Migrating MySQL On-Premises to Azure Database for MySQL

Technical white paper

Published: October 2020
Applies to: Azure Database for MySQL

Copyright

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. 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 to perform assessment, migration, and post-optimization activities for an Azure Database for MySQL environment.

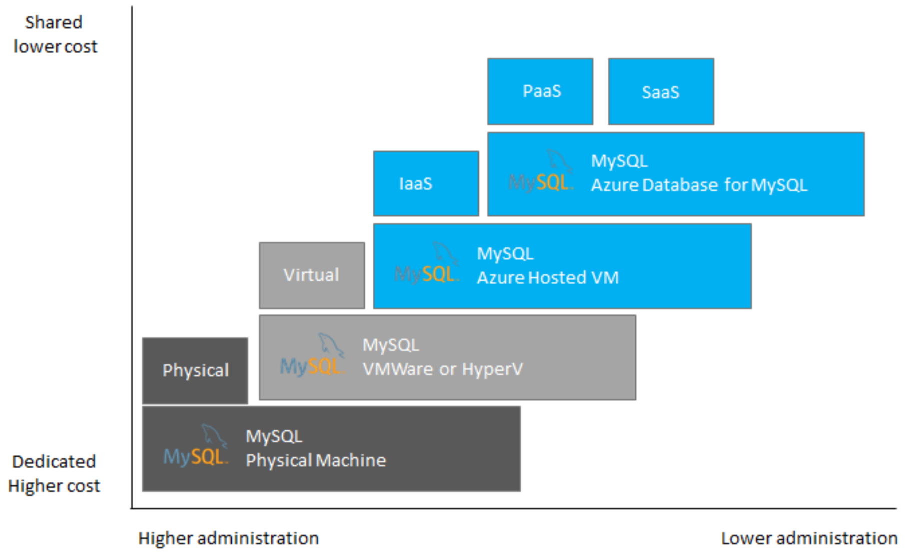
MySQL

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 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 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 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) 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 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), 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. 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.

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 and Memory database storage engines. Other storage engines, like MyISAM, 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 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_c
olumns}
```

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 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 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 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 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) to gather data on your virtual and physical machines. Additionally, consider using the Telegraf agent 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 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 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. 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, 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 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, Azure App Gateway, Azure Load Balancer or App Service 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 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 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
- Azure Database Migration Service (DMS)
- Express Route
- Azure Virtual Network with hub and spoke design with corresponding virtual network peerings establish.
- App Service
- Application Gateway
- Private endpoints 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 to the service. At creation time, you will be able to place the service inside a virtual network that has connectivity via ExpressRoute or over a site to site VPN.
- DMS can be configured in a hybrid-mode 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 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]() 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 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 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.

Dump and restore (mysqldump)

`mysqldump` is typically provided as part of the MySQL installation. It is a client utility 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.

mydumper and myloader

Environments with large databases requiring fast migration should use mydumper and myloader. 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. 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 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:

- 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 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 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) 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 allowing for significantly minimized downtime. Additionally, it supports on-premises data workloads and cloud-based workloads such as Amazon Relational Database Service (RDS) MySQL.

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 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: 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) : 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.

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
- wait_timeout

Additionally, review any parameters that will affect maximums:

- 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 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) 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

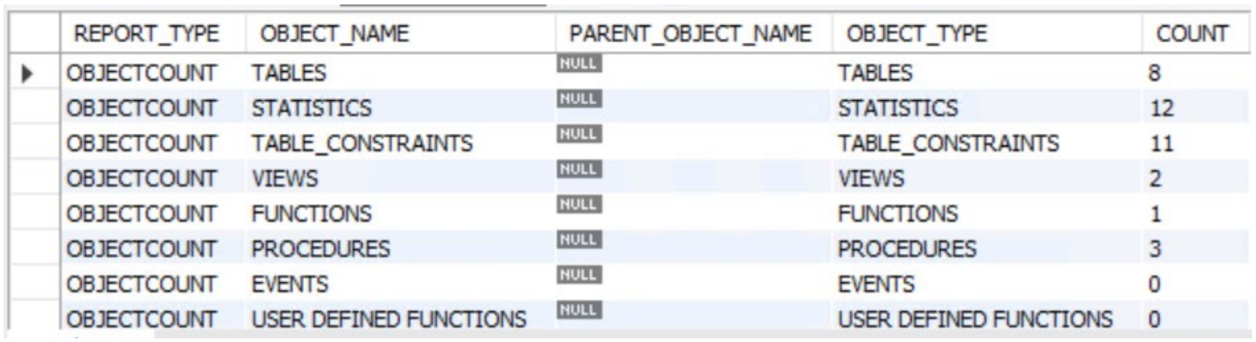
As outlined in the Test Plans section, an inventory of database objects should be done before and after the migration to ensure that you have migrated everything.

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 TABL
E_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;
```

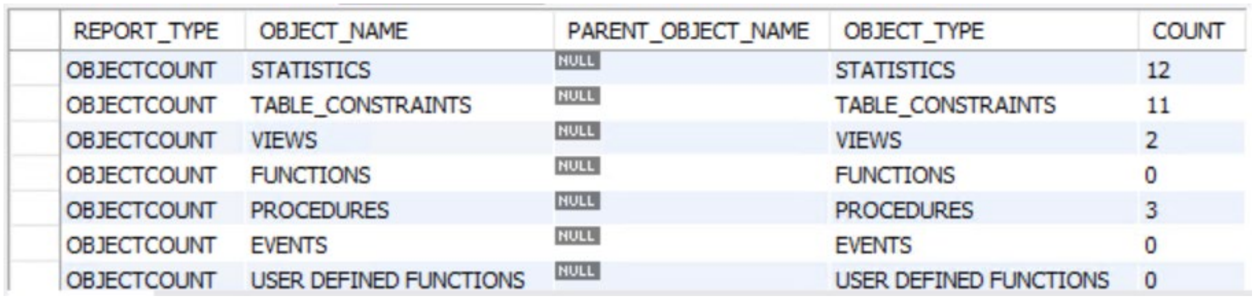
```
      INSERT INTO MIG_INVENTORY (REPORT_TYPE,OBJECT_NAME, OBJECT_TYPE, COUNT)
      SELECT
            'OBJECTCOUNT', 'PROCEDURES', 'PROCEDURES', COUNT(*)
      FROM
            information_schema.ROUTINES
      WHERE
            ROUTINE_TYPE = 'PROCEDURE' and
            ROUTINE_SCHEMA = schemaName;

      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, COUN
T)
            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):



REPORT_TYPE	OBJECT_NAME	PARENT_OBJECT_NAME	OBJECT_TYPE	COUNT
OBJECTCOUNT	TABLES	NULL	TABLES	8
OBJECTCOUNT	STATISTICS	NULL	STATISTICS	12
OBJECTCOUNT	TABLE_CONSTRAINTS	NULL	TABLE_CONSTRAINTS	11
OBJECTCOUNT	VIEWS	NULL	VIEWS	2
OBJECTCOUNT	FUNCTIONS	NULL	FUNCTIONS	1
OBJECTCOUNT	PROCEDURES	NULL	PROCEDURES	3
OBJECTCOUNT	EVENTS	NULL	EVENTS	0
OBJECTCOUNT	USER DEFINED FUNCTIONS	NULL	USER DEFINED FUNCTIONS	0

- 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.



REPORT_TYPE	OBJECT_NAME	PARENT_OBJECT_NAME	OBJECT_TYPE	COUNT
OBJECTCOUNT	STATISTICS	NULL	STATISTICS	12
OBJECTCOUNT	TABLE_CONSTRAINTS	NULL	TABLE_CONSTRAINTS	11
OBJECTCOUNT	VIEWS	NULL	VIEWS	2
OBJECTCOUNT	FUNCTIONS	NULL	FUNCTIONS	0
OBJECTCOUNT	PROCEDURES	NULL	PROCEDURES	3
OBJECTCOUNT	EVENTS	NULL	EVENTS	0
OBJECTCOUNT	USER DEFINED FUNCTIONS	NULL	USER DEFINED FUNCTIONS	0

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 G
RANTS FOR ''',user,'''@''',host,''';') FROM mysql.user WHERE user<>''" > Show
Grants.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\BaltimoreCyberTrus
tRoot.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 or by calling the Azure PowerShell for MySQL cmdlets to make the changes.

Execute the following PowerShell script to get all parameters:

```
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls
12

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 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 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 {u
sername}@{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 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 $rgna
me -ServerName $serverName -Value 1073741824
Update-AzMySqlConfiguration -Name innodb_buffer_pool_size -ResourceGroupName
$rgname -ServerName $serverName -Value 16106127360
Update-AzMySqlConfiguration -Name innodb_io_capacity -ResourceGroupName $rgna
me -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 -ResourceGr
oupName $rgname -ServerName $serverName -Value ON
```

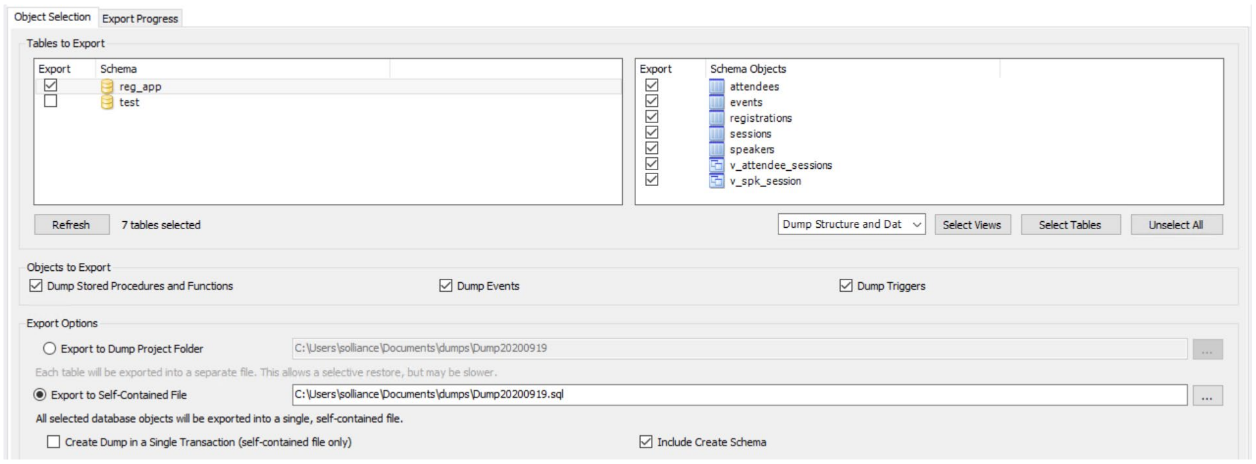
Data

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.



Object Selection | Export Progress

Tables to Export

Export	Schema
☑	reg_app
☐	test

Export	Schema Objects
☑	attendees
☑	events
☑	registrations
☑	sessions
☑	speakers
☑	v_attendee_sessions
☑	v_spk_session

Refresh | 7 tables selected | Dump Structure and Dat | Select Views | Select Tables | Unselect All

Objects to Export

☑ Dump Stored Procedures and Functions ☑ Dump Events ☑ Dump Triggers

Export Options

○ Export to Dump Project Folder C:\Users\solliance\Documents\dumps\Dump20200919

Each table will be exported into a separate file. This allows a selective restore, but may be slower.

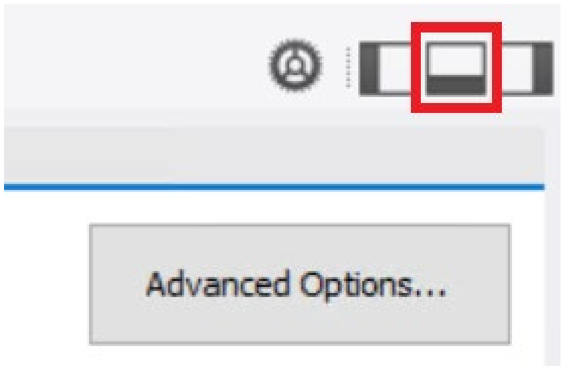
● Export to Self-Contained File C:\Users\solliance\Documents\dumps\Dump20200919.sql

All selected database objects will be exported into a single, self-contained file.

☐ Create Dump in a Single Transaction (self-contained file only) ☑ Include Create Schema

Test

- If any of these options appear unavailable, they are likely obstructed by the Output pane. Just change the editor layout.

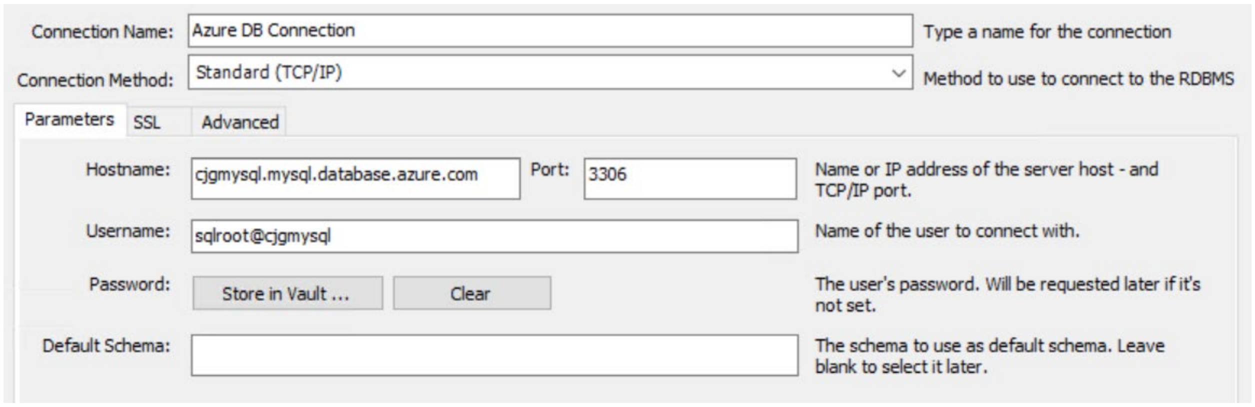


Test

- 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.**



Connection Name:	Azure DB Connection	Type a name for the connection
Connection Method:	Standard (TCP/IP)	Method to use to connect to the RDBMS

Hostname:	cjgmysql.mysql.database.azure.com Port: 3306	Name or IP address of the server host - and TCP/IP port.
Username:	sqlroot@cjgmysql	Name of the user to connect with.
Password:	Store in Vault ... Clear	The user's password. Will be requested later if it's not set.
Default Schema:		The schema to use as default schema. Leave blank to select it later.

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.

```
$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, Log Analytics and Azure Sentinel. 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 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 for more information.

Once log data is flowing, use the Kusto Query Language (KQL) query language to query the various log information. Administrators unfamiliar with KQL can find a SQL to KQL cheat sheet here or the Get started with log queries in Azure Monitor 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 based of these queries.

Server Parameters

As part of the migration, it is likely the on-premises 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
- How to back up and restore an Azure Database for MySQL server using PowerShell
- Configure server parameters in Azure Database for MySQL using PowerShell
- Auto grow storage in Azure Database for MySQL server using PowerShell
- How to create and manage read replicas in Azure Database for MySQL using PowerShell
- Restart Azure Database for MySQL server using 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 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. This workspace is configured to be a part of Azure Sentinel such that any Threat Analytics events would be surfaced, and incidents created.

The MySQL DBAs installed the Azure Database for MySQL Azure PowerShell cmdlets 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](). 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]() 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 , q
uery_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]() 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]().

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.

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 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 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 for more information.

Read Replicas

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 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 or Application Gateway 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. This security integration works by issuing tokens that act like passwords during the MySQL login process. Configuring Active Directory integration 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 to further surface any identity issues.

> **Note:** This security feature is supported by MySQL 5.7 and later. Most application drivers 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 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 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 and Log Analytics by turning on diagnostic logging.

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).

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 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. 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 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. 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.

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 to be configured.

Private Link

To limit access to the Azure Database for MySQL to internal Azure resources, enable 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 that must be taken into account that are not the focus of this guide.

Review a set of potential 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.
- To provide feedback or to request new features, create an entry via UserVoice.

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 or Microsoft MVP 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:

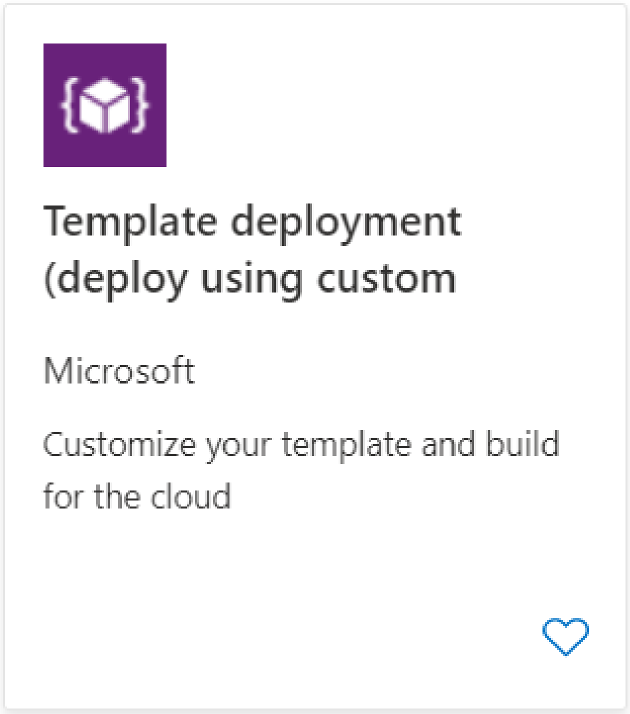
- Microsoft Community Forum
- Azure Database for MySQL Tech Community
- StackOverflow for Azure MySQL
- Azure Facebook Group
- LinkedIn Azure Group
- LinkedIn Azure Developers Group

Appendix A: Environment Setup

The following steps will configure an environment to perform the guide’s migration steps for the sample conference demo application.

Deploy the ARM template

- Open the Azure Portal
- Create a new resource group
- Select **+Add**, type **template**, select the **Template Deployment…**

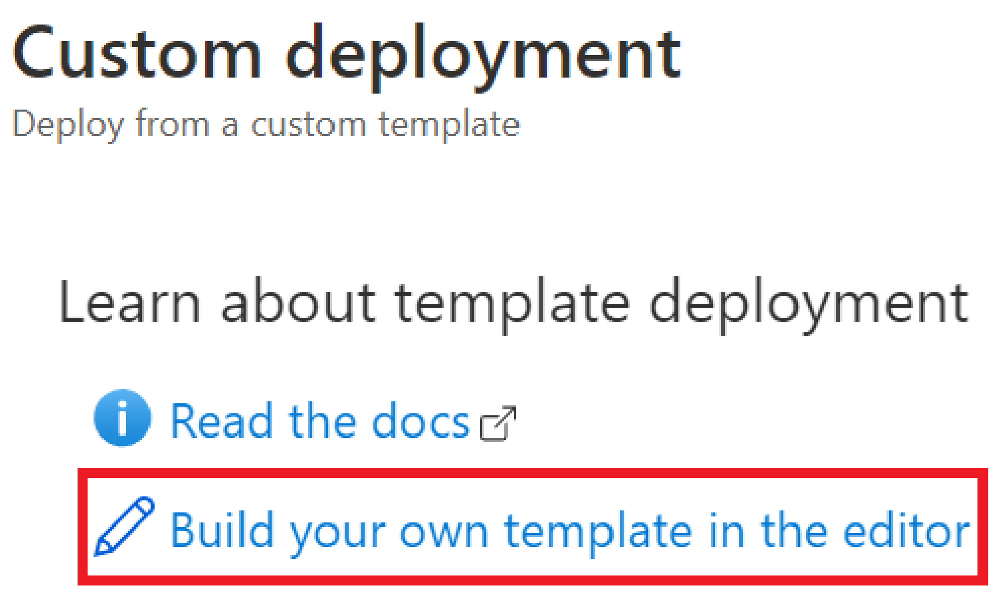


Template deployment (deploy using custom

Microsoft

Customize your template and build for the cloud

- Select **Create**
- Select **Build your own template in the editor**



Custom deployment

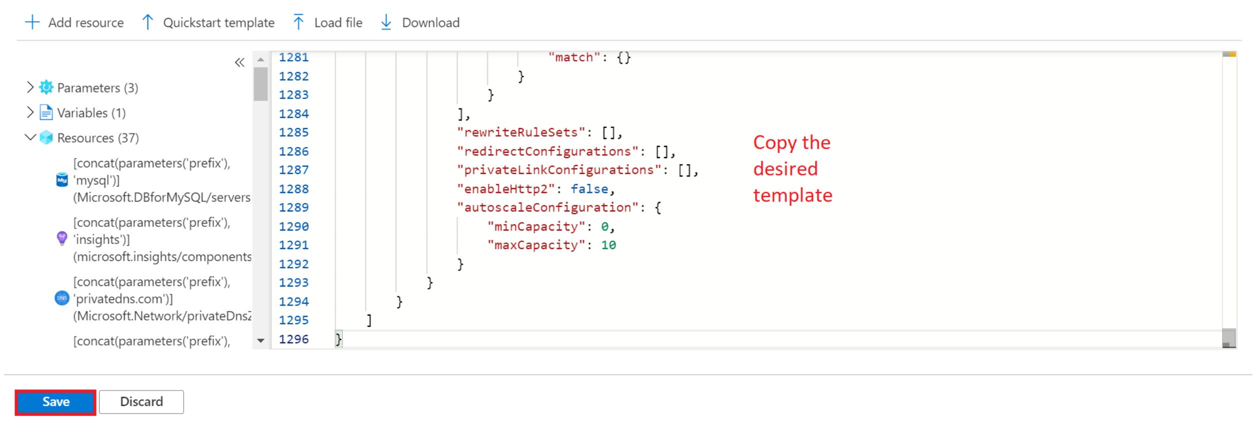
Deploy from a custom template

Learn about template deployment

Read the docs

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**



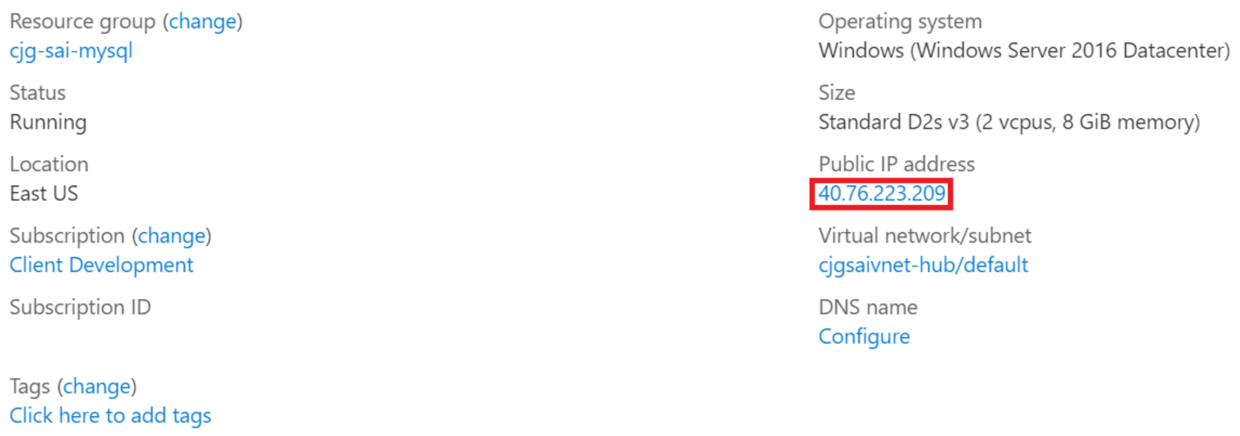
Add resource
Quickstart template
Load file
Download
Parameters (3)
Variables (1)
Resources (37)
[concat(parameters('prefix'), 'mysql')] (Microsoft.DBforMySQL/servers
[concat(parameters('prefix'), 'insights')] (microsoft.insights/components
[concat(parameters('prefix'), 'privatedns.com')] (Microsoft.Network/privateDnsZ
[concat(parameters('prefix'),
Copy the desired template
Save
Discard

```
                                "match": {}
                            }
                        }
                    ],
                    "rewriteRuleSets": [],
                    "redirectConfigurations": [],
                    "privateLinkConfigurations": [],
                    "enableHttp2": false,
                    "autoscaleConfiguration": {
                        "minCapacity": 0,
                        "maxCapacity": 10
                    }
                }
            }
        ]
}
```

- 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**



Resource group (change)
cjg-sai-mysql

Status
Running

Location
East US

Subscription (change)
Client Development

Subscription ID

Tags (change)
Click here to add tags

Operating system
Windows (Windows Server 2016 Datacenter)

Size
Standard D2s v3 (2 vcpus, 8 GiB memory)

Public IP address
40.76.223.209

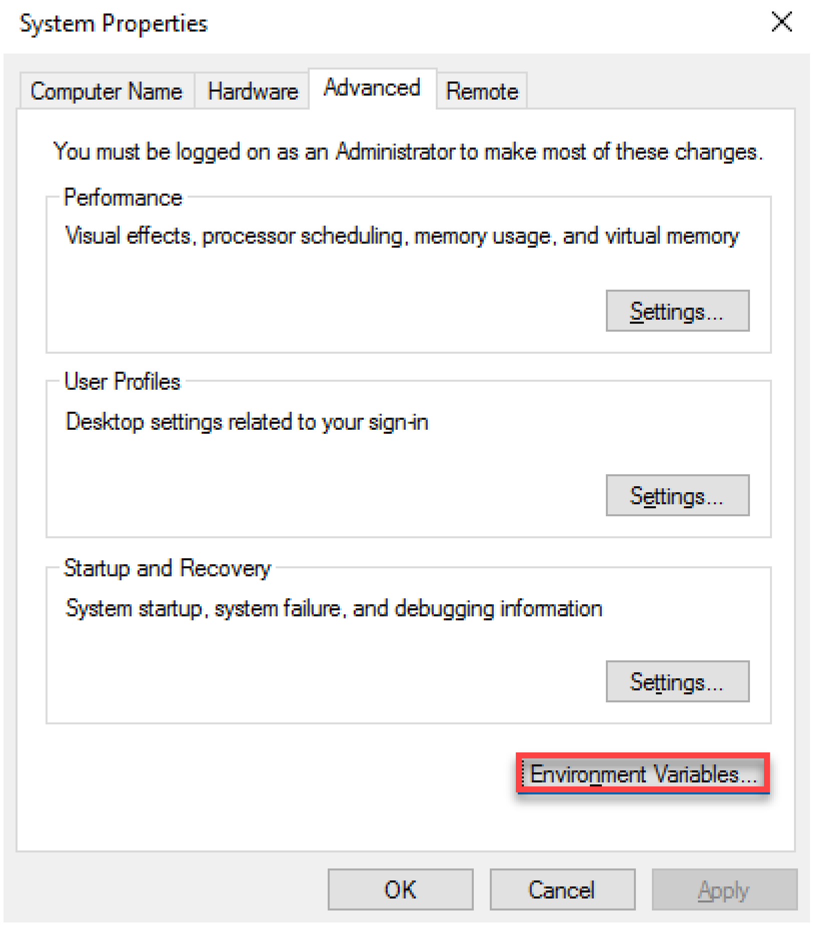
Virtual network/subnet
cjgsaivnet-hub/default

DNS name
Configure

  - Open Remote Desktop and connect to the VM IP Address.
  - Login using the username and password entered above.
- Download the following [MySQL community versions]
  - 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**


System Properties

Computer Name | Hardware | Advanced | Remote

You must be logged on as an Administrator to make most of these changes.

Performance
Visual effects, processor scheduling, memory usage, and virtual memory
Settings...

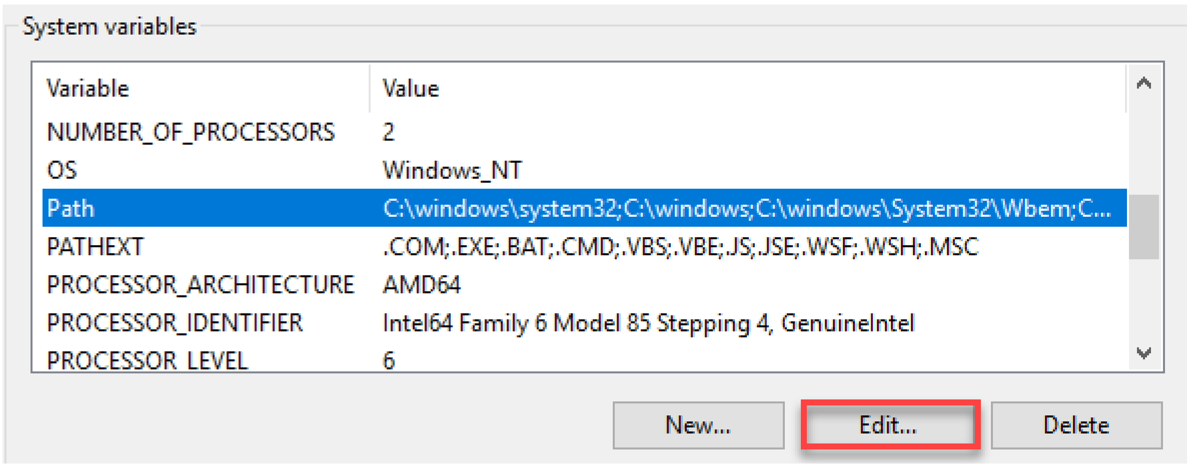
User Profiles
Desktop settings related to your sign-in
Settings...

Startup and Recovery
System startup, system failure, and debugging information
Settings...

Environment Variables...

OK | Cancel | Apply

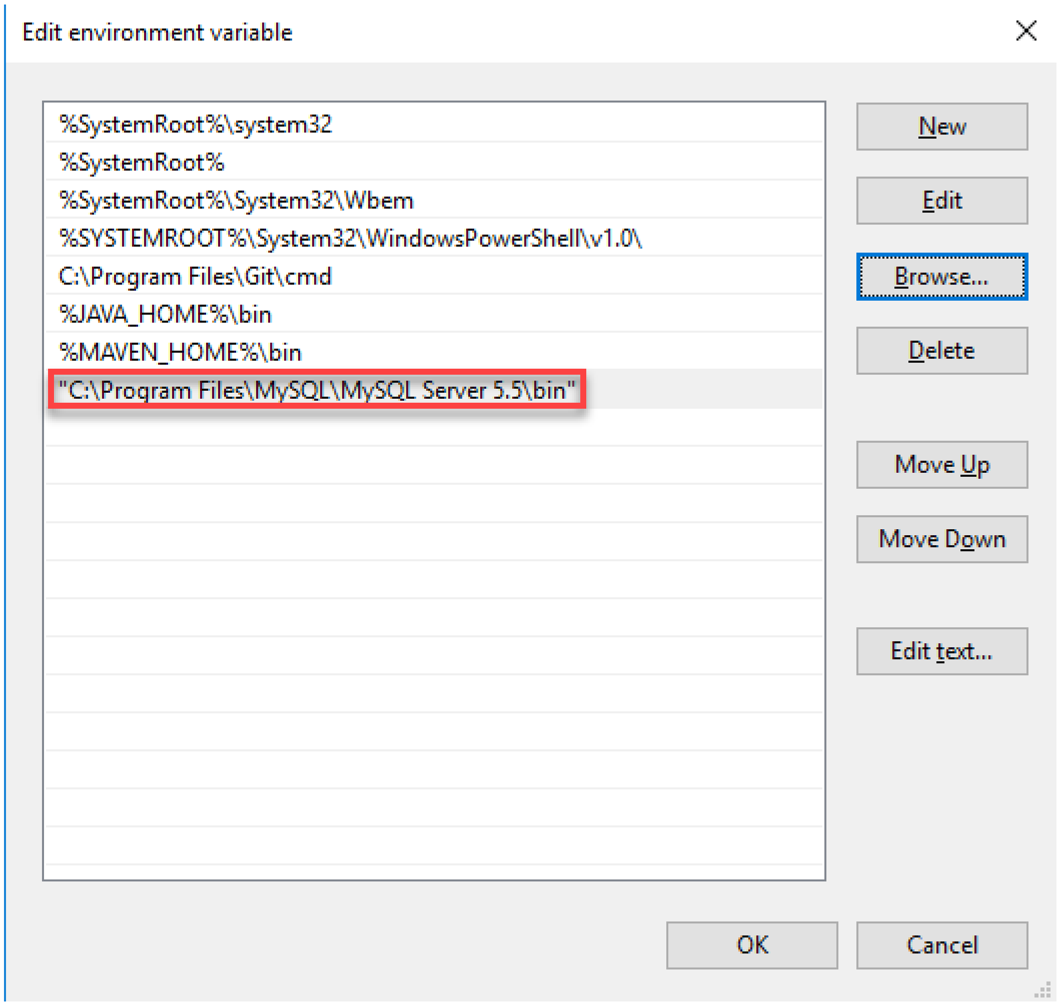
  - Under **System variables**, choose **Path.** Then, select **Edit...**



System variables

Variable	Value
NUMBER_OF_PROCESSORS	2
OS	Windows_NT
Path	C:\windows\system32;C:\windows;C:\windows\System32\Wbem;C...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE	AMD64
PROCESSOR_IDENTIFIER	Intel64 Family 6 Model 85 Stepping 4, GenuineIntel
PROCESSOR_LEVEL	6

– In the **Edit environment variable** dialog, select **New** and then **Browse…** Browse to `C:\Program Files\MySQL\MySQL Server 5.5\bin`.



Edit environment variable

```
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
C:\Program Files\Git\cmd
%JAVA_HOME%\bin
%MAVEN_HOME%\bin
"C:\Program Files\MySQL\MySQL Server 5.5\bin"
```

– Select **OK.**

Install MySQL Workbench

- Download the MySQL Workbench installer
- Install Visual C++ Redistributable for Visual Studio 2015, 2017, and 2019 from here. 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**

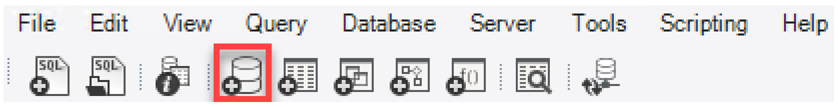
Download artifacts

- Download and Install Git
  - 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-migratio
n-guide
```

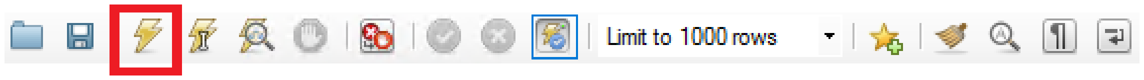
Deploy the Database

- 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

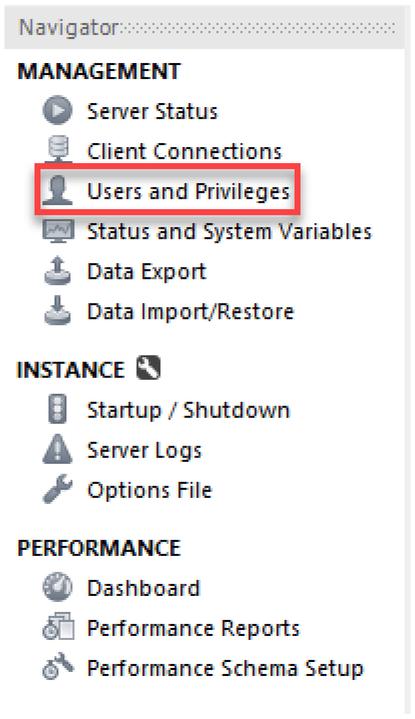


  - For the name, type **reg_app**
  - Select **Apply**
  - In the dialog, select **Apply**
  - Select **Finish**

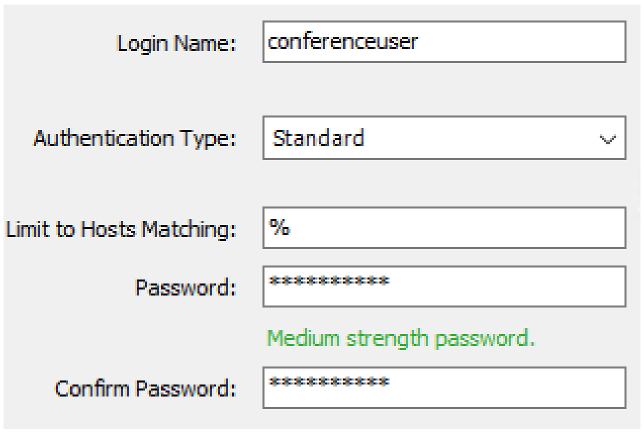
- Select **File->Open SQL Script**
- Browse to **C:\mysql-to-azuremysql-migration-guide-scripts**
- Select **conferencedemo-mysql** file, select **Open.**
- In the navigator, select the Schemas tab, double-click the **reg_app** schema.
- Select **Execute** in the query editor



- Create the database user
  - In the navigator, select the Administration tab.
  - Select **Users and Privileges**

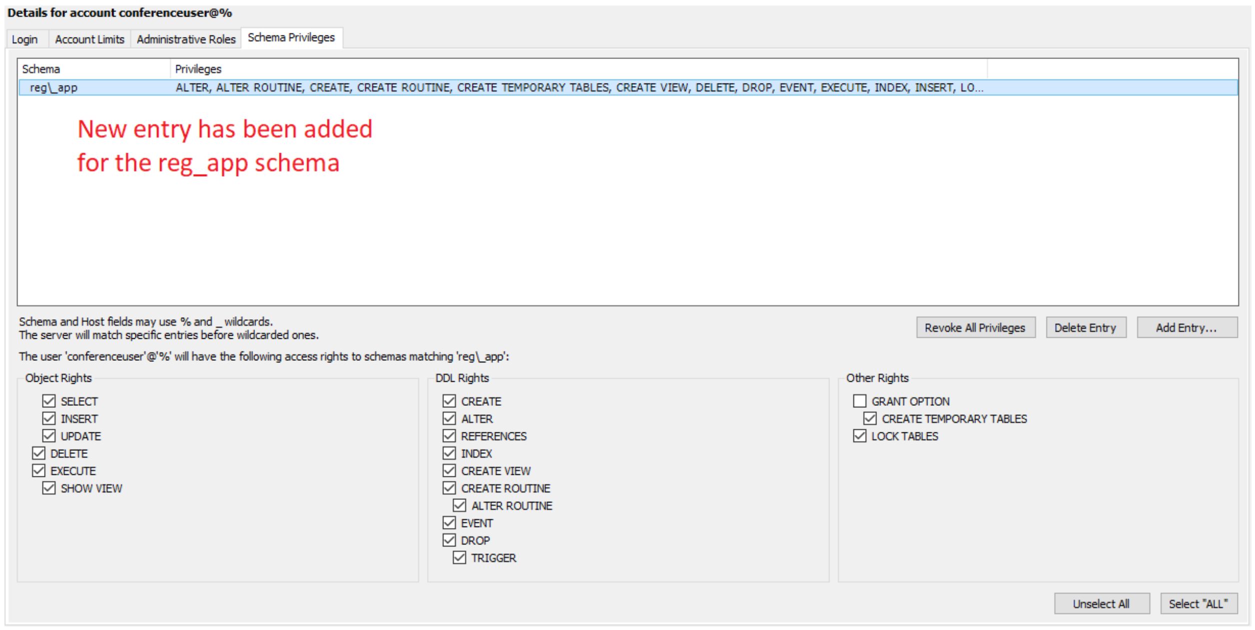


  - Select **Add Account**
  - For the name, type **conferenceuser**
  - For the password, type a password



Login Name:	conferenceuser
Authentication Type:	Standard
Limit to Hosts Matching:	%
Password:	***********
	Medium strength password.
Confirm 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.



Details for account conferenceuser@%

Schema	Privileges
reg_app	ALTER, ALTER ROUTINE, CREATE, CREATE ROUTINE, CREATE TEMPORARY TABLES, CREATE VIEW, DELETE, DROP, EVENT, EXECUTE, INDEX, INSERT, LO...

New entry has been added for the reg_app schema

Schema and Host fields may use % and _ wildcards.
The server will match specific entries before wildcarded ones.

The user 'conferenceuser'@'%' will have the following access rights to schemas matching 'reg_app':

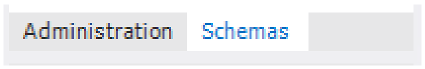
Object Rights
- ☑ SELECT
- ☑ INSERT
- ☑ UPDATE
- ☑ DELETE
- ☑ EXECUTE
- ☑ SHOW VIEW

DDL Rights
- ☑ CREATE
- ☑ ALTER
- ☑ REFERENCES
- ☑ INDEX
- ☑ CREATE VIEW
- ☑ CREATE ROUTINE
- ☑ ALTER ROUTINE
- ☑ EVENT
- ☑ DROP
- ☑ TRIGGER

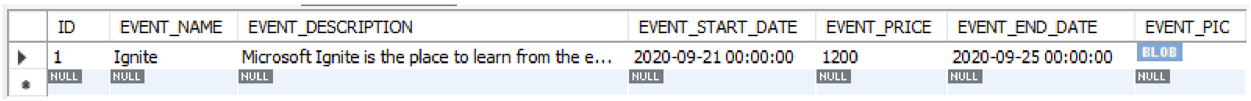
Other Rights
- ☐ GRANT OPTION
- ☑ CREATE TEMPORARY TABLES
- ☑ LOCK TABLES

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

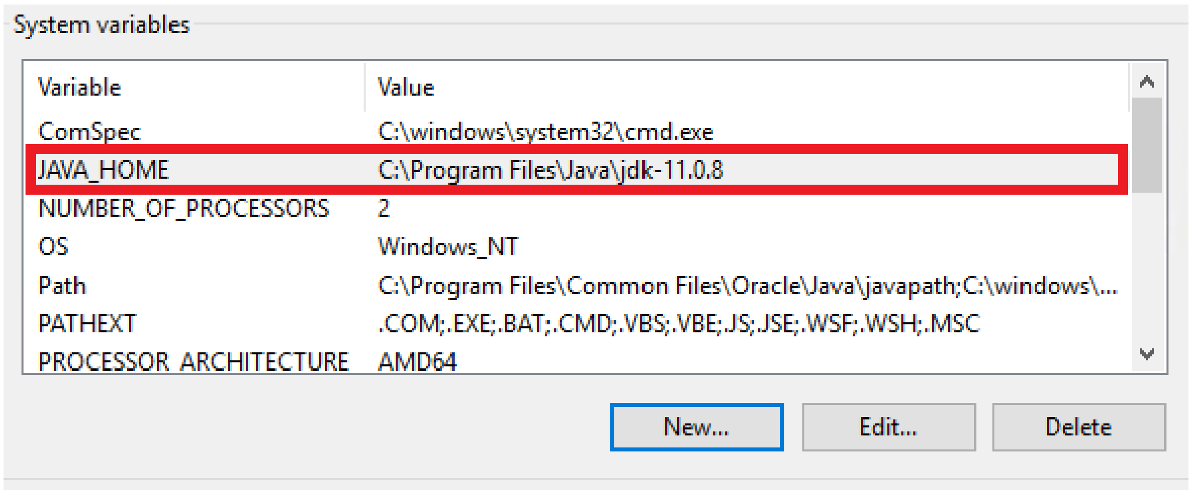


	ID	EVENT_NAME	EVENT_DESCRIPTION	EVENT_START_DATE	EVENT_PRICE	EVENT_END_DATE	EVENT_PIC
▶	1	Ignite	Microsoft Ignite is the place to learn from the e...	2020-09-21 00:00:00	1200	2020-09-25 00:00:00	BLOB
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Setup Java and Maven

- Download and Install Java Development Kit 11.0

  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 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.


System variables

Variable	Value
ComSpec	C:\windows\system32\cmd.exe
JAVA_HOME	C:\Program Files\Java\jdk-11.0.8
NUMBER_OF_PROCESSORS	2
OS	Windows_NT
Path	C:\Program Files\Common Files\Oracle\Java\javapath;C:\windows\...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE	AMD64

- Download and install the Java Runtime
- Download and configure Maven
  - 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

Install NodeJS

- Download and Install NodeJS. 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

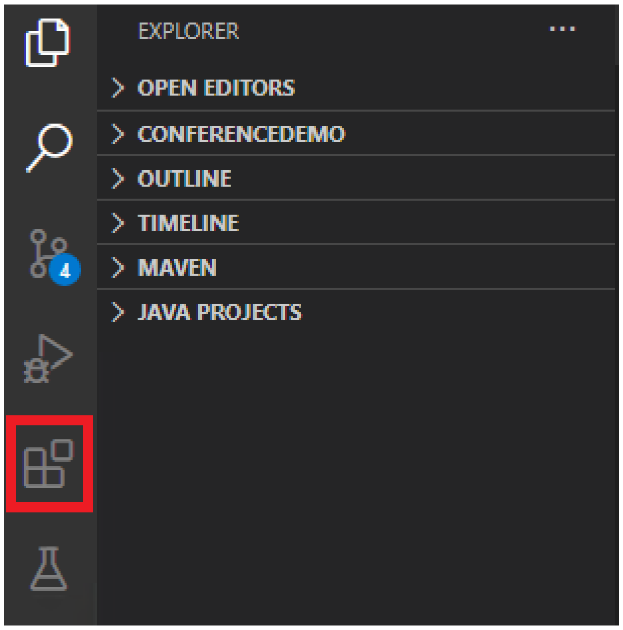
- Download and Install Visual Studio Code. Select the 64-bit Windows User Installer

Configure the Web Application API

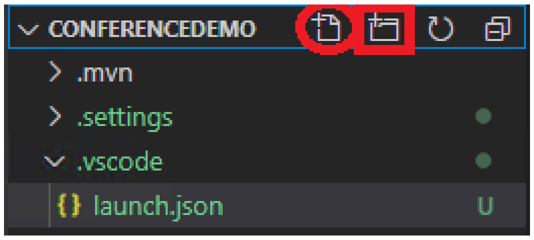
- Open Visual Studio Code

- 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.

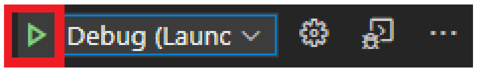


  - Copy the following into it:

```
{
    // 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.Conference
demoApplication",
            "env" :{
                "DB_CONNECTION_URL" : "jdbc:mysql://localhost:3306/
reg_app?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegac
yDatetimeCode=false&serverTimezone=UTC&noAccessToProcedureBodies=tr
ue",
                "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

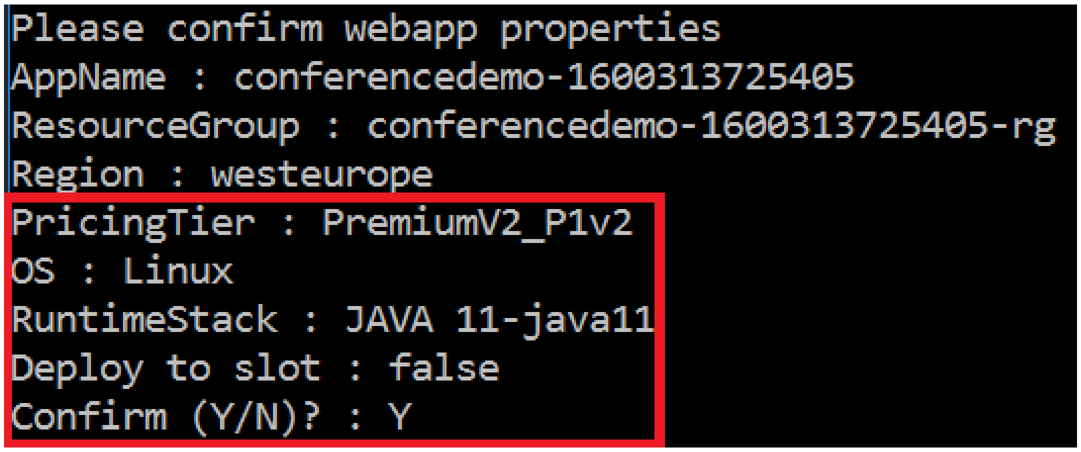
- Open a command prompt window.
- Run the following command to create the Maven configuration to deploy the app. Multiple packages will be installed from the Maven repository.

```
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-webap
p-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**



```
Please confirm webapp properties
AppName : conferencedemo-1600313725405
ResourceGroup : conferencedemo-1600313725405-rg
Region : westeurope
PricingTier : PremiumV2_P1v2
OS : Linux
RuntimeStack : JAVA 11-java11
Deploy to slot : false
Confirm (Y/N)? : Y
```

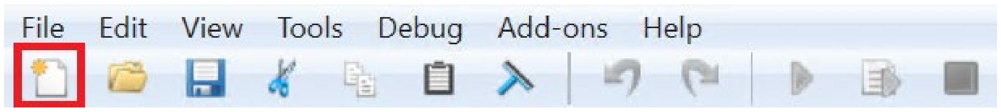
- Switch to Visual Studio and the **ConferenceDemo** project.

- 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]
- 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**.

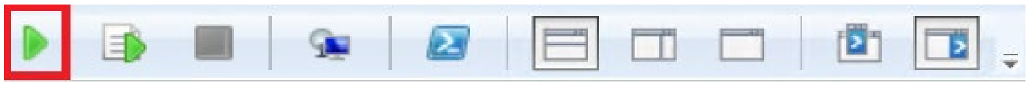
```
$prefix = "{PREFIX}";
$apiip = "{APIIP}";
$app_name = "$($prefix)api01";

$hostname = "$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.

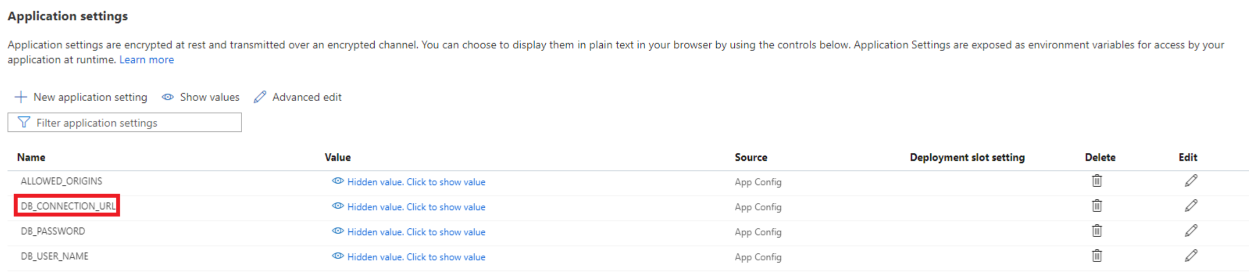
- When prompted, login to the Azure Portal.

```
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_CONNEC
TION_URL={DB_CONNECTION_URL}
az webapp config appsettings set -g $rgName -n $app_name --settings DB_USER_N
AME={DB_USER_NAME}
az webapp config appsettings set -g $rgName -n $app_name --settings DB_PASSWO
RD={DB_PASSWORD}
az webapp config appsettings set -g $rgName -n $app_name --settings ALLOWED_O
RIGINS=*
```

- **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.



Application settings

Application settings are encrypted at rest and transmitted over an encrypted channel. You can choose to display them in plain text in your browser by using the controls below. Application Settings are exposed as environment variables for access by your application at runtime. Learn more

Name	Value	Source	Deployment slot setting	Delete	Edit
ALLOWED_ORIGINS	Hidden value. Click to show value	App Config			
DB_CONNECTION_URL	Hidden value. Click to show value	App Config			
DB_PASSWORD	Hidden value. Click to show value	App Config			
DB_USER_NAME	Hidden value. Click to show value	App Config			

- 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 $subs
cription
az webapp deployment source config-zip --resource-group $rgName --name $app_n
ame --src "./confClient.zip" --subscription $subscription
az webapp start --name $app_name --resource-group $rgName --subscription $sub
scription
```

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

Non-Secure

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

ARM Template

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\\ch
arsets\\
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_p
ushdown=on,mrr=on,mrr_cost_based=on,block_nested_loop=on,batched_key_access=o
ff,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_ran
ge=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_clie
nt, 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_DA
TE,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
```