scale and why. Finally, when scalability is needed, you'll want to be able to scale your application hosting environment as easily as you can scale your database, using the same familiar mechanisms. All of these benefits are reasons why you should consider hosting your app on Azure as well.

Choosing an app hosting environment

First, let's examine some of the most popular application hosting environments on Azure and what they can bring to the table alongside Azure Database for MySQL.

Azure App Service

Customers often use Azure Database for MySQL together with Azure App Service, a fully managed HTTP-based service for building and hosting web apps, mobile back ends, and RESTful APIs—all without having to manage infrastructure. You can work in the programming language of your choice, including .NET, .NET Core, Java, Ruby, Node.js, PHP, and Python. App Service provides built-in auto-scaling and high availability, supports both Windows and Linux, and enables automated deployments from GitHub, Azure DevOps, or any Git repo.

With App Service, you pay only for the Azure compute resources you use, which are determined by the App Service plan that you choose. For apps that require superior levels of scalability, you can run in a fully dedicated and isolated App Service Environment. Along with Azure Spring Cloud (discussed next), App Service is a good choice for companies that want to migrate their on-premises databases off of Oracle and need a suitable application hosting environment.

If you'd like to try out using App Service and Azure Database for MySQL for yourself, Create an App Services Web App in a VNet with Azure Database for MySQL - Flexible Server provides step-by-step instructions for doing so. Build a PHP (Laravel) app in Azure App Service using Azure Database for MySQL - Flexible Server is another tutorial to try.

Azure Spring Cloud

Azure Spring Cloud makes it easy to deploy Java Spring Boot and ASP.NET Steeltoe apps to Azure without any code changes. The Spring Cloud service manages infrastructure so that you can focus on your code, with lifecycle management features that include monitoring and diagnostics, configuration management, service discovery, CI/CD integration, and blue-green deployments.

You can migrate existing Spring apps to Spring Cloud to manage cloud scaling and costs, or you can modernize apps with Spring Cloud patterns to improve agility and speed of delivery. Either way, Spring Cloud is a great way to run Java at cloud scale, without complicated infrastructure. You'll also benefit from rapid development and deployment, without containerization dependencies, along with efficient and effortless monitoring of your production workloads.

Use Spring Data JPA with Azure Database for MySQL demonstrates how to use the Spring Data Java Persistence API to store and retrieve information in Azure Database for MySQL.

Azure Kubernetes Service

Cloud-native apps that use Azure Database for MySQL are often deployed on Azure Kubernetes Service (AKS), a fully managed Kubernetes environment, which provides a great way to unite your development and operations teams. AKS supports integrated CI/CD along with enterprise-grade security and governance, providing a great way to rapidly build, deliver, and scale your apps with confidence.

AKS achieves this by offloading the complexity and operational overhead of managing a Kubernetes cluster to Azure, handling critical tasks like health monitoring and maintenance. Upon cluster deployment, the Kubernetes master and all nodes are deployed and configured for you. Advanced networking, Azure Active Directory (Azure AD) integration, monitoring, and other features also can be configured during the deployment process. You only manage and maintain (and pay for) the agent nodes within your cluster, not the masters, which are handled by Azure.

Best practices for Azure Kubernetes Service and Azure Database for MySQL covers some options to consider when using AKS with Azure Database for MySQL, including the use of accelerated networking and premium file shares on Azure. If you'd like to try out using AKS and Azure Database for MySQL for yourself, Deploy a WordPress app on AKS with Azure Database for MySQL - Flexible Server provides step-by-step instructions for doing so.

Designing an application architecture

Before building your app, you'll want to review our guidance for architecting solutions on Azure using established patterns and best practices. The Azure Architecture Center provides a wealth of information in this area—from designing for the cloud to choosing the right technology and optimizing for your workload.

Within the Azure Architecture Center, you'll find the Azure Application Architecture Guide, which presents a structured approach for designing applications on Azure that are scalable, secure, resilient, and highly available. It's based on proven practices that Microsoft has learned from customer engagements.

The Azure Architecture Center is also full of ideas for solutions that use Azure Database for MySQL, including digital marketing, finance management, gaming, and intelligent apps.

Common app development best practices

Regardless of which app hosting environment you use, you'll want to use established best practices for connecting your app to its database. Best practices for building an application with Azure Database for MySQL discusses configuration of application and database resources, things you can do to boost performance and resiliency, database deployment options, and considerations to keep in mind when building your database schema and queries.

GitHub actions for database deployment

GitHub provides many benefits for application developers, including comprehensive source control and the ability to track changes across multiple versions of the code base. Database schemas can evolve similarly,

with multiple versions across development, test, and production environments—and thus require similar robust source control capabilities.

GitHub Actions for Azure can be used to automate database development workflows within GitHub, including deployment of database updates to Azure Database for MySQL. Use GitHub Actions to connect to Azure Database for MySQL provides an overview of how it works, beginning with the workflow file that defines the various steps and parameters. The article also covers how to generate deployment credentials, specify a connection string, configure GitHub secrets, add your workflow, and review your deployment.

Additional resources

Beyond the deployment option-specific resources provided above, here are some additional resources that can help you get started with Azure Database for MySQL.

Samples

Azure CLI samples for Azure Database for MySQL provides links to sample Azure CLI scripts for Azure Database for MySQL—including scripts to create a server and firewall rule, scale a server, change server configurations, restore a server, and enable and download server logs.

References

Azure CLI reference for Azure Database for MySQL provides a list of Azure CLI commands for managing Azure Database for MySQL servers, with links to more information on each command.

Azure Database for MySQL REST API reference covers the REST operations you can use with Azure Database for MySQL. You can use these REST operations to create, delete, manage, and list servers, server configurations, databases, server logs, and firewall rules, and to list all available REST operations.

Azure Database for MySQL management stored procedures covers stored procedures to help manage your Azure Database for MySQL server, including stored procedures for data-in replication and other tasks.

Other useful links

- Online documentation for Azure Database for MySQL
- Pricing information for Azure Database for MySQL
- Azure pricing calculator for Azure Database for MySQL
- Azure Database for MySQL blog
- Microsoft Learn for online training courses
- Tech Community discussion forums for Azure Database for MySQL
- Azure Tips and Tricks for helpful tips for developers
- Azure service health dashboard
- Azure product availability by region
- Azure Community Support
- Azure Blog and Updates
- List of videos on Azure Database for MySQL

Conclusion

Microsoft Azure provides an end-to-end platform for open-source applications, including a fully managed MySQL database service that automatically handles all maintenance, patching, and updates to let you focus on application innovation. Azure Database for MySQL can be provisioned in minutes and scaled in seconds, with a simple yet flexible pricing model that includes automatic patching and backups, monitoring tools, essential security features, and more. And because it's a part of Azure, when you choose Azure Database for MySQL, you'll have everything you need to build great apps and deploy them across the globe.

With Azure Database for MySQL, you'll get support for the latest community versions of MySQL, a simplified developer experience, and AI-powered performance optimization. You'll also get a 99.99-percent service level agreement, advanced data security and compliance features, and multiple pricing options—including the ability to pay-as-you-go or to save money with Reserved Capacity. The new Flexible Server deployment option provides even more, including custom maintenance windows, more configuration parameters, zone-redundant high availability, and additional cost optimization controls.

Sign up for an Azure free account and get started with Azure Database for MySQL today. If you already have an Azure account, you can create a database on the Azure portal