

MYSQL @ AZURE

<https://vikasrajput.github.io>

MySQL is an [OSS](#) RDBMS, created by MySQL AB and now owned by Oracle®. Refer [Wikipedia](#) for MySQL history. Refer Stack Overflow [Dev Survey](#) for its Global DB Ranking (2022: #1).

This document (a) presents MySQL/Oracle/MSFT documentation pointers for Architects & other Tech roles (b) isn't warranted in any way and (c) not a replacement of vendor documentation.

HOW TO USE THIS FIELD MAP?

If new to MySQL, you can start with [Foundations](#), then check [References](#), and finally step through [WAF Pillars](#). Alternatively, feel free to jump through [Dev/Admin](#) or specific WAF [pillars](#).

FOUNDATIONS

[What's New in MySQL 8.0?](#), [Storage Engine Architecture](#)

Unlike most RDBMSs, MySQL offers option of DB Engines (default is innodb). Other engines aren't wasteful and have real-world applications.

[InnoDB Engine: Architecture, On-Disk Structures, Locking & Transaction, Compression, Disk management, Config, Params memcached plugin, Cluster, ClusterSet, ReplicaSet](#)

[Alternative Engines: Federated, MyISAM, Merge, Archive, CSV, Blackhole, Memory, NDB Cluster](#)

DEV

Not the absolute basic stuff when it comes to DB Dev.

[Objects, Functions, Partition, Optimization, Document Store INFORMATION SCHEMA, PERFORMANCE SCHEMA MySQL Plugins: Thread Pool, DDL, Query rewrite, Clone](#)

ADMIN

All these guides are very detailed. Refer or scan through, as preferred. [Secure Deployment](#), [*nix Guide](#), [Windows Guide](#), [Port Map MySQL Server Admin](#) + [Security](#) + [Backup](#) + [Replication Loadable Functions](#), [Multi-Instance](#), [Options and Variable Ref](#)

MYSQL TOOLS

Tools below are from MSFT or Oracle. 3rd Party products not included.

[MySQL Workbench](#): for [Dev](#), [Design](#), [Migrate](#), [Performance MySQL for VS](#), [MySQL Shell for VS Code](#), [MySQL Shell](#)

AZURE DATABASE FOR MYSQL (PAAS)

While you can host a self-managed MySQL VM (know its [merits and challenges](#)), MSFT has [Azure DB for MySQL](#) PaaS offering – with [Single](#) and [Flexible](#) Server modes, using MySQL Community Edition.

Check out deployment options for [Single vs Flexible vs MySQL VM](#). Compared to Self-managed MySQL, there are [Limits on Single](#) and [Limits on Flexible](#) Server.

Check what's new in [Single Server](#) and [Flexible Server](#). Plus pricing and service tiers for [Single Server](#) and [Flexible Server](#).

Take a note of [Version Support Policy](#) (to align with MySQL Community releases) and [Regional Availability](#) guidance.

REFERENCES

- [MySQL \(repo\) @ GitHub](#)
- [OSS @ Microsoft](#). [OSS Guidance for MSFT Teams](#)
- [Azure Database for MySQL](#)
 - [Reference Architecture](#), [MSFT Community Blog](#)
 - [MSFT Learning Path](#), [Developer Essentials Series](#)
 - [Azure Latest Updates](#), [QuickStart Templates](#)
- [Azure CLI for MySQL](#)
- [MySQL Dev Learning Journey \(MSFT e-book\)](#)
- Deploy Azure MySQL using [ARM](#), [Bicep](#), [PS](#), [Terraform](#)
- [MySQL Migration Sheet](#), [MySQL Migration Guide](#).
- [Economic Value of Migrating MySQL to Azure \(report\)](#)

SECURITY

Security is the most critical WAF pillar. Let's get to it and categorize Security measures as Identity Management, Access Management, Surface Management, Protect Data and finally, Monitor/Audit.

Identity: [AAD](#) (single only), [Managed Identity](#) (single). [azure_superuser](#).

Access: [Restrictions vs MySQL](#). [Gateway Architecture](#).

Surface: Public Access - [single](#), [flexible](#). [Private Link](#) (single). [VNet Integration](#) (flexible)

Encrypt: At Rest ([Customer Managed Key](#), [Microsoft Managed Keys](#)). In Transit ([SSL + TLS](#)).

Monitor: [Microsoft Defender](#) (single). [Audit Logs](#). [Azure Policy](#).

Azure [Security Benchmark](#). [Security Baseline](#) for [MySQL](#). Optional: [ProxySQL](#).

COST OPTIMIZATION

After Security, generally Cost holds the next priority for clients. Though it's very difficult to talk about this pillar in isolation. Firstly, every Org needs to establish a baseline Consumption (capacity, cost) Forecast, Budget & Ownership and then be able to Track and Report consumption. And then, we need to approach Architecture as such to elevate Demand Management (e.g., throttle) and Supply Management (e.g., scale).

[Azure Advisor](#). [Cost Management](#). Understand pricing for [Single Server](#), [Flexible Server](#). [Auto Start/Stop](#). [Burstable SKU](#) (flexible) + [Storage Scaling](#).

OPERATIONS EXCELLENCE

Ops Excellence proves the real-world agility and maturity of a business in managing Business Systems. At its core, Ops Excellence is all about how SDLC is managed, underscoring practices around Development, Deployment and Operation with Security, Monitoring and Automation embedded every step of the way.

[Azure DevOps for Azure MySQL](#). ARM Templates ([how-to](#), [repo](#)). [MySQL Test Framework](#). Use right [Azure MySQL Drivers](#), [Migration Tool](#). Considerations when using [AKS with MySQL](#).

Develop for [Resilient Connections](#). Resolve for [Connectivity](#), [DB Corruption](#), [Replication Latency](#). System Maintenance – [Single](#) (system controlled), [Flexible](#) (customer controlled). Follow [Azure MySQL Operations Best Practices](#).

RELIABILITY

Business should take lead on this and define must-have or preferred Availability and Recovery Metrics (SLA, MTTR, MTBF etc). This should inform Architecture – outlining the Failure Management (Failure Scenarios, Prevention measures, Recovery measures) and complemented by Recovery-Tests and Uptime Monitoring.

Single Server

[Backup Recovery](#): [Frequency](#), [Retention](#), [Redundancy](#), [Cost](#), [Point in Time](#) vs [Geo](#) Restore. Replication: [Data-in](#) (prem/multi-cloud sync) vs [Read Replicas \(Universal, Paired Regions\)](#) HA Options: Protection against [Planned Outages](#) and [Unplanned Outages](#).

Flexible Server

[Backup Recovery](#): [Frequency](#), [Retention](#), [Redundancy](#), [Cost](#), [Point in Time](#) vs [Geo](#) Restore. Replication: [Data-in](#) (prem/multi-cloud sync) vs [Read Replicas](#).

Monitoring: [Monitor](#), [App Insights](#), [Service Health](#), [Resource Health](#), [Policy](#), [Advisor](#), [Alerts](#) Reference Architecture – [Hybrid Availability and Performance Monitoring](#)

PERFORMANCE

Performance is an interesting pillar – for it can be negatively impacted by almost all other pillars, but positive impact must be weaved in! Low-cost commitment or subpar operations or security measures can limit it. For this, establish Monitoring (workload, resources, baseline), Design for Performance, and Remediate Contention.

Monitoring. Most critical input in performance tuning is Baseline – workload & consumption. Workload expectation is business driven, and Consumption Baseline should be established (and adjusted) over a period. [Monitor](#) ([audit logs](#), [slow query logs](#), [workbooks](#)), [App Insights](#), [Metrics](#), [Query Store \(Perf Insights, Perf Recommendation\)](#), [Advisor](#). [Core MySQL Performance Schema \(benchmark\)](#).

Design for Performance: Network ([Proximity](#), [Accelerate](#), [Connection](#), [Pooling](#)). IO ([Partition](#), [Segregate reads](#), [Scale IO](#)). Memory ([Right Size](#), [BufferPool Warmup](#)), CPU ([Size & Scale](#))

Remediate: [Query](#), [CPU](#), [Memory](#), Network ([connect](#), [replica](#)), [I/O](#) ([partition](#), [index](#), [IOPS](#))