Migrating MySQL On-Premises to Azure Database for MySQL

Technical white paper

**Published:** October 2020

**Applies to:** Azure Database for MySQL

Copyright

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. This content was developed prior to the product or service’ release and as such, we cannot guarantee that all details included herein will be exactly as what is found in the shipping product. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication. The information represents the product or service at the time this document was shared and should be used for planning purposes only.

This white paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED, OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in, or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property. Information subject to change at any time without prior notice.

Microsoft, Active Directory, Azure, Bing, Excel, Power BI, SharePoint, Silverlight, SQL Server, Visual Studio, Windows, and Windows Server are trademarks of the Microsoft group of companies.

All other trademarks are property of their respective owners.

© 2020 Microsoft Corporation. All rights reserved.

[Introduction 8](#_Toc53058557)

[MySQL 8](#_Toc53058558)

[Azure Database for MySQL 8](#_Toc53058559)

[Case Study 10](#_Toc53058560)

[Overview 10](#_Toc53058561)

[Assessment 11](#_Toc53058562)

[Limitations 11](#_Toc53058563)

[MySQL Versions 12](#_Toc53058564)

[Source Systems 14](#_Toc53058565)

[Tools 15](#_Toc53058566)

[WWI Case Study 18](#_Toc53058567)

[Assessment Checklist 18](#_Toc53058568)

[Planning 19](#_Toc53058569)

[Landing Zone 19](#_Toc53058570)

[Networking 19](#_Toc53058571)

[SSL/TLS Connectivity 20](#_Toc53058572)

[WWI Case Study 21](#_Toc53058573)

[Planning Checklist 21](#_Toc53058574)

[Migration Methods 22](#_Toc53058575)

[MySQL Workbench 22](#_Toc53058576)

[Dump and restore (mysqldump) 22](#_Toc53058577)

[mydumper and myloader 23](#_Toc53058578)

[Data-in replication (binlog) 23](#_Toc53058579)

[Azure Data Migration Service (DMS) 24](#_Toc53058580)

[Fastest/Minimum Downtime Migration 24](#_Toc53058581)

[Decision Table 25](#_Toc53058582)

[WWI Case Study 25](#_Toc53058583)

[Migration Methods Checklist 25](#_Toc53058584)

[Test Plans 26](#_Toc53058585)

[Overview 26](#_Toc53058586)

[Sample Queries 26](#_Toc53058587)

[Rollback Strategies 29](#_Toc53058588)

[WWI Case Study 29](#_Toc53058589)

[Checklist 29](#_Toc53058590)

[Performance Baselines 30](#_Toc53058591)

[Tools 30](#_Toc53058592)

[Server Parameters 30](#_Toc53058593)

[WWI Case Study 32](#_Toc53058594)

[Data Migration 33](#_Toc53058595)

[Back up the database 33](#_Toc53058596)

[Offline vs. Online 33](#_Toc53058597)

[Performance recommendations 35](#_Toc53058598)

[Performing the Migration 36](#_Toc53058599)

[Common Steps 36](#_Toc53058600)

[Migrate to latest MySQL version 36](#_Toc53058601)

[WWI Case Study 37](#_Toc53058602)

[Database Objects 37](#_Toc53058603)

[Users and permissions 41](#_Toc53058604)

[Execute migration 41](#_Toc53058605)

[Data Migration Checklist 42](#_Toc53058606)

[Data Migration with MySQL Workbench 43](#_Toc53058607)

[Setup 43](#_Toc53058608)

[Configuring Server Parameters (Source) 43](#_Toc53058609)

[Configuring Server Parameters (Target) 43](#_Toc53058610)

[Data 44](#_Toc53058611)

[Update Applications to support SSL 46](#_Toc53058612)

[Revert Server Parameters 47](#_Toc53058613)

[Change Connection String for the Java API 47](#_Toc53058614)

[Post Migration Management 48](#_Toc53058615)

[Monitoring and Alerts 48](#_Toc53058616)

[Server Parameters 49](#_Toc53058617)

[PowerShell Module 49](#_Toc53058618)

[Azure Database for MySQL Upgrade Process 49](#_Toc53058619)

[WWI Case Study 50](#_Toc53058620)

[Management Checklist 50](#_Toc53058621)

[Optimization 51](#_Toc53058622)

[Monitoring Hardware and Query Performance 51](#_Toc53058623)

[Query Performance Insight 51](#_Toc53058624)

[Upgrading the Tier 51](#_Toc53058625)

[Scale the Server 52](#_Toc53058626)

[Moving Regions 52](#_Toc53058627)

[WWI Case Study 52](#_Toc53058628)

[Optimization Checklist 52](#_Toc53058629)

[Business Continuity and Disaster Recovery (BCDR) 53](#_Toc53058630)

[Backup and Restore 53](#_Toc53058631)

[Read Replicas 53](#_Toc53058632)

[Deleted Servers 54](#_Toc53058633)

[Regional Failure 54](#_Toc53058634)

[WWI Case Study 54](#_Toc53058635)

[BCDR Checklist 55](#_Toc53058636)

[Security 55](#_Toc53058637)

[Authentication 55](#_Toc53058638)

[Threat Protection 55](#_Toc53058639)

[Audit Logging 55](#_Toc53058640)

[Encryption 56](#_Toc53058641)

[Firewall 56](#_Toc53058642)

[Private Link 57](#_Toc53058643)

[Security Checklist 57](#_Toc53058644)

[Summary 57](#_Toc53058645)

[Questions and Feedback 58](#_Toc53058646)

[Find a partner to assist in migrating 58](#_Toc53058647)

[Appendix A: Environment Setup 58](#_Toc53058648)

[Deploy the ARM template 58](#_Toc53058649)

[Install MySQL 5.5 60](#_Toc53058650)

[Install MySQL Workbench 63](#_Toc53058651)

[Download artifacts 63](#_Toc53058652)

[Deploy the Database 64](#_Toc53058653)

[Configure Blob Data 66](#_Toc53058654)

[Setup Java and Maven 67](#_Toc53058655)

[Install Azure CLI 67](#_Toc53058656)

[Install NodeJS 68](#_Toc53058657)

[Install and Configure Visual Studio Code 68](#_Toc53058658)

[Configure the Web Application API 68](#_Toc53058659)

[Test the Web Application 70](#_Toc53058660)

[Configure the Web Application Client 70](#_Toc53058661)

[Deploy the Java Server Application to Azure 70](#_Toc53058662)

[Deploy the Angular Web Application to Azure 73](#_Toc53058663)

[Configure Network Security (Secure path) 74](#_Toc53058664)

[Appendix B: ARM Templates 74](#_Toc53058665)

[Secure 74](#_Toc53058666)

[Non-Secure 74](#_Toc53058667)

[Appendix C: Default server parameters MySQL 5.5 74](#_Toc53058668)

[Appendix D: Default server parameters Azure 82](#_Toc53058669)

# Introduction

This migration guide is designed to provide snackable and actionable information for MySQL customers and software integrators seeking to migrate MySQL workloads to [Azure Database for MySQL](https://docs.microsoft.com/en-us/azure/mysql/overview). This guide will give applicable knowledge that will apply to a majority of cases and provide guidance that will lead the successful planning and execution of a MySQL migration to Azure.

The process of moving existing databases and MySQL workloads into the cloud can present challenges with respect to the workload functionality and the connectivity of existing applications. The information presented throughout this guide offers helpful links and recommendations focusing on a successful migration and ensure workloads and applications continue operating as originally intended.

The information provided will center on a customer journey using the Microsoft [Cloud Adoption Framework](https://docs.microsoft.com/en-us/azure/cloud-adoption-framework/get-started/) to perform assessment, migration, and post-optimization activities for an Azure Database for MySQL environment.

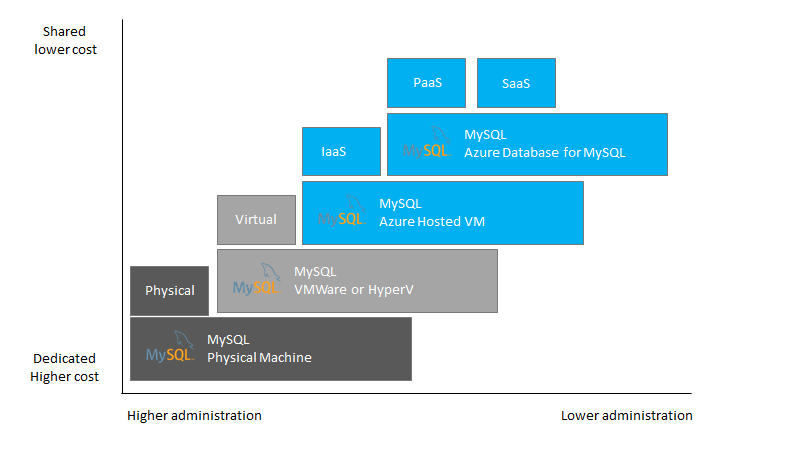
## MySQL

MySQL has a rich history in the open source community and has become very popular with corporations around the world for use in websites and other business critical applications. This guide will assist administrators who have been asked to scope, plan, and execute the migration. Administrators that are new to MySQL can also review the [MySQL Documentation](https://dev.mysql.com/doc/) for deeper information of the internal workings on MySQL. Additionally, this guide will also link to several reference articles through each of the sections to point you to helpful information and tutorials.

## Azure Database for MySQL

[Azure Database for MySQL](https://docs.microsoft.com/en-us/azure/mysql/overview) is a Platform as a Service (PaaS) offering by Microsoft, where the MySQL environment is fully managed. In this fully managed environment, the operating system and software updates are automatically applied, as well as the implementation of high availability and protection of the data.

In addition to the PaaS offering, it is still possible to run MySQL in Azure VMs. Reference the [Choose the right MySQL Server option in Azure](https://docs.microsoft.com/en-us/azure/mysql/select-right-deployment-type) article for more information on deciding what deployment type is most appropriate for the target data workload.



Comparison of MySQL environments.

This guide will focus entirely on migrating the on-premises MySQL workloads to the Platform as a Service Azure Database for MySQL offering due to its various advantages over Infrastructure as a Service (IaaS) such as scale-up and scale-out, pay-as-you-go, high availability, security and manageability features.

# Case Study

## Overview

World Wide Importers (WWI) is a San Francisco, California-based manufacturer and wholesale distributor of novelty goods. They began operations in 2002 and developed an effective business-to-business (B2B) model, selling the items they produce directly to retail customers throughout the United States. Its customers include specialty stores, supermarkets, computing stores, tourist attraction shops, and some individuals. This B2B model enables a streamlined distribution system of their products, allowing them to reduce costs and offer more competitive pricing on the items they manufacture. They also sell to other wholesalers via a network of agents who promote their products on WWI’s behalf.

Before launching into new areas, WWI wants to ensure its IT infrastructure can handle the expected growth. WWI currently hosts all its IT infrastructure on-premises at its corporate headquarters and believes moving these resources to the cloud enables future growth. They have tasked their CIO with overseeing the migration of their customer portal and the associated data workloads to the cloud.

WWI would like to continue to take advantage of the many advanced capabilities available in the cloud, and they are interested in migrating their databases and associated workloads into Azure. They want to do this quickly and without having to make any changes to their applications or databases. Initially, they plan on migrating their java-based customer portal web application and the associated MySQL databases and workloads to the cloud.

### Migration Goals

The primary goals for migrating their databases and associated SQL workloads to the cloud include:

* Improve their overall security posture with data at rest and in-transit.
* Enhance the high availability and disaster recovery (HA/DR) capabilities.
* Position the organization to leverage cloud-native capabilities and technologies such as point in time restore.
* Take advantage of administrative and performance optimizations features of Azure Database for MySQL.
* Create a scalable platform that they can use to expand their business into more geographic regions.
* Allow for enhanced compliance with various legal requirements where PII information is stored.

WWI used the [Cloud Adoption Framework (CAF)](https://docs.microsoft.com/azure/cloud-adoption-framework/) to educate their team on following best practices guidelines for cloud migration. Using CAF as a higher-level migration guide, WWI customized their migration into three main stages. Within each stage, they defined activities that needed to be addressed to ensure a successful lift and shift cloud migration.

These stages include:

|  |  |  |
| --- | --- | --- |
| Stage | Name | Activities |
| 1 | Pre-migration | Assessment, Planning, Migration Method Evaluation, Application Implications, Test Plans, Performance Baselines |
| 2 | Migration | Execute Migration, Execute Test Plans |
| 3 | Post-migration | Business Continuity, Disaster Recovery, Management, Security, Performance Optimization, Platform modernization |

WWI has several instances of MySQL running with varying versions ranging from 5.5 to 5.7. They would like to move their instances to the latest version as soon as possible but would like to ensure that their applications will still work if they move to the newer versions. They are comfortable with moving to the same version in the cloud and upgrading afterwards, but if they can accomplish two tasks at once, they would prefer that path.

They would also like to ensure that their data workloads are safe and available across multiple geographic regions in case of failure and are looking at the available configuration options.

WWI wants to start off with a simple application for the first migration, and then move to more business-critical applications in a later phase. This will provide the team with the knowledge and experience they need to prepare and plan for those future migrations.

# Assessment

Before jumping right into migrating a MySQL workload, there is a fair amount of due diligence that must be performed. This includes analyzing the data, hosting environment and application workloads to validate the Azure Landing zone is properly configured and prepared to host the soon to be migrated workloads.

## Limitations

Azure Database for MySQL is a fully supported version of the MySQL community edition running as a platform as a service. However, there are [some limitations](https://docs.microsoft.com/en-us/azure/mysql/concepts-limits) to become familiar with when doing an initial assessment.

The most important of which include:

* Storage engine support for InnoDB and Memory only
* Limited Privilege support (DBA, SUPER, DEFINER)
* Disabled data manipulation statements (SELECT ... INTO OUTFILE, LOAD DATA INFILE)
* Automatic major database migration (5.6 to 5.7, 5.7 to 8.0)
* When using [MySQL Server User-Defined Functions (UDFs)](https://dev.mysql.com/doc/refman/5.7/en/server-udfs.html), the only viable hosting option is Azure Hosted VMs, as there is no capability to upload the so or dll component to Azure Database for MySQL.

Many of the other items are simply operational aspects that administrators should become familiar with as part of the operational data workload lifecycle management. This guide will explore many of these operational aspects in the Post Migration Management section.

## MySQL Versions

MySQL has a rich history starting in 1995. Since then, it has since evolved into a widely used database management system. Azure Database for MySQL started with the support of MySQL version 5.6 and has continued to 5.7 and recently 8.0. For the latest on Azure Database for MySQL version support, reference [Supported Azure Database for MySQL server versions](https://docs.microsoft.com/en-us/azure/mysql/concepts-supported-versions). In the Post Migration Management section, we will review how upgrades (such as 5.7.20 to 5.7.21) are applied to the MySQL instances in Azure.

**Note:** The jump from 5.x to 8.0 was, in large part, due to the Oracle acquisition of MySQL. To read more about MySQL history, navigate to the [MySQL wiki page](https://en.wikipedia.org/wiki/MySQL).

Knowing the source MySQL version is important. The applications using the system may be using database objects and features that are specific to that version. Migrating a database to a lower version could cause compatibility issues and loss of functionality. It is also recommended the data and application instance are fully tested before migrating to a newer version as features introduced could break your application.

Examples that may influence the migration path and version:

* 5.6 : TIMESTAMP column for correct storage of milliseconds and full-text search
* 5.7 : Support for native JSON data type
* 8.0 : Support for NoSQL Document Store, atomic and crash safe DDL and JSON table functions

**Note:** MySQL 5.6 will lose general support in February of 2021. MySQL workloads will need to migrate to MySQL version of 5.7 or greater.

To check the MySQL server version:

SHOW VARIABLES LIKE "%version%";

### Database storage engines

Azure Database for MySQL only supports [InnoDB](https://dev.mysql.com/doc/refman/8.0/en/innodb-storage-engine.html) and [Memory](https://dev.mysql.com/doc/refman/8.0/en/memory-storage-engine.html) database storage engines. Other storage engines, like [MyISAM](https://dev.mysql.com/doc/refman/8.0/en/myisam-storage-engine.html), will need to be migrated to a supported storage engine. The difference between the MyISAM and InnoDB are the operational features and performance output. The higher-level tables and schema structure typically will not change between the engines, but the index and table column types may change for storage and performance reasons. Although InnoDB is known to have large data file sizes, these storage details are managed by the Azure Database for MySQL service.

#### Migrating from MyISAM to InnoDB

The MyISAM database and tables will need to be converted to InnoDB tables. After conversion, applications should be tested for compatibility and performance. In most cases, testing requires recreating the table and changing the target storage engine via DDL statements. It is unlikely this change will need to be performed during migration as it will occur during the schema creation in the Azure target. For more details, review the [converting tables developers documentation](https://dev.mysql.com/doc/refman/5.6/en/converting-tables-to-innodb.html) on the MySQL website.

To find useful table information, use this query:

SELECT  
 tab.table\_schema,  
 tab.table\_name,  
 tab.engine as engine\_type,  
 tab.auto\_increment,  
 tab.table\_rows,  
 tab.create\_time,  
 tab.update\_time,  
 tco.constraint\_type  
 FROM information\_schema.tables tab  
 LEFT JOIN information\_schema.table\_constraints tco  
 ON (tab.table\_schema = tco.table\_schema  
 AND tab.table\_name = tco.table\_name  
 )  
 WHERE  
 tab.table\_schema NOT IN ('mysql', 'information\_schema', 'performance\_schema', 'sys')  
 AND tab.table\_type = 'BASE TABLE';

**Note:** Running query against INFORMATION\_SCHEMA across multiple databases might impact performance. Run during low usage periods.

To convert a table from MyISAM to InnoDB, run the following:

ALTER TABLE {table\_name} ENGINE=InnoDB;

**Note:** This conversion approach will cause the table to lock, and all applications will wait until the operation is complete. The table locking will make the database appear offline for a short period of time.

Alternatively, the table can be created with the same structure, but with a different storage engine. Once created, copy the rows into the new table:

INSERT INTO {table\_name} SELECT \* FROM {myisam\_table} ORDER BY {primary\_key\_columns}

**Note:** For this approach to be successful, the original table would need to be deleted, then the new table renamed. This task will cause a short downtime period.

### Database data and objects

Data is one component of database migration. The database supporting objects need to be migrated and validated to ensure the applications will continue to run reliably.

Here is a list of items you should inventory before and after the migration:

* Tables (schema)
* Primary keys, foreign keys
* Indexes
* Functions
* Procedures
* Triggers
* Views

### User Defined Functions

MySQL allows for the usage of functions that call external code. Unfortunately, data workloads using User Defined Functions (UDFs) with external code cannot be migrated to Azure Database for MySQL. The required MySQL function's backing so or dll code cannot be uploaded to the Azure server.

Run the following query to find any UDFs that may be installed:

SELECT \* FROM mysql.func;

## Source Systems

The amount of migration preparation can vary depending on the source system and its location. In addition to the database objects, consider how to get the data from the source system to the target system. Migrating data can become challenging when there are firewalls and other networking components in between the source and target.

Additionally, moving data over the Internet can be slower than using dedicated circuits to Azure. When moving many gigabytes, terabytes, and petabytes of data, consider setting up an [ExpressRoute](https://docs.microsoft.com/en-us/azure/expressroute/expressroute-introduction) connection between the source network and the Azure network.

If ExpressRoute is already present, it is likely that connection is being used by other applications. Performing a migration over an existing route can cause strain on the network throughput and potentially cause a considerable performance hit for both the migration and other applications using the network.

Lastly, disk space must be evaluated. When exporting a very large database, consider the size of the data. Ensure the system where the tool is running, and ultimately the export location, has enough disk space to perform the export operation.

### Cloud Providers

Migrating databases from cloud services providers such as Amazon Web Services (AWS) may require an extra networking configuration steps in order to access the cloud hosted MySQL instances. Migration tools, like Data Migration Services, require access from outside IP ranges and may be otherwise blocked.

### On-premises

Like cloud providers, if the MySQL data workload is behind firewalls or other network security layers, a path will need to be created between the on-premises instance and Azure Database for MySQL.

## Tools

Many tools and methods can be used to assess the MySQL data workloads and environments. Each tool will provide a different set of assessment and migration features and functionality. As part of this guide, we will review the most commonly used tools for assessing MySQL data workloads.

#### Azure Migrate

Although [Azure Migrate](https://docs.microsoft.com/en-us/azure/migrate/migrate-services-overview) does not support migrating MySQL database workloads directly, it can be used when administrators are unsure of what users and applications are consuming the data, whether hosted in a virtual or hardware-based machine. [Dependency analysis](https://docs.microsoft.com/en-us/azure/migrate/concepts-dependency-visualization) can be accomplished by installing and running the monitoring agent on the machine hosting the MySQL workload. The agent will gather the information over a set period of time, such as a month. The dependency data can be analyzed to find unknown connections being made to the database. The connection data can help identify application owners that need to be notified of the pending migration.

In addition to the dependency analysis of applications and user connectivity data, Azure Migrate can also be used to analyze the [Hyper-V, VMWare, or physical servers](https://docs.microsoft.com/en-us/azure/migrate/migrate-appliance-architecture) to provide utilization patterns of the database workloads to help suggest the proper target environment.

#### Telgraf for Linux

Linux workloads can utilize the [Microsoft Monitoring Agent (MMA)](https://docs.microsoft.com/en-us/azure/azure-monitor/platform/agent-linux) to gather data on your virtual and physical machines. Additionally, consider using the [Telegraf agent](https://docs.microsoft.com/en-us/azure/azure-monitor/platform/collect-custom-metrics-linux-telegraf) and its wide array of plugins to gather your performance metrics.

### Service Tiers

Equipped with the assessment information (CPU, memory, storage, etc.), the migration user’s next choice is to decide which Azure Database for MySQL [pricing tier](https://docs.microsoft.com/en-us/azure/mysql/concepts-pricing-tiers) to start with.

There are currently three tiers:

* **Basic** : Workloads requiring light compute and I/O performance.
* **General Purpose** : Most business workloads requiring balanced compute and memory with scalable I/O throughput.
* **Memory Optimized** : High performance database workloads requiring in-memory performance for faster transaction processing and higher concurrency.

The tier decision can be influenced by the RTO and RPO requirements of the data workload. When the data workload requires over 4TB of storage, an extra step is required. Review and select [a region that supports](https://docs.microsoft.com/en-us/azure/mysql/concepts-pricing-tiers#storage) up to 16TB of storage.

**Note:** Contact the MySQL team (AskAzureDBforMySQL@service.microsoft.com) for regions that do not support your storage requirements.

Typically, the decision making will focus on the storage and IOPS, or Input/output Operations Per Second, needs. The target system will always need at least as much storage as in the source system. Additionally, since IOPS are allocated 3/GB, it is important to match up the IOPs needs to the final storage size.

|  |  |
| --- | --- |
| Factors | Tier |
| Basic | Development machine, no need for high performance with less than 1TB storage |
| General Purpose | Needs for IOPS more than what basic tier can provide, but for storage less than 16TB, and less than 4GB of memory |
| Memory Optimized | Data workloads that utilize high memory or high cache and buffer related server configuration such as high concurrency innodb\_buffer\_pool\_instances, large BLOB sizes, systems with many slaves for replication |

### Costs

After evaluating the entire WWI MySQL data workloads, WWI determined they would need at least 4 VCores and 20GB of memory and at least a 100GB of storage space with an IOP capacity of 450 IOPS. Because of the 450 IOPS requirement, they will need to allocate at least 150GB of storage due to [Azure Database for MySQLs IOPs allocation method](https://docs.microsoft.com/en-us/azure/mysql/concepts-pricing-tiers#storage). Additionally, they will require at least 7 days’ worth of backups and one read replica. They do not anticipate an outbound egress of more than 5GB/month.

Using the [Azure Database for MySQL pricing calculator](https://azure.microsoft.com/en-us/pricing/details/mysql/), WWI was able to determine the costs for the Azure Database for MySQL instance. As of 9/2020, the total costs of ownership (TCO) is displayed in the following table for the WWI Conference Database:

|  |  |  |  |
| --- | --- | --- | --- |
| Resource | Description | Quantity | Cost |
| Compute (General Purpose) | 4 vcores, 20GB | 1 @ $0.351/hr | $3074.76 / yr |
| Storage | 5GB | 12 x 5 @ $0.115 | $6.90 / yr |
| Backup | 7 full backups (1x free) | 6 \* 5(GB) \* .10 | $3.00 / yr |
| Read Replica | 1 second region replica | compute + storage | $3081.66 / yr |
| Network | < 5GB/month egress | Free |  |
| Total |  |  | $6166.32 / yr |

After reviewing the initial costs, WWI’s CIO confirmed they will be on Azure for a period much longer than 3 years. They made the decision to use 3-year [reserve instances](https://docs.microsoft.com/en-us/azure/mysql/concept-reserved-pricing) to save an extra ~$4K/yr:

|  |  |  |  |
| --- | --- | --- | --- |
| Resource | Description | Quantity | Cost |
| Compute (General Purpose) | 4 vcores | 1 @ $0.1375/hr | $1204.5 / yr |
| Storage | 5GB | 12 x 5 @ $0.115 | $6.90 / yr |
| Backup | 7 full backups (1x free) | 6 \* 5(GB) \* .10 | $3.00 / yr |
| Network | < 5GB/month egress | Free |  |
| Read Replica | 1 second region replica | compute + storage | $1211.40 / yr |
| Total |  |  | $2425.8 / yr |

As the table above shows, backups, network egress, and any read replicas must be considered in the total cost of ownership (TCO). As more databases are added, the storage and network traffic generated would be the only extra cost-based factor to consider.

**Note:** The estimates above do not include any [ExpressRoute](https://docs.microsoft.com/en-us/azure/expressroute/expressroute-introduction), [Azure App Gateway](https://docs.microsoft.com/en-us/azure/application-gateway/overview), [Azure Load Balancer](https://docs.microsoft.com/en-us/azure/load-balancer/load-balancer-overview) or [App Service](https://docs.microsoft.com/en-us/azure/app-service/overview) costs for the application layers.

The above pricing can change at any time and will vary based on region.

### Application Implications

When moving to Azure Database for MySQL, the conversion to secure sockets layer (SSL) based communication is likely to be one of the biggest changes for your applications. SSL is enabled by default in Azure Database for MySQL and it is likely the on-premises application and data workload is not set up to connect to MySQL using SSL. When enabled, SSL usage will add some additional processing overhead and should be monitored.

**Note** Although SSL is enabled by default, you do have the option to disable it.

Follow the activities in [Configure SSL connectivity in your application to securely connect to Azure Database for MySQL](https://docs.microsoft.com/en-us/azure/mysql/howto-configure-ssl) to reconfigure the application to support this strong authentication path.

Lastly, modify the server name in the application connection strings or switch the DNS to point to the new Azure Database for MySQL server.

## WWI Case Study

WWI started the assessment by gathering information about their MySQL data estate. They were able to compile the following:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Source | Db Engine | Size | IOPS | Version | Owner | Downtime |
| WwwDB | AWS (PaaS) | InnoDB | 1GB | 150 | 5.7 | Marketing Dept | 1 hr |
| BlogDB | AWS (Paas) | InnoDB | 1GB | 100 | 5.7 | Marketing Dept | 4 hrs |
| ConferenceDB | On-premises | InnoDB | 5GB | 50 | 5.5 | Sales Dept | 4 hrs |
| CustomerDB | On-premises | MyISAM | 10GB | 75 | 5.5 | Sales Dept | 2 hrs |
| SalesDB | On-premises | InnoDB | 20GB | 75 | 5.5 | Sales Dept | 1 hr |

Each database owner was contacted to determine the acceptable downtime period. The planning and migration method selected was based on the acceptable database downtime.

For the first phase, WWI focused solely on the ConferenceDB database. The team needed the migration experience to assist in the proceeding data workload migrations. The ConferenceDB database was selected because of the simple database structure and the large acceptable downtime. Once the database was migrated, the team focused on migrating the application into the secure Azure landing zone.

## Assessment Checklist

* Test the workload runs successfully on the target system.
* Ensure the right networking components are in place for the migration.
* Understand the data workload resource requirements.
* Estimate the total costs.
* Understand the downtime requirements.
* Be prepared to make application changes.

# Planning

## Landing Zone

An [Azure Landing zone](https://docs.microsoft.com/en-us/azure/cloud-adoption-framework/ready/landing-zone/) is the target environment defined as the final resting place of a cloud migration project. In most projects, the landing zone should be scripted via ARM templates for its initial setup. Finally, it should be customized with PowerShell or the Azure Portal to fit the workloads needs.

Since WWI is based in San Francisco, all resources for the Azure landing zone were created in the US West 2 region. The following resources were created to support the migration:

* [Azure Database for MySQL](https://docs.microsoft.com/en-us/azure/mysql/quickstart-create-mysql-server-database-using-azure-portal)
* [Azure Database Migration Service (DMS)](https://docs.microsoft.com/en-us/azure/dms/quickstart-create-data-migration-service-portal)
* [Express Route](https://docs.microsoft.com/en-us/azure/expressroute/expressroute-introduction)
* [Azure Virtual Network](https://docs.microsoft.com/en-us/azure/virtual-network/quick-create-portal) with [hub and spoke design](https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/hybrid-networking/hub-spoke) with corresponding [virtual network peerings](https://docs.microsoft.com/en-us/azure/virtual-network/virtual-network-peering-overview) establish.
* [App Service](https://docs.microsoft.com/en-us/azure/app-service/overview)
* [Application Gateway](https://docs.microsoft.com/en-us/azure/load-balancer/quickstart-load-balancer-standard-internal-portal?tabs=option-1-create-internal-load-balancer-standard)
* [Private endpoints](https://docs.microsoft.com/en-us/azure/private-link/private-endpoint-overview) for the App Services and MySQL instance

**Note:** As part of this guide, two ARM templates (one with private endpoints, one without) were provided in order to deploy a potential Azure landing zone for a MySQL migration project. The private endpoints ARM template provides a more secure and production like scenario. Additional manual Azure landing zone configuration may be necessary, depending on the requirements.

## Networking

Getting data from the source system to Azure Database for MySQL in a fast and optimal way is a vital component to consider in a migration project. Small unreliable connections may require administrators to restart the migration several times until a successful result is achieved. Restarting migrations due to network issues can lead to wasted effort.

Take the time to understand and evaluate the network connectivity between the source, tool, and destination environments. In some cases, it may be appropriate to upgrade the internet connectivity or configure an ExpressRoute connection from the on-premises environment to Azure. Once on-premises to Azure connectivity has been created, the next step is to validate that the selected migration tool can connect from the source to the destination.

The migration tool location will determine the network connectivity requirements. As shown in the table below, the selected migration tool must be able to connect to both the on-premises machine and to Azure. Azure should be configured to only accept network traffic from the migration tool location.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Migration Tool | Type | Location | Inbound Network Requirements | Outbound Network Requirements |
| Database Migration Service (DMS) | Online or Offline | Azure or Hybrid | Allow 3306 from external IP | A path to connect to the Azure MySQL database instance |
| Import/Export (MySQL Workbench, mysqldump) | Offline | On-premises | Allow 3306 from internal IP | A path to connect to the Azure MySQL database instance |
| Import/Export (MySQL Workbench, mysqldump) | Offline | Azure VM | Allow 3306 from external IP | A path to connect to the Azure MySQL database instance |
| mydumper/myloader | Offline | On-premises | Allow 3306 from internal IP | A path to connect to the Azure MySQL database instance |
| mydumper/myloader | Offline | Azure VM | Allow 3306 from external IP | A path to connect to the Azure MySQL database instance |
| binlog | Online | On-premises | Allow 3306 from external IP or private IP via Private endpoints | A path for each replication server to the master |

Other networking considerations include:

* DMS located in a VNET will be assigned a [dynamic public IP](https://docs.microsoft.com/en-us/azure/dms/faq#setup) to the service. At creation time, you will be able to place the service inside a virtual network that has connectivity via [ExpressRoute](https://docs.microsoft.com/en-us/azure/expressroute/expressroute-introduction) or over a [site to site VPN](https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-howto-site-to-site-resource-manager-portal).
* DMS can be configured in a [hybrid-mode](https://docs.microsoft.com/en-us/azure/dms/quickstart-create-data-migration-service-hybrid-portal) with a worker installed on-premises to proxy the data to DMS.
* When using an Azure Virtual Machine to run the migration tools, assign it a public IP address and then only allow it to connect to the on-premises MySQL instance.
* Outbound firewalls must ensure outbound connectivity to Azure Database for MySQL. The MySQL gateway IP addresses are available on the [Connectivity Architecture in Azure Database for MySQL](https://docs.microsoft.com/en-us/azure/mysql/concepts-connectivity-architecture#azure-database-for-mysql-gateway-ip-addresses) page.

## SSL/TLS Connectivity

In addition to the application implications of migrating to SSL based communication, the SSL/TLS connection types are also something that needs to be considered. After creating the Azure Database for MySQL database, review the SSL settings, and read the [SSL/TLS connectivity in Azure Database for MySQL](https://docs.microsoft.com/en-us/azure/mysql/concepts-ssl-connection-security) article to understand how the TLS settings can affect the security posture.

**Note:** Pay attention to the disclaimer on the page. Enforcement of TLS version will not be enabled by default. Once TLS is enabled, the only way to disable it is to re-enable SSL.

## WWI Case Study

WWI’s cloud team has created the necessary Azure landing zone resources in a specific resource group for the Azure Database for MySQL. Additional resources will be included to support the applications. To create the landing zone, WWI decided to script the setup and deployment using ARM templates. By using ARM templates, they would be able to quickly tear down and re-setup the environment, if needed.

As part of the ARM template, all connections between virtual networks will be configured with peering in a hub and spoke architecture. The database and application will be placed into separate virtual networks. An Azure App Gateway will be placed in front of the app service to allow the app service to be isolated from the Internet. The Azure App Service will connect to the Azure Database for MySQL using a private endpoint.

WWI originally wanted to test an online migration, but the required network setup for DMS to connect to their on-premises environment made this infeasible. WWI chose to do an offline migration instead. The MySQL Workbench tool was used to export the on-premises data and then was used to import the data into the Azure Database for MySQL instance.

## Planning Checklist

* Prepare the Azure landing zone. Consider using ARM template deployment in case the environment must be torn down and rebuilt quickly.
* Verify the networking setup. Verification should include: connectivity, bandwidth, latency and firewall configurations.
* Determine if you are going to use the online or offline data migration strategy.
* Decide on the SSL certificate strategy.

# Migration Methods

Getting the data from the source to target will require using tools or features of MySQL to accomplish the migration.

It is important to complete the entire assessment and planning stages before starting the next stages. The decisions and data collected are migration path and tool selection dependencies.

We explore the following commonly used tools in this section:

* MySQL Workbench
* mysqldump
* mydumper and myloader
* Data-in replication (binlog)

## MySQL Workbench

[MySQL Workbench](https://www.mysql.com/products/workbench/) provides a rich GUI experience that allows developers and administrators to design, develop, and manage their MySQL instances.

The latest version of the MySQL Workbench provides sophisticated [object migration capabilities](https://www.mysql.com/products/workbench/migrate/) when moving a database from a source to target.

### Data Import and Export

MySQL Workbench provides a wizard-based UI to do full or partial export and import of tables and database objects. For an example of how to use the MySQL Workbench, see [Migrate your MySQL database using import and export](https://docs.microsoft.com/en-us/azure/mysql/concepts-migrate-import-export).

## Dump and restore (mysqldump)

mysqldump is typically provided as part of the MySQL installation. It is a [client utility](https://dev.mysql.com/doc/refman/5.7/en/mysqldump.html) that can be run to create logical backups that equate to a set of SQL statements that can be replayed to rebuild the database to a point in time. mysqldump is not intended as a fast or scalable solution for backing up or migrating very large amounts of data. Executing a large set of SQL insert statements can perform poorly due to the disk I/O required to update indexes. However, when combined with other tools that require the original schema, mysqldump is a great tool for generating the database and table schemas. The schemas can create the target landing zone environment.

The mysqldump utility provides useful features during the data migration phase. Performance considerations need to be evaluated before running the utility. See [Performance considerations](https://docs.microsoft.com/en-us/azure/mysql/concepts-migrate-dump-restore#performance-considerations).

## mydumper and myloader

Environments with large databases requiring fast migration should use [mydumper and myloader](https://github.com/maxbube/mydumper). These tools are written in C++ and utilize multi-threaded techniques to send the data as fast as possible to the target MySQL instance. mydumper and myloader take advantage of parallelism and can speed up the migration by a factor of 10x or more.

The tools’ binary releases available for public download have been compiled for Linux. To run these tools on Windows, the open source projects would need to be recompiled. Compiling source code and creating releases is not a trivial task for most users.

## Data-in replication (binlog)

Similar to other database management systems, MySQL provides for a log replication feature called [binlog replication](https://dev.mysql.com/doc/refman/5.7/en/binlog-replication-configuration-overview.html). The binlog replication feature helps with data migration and the creation of read replicas.

Utilize binlog replication to [migrate your data to Azure Database for MySQL](https://docs.microsoft.com/en-us/azure/mysql/concepts-data-in-replication) in an online scenario. The data replication will help to reduce the downtime required to make the final target data changes.

In order to use the binlog replication feature there are some setup [requirements](https://docs.microsoft.com/en-us/azure/mysql/howto-data-in-replication#link-master-and-replica-servers-to-start-data-in-replication):

* Then master server is recommended to use the MySQL InnoDB engine. If you are using a storage engine other than InnoDB, you will need to migrate those tables to InnoDB.
* Migration users must have permissions to configure binary logging and create new users on the master server.
* If the master server has SSL enabled, ensure the SSL CA certificate provided for the domain has been included in the mysql.az\_replication\_change\_master stored procedure. Refer to the following [examples](https://docs.microsoft.com/en-us/azure/mysql/howto-data-in-replication#link-master-and-replica-servers-to-start-data-in-replication) and the master\_ssl\_ca parameter.
* Ensure the master server’s IP address has been added to the Azure Database for MySQL replica server’s firewall rules. Update firewall rules using the Azure portal or Azure CLI.
* Ensure the machine hosting the master server allows both inbound and outbound traffic on port 3306.
* Ensure the master server has an accessible IP address (public or private) from the source to the targets.

To perform a migration using replication, review [How to configure Azure Database for MySQL Data-in Replication](https://docs.microsoft.com/en-us/azure/mysql/howto-data-in-replication#link-master-and-replica-servers-to-start-data-in-replication) for details.

The binlog replication method has high CPU and extra storage requirements. Migration users should test the load placed on the source system during online migrations and determine if it is acceptable.

## Azure Data Migration Service (DMS)

The [Azure Database Migration Services (DMS)](https://azure.microsoft.com/en-us/services/database-migration/) is an Azure cloud-based tool that allows administrators to keep track of the various settings for migration and reuse them if necessary. DMS works by creating migration projects with settings that point to various sources and destinations. It supports both online and offline migrations, with the [online option](https://docs.microsoft.com/en-us/azure/mysql/howto-migrate-online) allowing for significantly minimized downtime. Additionally, it supports [on-premises data workloads](https://docs.microsoft.com/en-us/azure/dms/tutorial-mysql-azure-mysql-online) and cloud-based workloads such as [Amazon Relational Database Service (RDS) MySQL](https://docs.microsoft.com/en-us/azure/dms/tutorial-rds-mysql-server-azure-db-for-mysql-online).

Although the DMS service is an online tool, it does rely on the binlog replication feature of MySQL to complete its tasks. Currently, DMS partially automates the offline migration process. DMS requires the generation and application of the matching schema in the target Azure Database for MySQL instance. Schemas can be exported using the mysqldump client utility.

## Fastest/Minimum Downtime Migration

There are plenty of paths for migrating the data. Deciding which path to take is a function of the migration team’s skill set, and the amount of downtime the database and application owners are willing to accept. Some tools support multi-threaded parallel data migration approaches while other tools were designed for simple migrations of table data only.

The fastest and most complete path is to use binlog replication (either directly with MySQL or via DMS), but it comes with the costs of adding primary keys.

## Decision Table

There are many paths WWI can take to migrate their MySQL workloads. We have provided a table of the potential paths and the advantages and disadvantages of each:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Objective | Description | Tool | Prerequisites | Advantages | Disadvantages |
| Fastest migration possible | Parallel approach | mydumper and myloader | Linux | Highly parallelized | Target throttling |
| Online migration | Keep the source up for as long as possible | binlog | None | Seamless | Extra processing and storage |
| Online migration | Keep the source up for as long as possible | Database Migration Service (DMS) | None | Repeatable process | Limited to data only, supports only 5.5 and 5.7 |
| Highly Customized Offline Migration | Selectively export objects | mysqldump | None | Highly customizable | Manual |
| Offline Migration Semi-automated | UI based export and import | MySQL Workbench | Download and Install | Semi-automated | Only common sets of switches are supported |

**Note:** Let’s use Azure terms. This is data-in replication or binlog replication.

## WWI Case Study

WWI has selected their conference database as their first migration workload. The workload was selected because it had the least risk and the most available downtime due to the gap in the annual conference schedule. Based on the migration team’s assessment, they determined that they will attempt to perform an offline migration using MySQL Workbench.

## Migration Methods Checklist

* Ensure the right method is selected given the target and source environments.
* Ensure the method can meet the business requirements.
* Always verify if the data workload will support the method.

# Test Plans

## Overview

WWI created a test plan that included a set of IT and the Business tasks. Successful migrations require all the tests to be executed.

Tests:

* Ensure the migrated database has consistency (same record counts and query results) with on-premises tables.
* Ensure the performance is acceptable (it should match the same performance as if it were running on-premises).
* Ensure the performance of target queries meet stated requirements.
* Ensure acceptable network connectivity between on-premises and the Azure network.
* Ensure all identified applications and users can connect to the migrated data instance.

WWI has identified a migration weekend and time window that started at 10 pm and ended at 2 am Pacific Time. If the migration did not complete before the 2 am target (the 4hr downtime target) with all tests passing, the rollback procedures were started. Issues were documented for future migration attempts. All migrations windows were pushed forward and rescheduled based on acceptable business timelines.

## Sample Queries

A series of queries were executed on the source and target to verify the database migration success. The following queries and scripts will help determine if the migration actions moved all required database objects from the source to the target.

### Table rows

You can use this query to get all the tables and an estimated row count:

SELECT SUM(TABLE\_ROWS)  
FROM INFORMATION\_SCHEMA.TABLES  
WHERE TABLE\_SCHEMA = '{SchemaName}';

**Note:** The INFORMATION\_SCHEMA table will provide an estimated set of values across the tables. To get a more accurate view of metrics like table size, increase the page sample size with the innodb\_stats\_transient\_sample\_pages server parameter. Increasing this server value will force more pages to be analyzed and provide more accurate results.

Execute the count(\*) SQL statement against every table to get an accurate count of rows. Running this command can take a large amount of time on very large tables. The following script will generate a set of SQL statements that can be executed to get the exact counts:

SELECT CONCAT(  
 'SELECT "',   
 table\_name,   
 '" AS table\_name, COUNT(\*) AS exact\_row\_count FROM `',   
 table\_schema,  
 '`.`',  
 table\_name,   
 '` UNION '  
)   
FROM INFORMATION\_SCHEMA.TABLES   
WHERE table\_schema = '\*\*my\_schema\*\*';

### Table Fragmentation

The data tables are likely to continue to grow larger with continued application use. In some cases, the data may not grow, but it constantly changing through deletions and updates. If this is the case, it is possible there will be a lot of fragmentation in the database files. The MySQL OPTIMIZE TABLE statement can reduce physical storage space needs and improve I/O efficiency.

You can [optimize the MySQL tables](https://dev.mysql.com/doc/refman/8.0/en/optimize-table.html) by running the following:

optimize table TABLE\_NAME

### Database objects

Use the following query to ensure that all other database objects are accounted for. Although these queries can calculate the table row counts, they may not account for the version of the particular database object. For example, even though a stored procedure may exist, it could have changed between the start and end of the migration. Freezing database object development during migration is recommended.

#### Users

SELECT DISTINCT  
 USER  
FROM  
 mysql.user  
WHERE  
 user <> '' order by user

#### Indexes

SELECT DISTINCT  
 INDEX\_NAME  
FROM  
 information\_schema.STATISTICS  
WHERE  
 TABLE\_SCHEMA = '{SchemaName}'

#### Constraints

SELECT DISTINCT  
 CONSTRAINT\_NAME  
FROM  
 information\_schema.TABLE\_CONSTRAINTS  
WHERE  
 CONSTRAINT\_SCHEMA = '{SchemaName}'

#### Views

SELECT  
 TABLE\_NAME  
FROM  
 information\_schema.VIEWS  
WHERE  
 TABLE\_SCHEMA = '{SchemaName}'

#### Stored Procedures

SELECT  
 ROUTINE\_NAME  
FROM  
 information\_schema.ROUTINES  
WHERE  
 ROUTINE\_TYPE = 'FUNCTION' and  
 ROUTINE\_SCHEMA = '{SchemaName}'

#### Functions

SELECT  
 ROUTINE\_NAME  
FROM  
 information\_schema.ROUTINES  
WHERE  
 ROUTINE\_TYPE = 'PROCEDURE' and  
 ROUTINE\_SCHEMA = '{SchemaName}'

#### Events

SELECT  
 EVENT\_NAME  
FROM  
 INFORMATION\_SCHEMA.EVENTS  
WHERE  
 EVENT\_SCHEMA = '{SchemaName}'

## Rollback Strategies

The queries above will provide a list of object names and counts that can be used in a rollback decision. Migration users can take the first object verification step by checking the source and target object counts. A discrepancy in object counts may not necessarily mean a rollback is needed. Preforming an in-depth evaluation could point out the discrepancy is small and easily recoverable. A manual migration of a few failed objects could be the solution. For example, if all tables and data rows were migrated, but only a few of the functions were missed, remediate those failed objects and finalize the migration. If the database is relatively small, it could be possible to clear the Azure Database for MySQL instance and restart the migration again. Large databases may need more time than available in the migration window to determine missing objects. The migration will need to stop and rollback.

Identifying missing database objects needs to occur quickly during a migration window. Every minute counts. One option could be, export the environment object names to a file and use a data comparison tool to quickly identify missing objects. Another option could be, export the source database object names and import the data into a target database environment temp table. Compare the data using a **scripted** and **tested** SQL statement. Data verification speed and accuracy are critical to the migration process. Do not rely on manually reading and verifying a long list of database objects during a migration window. The possibility of human error is too great. You should manage by exception using tools.

## WWI Case Study

The WWI CIO received a confirmation report that all database objects were migrated from the on-premises database to the Azure Database for MySQL instance. The database team ran the above queries against the database before the beginning of the migration and saved all the results to a spreadsheet.

The source database schema information was used to verify the target migration object fidelity.

## Checklist

* Have test queries scripted, tested, and ready to execute.
* Know how long test queries take to run and make it a part of the documented migration timeline.
* Have a mitigation and rollback strategy ready for different potential outcomes.
* Have a well-defined timeline of events for the migration.

# Performance Baselines

Understanding the existing MySQL workload is one of the best investments that can be made to ensure a successful migration. Excellent system performance depends on adequate hardware and great application design. Items such as CPU, memory, disk, and networking need to be sized and configured appropriately for the anticipated load. Hardware and configuration are part of the system performance equation. The developer must understand the database query load and the most expensive queries to execute. Focusing on the most expensive queries can make a substantial difference in the overall performance metrics.

Creating baselines of query performance is vital to a migration project. The performance baselines can be used to verify the Azure landing zone configuration for the migrated data workloads. Most systems will be run 24/7 and have different peak load times. It is important to capture the peak workloads for the baseline. Metrics should captured several times. Later in the document, we will explore the source server parameters and how they are essential to the overall performance baseline picture. The server parameters should not be overlooked during a migration project.

## Tools

Below are tools used to gather server metrics and database workload information. Use the captured metrics to determine the appropriate Azure Database for MySQL tier and the associated scaling options.

* [MySQL Enterprise Monitor](https://www.mysql.com/products/enterprise/monitor.html): This non-free, enterprise edition tool can provide a sorted list of the most expensive queries, server metrics, file I/O and topology information
* [Percona Monitoring and Management (PMM)](https://www.percona.com/software/database-tools/percona-monitoring-and-management) : a best-of-breed open source database monitoring solution. It helps to reduce complexity, optimize performance, and improve the security of business-critical database environments, no matter the deployed location.

## Server Parameters

MySQL server default configurations may not adequately support a workload. There are a plethora of server parameters in MySQL, but in most cases the migration team should focus on a handful. The following parameters should be evaluated in the **source** and **target** environments. Incorrect configurations can affect the speed of the migration. We will revisit these parameters again when we execute the migration steps.

* **innodb\_buffer\_pool\_size**: A large value will ensure in-memory resources are used first before utilizing disk I/O. Typical values will range from 80-90% of the available memory. For example, a system with 8GB of memory should allocate 5-6GB for the pool size.
* **innodb\_log\_file\_size**: The redo logs are used to ensure fast durable writes. This transactional backup is helpful during a system crash. Starting with innodb\_log\_file\_size = 512M (giving 1GB of redo logs) should give plenty of room for writes. Write-intensive applications using MySQL 5.6 or higher should start with innodb\_log\_file\_size = 4G.
* **max\_connections**: This parameter can help alleviate the Too many connections error. The default is 151 connections. Using a connection pool at the application level is preferred, but the server connection configuration may need to increase as well.
* **innodb\_file\_per\_table**: This setting will tell InnoDB if it should store data and indexes in the shared tablespace or in a separate .ibd file for each table. Having a file per table enables the server to reclaim space when tables are dropped, truncated, or rebuilt. Databases containing a large number of tables should not use the table per file configuration. As of MySQL 5.6, the default value is ON. Earlier database versions should set the configuration to ON prior to loading data. This setting only affects newly created tables.
* **innodb\_flush\_log\_at\_trx\_commit**: The default setting of 1 means that InnoDB is fully ACID compliant. This lower risk transaction configuration can have a significant overhead on systems with slow disks because of the extra fsyncs are needed to flush each change to the redo logs. Setting the parameter to 2 is a bit less reliable because committed transactions will be flushed to the redo logs only once a second. The risk can be acceptable in some master situations, and it is definitely a good value for a replica. A value of 0 allows for better system performance, but the database server is more likely to lose some data during a failure. The bottom line, use the 0 value for a replica only.
* **innodb\_flush\_method**: This setting controls how data and logs are flushed to disk. Use O\_DIRECT when in presence of a hardware RAID controller with a battery-protected write-back cache. Use fdatasync (default value) for other scenarios.
* **innodb\_log\_buffer\_size**: This setting is the size of the buffer for transactions that have not been committed yet. The default value (1MB) is usually fine. Transactions with large blob/text fields can fill up the buffer quickly and trigger extra I/O load. Look at the Innodb\_log\_waits status variable and if it is not 0, increase innodb\_log\_buffer\_size.
* **query\_cache\_size**: The query cache is a well-known bottleneck that can be seen during moderate concurrency. The initial value should be set to 0 to disable cache. e.g. query\_cache\_size = 0. This is the default on MySQL 5.6 and later.
* **log\_bin**: This setting will enable binary logging. Enabling binary logging is mandatory if the server is to act as a replication master.
* **server\_id**: This setting will be a unique value to identity servers in replication topologies.
* **expire\_logs\_days**: This setting will control how many days the binary logs will be automatically purged.
* **skip\_name\_resolve**: user to perform client hostname resolution. If the DNS is slow, the connection will be slow. When disabling name resolution, the GRANT statements must use IP addresses only. Any GRANT statements made previously would need to be redone to use the IP.

Run the following command to export the server parameters to a file for review. Using some simple parsing, the output can be used to reapply the same server parameters after the migration, if appropriate to the Azure Database for MySQL server. Reference [Configure server parameters in Azure Database for MySQL using the Azure portal](https://docs.microsoft.com/en-us/azure/mysql/howto-server-parameters).

mysql -u root -p -A -e "SHOW GLOBAL VARIABLES;" > settings.txt

The MySQL 5.5.60 default installed server parameters can be found in Appendix C.

Before migration begins, export the source MySQL configuration settings. Compare those values to the Azure landing zone instance settings after the migration. If any settings were modified from the default in the target Azure landing zone instance, ensure that these are set back after the migration. Also, the migration user should verify the server parameters can be set before the migration.

For a list of server parameters that cannot be configured, reference [Non-configurable server parameters](https://docs.microsoft.com/en-us/azure/mysql/concepts-server-parameters#non-configurable-server-parameters).

### Egress and Ingress

For each respective data migration tool and path, the source and the target MySQL server parameters will need to be modified to support the fastest possible egress and ingress. Depending on the tool, the parameters could be different. For example, a tool that performs a migration in parallel may need more connections on the source and the target versus a single threaded tool.

Review any timeout parameters that may be affected by the datasets. These include:

* [connect\_timeout](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_connect_timeout)
* [wait\_timeout](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_wait_timeout)

Additionally, review any parameters that will affect maximums:

* [max\_allowed\_packet](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_max_allowed_packet)

**Note:** A common migration error is MySQL server has gone away. The parameters mentioned here are the typical culprits for resolving this error.

## WWI Case Study

WWI reviewed their Conference database workload and determined it had a very small load. Although a basic tier server would work for them, they did not want to perform work later to migrate to another tier. The server being deployed will eventually host the other MySQL data workloads and so they picked the General Performance tier.

In reviewing the MySQL database, the MySQL 5.5 server is running with the defaults server parameters that are set during the initial install.

# Data Migration

## Back up the database

As a prudent step before upgrade or migrate data, export the database before the upgrade using MySQL Workbench or manually via the mysqldump command.

## Offline vs. Online

Before a migration tool is selected, it should be determined if the migration should be online or offline.

* **Offline migrations** will cause the system to be down while the migration takes place. This option ensures that no transactions are occurring and the state of the data will be exactly what is expected when restored in Azure.
* **Online migrations** will migrate the data in near real-time. This option is appropriate when there is little downtime for the users or application consuming the data workload. The process involves replicating the data using a replication method such as binlog or similar.

In the case of WWI, their environment has some complex networking and security requirements that will not allow for the appropriate changes to be applied for inbound and outbound connectivity in the target migration time frame. These complexities and requirements essentially eliminate the online approach from consideration.

**Note:** Review the Planning and Assessment sections for more details on Offline vs Online migration.

## Data Drift

Offline migration strategies have the potential for data drift. Data drift occurs when newly modified source data becomes out of sync with migrated data. When this happens, a full export or a delta export will be needed. You can mitigate this problem by stopping all traffic to the database and then performing your export. If stopping all data modification traffic is not possible, you will need to account for the drift.

Determining the changes can become complicated if the database tables don’t have columns such as a numeric based primary keys, or some type of modify and create date in every table that needs to be migrated.

For example, if a numeric based primary key is present and the migration is importing in sort order, it will be relatively simple to determine where the import stopped and restart it from that position. If no numeric based key is present, then it could be possible to utilize modify and create date, and again, import in a sorted manner to be able to restart the migration from the last date seen in the target.

## Performance recommendations

### Exporting

* Use an export tool that can run in a multi-threaded mode such as mydumper
* When using MySQL 8.0, use [partitioned tables](https://dev.mysql.com/doc/refman/8.0/en/partitioning-overview.html) when appropriate to increase the speed of exports.

### Importing

* Create clustered indexes and primary keys after loading data. Load data in primary key order, or other if primary key some date column (such as modify date or create date) in sorted order.
* Delay the creation of secondary indexes until after data is loaded. Create all secondary indexes after loading.
* Disable foreign key constraints before loading. Disabling foreign key checks provides significant performance gains. Enable the constraints and verify the data after the load to ensure referential integrity.
* Load data in parallel. Avoid too much parallelism that could cause resource contention and monitor resources by using the metrics available in the Azure portal.

## Performing the Migration

* Back up the database
* Create and verify the Azure Landing zone
* Configure Source Server Parameters
* Configure Target Server Parameters
* Export the database objects (schema, users, etc.)
* Export the data
* Import the database objects
* Import the data
* Validation
* Reset Target Server Parameters
* Migrate the Application(s)

## Common Steps

Despite what path is taken, there are common steps that must be performed:

* Upgrade to a supported Azure MySQL version
* Inventory Database Objects
* Export users and permissions

## Migrate to latest MySQL version

Since the WWI Conference database is running 5.5 it will be necessary to perform an upgrade. The CIO has asked that they upgrade to the latest version of MySQL (currently 8.0).

There are two ways to upgrade to 8.0:

* In-place
* Export/Import

When deciding to do an upgrade, it is important that to run the **upgrade checker** tool to determine if there are any conflicts. For example, when upgrading to MySQL Server 8.0, the tool will check for the following conflicts:

* Database object names that conflict with reserve words in MySQL 8.0
* Usage of the utf8mb3 charset
* Usage of the ZEROFILL/display length type attributes
* Table names that conflict with tables in 8.0
* Temporal type usage
* Foreign key constraint names longer than 64 characters

If the upgrade checker reports no issues, it is safe to do an in-place upgrade by replacing MySQL binaries. Databases with issues will need to be exported and the issues addressed.

## WWI Case Study

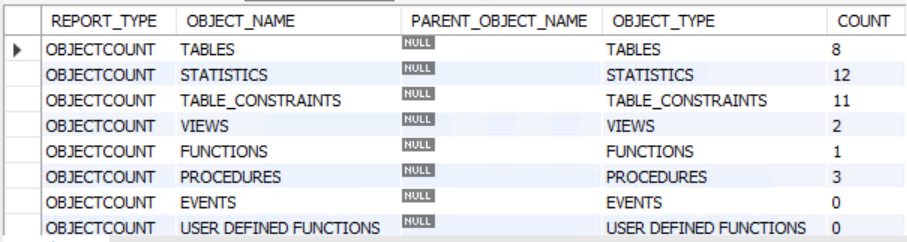
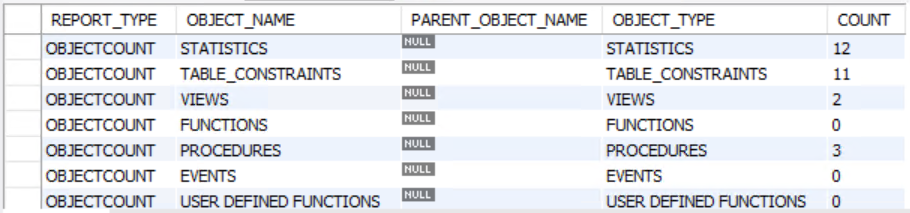
After successfully migrating the MySQL instance to 8.0, the WWI migration team realized the original [Database Migration Service (DMS)](https://datamigration.microsoft.com/scenario/mysql-to-azuremysql?step=1) migration path could no longer be used as the DMS tool currently only supports 5.6 and 5.7. DMS required network access. The WWI migration team was not ready to handle their complex network issues. These environmental issues narrowed their migration tool choice to MySQL Workbench.

## Database Objects

As outlined in the Test Plans section, an inventory of database objects should be done before and after the migration to ensure that you have migrated everything.

If you would like to execute a stored procedure to generate this information, you can use something similar to the following:

DELIMITER //  
  
CREATE PROCEDURE `Migration\_PerformInventory`(IN schemaName CHAR(64))  
BEGIN  
   
 DECLARE finished INTEGER DEFAULT 0;  
 DECLARE tableName varchar(100) DEFAULT "";  
   
 #get all tables  
 DECLARE curTableNames  
 CURSOR FOR   
 SELECT TABLE\_NAME FROM information\_schema.tables where TABLE\_SCHEMA = schemaName;  
  
 -- declare NOT FOUND handler  
 DECLARE CONTINUE HANDLER   
 FOR NOT FOUND SET finished = 1;  
   
 DROP TABLE IF EXISTS MIG\_INVENTORY;  
   
 CREATE TABLE MIG\_INVENTORY  
 (   
 REPORT\_TYPE VARCHAR(1000),   
 OBJECT\_NAME VARCHAR(1000),   
 PARENT\_OBJECT\_NAME VARCHAR (1000),  
 OBJECT\_TYPE VARCHAR(1000),   
 COUNT INT  
 )  
 ROW\_FORMAT=DYNAMIC,  
 ENGINE='InnoDB';  
   
 INSERT INTO MIG\_INVENTORY (REPORT\_TYPE,OBJECT\_NAME, OBJECT\_TYPE, COUNT)  
 SELECT  
 'OBJECTCOUNT', 'TABLES', 'TABLES', COUNT(\*)  
 FROM   
 information\_schema.tables   
 where   
 TABLE\_SCHEMA = schemaName;  
   
 INSERT INTO MIG\_INVENTORY (REPORT\_TYPE,OBJECT\_NAME, OBJECT\_TYPE, COUNT)  
 SELECT  
 'OBJECTCOUNT', 'STATISTICS', 'STATISTICS', COUNT(\*)  
 FROM  
 information\_schema.STATISTICS  
 WHERE  
 TABLE\_SCHEMA = schemaName;  
   
 #### Constraints  
 INSERT INTO MIG\_INVENTORY (REPORT\_TYPE,OBJECT\_NAME, OBJECT\_TYPE, COUNT)  
 SELECT  
 'OBJECTCOUNT', 'TABLE\_CONSTRAINTS', 'TABLE\_CONSTRAINTS', COUNT(\*)  
 FROM  
 information\_schema.TABLE\_CONSTRAINTS  
 WHERE  
 CONSTRAINT\_SCHEMA = schemaName;  
   
 INSERT INTO MIG\_INVENTORY (REPORT\_TYPE,OBJECT\_NAME, OBJECT\_TYPE, COUNT)  
 SELECT  
 'OBJECTCOUNT', 'VIEWS', 'VIEWS', COUNT(\*)  
 FROM  
 information\_schema.VIEWS  
 WHERE  
 TABLE\_SCHEMA = schemaName;  
   
 INSERT INTO MIG\_INVENTORY (REPORT\_TYPE,OBJECT\_NAME, OBJECT\_TYPE, COUNT)  
 SELECT  
 'OBJECTCOUNT', 'FUNCTIONS', 'FUNCTIONS', COUNT(\*)  
 FROM  
 information\_schema.ROUTINES  
 WHERE  
 ROUTINE\_TYPE = 'FUNCTION' and  
 ROUTINE\_SCHEMA = schemaName;  
   
 INSERT INTO MIG\_INVENTORY (REPORT\_TYPE,OBJECT\_NAME, OBJECT\_TYPE, COUNT)  
 SELECT  
 'OBJECTCOUNT', 'PROCEDURES', 'PROCEDURES', COUNT(\*)  
 FROM  
 information\_schema.ROUTINES  
 WHERE  
 ROUTINE\_TYPE = 'PROCEDURE' and  
 ROUTINE\_SCHEMA = schemaName;  
   
 INSERT INTO MIG\_INVENTORY (REPORT\_TYPE,OBJECT\_NAME, OBJECT\_TYPE, COUNT)  
 SELECT  
 'OBJECTCOUNT', 'EVENTS', 'EVENTS', COUNT(\*)  
 FROM  
 information\_schema.EVENTS  
 WHERE  
 EVENT\_SCHEMA = schemaName;  
   
 INSERT INTO MIG\_INVENTORY (REPORT\_TYPE,OBJECT\_NAME, OBJECT\_TYPE, COUNT)  
 SELECT  
 'OBJECTCOUNT', 'USER DEFINED FUNCTIONS', 'USER DEFINED FUNCTIONS', COUNT(\*)  
 FROM  
 mysql.func;  
   
 INSERT INTO MIG\_INVENTORY (REPORT\_TYPE,OBJECT\_NAME, OBJECT\_TYPE, COUNT)  
 SELECT  
 'OBJECTCOUNT', 'USERS', 'USERS', COUNT(\*)  
 FROM  
 mysql.user  
 WHERE  
 user <> '' order by user;  
  
 OPEN curTableNames;  
  
 getTableName: LOOP  
 FETCH curTableNames INTO tableName;  
 IF finished = 1 THEN   
 LEAVE getTableName;  
 END IF;  
   
 SET @s = CONCAT('SELECT COUNT(\*) into @TableCount FROM ', schemaName, '.', tableName);  
 #SELECT @s;  
 PREPARE stmt FROM @s;  
   
 EXECUTE stmt;  
   
 INSERT INTO MIG\_INVENTORY (REPORT\_TYPE,OBJECT\_NAME, OBJECT\_TYPE, COUNT)  
 SELECT  
 'TABLECOUNT', tableName, 'TABLECOUNT', @TableCount;  
   
 DEALLOCATE PREPARE stmt;  
   
 END LOOP getTableName;  
 CLOSE curTableNames;  
   
 SELECT \* FROM MIG\_INVENTORY;  
  
END //  
  
DELIMITER ;  
  
CALL Migration\_PerformInventory('reg\_app');

* Calling this procedure on the source DB reveals the following (truncated output):
* 
* The target database procedure result should resemble the image below after completing the migration. Notice there are no functions in the DB.Functions were eliminated before the migration.
* 

## Users and permissions

A successful migration requires migrating associated users and permissions to the target environment.

Export all users and their grants using the following PowerShell script:

$username = "yourusername";  
$password = "yourpassword";  
mysql -u$username -p$password --skip-column-names -A -e"SELECT CONCAT('SHOW GRANTS FOR ''',user,'''@''',host,''';') FROM mysql.user WHERE user<>''" > ShowGrants.sql;  
  
$lines = get-content "ShowGrants.sql"  
  
foreach ($line in $lines)  
{  
 mysql -u$username -p$password --skip-column-names -A -e"$line" >> AllGrants.sql  
}

* Create a new PowerShell script using PowerShell ISE (refer to the Setup document)
* Set **yourusername** to root and **yourpassword** to the root user’s password

You can then run the AllGrants.sql script targeting the new Azure Database for MySQL:

$username = "yourusername";  
$password = "yourpassword";  
$server = "serverDNSname";  
$lines = get-content "AllGrants.sql"  
  
foreach ($line in $lines)  
{  
 mysql -u$username -p$password -h$server --ssl-ca=c:\temp\BaltimoreCyberTrustRoot.crt.cer --skip-column-names -A -e"$line"  
}

You can also create users in Azure Database for MySQL using PowerShell: https://docs.microsoft.com/en-us/azure/mysql/howto-create-users

## Execute migration

With the basic migration components in place, it is now possible to proceed with the data migration. There were several tools and methods introduced previously. For WWI, they are going to utilize the MySQL Workbench path to export the data and then import it into Azure Database for MySQL.

## Data Migration Checklist

* Understand the complexity of the environment and if an online approach is feasible.
* Account for data drift. Stopping the database service can eliminate potential data drift.
* Configure source parameters for fast export.
* Configure target parameters for fast import.
* Test any migrations that have a different source version vs the target.
* Migrate any non-data based objects such as user names and privileges.
* Make sure all tasks are documented and checked off as the migration is executed.

# Data Migration with MySQL Workbench

## Setup

Follow all the steps in the Setup guide to create an environment to support the following steps.

## Configuring Server Parameters (Source)

Depending on the type of migration you have chosen (offline vs. online), you will want to evaluate if you are going to modify the server parameters to support a fast egress of the data. If you are doing online, it may not be necessary to do anything to server parameters as you will likely be performing binlog replication and have the data syncing on its own. However, if you are doing an offline migration, once you stop application traffic, you can switch the server parameters from supporting the workload to supporting the export.

## Configuring Server Parameters (Target)

Review the server parameters before starting the import process into Azure Database for MySQL. Server parameters can be retrieved and set using the [Azure Portal](https://docs.microsoft.com/en-us/azure/mysql/howto-server-parameters) or by calling the [Azure PowerShell for MySQL cmdlets](https://docs.microsoft.com/en-us/azure/mysql/howto-configure-server-parameters-using-powershell) to make the changes.

Execute the following PowerShell script to get all parameters:

[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12  
  
Install-Module -Name Az.MySql  
Connect-AzAccount  
$rgName = "{RESOURCE\_GROUP\_NAME}";  
$serverName = "{SERVER\_NAME}";  
Get-AzMySqlConfiguration -ResourceGroupName $rgName -ServerName $serverName

* To do the same with the mysql tool, download the [CA root certification](https://www.digicert.com/CACerts/BaltimoreCyberTrustRoot.crt.pem) to c:\temp (make this directory).

**Note** The certificate is subject to change. Reference [Configure SSL connectivity in your application to securely connect to Azure Database for MySQL](https://docs.microsoft.com/en-us/azure/mysql/howto-configure-ssl) for the latest certificate information.

* Run the following in a command prompt, be sure to update the tokens:

mysql --host {servername}.mysql.database.azure.com --database mysql --user {username}@{servername} -p --ssl-ca=c:\temp\BaltimoreCyberTrustRoot.crt.cer -A -e "SHOW GLOBAL VARIABLES;" > c:\temp\settings\_azure.txt

In the new `settings\_azure.txt` file, you will see the default Azure Database for MySQL server parameters as shown in Appendix A.

To support the migration, set the target MySQL instance parameters to allow for a faster ingress. The following server parameters should be set before starting the data migration:

* max\_allowed\_packet – set the parameter to 1073741824 (i.e. 1GB) or the largest size of a row in the database to prevent any overflow issue due to long rows. Consider adjusting this parameter if there are large BLOB rows that need to be pulled out (or read).
* innodb\_buffer\_pool\_size – Scale up the server to 32 vCore Memory Optimized SKU from the Pricing tier of the portal during migration to increase the innodb\_buffer\_pool\_size. Innodb\_buffer\_pool\_size can only be increased by scaling up compute for Azure Database for MySQL server. Reference [Server parameters in Azure Database for MySQL](https://docs.microsoft.com/en-us/azure/mysql/concepts-server-parameters#innodb_buffer_pool_size) for the max value for the tier. The maximum value in a Memory Optimized 32 vCore system is 132070244352.
* innodb\_io\_capacity & innodb\_io\_capacity\_max - Change the parameter to 9000 to improve the IO utilization to optimize for migration speed.
* max\_connections - If using a tool that generates multiple threads to increase throughput, increase the connections to support that tool. Default is 151, max is 5000.

**Note:** Take care when performing scaling. Some operations cannot be undone, such as storage scaling.

These settings can be updated using the Azure PowerShell cmdlets below:

Install-Module -Name Az.MySql  
$rgName = " {RESOURCE\_GROUP\_NAME}";  
$serverName = "{SERVER\_NAME}";

Select-AzSubscription -Subscription "{SUBSCRIPTION\_ID}"  
Update-AzMySqlConfiguration -Name max\_allowed\_packet -ResourceGroupName $rgname -ServerName $serverName -Value 1073741824  
Update-AzMySqlConfiguration -Name innodb\_buffer\_pool\_size -ResourceGroupName $rgname -ServerName $serverName -Value 16106127360  
Update-AzMySqlConfiguration -Name innodb\_io\_capacity -ResourceGroupName $rgname -ServerName $serverName -Value 9000  
Update-AzMySqlConfiguration -Name innodb\_io\_capacity\_max -ResourceGroupName $rgname -ServerName $serverName -Value 9000

#required if you have functions

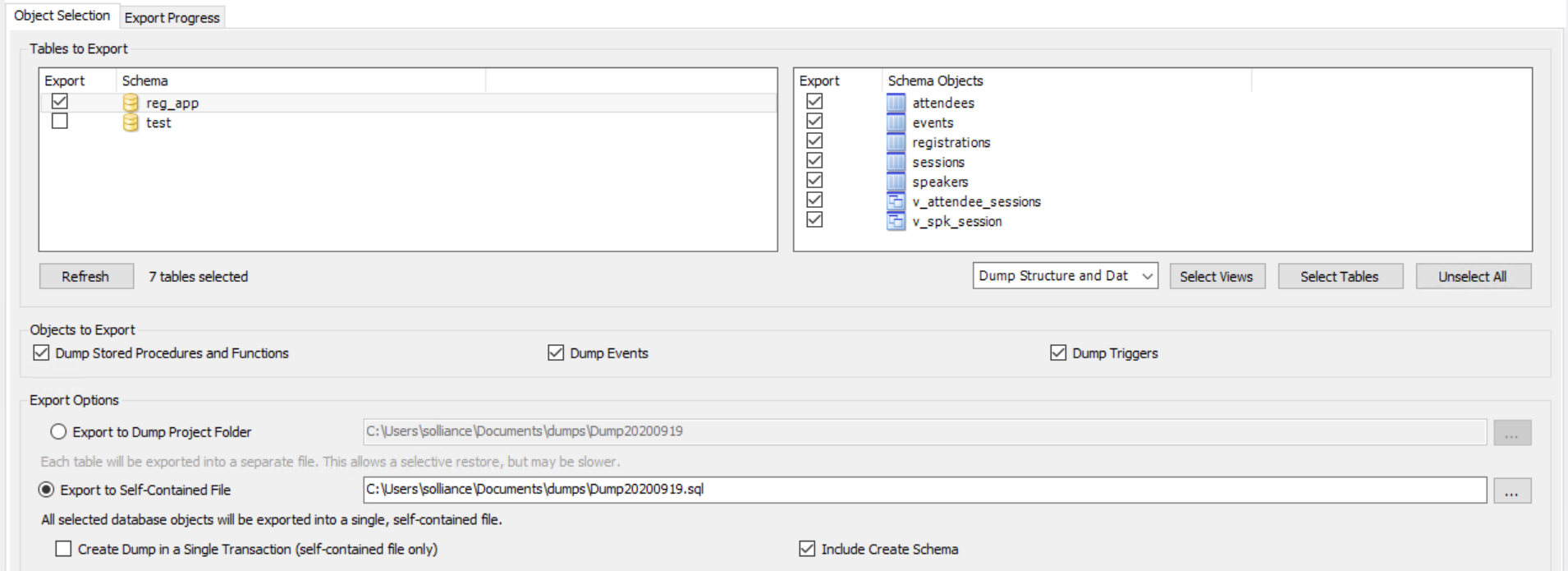
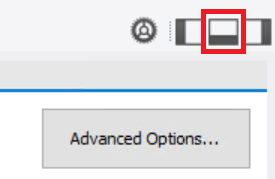
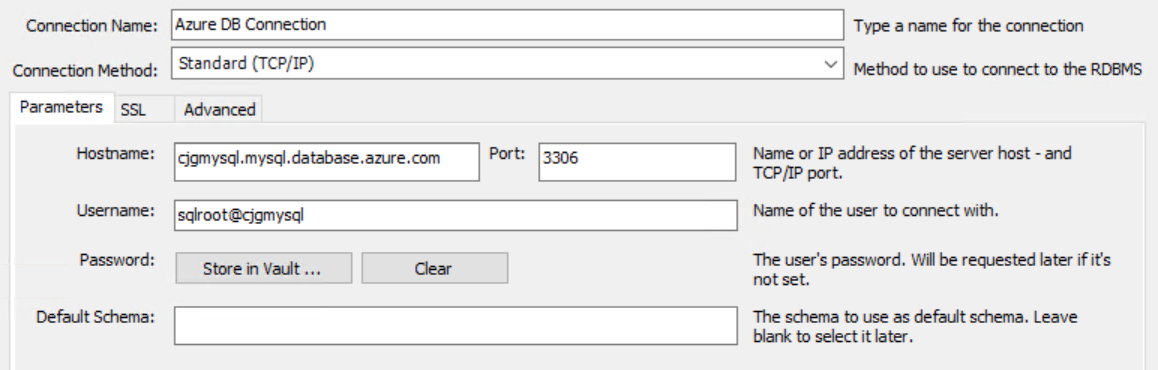
Update-AzMySqlConfiguration -Name log\_bin\_trust\_function\_creators -ResourceGroupName $rgname -ServerName $serverName -Value ON

## Data

### Tool Choice

With the database objects and users from the source system migrated, the migration can begin. Databases running on MySQL version 8.0 cannot use Azure DMS to migrate the workload. Instead, migration users should use MySQL Workbench.

### Manual Import and Export Steps

* Open MySQL Workbench and connect as the local database’s root user.
* Under \*\*Management\*\*, select \*\*Data Export\*\*. Select the **reg\_app** schema.
* In **Objects to Export**, select **Dump Stored Procedures and Functions**, **Dump Events** and **Dump Triggers**.
* Under **Export Options**, select **Export to Self-Contained File**.
* Also, select the **Include Create Schema** checkbox. Refer to the image below to observe the correct mysqldump configuration.
* 
* Test
* If any of these options appear unavailable, they are likely obstructed by the Output pane. Just change the editor layout.
* 
* Test
* Select the **Export Progress** tab.
* Select **Start Export**, notice MySQL Workbench makes calls to the mysqldump tool.
* Open the newly created export script.
* Find any DEFINER statements and either change to a valid user or remove them completely.
* **Note:** This can be done by passing the --skip-definer in the mysqldump command. This is not an option in the MySQL Workbench; therefore, the lines will need to be manually removed in the export commands. Although we point out four items to remove here, there can be other items that could fail when migrating from one MySQL version to another (such as new reserved words).
* Find SET GLOBAL statements and either change to a valid user or remove them completely.
* Ensure sql\_mode isn’t set to NO\_AUTO\_CREATE\_USER.
* Remove the hello\_world function.
* In MySQL Workbench, create a new connection to the Azure Database for MySQL.
  + For Hostname, enter the full server DNS (ex: servername.mysql.database.azure.com).
  + Enter the username (ex: sqlroot@servername).
  + Select the **SSL** tab.
  + For the SSL CA File, browse to the **BaltimoreCyberTrustRoot.crt.cer** key file.
  + Select **Test Connection**, ensure the connection completes.
  + Select **OK**.
  + 
  + MySQL connection dialog box is displayed.
* Select **File->Open SQL Script**.
* Browse to the dump file, select **Open**.
* Select **Execute**.

## Update Applications to support SSL

* Switch to the Java Server API in Visual Studio code.
* Open the **launch.json** file.
* Update the **DB\_CONNECTION\_URL** to jdbc:mysql://serverDNSname:3306/reg\_app?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC&verifyServerCertificate=true&useSSL=true&requireSSL=true&noAccessToProcedureBodies=true. Note the additional SSL parameters.
* Update **DB\_USER\_NAME** to **conferenceuser@servername**.
* Start the debug configuration, and ensure that the application works locally with the new database.

## Revert Server Parameters

The following parameters can be changed on the Azure Database for MySQL target instance. These parameters can be set through the Azure Portal or by using the [Azure PowerShell for MySQL cmdlets](https://docs.microsoft.com/en-us/azure/mysql/howto-configure-server-parameters-using-powershell).

$rgName = "YourRGName";  
$serverName = "servername";  
Update-AzMySqlConfiguration -Name max\_allowed\_packet -ResourceGroupName $rgname -ServerName $serverName -Value 536870912  
Update-AzMySqlConfiguration -Name innodb\_buffer\_pool\_size -ResourceGroupName $rgname -ServerName $serverName -Value 16106127360  
Update-AzMySqlConfiguration -Name innodb\_io\_capacity -ResourceGroupName $rgname -ServerName $serverName -Value 200  
Update-AzMySqlConfiguration -Name innodb\_io\_capacity\_max -ResourceGroupName $rgname -ServerName $serverName -Value 2000

## Change Connection String for the Java API

* Use the following commands to change the connection string for the App Service Java API

$rgName = "YourRGName";  
$app\_name = "servername";  
az webapp config appsettings set -g $rgName -n $app\_name --settings DB\_CONNECTION\_URL={DB\_CONNECTION\_URL}

**Note:** Remember that you can use the Portal to set the connection string.

* Restart the App Service API

az webapp restart -g $rgName -n $app\_name

You have successfully completed an on-premises to Azure Database for MySQL migration!

# Post Migration Management

## 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 operational 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.

## WWI Case Study

WWI decided to utilize the Azure Activity logs and enable MySQL logging to flow to a [Log Analytics workspace](https://docs.microsoft.com/en-us/azure/azure-monitor/platform/design-logs-deployment). This workspace is configured to be a part of [Azure Sentinel](https://docs.microsoft.com/en-us/azure/sentinel/) such that any [Threat Analytics](https://docs.microsoft.com/en-us/azure/mysql/concepts-data-access-and-security-threat-protection) events would be surfaced, and incidents created.

The MySQL DBAs installed the Azure Database for [MySQL Azure PowerShell cmdlets](https://docs.microsoft.com/en-us/azure/mysql/quickstart-create-mysql-server-database-using-azure-powershell) to make managing the MySQL Server automated versus having to log to the Azure Portal each time.

## Management Checklist

* Create resource alerts for common things like CPU and Memory.
* Ensure the server parameters are configured for the target data workload after migration.
* Script common administrative tasks.
* Set up notifications for maintenance events such as upgrades and patches. Notify users as necessary.

# 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 section.

## WWI Case Study

WWI business and application users expressed a high level of excitement regarding the ability to scale the database on-demand. They were also interested in using the Query Performance Insight to determine if long running queries performance needed to be addressed.

They opted to utilize a read replica server for any potential failover or read-only needed scenarios.

The migration team, working with the Azure engineers, set up KQL queries to monitor for any potential issues with the MySQL server performance. The KQY queries were set up with alerts to email event issues to the database and conference team.

They elected to monitor any potential issues for now and implement Azure Automation run books at a later time, if needed, to improve operational efficiency.

## 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.

# 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.

# 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.

# Summary

This document has covered several topics related to migrating an application from on-premises MySQL to Azure Database for MySQL. We covered how to begin and assess the project all the way to application cut over.

The migration team will need to review the topics carefully as the choices made can have project timeline effects. The total cost of ownership is very attractive given the many enterprise ready features provided.

The migration project approach is very important. The team will need to assess the application and database complexity to determine the amount of conversion time. Conversion tools will help make the transition easier, but there will always be an element of manual review and updates required. Scripting out pre-migration tasks and post migration testing is important.

Application architecture and design can provide strong indicators as to the level of effort required. For example, applications utilizing ORM frameworks can be great candidates, especially if the business logic is contained in the application instead of database objects.

In the end, several tools exist in the marketplace ranging from free to commercial. This document covered the steps required if the team plans a database migration using one of the more popular open source tool options. Whichever path that is chosen, Microsoft and the MySQL community have the tools and expertise to make the database migration successful.

## Questions and Feedback

For any questions or suggestions about working with Azure Database for MySQL, send an email to the Azure Database for MySQL Team (AskAzureDBforMySQL@service.microsoft.com). Please note that this address is for general questions rather than support tickets.

In addition, consider these points of contact as appropriate:

* To contact Azure Support or fix an issue with your account, [file a ticket from the Azure portal](https://portal.azure.com/#blade/Microsoft_Azure_Support/HelpAndSupportBlade/overview).
* To provide feedback or to request new features, create an entry via [UserVoice](https://feedback.azure.com/forums/597982-azure-database-for-mysql).

## Find a partner to assist in migrating

This guide can be overwhelming, but don’t fret! There are many experts in the community with a proven migration track record. [Search for a Microsoft Partner](https://www.microsoft.com/solution-providers/home) or [Microsoft MVP](https://mvp.microsoft.com/MvpSearch) to help with finding the most appropriate migration strategy. You are not alone!

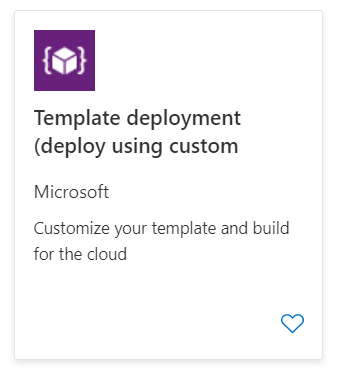
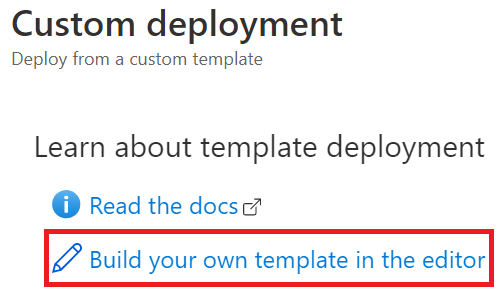
You can also browse the technical forums and social groups for more detailed real-world information:

* [Microsoft Community Forum](https://docs.microsoft.com/en-us/answers/topics/azure-database-mysql.html)
* [Azure Database for MySQL Tech Community](https://techcommunity.microsoft.com/t5/azure-database-for-mysql/bg-p/ADforMySQL)
* [StackOverflow for Azure MySQL](https://stackoverflow.com/questions/tagged/azure-database-mysql)
* [Azure Facebook Group](https://www.facebook.com/groups/MsftAzure)
* [LinkedIn Azure Group](https://www.linkedin.com/groups/2733961/)
* [LinkedIn Azure Developers Group](https://www.linkedin.com/groups/1731317/)

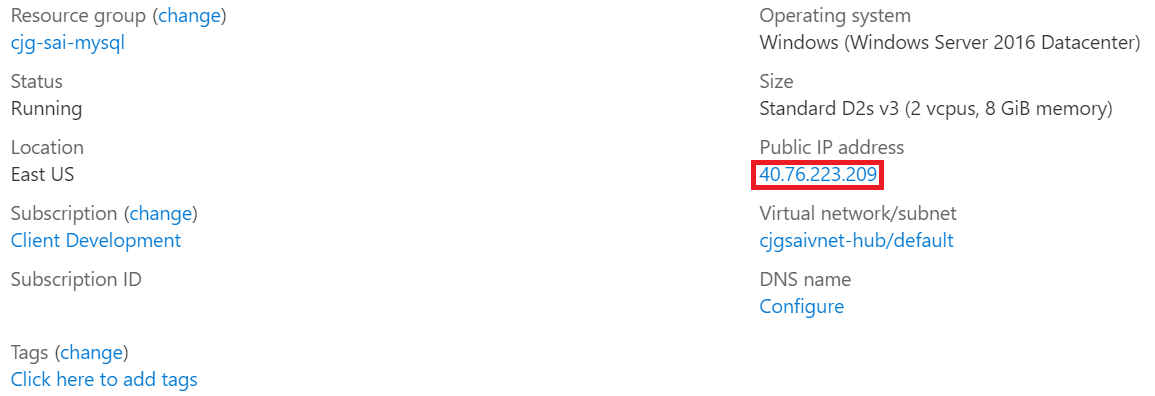
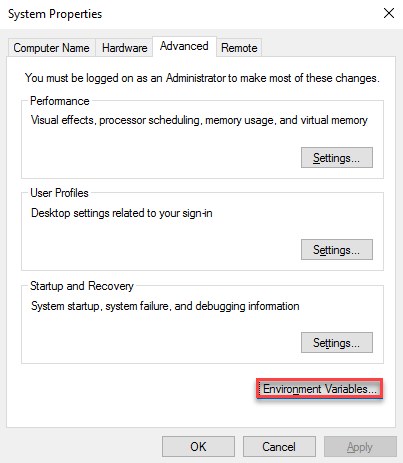
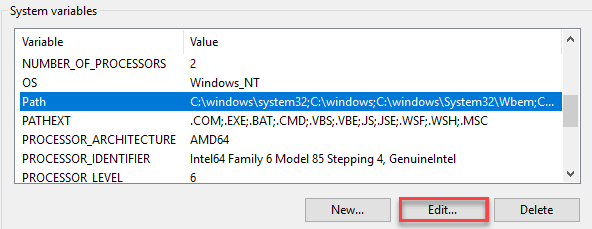
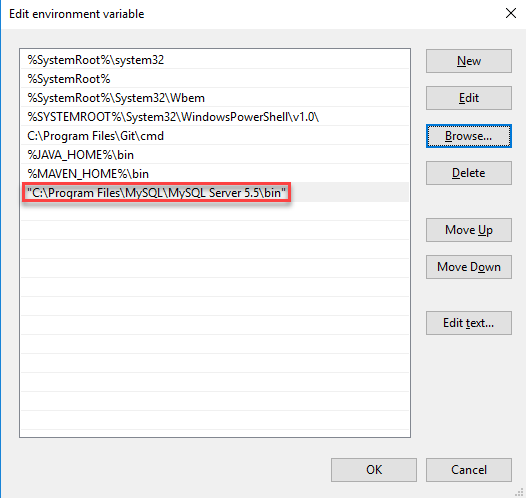
# Appendix A: Environment Setup

The following steps will configure an environment to perform the guide’s migration steps for the sample [conference demo application](https://github.com/Azure/azure-mysql/tree/master/MigrationGuide/sample-app).

## Deploy the ARM template

* Open the Azure Portal
* Create a new resource group
* Select **+Add**, type **template**, select the **Template Deployment…**
* 
* Select **Create**
* Select **Build your own template in the editor**
* 
* Choose between the secure or the non-secure ARM template. The difference between the two options is the secured option’s resources are hidden behind an App Gateway with private endpoints, whereas the other is directly exposed to the internet.
* Copy the json into the window
* Select **Save**
* 
* Fill in the parameters
  + Be sure to record your prefix and password, they are needed later
* Select **Review + create**
* Select the **I agree…** checkbox
* Select **Create**, after about 20 minutes the landing zone will be deployed

## Install MySQL 5.5

* Login to the deployed Dev VM
  + Browse to the Azure Portal
  + Select the VM
  + Copy the **Public IP Address**
  + 
  + Open Remote Desktop and connect to the VM IP Address.
  + Login using the username and password entered above.
* Download the following [MySQL community versions](https://dev.mysql.com/downloads/mysql/)
  + MySQL 5.5.x
  + MySQL 8.0.x
* Install MySQL 5.5
  + Select **Run.**
  + On the Welcome dialog, select **Next**.
  + Select **I accept…**, select **Next.**
  + On the Choose Setup Type dialog, select **Complete**, select **Next.**
  + Select **Install.**
  + When complete, select **Finish.**
  + The configuration wizard will start, select **Next.**
  + Select **Detailed Configuration,** then select **Next.**
  + Select **Server Machine**, select **Next.**
  + Select **Multifunctional database**, select **Next**
  + On the **InnoDB Tablespace Settings**, select **Next**
  + Select **Online Transaction Processing (OLTP)**, select **Next**
  + On the networking options, select **Next**
  + For the character set, select **Next**
  + On the windows service, select **Next**
  + Enter the root password, select the **Enable root access from remote machines**, select **Next**
  + Select **Execute**
  + Select **Finish**
* Add the \*\*C:FilesServer 5.5\* path to the PATH environment variable
  + Open Windows Explorer
  + Right-click **This PC**, select **Properties** and then **Advanced system settings**
  + Select **Environment Variables**
  + 
  + Under **System variables**, choose **Path**. Then, select **Edit…**
  + 
  + In the **Edit environment variable** dialog, select **New** and then **Browse…** Browse to C:\Program Files\MySQL\MySQL Server 5.5\bin.
  + 
  + Select **OK**.

## Install MySQL Workbench

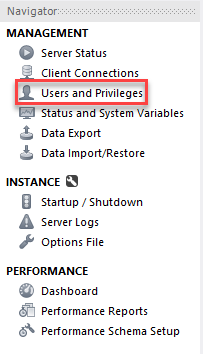
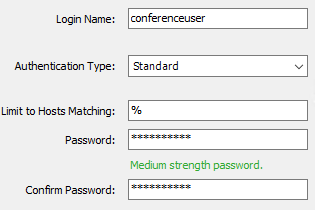
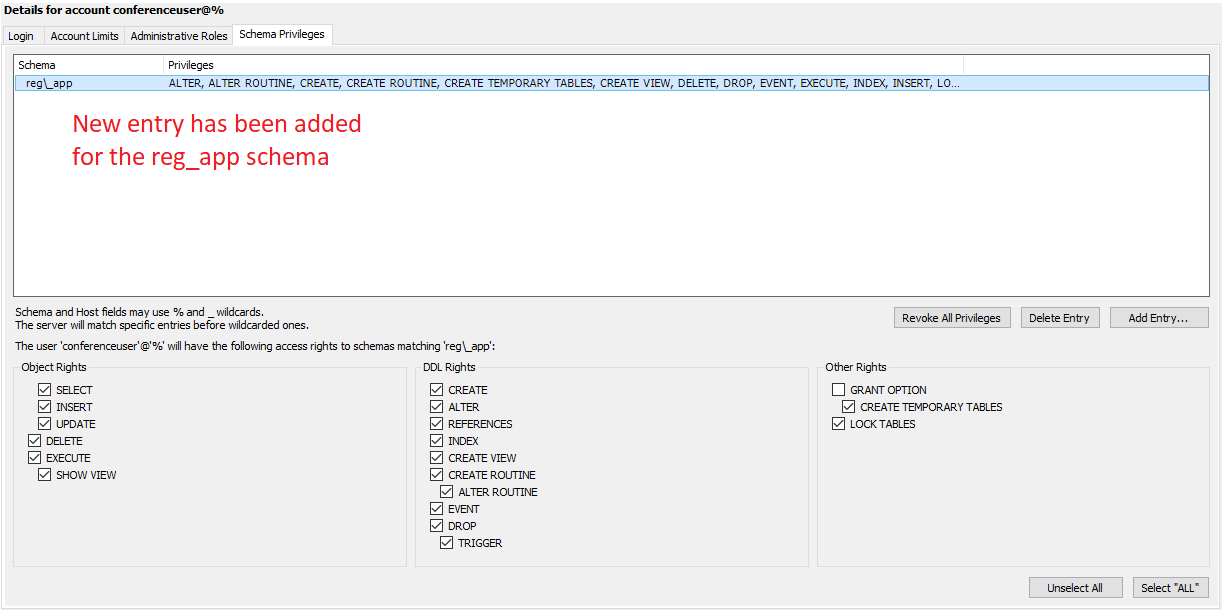
* Download the [MySQL Workbench installer](https://downloads.mysql.com/archives/workbench/)
* Install Visual C++ Redistributable for Visual Studio 2015, 2017, and 2019 from [here](https://support.microsoft.com/en-us/help/2977003/the-latest-supported-visual-c-downloads). Make sure to select the x64 installer. Run the installer, accept the software’s license terms, and close the installer.
* Install MySQL Workbench
  + Start the installer, select **Run.**
  + Select **Next**
  + Select **Next**
  + Select **Install**
  + Select **Finish**

## Download artifacts

* Download and Install [Git](https://git-scm.com/download/win)
  + Download and run the 64-bit installer
  + Click **Next** through all prompts
* Open a Windows PowerShell window (just by entering “PowerShell” into the Start menu) and run the following commands

mkdir c:\mysqlguide  
cd c:\mysqlguide  
git clone https://github.com/solliancenet/onprem-mysql-to-azuremysql-migration-guide

## Deploy the Database

* Open SQL Workbench
* Connect to your local MySQL instance (just select **Local instance MySQL** on the Welcome page)
* If prompted, select **Continue anyway**
* Create a new schema called **reg\_app**
  + Select the **Create schema** button
  + 
  + For the name, type **reg\_app**
  + Select **Apply**
  + In the dialog, select **Apply**
  + Select **Finish**
* Select **File->Open SQL Script**
* Browse to **C:\mysql-to-azuremysql-migration-guide-scripts**
* Select **conferencedemo-mysql** file, select **Open.**
* In the navigator, select the Schemas tab, double-click the **reg\_app** schema.
* Select **Execute** in the query editor
* 
* Create the database user
  + In the navigator, select the Administration tab.
  + Select **Users and Privileges**
  + 
  + Select **Add Account**
  + For the name, type **conferenceuser**
  + For the password, type a password
  + 
  + Select **Apply**
  + Select the **Schema Privileges** tab
  + Select **Add Entry**
  + Select the **reg\_app** schema, then select **OK**
  + Select **Select ALL**
  + Select **Apply.** View the following image to ensure it is configured correctly.
  + 

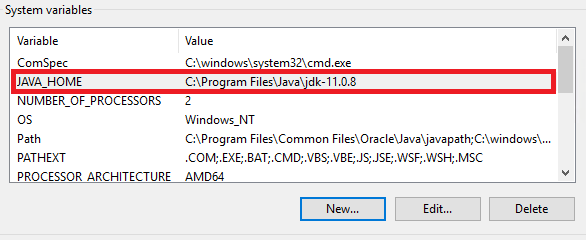
## Configure Blob Data

* Navigate to the **Schemas** tab under the **Navigator** window
* 
* If the **reg\_app** schema does not appear, then right-click anywhere within the pane and select **Refresh All**
* Expand **reg\_app > Tables**. Hover over the **speakers** table. Select the third button which appears. The **Result Grid** will show every record in the table
* For every record, change the value of the **SPEAKER\_PIC** field
  + Right-click each NULL entry and select **Load Value From File…**
  + Select **C:\mysql-to-azuremysql-migration-guide-scripts-images-bio-pic1.png**
  + Select **Open**
* Select **Apply** at the bottom right-hand corner of the page, and **Apply** again to confirm the query
* Repeat this process for the **EVENT\_PIC** field of the **events** table
  + This time, load **C:-mysql-to-azuremysql-migration-guide-scripts-images-pic0.png** (or **event-pic1.png**)
  + The image below shows BLOB data successfully loaded for the **events** table
  + 

## Setup Java and Maven

* Download and Install [Java Development Kit 11.0](https://www.oracle.com/java/technologies/javase-jdk11-downloads.html)

**Note:** Be sure to check what the highest SDK/JRE that is supported in Azure App Service before downloading the latest.

* Set the JAVA\_HOME environment variable to the **C:\Files-{version}** folder
  + Open the **Environment Variables** dialog box again (refer [here](#Install-MySQL-5.5) for more details)
  + Select **New** under **System variables**
  + Type **JAVA\_HOME**
  + Copy the path shown below, then select **OK.** The image below shows the correct configuration for the **JAVA\_HOME** environment variable.
  + 
* Download and install the Java Runtime
* Download and configure [Maven](https://maven.apache.org/download.cgi)
  + Download the zip archive
  + From the download location, right-click the zip archive and select **Extract All…**
  + Set the destination to **C:Files**. Then, select **Extract**
  + Set the M2\_HOME environment variable to the **C:Files-maven-{version}** folder
  + Add the **C:\Files-maven-3.6.3** path to the PATH environment variable

## Install Azure CLI

* Download and Install the [Azure CLI](https://aka.ms/installazurecliwindows)

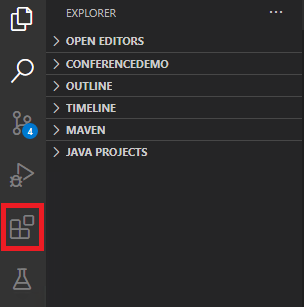
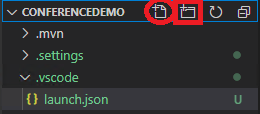
## Install NodeJS

* Download and Install [NodeJS](https://nodejs.org/en/download/). Select the LTS 64-bit MSI Installer.
  + Accept the default installation location
  + Make sure that the **Automatically install the necessary tools** box is not selected

## Install and Configure Visual Studio Code

* Download and Install [Visual Studio Code](https://code.visualstudio.com/download). Select the 64-bit Windows User Installer

## Configure the Web Application API

* Open Visual Studio Code
* Open the **C:\mysql-to-azuremysql-migration-guide\* folder (Ctrl+K and Ctrl+O, or** File->Open Folder…\*\*)
* Select the **Extensions** tab
* 
* Search for and install the following extensions
  + Java Extension Pack
  + Spring Initializer Java Support
* When prompted, select **Yes** to trust the **Maven Wrapper**
* Update the .vscode\launch.json file
  + If a launch.json does not exist, create a .vscode folder, and then create a new file called launch.json. The rectangle highlights the tool used to create a new folder, while the oval indicates the tool to create a new file.
  + 
  + Copy the following into it:
  + {  
     // Use IntelliSense to learn about possible attributes.  
     // Hover to view descriptions of existing attributes.  
     // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387  
     "version": "0.2.0",  
     "configurations": [  
     {  
     "type": "java",  
     "name": "Debug (Launch)",  
     "request": "launch",  
     "mainClass": "com.yourcompany.conferencedemo.ConferencedemoApplication",  
     "env" :{  
     "DB\_CONNECTION\_URL" : "jdbc:mysql://localhost:3306/reg\_app?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC&noAccessToProcedureBodies=true",  
     "DB\_USER\_NAME" : "conferenceuser",  
     "DB\_PASSWORD" : "Seattle123",  
     "ALLOWED\_ORIGINS" : "\*",  
     }  
     }  
     ]  
    }
  + Update the **{DB\_CONNECTION\_URL}** environment variable to the MySQL Connections string jdbc:mysql://localhost:3306/reg\_app?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC&noAccessToProcedureBodies=true
  + Update the **{DB\_USER\_NAME}** environment variable to the MySQL Connections string conferenceuser
  + Update the **{DB\_PASSWORD}** environment variable to the MySQL Connections string Seattle123
  + Update the **{ALLOWED\_ORIGINS}** environment variable to \*
* Select the **Debug** tab (directly above the **Extensions** tab from earlier), then select the debug option to start a debug session
* 
* If prompted, select **Yes** to switch to standard mode

## Test the Web Application

* Open a browser window, browse to **http://localhost:8888.**
* Ensure the application started on port 8888 and displays results.

## Configure the Web Application Client

* Open a new Visual Studio Code window to **C:\mysql-to-azuremysql-migration-guide-client**
* Open a terminal window (**Terminal**->**New Terminal**).
* Run the following commands in the terminal window to install all the needed packages, if prompted, select **N**

$env:Path = [System.Environment]::GetEnvironmentVariable("Path","Machine")

npm install  
npm install -g @angular/cli

**Note**: If PowerShell indicates that npm is not a recognized command, try restarting VS Code.

* Run the following commands to run the client application.

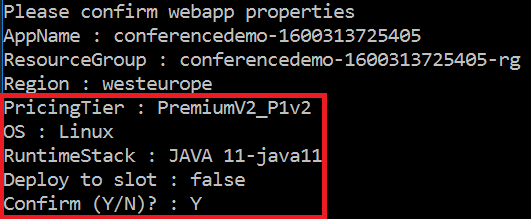
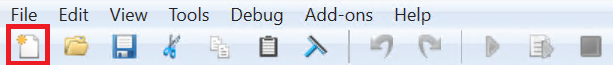
npm start

* Open a browser to the node site **http://localhost:{port}.**
* Browse the conference site, ensure sessions and speaker pages load.

## Deploy the Java Server Application to Azure

* Open a command prompt window.
* Run the following command to create the Maven configuration to deploy the app. Multiple packages will be installed from the Maven repository.

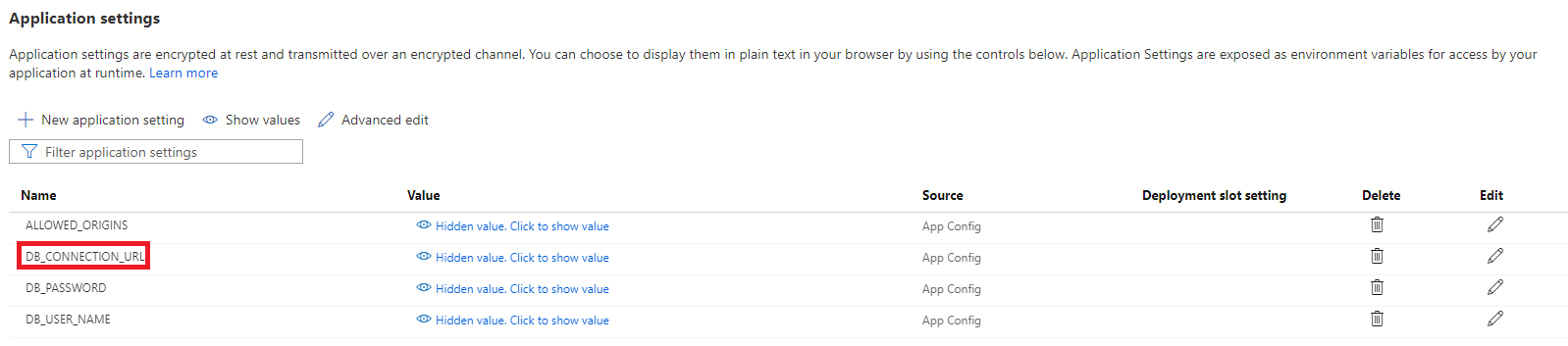
cd C:\mysqlguide\onprem-mysql-to-azuremysql-migration-guide\artifacts\testapp\conferencedemo  
"C:\Program Files\apache-maven-3.6.3\bin\mvn" com.microsoft.azure:azure-webapp-maven-plugin:1.9.1:config

* For the`Define value for OS(Default:Linux), select the option that corresponds to linux or press \*\*ENTER\*\*
* Select Java 11
* Type **Y** to confirm the settings, then press **ENTER**
* 
* Switch to Visual Studio and the \*\*ConferenceDemo\*\* project.
* Switch to the pom.xml file, notice the **com.microsoft.azure** groupId is now added
* Modify the resource group, appName and region to match the ones deployed in the ARM template
* If there is more than one subscription, set the specific subscriptionId in the [maven configuration](https://github.com/microsoft/azure-maven-plugins/wiki/Authentication#subscription)
* If the secure landing zone has been deployed, set the hosts file
  + Browse to your resource group, select the **PREFIXapi01** app service
  + Select **Networking**
  + Select **Configure your Private Endpoint connections**
  + Select the **PREFIXapi-pe** private endpoint.
  + Record the private IP Address.
  + Repeat for the **PREFIXapp01** app service
  + Open a Windows PowerShell ISE window.
  + Copy in the code from below, be sure to replace tokens, and save to **C:\mysqlguide\onprem-mysql-to-azuremysql-migration-guide\artifacts** as **ConfiguringHostsFile.ps1**.
  + $prefix = "{PREFIX}";  
    $apiip = "{APIIP}";  
    $app\_name = "$($prefix)api01";  
      
    $hostname = "$app\_name.azurewebsites.net"  
    $line = "$apiip`t$hostname"  
    add-content "c:\windows\system32\drivers\etc\hosts" $line  
      
    $hostname = "$app\_name.scm.azurewebsites.net"  
    $line = "$apiip`t$hostname"  
    add-content "c:\windows\system32\drivers\etc\hosts" $line  
      
    $appip = "{APPIP}"  
    $app\_name = "$($prefix)app01";  
    $hostname = "$app\_name.azurewebsites.net"  
    $line = "$appip`t$hostname"  
    add-content "c:\windows\system32\drivers\etc\hosts" $line  
      
    $hostname = "$app\_name.scm.azurewebsites.net"  
    $line = "$appip`t$hostname"  
    add-content "c:\windows\system32\drivers\etc\hosts" $line
    - To run the PowerShell script, open the PowerShell ISE
    - Create a new file
    - 
    - Copy in the code from above and save to **C:-mysql-to-azuremysql-migration-guide\* as** ConfiguringHostsFile.ps1\*\*
    - Run the file
    - 
* In the command prompt window from earlier, run the following to deploy the application.
* When prompted, login to the Azure Portal.

C:\Program Files\apache-maven-3.6.3\bin\mvn" package azure-webapp:deploy

* Update the App Service configuration variables by running the following, be sure to replace the tokens:

$prefix = "{PREFIX}";  
$app\_name = "$($prefix)api01";  
$rgName = "{RESOURCE-GROUP-NAME}";  
az login  
az account set --subscription "{SUBSCRIPTION-ID}"  
az webapp config appsettings set -g $rgName -n $app\_name --settings DB\_CONNECTION\_URL={DB\_CONNECTION\_URL}  
az webapp config appsettings set -g $rgName -n $app\_name --settings DB\_USER\_NAME={DB\_USER\_NAME}  
az webapp config appsettings set -g $rgName -n $app\_name --settings DB\_PASSWORD={DB\_PASSWORD}  
az webapp config appsettings set -g $rgName -n $app\_name --settings ALLOWED\_ORIGINS=\*

* **Note:** You will need to escape the ampersands in the connection string. You may consider inputting the value through Azure Portal as well. Navigate to the API App Service, and select **Configuration** under **Settings**. Then, under **Application settings**, manually enter the value.
* 
* Restart the Java API App Service by running the following.

az webapp restart -g $rgName -n $app\_name

## Deploy the Angular Web Application to Azure

* Switch to the Visual Studio Code window for the Angular app (Conferencedemo-client)
* Navigate to **src.prod.ts**.
* Set **webApiUrl** to **[JAVA APP SERVICE URL]/api/v1**

**Note**: the App service url will come from the App Gateway service blade if using the secure deployment, or the App Service blade if not using the secure deployment.

* Run the following command to package the client app:

ng build --prod

* Run the following commands in the Visual Code terminal window or a new PowerShell Window to zip and publish the client application, be sure to replace the tokens:

cd C:\mysqlguide\onprem-mysql-to-azuremysql-migration-guide\artifacts\testapp\conferencedemo-client\dist  
Compress-Archive -Path "./conference-client/\*" -DestinationPath "./confClient.zip"  
$prefix = "{PREFIX}";  
$app\_name = "$($prefix)app01";  
$rgName = "{RESOURCE-GROUP-NAME}";  
$subscription = "{SUBSCRIPTION-NAME}";  
  
az login

az account set --subscription "{SUBSCRIPTION-ID}"

az webapp stop --name $app\_name --resource-group $rgName --subscription $subscription  
az webapp deployment source config-zip --resource-group $rgName --name $app\_name --src "./confClient.zip" --subscription $subscription  
az webapp start --name $app\_name --resource-group $rgName --subscription $subscription

## Configure Network Security (Secure path)

* When attempting to connect to the database from the app service, an access denied message should be displayed. Add the app virtual network to the firewall of the Azure Database for MySQL
  + Browse to the Azure Portal
  + Select the target resource group
  + Select the {PREFIX}mysql resource
  + Select **Connection security**
  + Select the Allow access to all Azure Services toggle to Yes
  + Select **Save**

# Appendix B: ARM Templates

## Secure

This template will deploy all resources with private endpoints. This effectively removes any access to the PaaS services from the internet.

[ARM Template](https://github.com/Azure/azure-mysql/tree/master/MigrationGuide/arm-templates/ExampleWithMigration)

## Non-Secure

This template will deploy resources using standard deployment where all resources are available from the internet.

[ARM Template](https://github.com/Azure/azure-mysql/tree/master/MigrationGuide/arm-templates/ExampleWithMigrationSecure)

# Appendix C: Default server parameters MySQL 5.5

Variable\_name Value  
auto\_increment\_increment 1  
auto\_increment\_offset 1  
autocommit ON  
automatic\_sp\_privileges ON  
back\_log 50  
basedir C:\\Program Files\\MySQL\\MySQL Server 5.5\\  
big\_tables OFF  
binlog\_cache\_size 32768  
binlog\_direct\_non\_transactional\_updates OFF  
binlog\_format STATEMENT  
binlog\_stmt\_cache\_size 32768  
bulk\_insert\_buffer\_size 8388608  
character\_set\_client latin1  
character\_set\_connection latin1  
character\_set\_database latin1  
character\_set\_filesystem binary  
character\_set\_results latin1  
character\_set\_server latin1  
character\_set\_system utf8  
character\_sets\_dir C:\\Program Files\\MySQL\\MySQL Server 5.5\\share\\charsets\\  
collation\_connection latin1\_swedish\_ci  
collation\_database latin1\_swedish\_ci  
collation\_server latin1\_swedish\_ci  
completion\_type NO\_CHAIN  
concurrent\_insert AUTO  
connect\_timeout 10  
datadir C:\\ProgramData\\MySQL\\MySQL Server 5.5\\Data\\  
date\_format %Y-%m-%d  
datetime\_format %Y-%m-%d %H:%i:%s  
default\_storage\_engine InnoDB  
default\_week\_format 0  
delay\_key\_write ON  
delayed\_insert\_limit 100  
delayed\_insert\_timeout 300  
delayed\_queue\_size 1000  
div\_precision\_increment 4  
engine\_condition\_pushdown ON  
event\_scheduler OFF  
expire\_logs\_days 0  
flush OFF  
flush\_time 1800  
foreign\_key\_checks ON  
ft\_boolean\_syntax + -><()~\*:""&|  
ft\_max\_word\_len 84  
ft\_min\_word\_len 4  
ft\_query\_expansion\_limit 20  
ft\_stopword\_file (built-in)  
general\_log OFF  
general\_log\_file cjg-vm-dev.log  
group\_concat\_max\_len 1024  
have\_compress YES  
have\_crypt NO  
have\_csv YES  
have\_dynamic\_loading YES  
have\_geometry YES  
have\_innodb YES  
have\_ndbcluster NO  
have\_openssl DISABLED  
have\_partitioning YES  
have\_profiling YES  
have\_query\_cache YES  
have\_rtree\_keys YES  
have\_ssl DISABLED  
have\_symlink YES  
hostname cjg-vm-dev  
ignore\_builtin\_innodb OFF  
init\_connect   
init\_file   
init\_slave   
innodb\_adaptive\_flushing ON  
innodb\_adaptive\_hash\_index ON  
innodb\_additional\_mem\_pool\_size 2097152  
innodb\_autoextend\_increment 8  
innodb\_autoinc\_lock\_mode 1  
innodb\_buffer\_pool\_instances 1  
innodb\_buffer\_pool\_size 85983232  
innodb\_change\_buffering all  
innodb\_checksums ON  
innodb\_commit\_concurrency 0  
innodb\_concurrency\_tickets 500  
innodb\_data\_file\_path ibdata1:10M:autoextend  
innodb\_data\_home\_dir   
innodb\_doublewrite ON  
innodb\_fast\_shutdown 1  
innodb\_file\_format Barracuda  
innodb\_file\_format\_check ON  
innodb\_file\_format\_max Barracuda  
innodb\_file\_per\_table ON  
innodb\_flush\_log\_at\_trx\_commit 1  
innodb\_flush\_method   
innodb\_force\_load\_corrupted OFF  
innodb\_force\_recovery 0  
innodb\_io\_capacity 200  
innodb\_large\_prefix OFF  
innodb\_lock\_wait\_timeout 50  
innodb\_locks\_unsafe\_for\_binlog OFF  
innodb\_log\_buffer\_size 1048576  
innodb\_log\_file\_size 44040192  
innodb\_log\_files\_in\_group 2  
innodb\_log\_group\_home\_dir .\\  
innodb\_max\_dirty\_pages\_pct 75  
innodb\_max\_purge\_lag 0  
innodb\_mirrored\_log\_groups 1  
innodb\_old\_blocks\_pct 37  
innodb\_old\_blocks\_time 0  
innodb\_open\_files 300  
innodb\_print\_all\_deadlocks OFF  
innodb\_purge\_batch\_size 20  
innodb\_purge\_threads 0  
innodb\_random\_read\_ahead OFF  
innodb\_read\_ahead\_threshold 56  
innodb\_read\_io\_threads 4  
innodb\_replication\_delay 0  
innodb\_rollback\_on\_timeout OFF  
innodb\_rollback\_segments 128  
innodb\_spin\_wait\_delay 6  
innodb\_stats\_method nulls\_equal  
innodb\_stats\_on\_metadata ON  
innodb\_stats\_sample\_pages 8  
innodb\_strict\_mode OFF  
innodb\_support\_xa ON  
innodb\_sync\_spin\_loops 30  
innodb\_table\_locks ON  
innodb\_thread\_concurrency 8  
innodb\_thread\_sleep\_delay 10000  
innodb\_use\_native\_aio ON  
innodb\_use\_sys\_malloc ON  
innodb\_version 5.5.60  
innodb\_write\_io\_threads 4  
interactive\_timeout 28800  
join\_buffer\_size 131072  
keep\_files\_on\_create OFF  
key\_buffer\_size 8388608  
key\_cache\_age\_threshold 300  
key\_cache\_block\_size 1024  
key\_cache\_division\_limit 100  
large\_files\_support ON  
large\_page\_size 0  
large\_pages OFF  
lc\_messages en\_US  
lc\_messages\_dir C:\\Program Files\\MySQL\\MySQL Server 5.5\\share\\  
lc\_time\_names en\_US  
license GPL  
local\_infile ON  
lock\_wait\_timeout 31536000  
log OFF  
log\_bin OFF  
log\_bin\_trust\_function\_creators OFF  
log\_error .\\cjg-vm-dev.err  
log\_output FILE  
log\_queries\_not\_using\_indexes OFF  
log\_slave\_updates OFF  
log\_slow\_queries ON  
log\_warnings 1  
long\_query\_time 10.000000  
low\_priority\_updates OFF  
lower\_case\_file\_system ON  
lower\_case\_table\_names 1  
max\_allowed\_packet 4194304  
max\_binlog\_cache\_size 18446744073709547520  
max\_binlog\_size 1073741824  
max\_binlog\_stmt\_cache\_size 18446744073709547520  
max\_connect\_errors 10  
max\_connections 100  
max\_delayed\_threads 20  
max\_error\_count 64  
max\_heap\_table\_size 16777216  
max\_insert\_delayed\_threads 20  
max\_join\_size 18446744073709551615  
max\_length\_for\_sort\_data 1024  
max\_long\_data\_size 4194304  
max\_prepared\_stmt\_count 16382  
max\_relay\_log\_size 0  
max\_seeks\_for\_key 4294967295  
max\_sort\_length 1024  
max\_sp\_recursion\_depth 0  
max\_tmp\_tables 32  
max\_user\_connections 0  
max\_write\_lock\_count 4294967295  
metadata\_locks\_cache\_size 1024  
min\_examined\_row\_limit 0  
multi\_range\_count 256  
myisam\_data\_pointer\_size 6  
myisam\_max\_sort\_file\_size 107374182400  
myisam\_mmap\_size 18446744073709551615  
myisam\_recover\_options OFF  
myisam\_repair\_threads 1  
myisam\_sort\_buffer\_size 36700160  
myisam\_stats\_method nulls\_unequal  
myisam\_use\_mmap OFF  
named\_pipe OFF  
net\_buffer\_length 16384  
net\_read\_timeout 30  
net\_retry\_count 10  
net\_write\_timeout 60  
new OFF  
old OFF  
old\_alter\_table OFF  
old\_passwords OFF  
open\_files\_limit 6158  
optimizer\_prune\_level 1  
optimizer\_search\_depth 62  
optimizer\_switch index\_merge=on,index\_merge\_union=on,index\_merge\_sort\_union=on,index\_merge\_intersection=on,engine\_condition\_pushdown=on  
performance\_schema OFF  
performance\_schema\_events\_waits\_history\_long\_size 10000  
performance\_schema\_events\_waits\_history\_size 10  
performance\_schema\_max\_cond\_classes 80  
performance\_schema\_max\_cond\_instances 1000  
performance\_schema\_max\_file\_classes 50  
performance\_schema\_max\_file\_handles 32768  
performance\_schema\_max\_file\_instances 10000  
performance\_schema\_max\_mutex\_classes 200  
performance\_schema\_max\_mutex\_instances 1000000  
performance\_schema\_max\_rwlock\_classes 30  
performance\_schema\_max\_rwlock\_instances 1000000  
performance\_schema\_max\_table\_handles 100000  
performance\_schema\_max\_table\_instances 50000  
performance\_schema\_max\_thread\_classes 50  
performance\_schema\_max\_thread\_instances 1000  
pid\_file C:\\ProgramData\\MySQL\\MySQL Server 5.5\\Data\\cjg-vm-dev.pid  
plugin\_dir C:\\Program Files\\MySQL\\MySQL Server 5.5\\lib\\plugin\\  
port 3306  
preload\_buffer\_size 32768  
profiling OFF  
profiling\_history\_size 15  
protocol\_version 10  
query\_alloc\_block\_size 8192  
query\_cache\_limit 1048576  
query\_cache\_min\_res\_unit 4096  
query\_cache\_size 0  
query\_cache\_type ON  
query\_cache\_wlock\_invalidate OFF  
query\_prealloc\_size 8192  
range\_alloc\_block\_size 4096  
read\_buffer\_size 65536  
read\_only OFF  
read\_rnd\_buffer\_size 262144  
relay\_log   
relay\_log\_index   
relay\_log\_info\_file relay-log.info  
relay\_log\_purge ON  
relay\_log\_recovery OFF  
relay\_log\_space\_limit 0  
report\_host   
report\_password   
report\_port 3306  
report\_user   
rpl\_recovery\_rank 0  
secure\_auth OFF  
secure\_file\_priv C:\\ProgramData\\MySQL\\MySQL Server 5.5\\Uploads\\  
server\_id 1  
shared\_memory OFF  
shared\_memory\_base\_name MYSQL  
skip\_external\_locking ON  
skip\_name\_resolve OFF  
skip\_networking OFF  
skip\_show\_database OFF  
slave\_compressed\_protocol OFF  
slave\_exec\_mode STRICT  
slave\_load\_tmpdir C:\\Windows\\SERVIC~2\\NETWOR~1\\AppData\\Local\\Temp  
slave\_max\_allowed\_packet 1073741824  
slave\_net\_timeout 3600  
slave\_skip\_errors OFF  
slave\_transaction\_retries 10  
slave\_type\_conversions   
slow\_launch\_time 2  
slow\_query\_log ON  
slow\_query\_log\_file cjg-vm-dev-slow.log  
socket MySQL  
sort\_buffer\_size 262144  
sql\_auto\_is\_null OFF  
sql\_big\_selects ON  
sql\_big\_tables OFF  
sql\_buffer\_result OFF  
sql\_log\_bin ON  
sql\_log\_off OFF  
sql\_low\_priority\_updates OFF  
sql\_max\_join\_size 18446744073709551615  
sql\_mode STRICT\_TRANS\_TABLES,NO\_AUTO\_CREATE\_USER,NO\_ENGINE\_SUBSTITUTION  
sql\_notes ON  
sql\_quote\_show\_create ON  
sql\_safe\_updates OFF  
sql\_select\_limit 18446744073709551615  
sql\_slave\_skip\_counter 0  
sql\_warnings OFF  
ssl\_ca   
ssl\_capath   
ssl\_cert   
ssl\_cipher   
ssl\_key   
storage\_engine InnoDB  
stored\_program\_cache 256  
sync\_binlog 0  
sync\_frm ON  
sync\_master\_info 0  
sync\_relay\_log 0  
sync\_relay\_log\_info 0  
system\_time\_zone Coordinated Universal Time  
table\_definition\_cache 400  
table\_open\_cache 2000  
thread\_cache\_size 8  
thread\_concurrency 10  
thread\_handling one-thread-per-connection  
thread\_stack 262144  
time\_format %H:%i:%s  
time\_zone SYSTEM  
timed\_mutexes OFF  
tmp\_table\_size 18874368  
tmpdir C:\\Windows\\SERVIC~2\\NETWOR~1\\AppData\\Local\\Temp  
transaction\_alloc\_block\_size 8192  
transaction\_prealloc\_size 4096  
tx\_isolation REPEATABLE-READ  
unique\_checks ON  
updatable\_views\_with\_limit YES  
version 5.5.60-log  
version\_comment MySQL Community Server (GPL)  
version\_compile\_machine AMD64  
version\_compile\_os Win64  
wait\_timeout 28800

# Appendix D: Default server parameters Azure

Variable\_name Value  
aad\_audience https://ossrdbms-aad.database.windows.net  
aad\_auth\_validate\_oids\_in\_tenant ON  
aad\_min\_fetch\_signature\_sec 1800  
aad\_tenant\_id   
aad\_token\_endpoint https://sts.windows.net/  
activate\_all\_roles\_on\_login OFF  
admin\_address   
admin\_port 33062  
allowed\_ip\_cache\_size 1000  
audit\_log\_enabled OFF  
audit\_log\_events CONNECTION  
audit\_log\_exclude\_users azure\_superuser  
audit\_log\_include\_users   
audit\_pii\_login\_enabled ON  
audit\_slow\_log\_enabled ON  
auto\_generate\_certs ON  
auto\_increment\_increment 1  
auto\_increment\_offset 1  
autocommit ON  
automatic\_sp\_privileges ON  
avoid\_temporal\_upgrade OFF  
azure\_ia\_enabled ON  
azure\_ip\_address\_file C:\\work\\azure\_ip\_address.csv  
azure\_primary\_server\_ca\_file C:\\work\\primary\_ca.pem  
azure\_primary\_server\_info\_file C:\\work\\primary\_info  
azure\_replication\_enabled ON  
back\_log 725  
basedir c:\\mysql\\  
big\_tables OFF  
bind\_address 0.0.0.0  
binlog\_cache\_size 1048576  
binlog\_checksum CRC32  
binlog\_direct\_non\_transactional\_updates OFF  
binlog\_encryption OFF  
binlog\_error\_action ABORT\_SERVER  
binlog\_expire\_logs\_seconds 0  
binlog\_format ROW  
binlog\_group\_commit\_sync\_delay 0  
binlog\_group\_commit\_sync\_no\_delay\_count 0  
binlog\_gtid\_simple\_recovery ON  
binlog\_index\_doublewrite ON  
binlog\_max\_flush\_queue\_time 0  
binlog\_order\_commits ON  
binlog\_rotate\_encryption\_master\_key\_at\_startup OFF  
binlog\_row\_event\_max\_size 1048576  
binlog\_row\_image MINIMAL  
binlog\_row\_metadata MINIMAL  
binlog\_row\_value\_options   
binlog\_rows\_query\_log\_events OFF  
binlog\_stmt\_cache\_size 32768  
binlog\_thd\_synclock\_enabled ON  
binlog\_transaction\_dependency\_history\_size 25000  
binlog\_transaction\_dependency\_tracking COMMIT\_ORDER  
block\_encryption\_mode aes-128-ecb  
bulk\_insert\_buffer\_size 8388608  
bypass\_firewall\_rule\_from\_socket\_option\_enabled ON  
caching\_sha2\_password\_auto\_generate\_rsa\_keys ON  
caching\_sha2\_password\_private\_key\_path private\_key.pem  
caching\_sha2\_password\_public\_key\_path public\_key.pem  
character\_set\_client latin1  
character\_set\_connection latin1  
character\_set\_database latin1  
character\_set\_filesystem binary  
character\_set\_results latin1  
character\_set\_server latin1  
character\_set\_system utf8  
character\_sets\_dir c:\\mysql\\share\\charsets\\  
check\_proxy\_users OFF  
collation\_connection latin1\_swedish\_ci  
collation\_database latin1\_swedish\_ci  
collation\_server latin1\_swedish\_ci  
completion\_type NO\_CHAIN  
concurrent\_insert AUTO  
connect\_timeout 10  
connection\_log\_enabled OFF  
core\_file OFF  
create\_admin\_listener\_thread OFF  
cte\_max\_recursion\_depth 1000  
datadir c:\\mysqldata\\  
deactivate\_idle\_server\_disabled ON  
default\_authentication\_plugin mysql\_native\_password  
default\_collation\_for\_utf8mb4 utf8mb4\_0900\_ai\_ci  
default\_password\_lifetime 0  
default\_storage\_engine InnoDB  
default\_tmp\_storage\_engine InnoDB  
default\_week\_format 0  
delay\_key\_write ON  
delayed\_insert\_limit 100  
delayed\_insert\_timeout 300  
delayed\_queue\_size 1000  
disabled\_storage\_engines MyISAM,MRG\_MyISAM  
disconnect\_on\_expired\_password ON  
div\_precision\_increment 4  
end\_markers\_in\_json OFF  
enforce\_gtid\_consistency OFF  
eq\_range\_index\_dive\_limit 200  
event\_scheduler OFF  
expire\_logs\_days 0  
expire\_logs\_disabled ON  
explicit\_defaults\_for\_timestamp ON  
firewall\_rule\_enabled ON  
flush OFF  
flush\_time 0  
foreign\_key\_checks ON  
ft\_boolean\_syntax + -><()~\*:""&|  
ft\_max\_word\_len 84  
ft\_min\_word\_len 4  
ft\_query\_expansion\_limit 20  
ft\_stopword\_file (built-in)  
general\_log OFF  
general\_log\_file c:\\work\\serverlogs\\mysql-general-cjgmysql-2020090902.log  
group\_concat\_max\_len 1024  
group\_replication\_consistency EVENTUAL  
gtid\_executed   
gtid\_executed\_compression\_period 1000  
gtid\_mode OFF  
gtid\_owned   
gtid\_purged   
have\_compress YES  
have\_dynamic\_loading YES  
have\_geometry YES  
have\_openssl YES  
have\_profiling YES  
have\_query\_cache NO  
have\_rtree\_keys YES  
have\_schannel ON  
have\_ssl YES  
have\_statement\_timeout YES  
have\_symlink DISABLED  
histogram\_generation\_max\_mem\_size 20000000  
host\_cache\_size 634  
hostname CLIENT  
information\_schema\_stats\_expiry 86400  
init\_connect   
init\_file   
init\_slave   
innodb\_adaptive\_flushing ON  
innodb\_adaptive\_flushing\_lwm 10  
innodb\_adaptive\_hash\_index ON  
innodb\_adaptive\_hash\_index\_parts 8  
innodb\_adaptive\_max\_sleep\_delay 150000  
innodb\_aggressive\_pc ON  
innodb\_aio\_write\_slots 40  
innodb\_api\_bk\_commit\_interval 5  
innodb\_api\_disable\_rowlock OFF  
innodb\_api\_enable\_binlog OFF  
innodb\_api\_enable\_mdl OFF  
innodb\_api\_trx\_level 0  
innodb\_autoextend\_increment 64  
innodb\_autoinc\_lock\_mode 1  
innodb\_avoid\_logwrite ON  
innodb\_buffer\_pool\_chunk\_size 67108864  
innodb\_buffer\_pool\_dump\_at\_shutdown OFF  
innodb\_buffer\_pool\_dump\_now OFF  
innodb\_buffer\_pool\_dump\_pct 25  
innodb\_buffer\_pool\_filename ib\_buffer\_pool  
innodb\_buffer\_pool\_in\_core\_file ON  
innodb\_buffer\_pool\_instances 8  
innodb\_buffer\_pool\_load\_abort OFF  
innodb\_buffer\_pool\_load\_at\_startup OFF  
innodb\_buffer\_pool\_load\_now OFF  
innodb\_buffer\_pool\_size 16106127360  
innodb\_change\_buffer\_max\_size 25  
innodb\_change\_buffering all  
innodb\_checksum\_algorithm crc32  
innodb\_chunk\_write\_enabled ON  
innodb\_cmp\_per\_index\_enabled OFF  
innodb\_commit\_concurrency 0  
innodb\_compression\_failure\_threshold\_pct 5  
innodb\_compression\_level 6  
innodb\_compression\_pad\_pct\_max 50  
innodb\_concurrency\_tickets 5000  
innodb\_data\_file\_path ibdata1:12M:autoextend  
innodb\_data\_home\_dir   
innodb\_deadlock\_detect ON  
innodb\_dedicated\_server OFF  
innodb\_default\_row\_format dynamic  
innodb\_delay\_datasync ON  
innodb\_delay\_logwrite ON  
innodb\_directories   
innodb\_dirty\_page\_fflush 10  
innodb\_disable\_sort\_file\_cache OFF  
innodb\_diskfull\_wa\_enabled ON  
innodb\_doublewrite OFF  
innodb\_dynamic\_page\_cleaners ON  
innodb\_fast0ext\_enabled ON  
innodb\_fast\_shutdown 1  
innodb\_fatal\_semaphore\_wait\_threshold 600  
innodb\_file\_per\_table ON  
innodb\_filesize\_update\_interval 30  
innodb\_fill\_factor 100  
innodb\_flush\_data\_file\_before\_checkpoint ON  
innodb\_flush\_log\_at\_timeout 1  
innodb\_flush\_log\_at\_trx\_commit 1  
innodb\_flush\_log\_checkpoint ON  
innodb\_flush\_method unbuffered  
innodb\_flush\_neighbors 0  
innodb\_flush\_sync ON  
innodb\_flushing\_avg\_loops 30  
innodb\_force\_load\_corrupted OFF  
innodb\_force\_pacing ON  
innodb\_force\_recovery 0  
innodb\_fs\_block\_size 8192  
innodb\_fsync\_threshold 0  
innodb\_ft\_aux\_table   
innodb\_ft\_cache\_size 8000000  
innodb\_ft\_enable\_diag\_print OFF  
innodb\_ft\_enable\_stopword ON  
innodb\_ft\_max\_token\_size 84  
innodb\_ft\_min\_token\_size 3  
innodb\_ft\_num\_word\_optimize 2000  
innodb\_ft\_result\_cache\_limit 2000000000  
innodb\_ft\_server\_stopword\_table   
innodb\_ft\_sort\_pll\_degree 2  
innodb\_ft\_total\_cache\_size 640000000  
innodb\_ft\_user\_stopword\_table   
innodb\_harness\_checkpoint ON  
innodb\_ignore\_fsctl\_set\_sparse ON  
innodb\_ignore\_fsctl\_set\_zero\_data ON  
innodb\_inflight\_syncio 100  
innodb\_io\_capacity 200  
innodb\_io\_capacity\_calibceil 300  
innodb\_io\_capacity\_calibrate 2  
innodb\_io\_capacity\_max 2000  
innodb\_io\_hosed\_threshold 2048000  
innodb\_io\_throttled\_threshold 2048000  
innodb\_list\_dir\_files\_name\_only ON  
innodb\_lock\_wait\_timeout 50  
innodb\_log\_buffer\_size 16777216  
innodb\_log\_checkpoint\_every 1000  
innodb\_log\_checksums ON  
innodb\_log\_closer\_spin\_delay 0  
innodb\_log\_closer\_timeout 1000  
innodb\_log\_compressed\_pages ON  
innodb\_log\_file\_size 268435456  
innodb\_log\_files\_in\_group 2  
innodb\_log\_flush\_events 2048  
innodb\_log\_flush\_notifier\_spin\_delay 0  
innodb\_log\_flush\_notifier\_timeout 10  
innodb\_log\_flusher\_spin\_delay 250000  
innodb\_log\_flusher\_timeout 10  
innodb\_log\_group\_home\_dir .\\  
innodb\_log\_recent\_closed\_size 2097152  
innodb\_log\_recent\_written\_size 33554432  
innodb\_log\_spin\_cpu\_abs\_lwm 80  
innodb\_log\_spin\_cpu\_pct\_hwm 50  
innodb\_log\_wait\_for\_flush\_spin\_delay 25000  
innodb\_log\_wait\_for\_flush\_spin\_hwm 400  
innodb\_log\_wait\_for\_flush\_timeout 1000  
innodb\_log\_wait\_for\_write\_spin\_delay 25000  
innodb\_log\_wait\_for\_write\_timeout 1000  
innodb\_log\_write\_ahead\_size 16384  
innodb\_log\_write\_backoff ON  
innodb\_log\_write\_events 2048  
innodb\_log\_write\_max\_size 16777216  
innodb\_log\_write\_notifier\_spin\_delay 0  
innodb\_log\_write\_notifier\_timeout 10  
innodb\_log\_writer\_spin\_delay 250000  
innodb\_log\_writer\_timeout 10  
innodb\_logwr\_prio\_checkpoint 2  
innodb\_lru\_scan\_depth 1024  
innodb\_max\_dirty\_pages\_pct 90.000000  
innodb\_max\_dirty\_pages\_pct\_lwm 10.000000  
innodb\_max\_purge\_lag 0  
innodb\_max\_purge\_lag\_delay 0  
innodb\_max\_undo\_log\_size 1073741824  
innodb\_min\_tsc\_freq 2000  
innodb\_monitor\_disable   
innodb\_monitor\_enable   
innodb\_monitor\_reset   
innodb\_monitor\_reset\_all   
innodb\_old\_blocks\_pct 37  
innodb\_old\_blocks\_time 1000  
innodb\_online\_alter\_log\_max\_size 134217728  
innodb\_open\_files 5000  
innodb\_optimize\_fulltext\_only OFF  
innodb\_page\_cleaner\_priority 4  
innodb\_page\_cleaners 4  
innodb\_page\_cleaners\_enforced 3  
innodb\_page\_size 16384  
innodb\_parallel\_read\_threads 4  
innodb\_print\_all\_deadlocks OFF  
innodb\_print\_ddl\_logs OFF  
innodb\_purge\_batch\_size 300  
innodb\_purge\_rseg\_truncate\_frequency 128  
innodb\_purge\_threads 4  
innodb\_random\_read\_ahead OFF  
innodb\_read\_ahead\_threshold 56  
innodb\_read\_io\_threads 4  
innodb\_read\_io\_threads\_enforced 1  
innodb\_read\_only OFF  
innodb\_read\_throttle\_hint OFF  
innodb\_redo\_log\_encrypt OFF  
innodb\_replication\_delay 0  
innodb\_rollback\_on\_timeout OFF  
innodb\_rollback\_segments 128  
innodb\_skip\_check\_ibd OFF  
innodb\_sort\_buffer\_size 1048576  
innodb\_spin\_wait\_delay 6  
innodb\_standalone\_systmp ON  
innodb\_stats\_auto\_recalc ON  
innodb\_stats\_include\_delete\_marked OFF  
innodb\_stats\_method nulls\_equal  
innodb\_stats\_on\_metadata OFF  
innodb\_stats\_persistent ON  
innodb\_stats\_persistent\_sample\_pages 20  
innodb\_stats\_transient\_sample\_pages 8  
innodb\_status\_output OFF  
innodb\_status\_output\_locks OFF  
innodb\_strict\_mode ON  
innodb\_sync\_array\_size 1  
innodb\_sync\_log\_slots 1  
innodb\_sync\_spin\_loops 30  
innodb\_table\_locks ON  
innodb\_temp\_data\_file\_path C:\\temp\\ibtmp1:12M:autoextend  
innodb\_temp\_tablespaces\_dir C:\\temp\\  
innodb\_thread\_concurrency 0  
innodb\_thread\_sleep\_delay 10000  
innodb\_tmpdir C:\\temp  
innodb\_undo\_directory .\\  
innodb\_undo\_log\_encrypt OFF  
innodb\_undo\_log\_truncate ON  
innodb\_undo\_tablespaces 2  
innodb\_use\_enforced ON  
innodb\_use\_native\_aio ON  
innodb\_version 8.0.15  
innodb\_write\_io\_threads 4  
innodb\_write\_io\_threads\_enforced 2  
innodb\_write\_throttle\_hint OFF  
interactive\_timeout 28800  
internal\_tmp\_disk\_storage\_engine InnoDB  
internal\_tmp\_mem\_storage\_engine TempTable  
join\_buffer\_size 262144  
keep\_files\_on\_create OFF  
key\_buffer\_size 16777216  
key\_cache\_age\_threshold 300  
key\_cache\_block\_size 1024  
key\_cache\_division\_limit 100  
keyring\_operations ON  
large\_files\_support ON  
large\_page\_size 0  
large\_pages OFF  
lc\_messages en\_US  
lc\_messages\_dir c:\\mysql\\share\\  
lc\_time\_names en\_US  
license GPL  
list\_dir\_files\_name\_only ON  
local\_infile ON  
lock\_wait\_timeout 31536000  
log\_bin ON  
log\_bin\_basename c:\\work\\binlogs\\mysql-bin  
log\_bin\_index c:\\mysqldata\\mysql-bin.index  
log\_bin\_trust\_function\_creators OFF  
log\_bin\_use\_v1\_row\_events OFF  
log\_error stderr  
log\_error\_services log\_filter\_internal; log\_sink\_internal  
log\_error\_suppression\_list   
log\_error\_verbosity 3  
log\_output FILE  
log\_queries\_not\_using\_indexes OFF  
log\_slave\_updates ON  
log\_slow\_admin\_statements OFF  
log\_slow\_extra OFF  
log\_slow\_slave\_statements OFF  
log\_statements\_unsafe\_for\_binlog ON  
log\_throttle\_queries\_not\_using\_indexes 0  
log\_timestamps UTC  
logical\_server\_name cjgmysql  
login\_bucket\_factor\_bias 0.400000  
login\_bucket\_reset\_time 120  
login\_bucket\_restore\_factor 2.000000  
login\_bucket\_update\_time 20  
long\_query\_time 10.000000  
low\_priority\_updates OFF  
lower\_case\_file\_system ON  
lower\_case\_table\_names 1  
mandatory\_roles   
master\_info\_repository TABLE  
master\_verify\_checksum OFF  
max\_allowed\_packet 536870912  
max\_binlog\_cache\_size 18446744073709547520  
max\_binlog\_size 104857600  
max\_binlog\_stmt\_cache\_size 18446744073709547520  
max\_connect\_errors 100  
max\_connections 635  
max\_delayed\_threads 20  
max\_digest\_length 1024  
max\_error\_count 1024  
max\_execution\_time 0  
max\_heap\_table\_size 16777216  
max\_insert\_delayed\_threads 20  
max\_join\_size 18446744073709551615  
max\_length\_for\_sort\_data 1024  
max\_login\_bucket\_limit 2147483647  
max\_points\_in\_geometry 65536  
max\_prepared\_stmt\_count 16382  
max\_relay\_log\_size 104857600  
max\_seeks\_for\_key 4294967295  
max\_sort\_length 1024  
max\_sp\_recursion\_depth 0  
max\_super\_acl\_connections 30  
max\_superuser\_connections 0  
max\_user\_connections 0  
max\_write\_lock\_count 4294967295  
mem\_table\_binlog\_dml\_disabled OFF  
min\_examined\_row\_limit 0  
mts\_crash\_safety\_enabled OFF  
multiple\_processor\_group\_enabled ON  
myisam\_data\_pointer\_size 6  
myisam\_max\_sort\_file\_size 2146435072  
myisam\_mmap\_size 18446744073709551615  
myisam\_recover\_options OFF  
myisam\_repair\_threads 1  
myisam\_sort\_buffer\_size 8388608  
myisam\_stats\_method nulls\_unequal  
myisam\_use\_mmap OFF  
mysql\_native\_password\_proxy\_users OFF  
mysql\_stack\_trace\_enabled OFF  
named\_pipe OFF  
named\_pipe\_full\_access\_group \*everyone\*  
net\_buffer\_length 8192  
net\_read\_timeout 120  
net\_retry\_count 10  
net\_write\_timeout 240  
new OFF  
ngram\_token\_size 2  
nonblocking\_socket\_enabled ON  
normal\_log\_header\_enabled OFF  
offline\_mode OFF  
old OFF  
old\_alter\_table OFF  
open\_files\_limit 12693  
optimize\_create\_log\_file0\_enabled OFF  
optimize\_create\_log\_file1\_enabled OFF  
optimizer\_prune\_level 1  
optimizer\_search\_depth 62  
optimizer\_switch index\_merge=on,index\_merge\_union=on,index\_merge\_sort\_union=on,index\_merge\_intersection=on,engine\_condition\_pushdown=on,index\_condition\_pushdown=on,mrr=on,mrr\_cost\_based=on,block\_nested\_loop=on,batched\_key\_access=off,materialization=on,semijoin=on,loosescan=on,firstmatch=on,duplicateweedout=on,subquery\_materialization\_cost\_based=on,use\_index\_extensions=on,condition\_fanout\_filter=on,derived\_merge=on,use\_invisible\_indexes=off,skip\_scan=on  
optimizer\_trace enabled=off,one\_line=off  
optimizer\_trace\_features greedy\_search=on,range\_optimizer=on,dynamic\_range=on,repeated\_subselect=on  
optimizer\_trace\_limit 1  
optimizer\_trace\_max\_mem\_size 1048576  
optimizer\_trace\_offset -1  
outbound\_use\_schannel ON  
parser\_max\_mem\_size 18446744073709551615  
password\_history 0  
password\_require\_current OFF  
password\_reuse\_interval 0  
performance\_schema ON  
performance\_schema\_accounts\_size -1  
performance\_schema\_az\_digest\_exclude\_user azure\_superuser  
performance\_schema\_az\_digests\_size 512  
performance\_schema\_az\_max\_sql\_text\_length 1024  
performance\_schema\_consumer\_az\_statements\_digest OFF  
performance\_schema\_digests\_size 10000  
performance\_schema\_error\_size 4415  
performance\_schema\_events\_stages\_history\_long\_size 10000  
performance\_schema\_events\_stages\_history\_size 10  
performance\_schema\_events\_statements\_history\_long\_size 10000  
performance\_schema\_events\_statements\_history\_size 10  
performance\_schema\_events\_transactions\_history\_long\_size 10000  
performance\_schema\_events\_transactions\_history\_size 10  
performance\_schema\_events\_waits\_history\_long\_size 10000  
performance\_schema\_events\_waits\_history\_size 10  
performance\_schema\_expand\_query\_sample OFF  
performance\_schema\_hosts\_size -1  
performance\_schema\_max\_cond\_classes 80  
performance\_schema\_max\_cond\_instances -1  
performance\_schema\_max\_digest\_length 1024  
performance\_schema\_max\_digest\_sample\_age 60  
performance\_schema\_max\_file\_classes 80  
performance\_schema\_max\_file\_handles 32768  
performance\_schema\_max\_file\_instances -1  
performance\_schema\_max\_index\_stat -1  
performance\_schema\_max\_memory\_classes 450  
performance\_schema\_max\_metadata\_locks -1  
performance\_schema\_max\_mutex\_classes 300  
performance\_schema\_max\_mutex\_instances -1  
performance\_schema\_max\_prepared\_statements\_instances -1  
performance\_schema\_max\_program\_instances -1  
performance\_schema\_max\_rwlock\_classes 40  
performance\_schema\_max\_rwlock\_instances -1  
performance\_schema\_max\_socket\_classes 10  
performance\_schema\_max\_socket\_instances -1  
performance\_schema\_max\_sql\_text\_length 1024  
performance\_schema\_max\_stage\_classes 150  
performance\_schema\_max\_statement\_classes 218  
performance\_schema\_max\_statement\_stack 10  
performance\_schema\_max\_table\_handles -1  
performance\_schema\_max\_table\_instances -1  
performance\_schema\_max\_table\_lock\_stat -1  
performance\_schema\_max\_thread\_classes 100  
performance\_schema\_max\_thread\_instances -1  
performance\_schema\_session\_connect\_attrs\_size 512  
performance\_schema\_setup\_actors\_size -1  
performance\_schema\_setup\_objects\_size -1  
performance\_schema\_users\_size -1  
persist\_only\_admin\_x509\_subject   
persisted\_globals\_load ON  
pid\_file c:\\mysqldata\\CLIENT.pid  
plugin\_dir c:\\mysql\\lib\\plugin\\  
port 20193  
preload\_buffer\_size 32768  
profiling OFF  
profiling\_history\_size 15  
protocol\_version 10  
query\_alloc\_block\_size 8192  
query\_cache\_limit 1048576  
query\_cache\_min\_res\_unit 4096  
query\_cache\_size 1048576  
query\_cache\_type OFF  
query\_cache\_wlock\_invalidate OFF  
query\_prealloc\_size 8192  
range\_alloc\_block\_size 4096  
range\_optimizer\_max\_mem\_size 8388608  
rbr\_exec\_mode STRICT  
read\_buffer\_size 262144  
read\_only OFF  
read\_rnd\_buffer\_size 524288  
redirect\_enabled OFF  
redirect\_flag c05304644d44  
redirect\_server\_host c05304644d44.tr6437.eastus1-a.worker.database.windows.net  
redirect\_server\_port 16016  
redirect\_server\_ttl 0  
regexp\_stack\_limit 8000000  
regexp\_time\_limit 32  
relay\_log C:\\temp\\relaylogs\\relay\_bin  
relay\_log\_basename C:\\temp\\relaylogs\\relay\_bin  
relay\_log\_index C:\\temp\\relaylogs\\relay\_bin.index  
relay\_log\_info\_file relay-log.info  
relay\_log\_info\_repository TABLE  
relay\_log\_purge ON  
relay\_log\_recovery ON  
relay\_log\_space\_limit 1073741824  
report\_host   
report\_password   
report\_port 20193  
report\_user   
require\_secure\_transport OFF  
rpl\_read\_size 8192  
rpl\_stop\_slave\_timeout 31536000  
schema\_definition\_cache 256  
scrub\_hash\_length 5  
scrub\_user\_info ON  
secure\_file\_priv   
server\_id 1526380112  
server\_id\_bits 32  
server\_uuid 5d3c2c36-f240-11ea-9abb-9f662278f45f  
session\_track\_gtids OFF  
session\_track\_schema ON  
session\_track\_state\_change OFF  
session\_track\_system\_variables time\_zone, autocommit, character\_set\_client, character\_set\_results, character\_set\_connection  
session\_track\_transaction\_info OFF  
sha224\_enabled ON  
sha256\_password\_auto\_generate\_rsa\_keys ON  
sha256\_password\_private\_key\_path private\_key.pem  
sha256\_password\_proxy\_users OFF  
sha256\_password\_public\_key\_path public\_key.pem  
shared\_memory OFF  
shared\_memory\_base\_name MYSQL  
show\_create\_table\_verbosity OFF  
show\_old\_temporals OFF  
skip\_external\_locking ON  
skip\_name\_resolve ON  
skip\_networking OFF  
skip\_show\_database OFF  
slave\_allow\_batching OFF  
slave\_checkpoint\_group 512  
slave\_checkpoint\_period 300  
slave\_compressed\_protocol OFF  
slave\_exec\_mode STRICT  
slave\_load\_tmpdir C:\\temp  
slave\_max\_allowed\_packet 1073741824  
slave\_net\_timeout 60  
slave\_parallel\_type LOGICAL\_CLOCK  
slave\_parallel\_workers 0  
slave\_pending\_jobs\_size\_max 16777216  
slave\_preserve\_commit\_order OFF  
slave\_rows\_search\_algorithms INDEX\_SCAN,HASH\_SCAN  
slave\_skip\_errors OFF  
slave\_sql\_verify\_checksum ON  
slave\_transaction\_retries 10  
slave\_type\_conversions   
slow\_launch\_time 2  
slow\_query\_log OFF  
slow\_query\_log\_file c:\\work\\serverlogs\\mysql-slow-cjgmysql-2020090902.log  
socket /tmp/mysql.sock  
sort\_buffer\_size 524288  
sql\_auto\_is\_null OFF  
sql\_big\_selects ON  
sql\_buffer\_result OFF  
sql\_log\_off OFF  
sql\_mode ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION  
sql\_notes ON  
sql\_quote\_show\_create ON  
sql\_require\_primary\_key OFF  
sql\_safe\_updates OFF  
sql\_select\_limit 18446744073709551615  
sql\_slave\_skip\_counter 0  
sql\_warnings OFF  
ssl\_ca c:\\work\\ca.pem  
ssl\_capath   
ssl\_cert c:\\work\\cert.pem  
ssl\_cipher   
ssl\_crl   
ssl\_crlpath   
ssl\_fips\_mode OFF  
ssl\_key c:\\work\\key.pem  
stored\_program\_cache 256  
stored\_program\_definition\_cache 256  
super\_read\_only OFF  
sync\_binlog 1  
sync\_master\_info 0  
sync\_relay\_log 0  
sync\_relay\_log\_info 1  
system\_time\_zone Coordinated Universal Time  
table\_definition\_cache 5000  
table\_open\_cache 5000  
table\_open\_cache\_instances 1  
tablespace\_definition\_cache 256  
temptable\_max\_ram 1073741824  
test\_mode\_mask 0  
thread\_cache\_size 14  
thread\_handling one-thread-per-connection  
thread\_pool\_batch\_max\_time 0  
thread\_pool\_batch\_wait\_timeout 0  
thread\_pool\_max\_threads 65535  
thread\_pool\_min\_threads 1  
thread\_stack 286720  
threat\_detection\_enabled OFF  
time\_zone SYSTEM  
tls\_version TLSv1,TLSv1.1,TLSv1.2  
tmp\_table\_size 16777216  
tmpdir C:\\temp  
transaction\_alloc\_block\_size 8192  
transaction\_isolation REPEATABLE-READ  
transaction\_prealloc\_size 4096  
transaction\_read\_only OFF  
transaction\_write\_set\_extraction XXHASH64  
tx\_isolation REPEATABLE-READ  
tx\_read\_only OFF  
unique\_checks ON  
updatable\_views\_with\_limit YES  
version 8.0.15  
version\_comment Source distribution  
version\_compile\_machine x86\_64  
version\_compile\_os Win64  
version\_compile\_zlib 1.2.11  
wait\_timeout 120  
windowing\_use\_high\_precision ON