Migrate MySQL Apps with Minimum Downtime Migration

Overview

These files will setup a migration demo which showcase seamless MySQL application migration using minimum downtime capability in Azure Database Migration Service. This capability will help customer with mission critical application to achieve minimum downtime during migration.

This is the also the same demo in this video which was presented at \\Build 2018 conference. https://medius.studios.ms/Embed/Video/THR2201?sid=THR2201

To learn more about Azure Database Migration Service and what its capabilities, visit here.

When you have completed setup of this demo, you will be able to demo a Star Wars collectible web application hosted on MySQL locally and perform data migration from local database to Azure Database for MySQL. In addition, you will also be able to show data continuously syncing from source to target to achieve minimum downtime during migration. Lastly, you will be able to perform application cutover by updating connection string pointing to Azure Database for MySQL and the web application should be able to connect and display the most updated data.

This demo consists of:

Web application

1. Star Wars collectable python web application – which can be hosted in both MySQL or PostgreSQL.
2. This webapp also has feature a cutover button, which will take valid Azure Database for MySQL or Azure Database for Postgresql and connect to the connection string specified.

Reading from MySQL server : mysqlbuilddemo.mysql.database.azure.com

Cutover

Orders

Order Id	Product Name	Order Date	Quantity
1	Obi-Wan Kenobi Light Saber	2018-05-06 01:14:49	25
2	Star Wars: Return of the Jedi	2018-05-06 01:14:54	25
3	Star Wars: Attack of the Clones	2018-05-06 01:14:59	5
4	Star Wars: Revenge of the Sith	2018-05-06 01:15:04	10
5	Obi-Wan Kenobi Light Saber	2018-05-06 01:15:09	5
6	Obi-Wan Kenobi Light Saber	2018-05-06 01:15:14	15
7	Star Wars: Return of the Jedi	2018-05-06 04:30:49	15
8	Star Wars: The Empire Strikes Back	2018-05-06 04:30:54	15
9	Star Wars: The Empire Strikes Back	2018-05-06 06:49:51	10

Figure 1: Screenshot of Star Wars web application

MySQL as the database

1. The local MySQL or Postgresql will get populated with a simple schema file which will create two tables: Catalog and Orders
2. It will also create a view which will be displayed on the web application.
3. The same schema will get populated in Azure Database for MySQL and Azure Database for Postgresql before using Database Migration Service to move the schema from on-premise to Azure targets.
4. To populate with some prefix data, follow section in Detailed Instruction section below.

Console applications to insert and query orders from source and target

1. There is a console application (through command prompt) to query from the target. Target here means querying from database hosted in Azure Database for MySQL. It will always select top 1 order from the orders table in descending order.

```
C:\BuildDemoSetup\InventoryConsole>queryorders.py --host mysqlbuilddemo.mysql
.database.azure.com --user dms@mysqlbuilddemo --database inventory --platform
 mysql
Password:
Connected to server. Now querying orders
Latest order # 211. Order date - 2018-05-31 17:39:01
Latest order # 211. Order date - 2018-05-31 17:39:01
Latest order # 211. Order date - 2018-05-31 17:39:01
Latest order # 211. Order date - 2018-05-31 17:39:01
Latest order # 211. Order date - 2018-05-31 17:39:01
Latest order # 211. Order date - 2018-05-31 17:39:01
Latest order # 211. Order date - 2018-05-31 17:39:01
Latest order # 211. Order date - 2018-05-31 17:39:01
Latest order # 211. Order date - 2018-05-31 17:39:01
Latest order # 211. Order date - 2018-05-31 17:39:01
Latest order # 211. Order date - 2018-05-31 17:39:01
Latest order # 211. Order date - 2018-05-31 17:39:01
Latest order # 211. Order date - 2018-05-31 17:39:01
Latest order # 211. Order date - 2018-05-31 17:39:01
Latest order # 211. Order date - 2018-05-31 17:39:01
Latest order # 211. Order date - 2018-05-31 17:39:01
Latest order # 214. Order date - 2018-06-07 22:03:12
Latest order # 215. Order date - 2018-06-07 22:03:17
Latest order # 216. Order date - 2018-06-07 22:03:22
```

Figure 2: Console application to select orders from Azure Database for MySQL

2. The same console application (through command prompt) can also do insert orders in the source database. Source here means MySQL hosted on localhost. It will insert one order every 5 seconds into the orders table.

```
C:\BuildDemoSetup\InventoryConsole>createorders.py --host localhost --user dm
s --database inventory --platform Mysql
Password:
Connected to server. Now creating new orders
Created order # 212
Created order # 213
Created order # 214
Created order # 215
Created order # 216
Created order # 217
Created order # 218
Created order # 219
Created order # 220
Created order # 221
Created order # 222
Created order # 223
Created order # 224
Created order # 225
Created order # 226
Created order # 227
Created order # 228
```

Figure 3: Console application to insert orders in the local MySQL

3. The goal is to show that orders coming into application will be syncing live to the target MySQL instance using minimum downtime feature in Azure Database Migration Service (DMS).

Azure Database Migration Service (DMS) Pipeline

1. Given migrating from MySQL to Azure Database for MySQL scenario is at private preview, to access this service, please contact dmsfeedback@microsoft.com. After going through the onboarding process, you will be able to setup a continuous sync migration pipeline in Azure portal like the following:

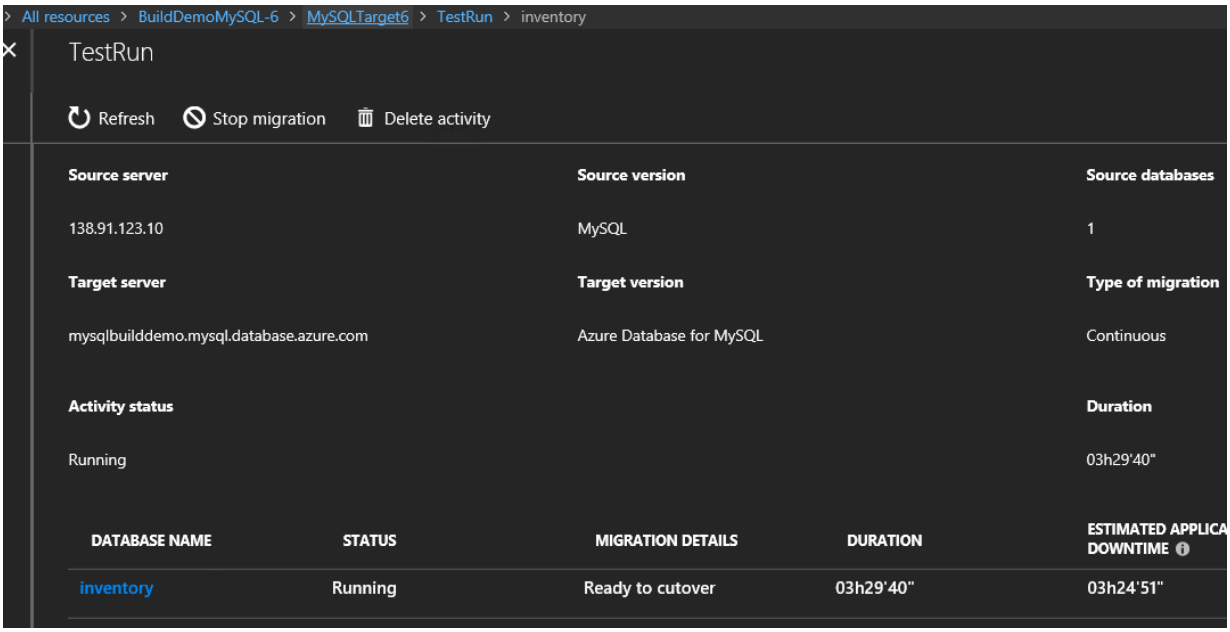


Figure 4: DMS as shown in Azure portal

Detailed Setup Instruction

Pre-requisite Installation and files

Star Wars collectible Web Application

1. The web application is a python flask app. Install Python 3.6.5 from this site - https://www.python.org/downloads/
2. After successful install, you should see Python 36 installed in c:\Python folder.

Install MySQL Server

1. We are using MySQL Server 5.7. Install MySQL Server from https://dev.mysql.com/downloads/mysql/5.7.html#downloads
2. Create a BinLogs folder in a second drive for example D drive. This is where MySQL bin log files will go. Ensure the **service running MySQL has appropriate permission to read + write** in this folder.
3. Change the my.ini file (C:\ProgramData\MySQL\MySQL Server 5.7) to include the following settings:-

   log-bin=D:/BinLogs

   binlog_format=row

4. Save the my.ini file and restart MySQL service.

Create MySQL database, tables and pre-load with data

1. Copy BuildDemoSetup folder to c:\
2. In BuildDemoSetup folder, you should see MySQL-CreateDatabase.sql file. Run that sql file in MySQL Workbench.

3. You should be able to see an inventory database created, with inventory.catalog and inventory.orders tables created.
4. To load some pre-fix data, run the insert script from c:\BuildDemoSetup\PostgreSql-CreateTables.

Install create and query order console application

1. Copy BuildDemoSetup folder to C:\
2. Run the console application in command prompt and navigate to c:\BuildDemoSetup\InventoryConsole\ folder.
3. See example of command in "Putting things together" section below.

Setup Continuous Sync in DMS

1. After confirmation of whitelisting from dmsfeedback@microsoft.com team, please refer to Appendix at the end of this document to setup DMS service.

Putting Things Together

To Start Star Wars collectible web application

1. Navigate to c:\BuildDemoSetup\Inventory\. Click on runserver.py file. This the python script that will start the web application. Keep the python script running whenever the Star Wars application is running.
2. In browser (Edge or Internet Explorer), navigate to http://localhost:5555
3. Click on MySQL Demo to launch the app, enter localhost, username and password to connect.
4. Upon successful of connection, you should see the following screenshot.

Reading from MySQL server : localhost

Cutover

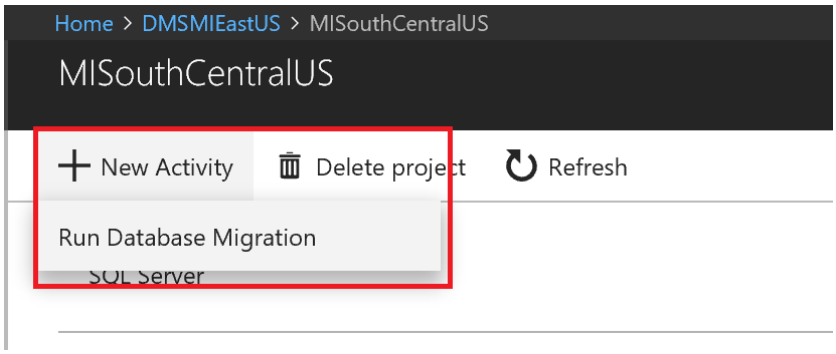
Orders

Order Id	Product Name	Order Date	Quantity
1	Obi-Wan Kenobi Light Saber	2018-05-06 01:14:49	25
2	Star Wars: Return of the Jedi	2018-05-06 01:14:54	25
3	Star Wars: Attack of the Clones	2018-05-06 01:14:59	5
4	Star Wars: Revenge of the Sith	2018-05-06 01:15:04	10
5	Obi-Wan Kenobi Light Saber	2018-05-06 01:15:09	5
6	Obi-Wan Kenobi Light Saber	2018-05-06 01:15:14	15
7	Star Wars: Return of the Jedi	2018-05-06 04:30:49	15

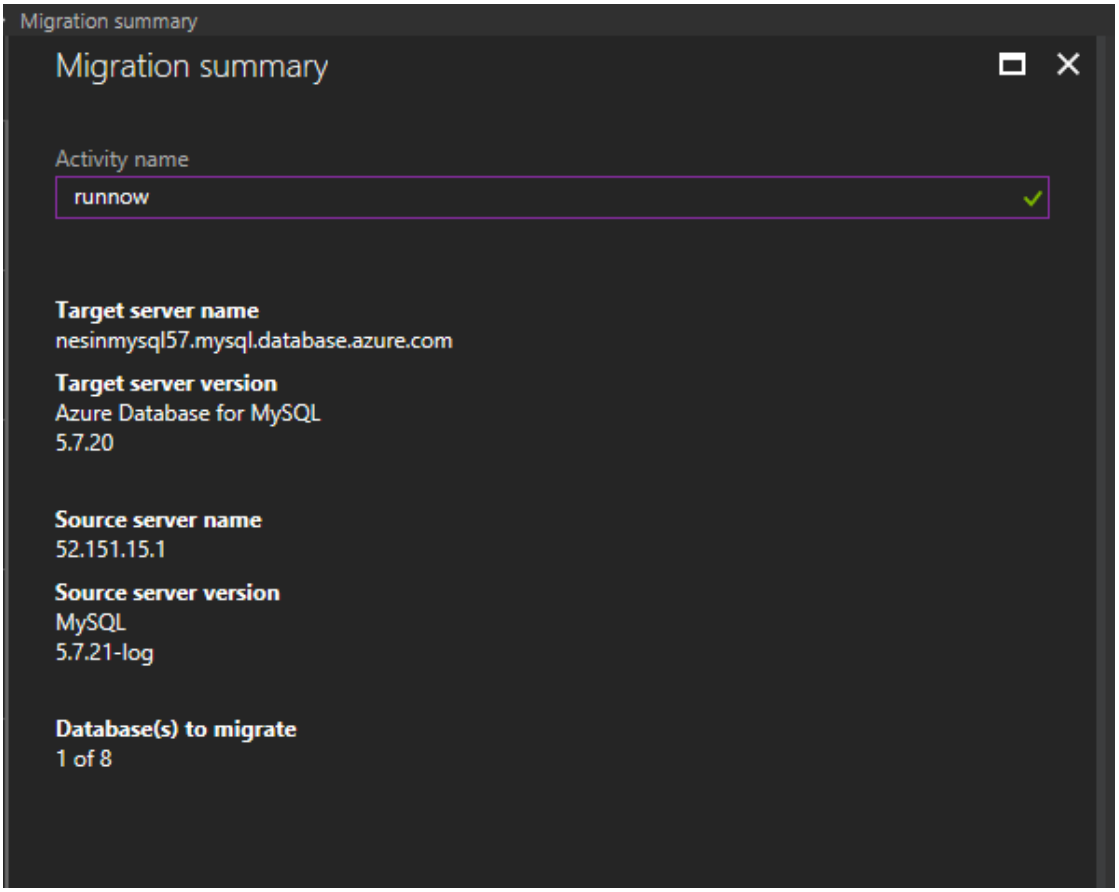
To Start Continuous Sync in DMS

1. In Azure portal, navigate to DMS service you have created. You should be able to create a project by following step #8-#10 in the Appendix section.

2. To start the migration activity, click on +New Activity. Over the two screens, enter the password to connect to both source and target again.



3. Select Inventory as the database to migrate.
4. Final step is to run this activity (migration). Give an activity name, and verify all the information in the summary blade:-



5. Click on Run migration to start migration.
6. The DMS pipeline is started. It will show initializing to running. This means the initial load has started and any new traffic coming in will get replicated to the target as well.

To show continuous sync through inserting new records through web console

1. In command prompt, navigate to C:\BuildDemoSetup\InventoryConsole\
2. Run this command to query for orders from Azure Database for MySQL

queryorders.py --host [Azure Database for MySQL instance connection string] --user [UserName for Azure Database for MySQL] --database inventory --platform mysql

For example:
queryorders.py --host mysqlbuilddemo.mysql.database.azure.com --user dms@mysqlbuilddemo --database inventory --platform mysql

3. Enter password to connect
4. It should give you the top 1 order from the orders table like the following:-

```
C:\BuildDemoSetup\InventoryConsole>queryorders.py --host mysqlbuilddemo.mysql
.database.azure.com --user dms@mysqlbuilddemo --database inventory --platform
 mysql
Password:
Connected to server. Now querying orders
Latest order # 211. Order date - 2018-05-31 17:39:01
Latest order # 211. Order date - 2018-05-31 17:39:01
Latest order # 211. Order date - 2018-05-31 17:39:01
Latest order # 211. Order date - 2018-05-31 17:39:01
Latest order # 211. Order date - 2018-05-31 17:39:01
Latest order # 211. Order date - 2018-05-31 17:39:01
Latest order # 211. Order date - 2018-05-31 17:39:01
Latest order # 211. Order date - 2018-05-31 17:39:01
Latest order # 211. Order date - 2018-05-31 17:39:01
Latest order # 211. Order date - 2018-05-31 17:39:01
Latest order # 211. Order date - 2018-05-31 17:39:01
Latest order # 211. Order date - 2018-05-31 17:39:01
Latest order # 211. Order date - 2018-05-31 17:39:01
Latest order # 211. Order date - 2018-05-31 17:39:01
Latest order # 211. Order date - 2018-05-31 17:39:01
Latest order # 211. Order date - 2018-05-31 17:39:01
Latest order # 214. Order date - 2018-06-07 22:03:12
Latest order # 215. Order date - 2018-06-07 22:03:17
Latest order # 216. Order date - 2018-06-07 22:03:22
```

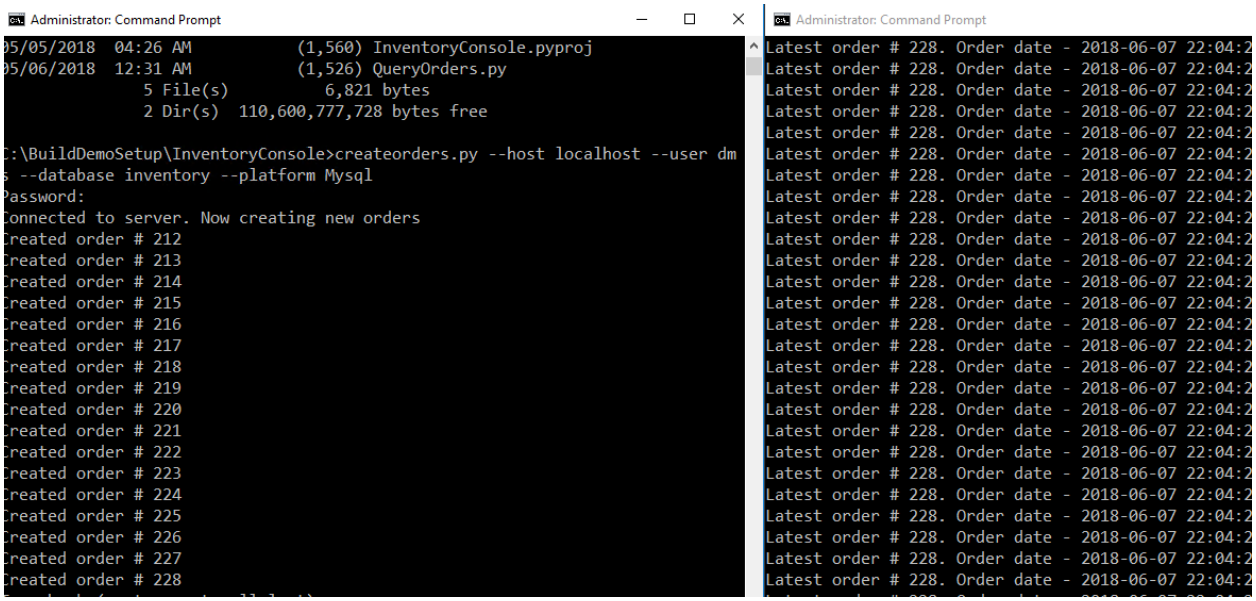
5. To insert the order to localhost to show that new records coming in are continuously replicated to Azure Database for MySQL, open a second command prompt window.
6. Navigate to C:\BuildDemoSetup\InventoryConsole\
7. Run this command to insert one order at every 5 seconds interval into MySQL installed in localhost.

   createorders.py --host [local host of MySQL] --user [username] --database inventory --platform Mysql

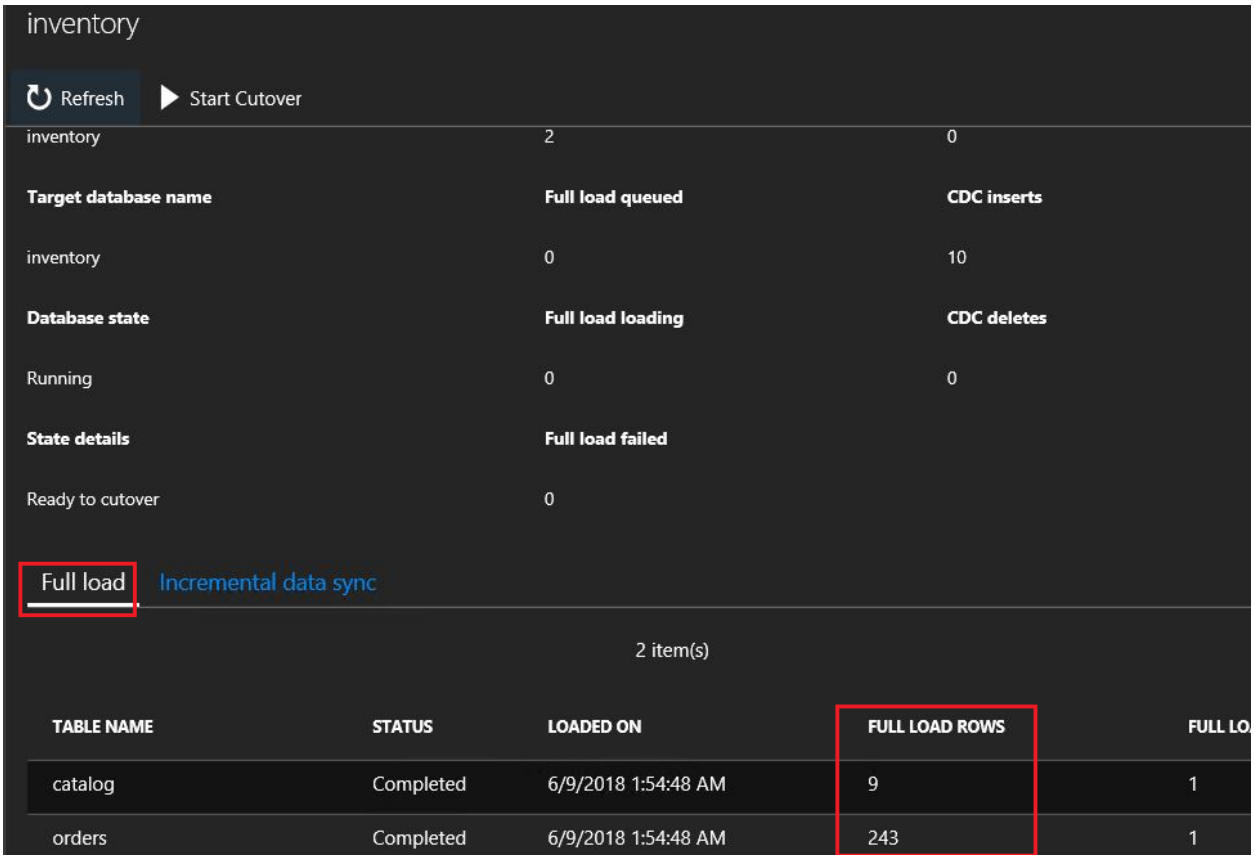
   For example:
   createorders.py --host localhost --user dms --database inventory --platform Mysql
8. Enter password to connect to localhost

```
C:\BuildDemoSetup\InventoryConsole>createorders.py --host localhost --user dm
s --database inventory --platform Mysql
Password:
Connected to server. Now creating new orders
Created order # 212
Created order # 213
Created order # 214
Created order # 215
Created order # 216
Created order # 217
Created order # 218
Created order # 219
Created order # 220
Created order # 221
Created order # 222
Created order # 223
Created order # 224
Created order # 225
Created order # 226
Created order # 227
Created order # 228
```

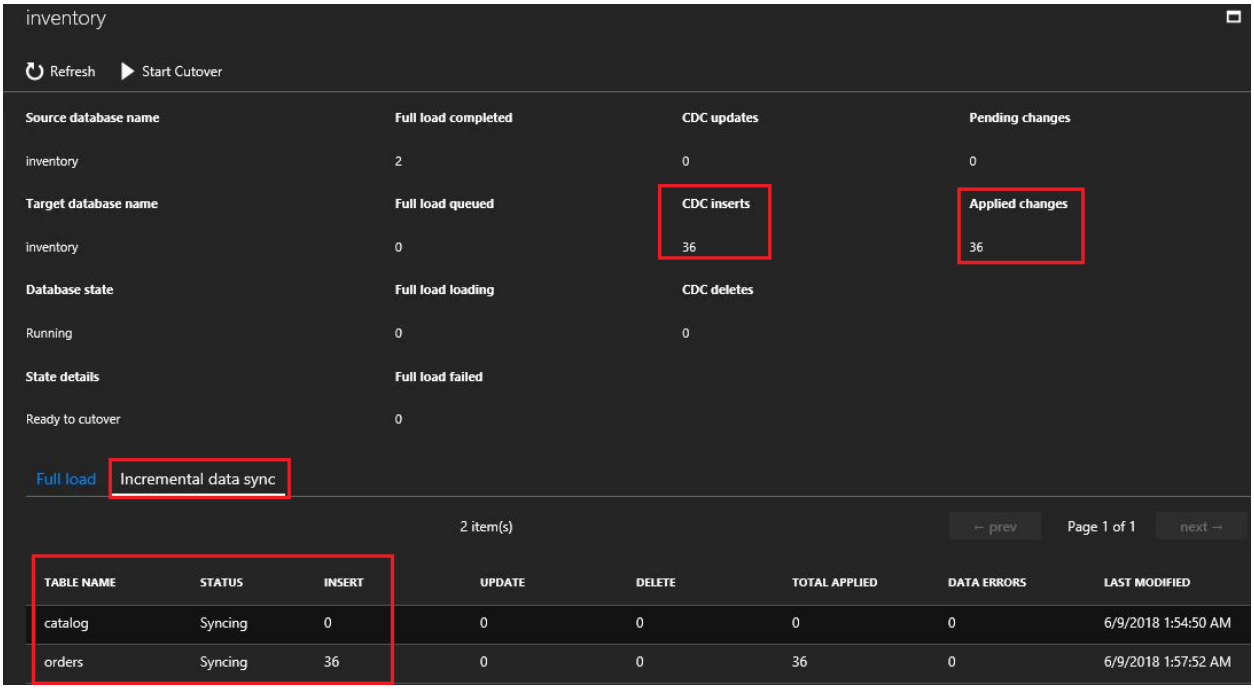
9. I usually like to put two command prompt windows side-by-side to show the select query is showing the updated orders coming in like the following:-



10. I will usually go back to Azure portal to show how many rows get inserted in Full load and Incremental Data Sync in DMS.
11. For example, to show full load, and how many rows are migrated to target:-

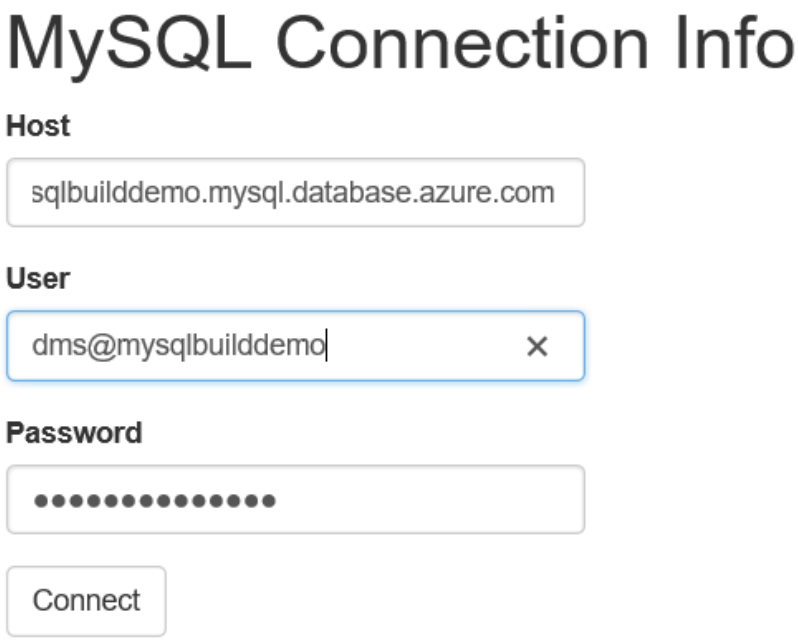


12. To show incremental data sync that the records are inserted:-

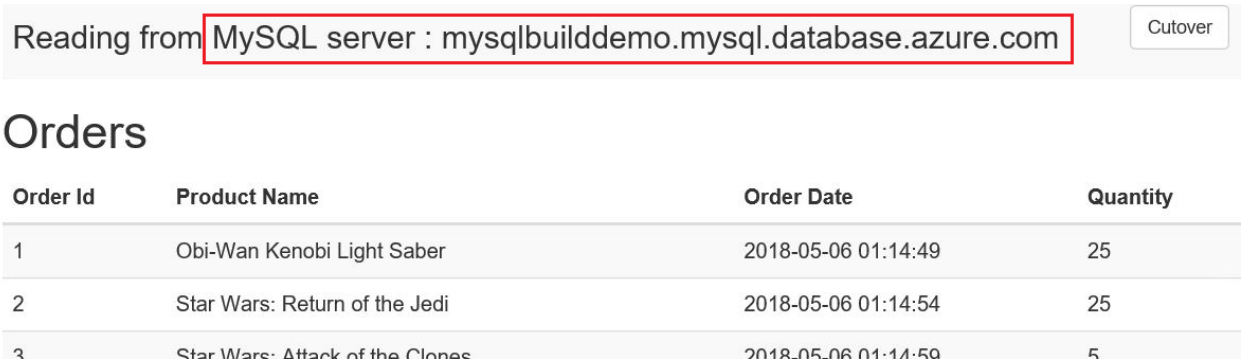


To show application cutover without any code change

1. Navigate to http://localhost:5555/orders website, the website should still be running.
2. Click on the Cutover button
3. Enter the Host connection string that points to Azure Database for MySQL, username and password
4. Click Connect



5. The application should connect to the database hosting on Azure Database for MySQL, and it should have the latest rows you have inserted.



Congratulations! You have completed the building and showing the demo of migrating Star Wars collectible web application from local VM to Azure Database for MySQL using continuous sync capability in Azure Database Migration Service.

Appendix

Migrating from MySQL to Azure Database for MySQL with Continuous Sync

Pre-requites: -

1. The source MySQL Server version must be version 5.6.35, 5.7.18 or later
2. Azure Database for MySQL supports the following:-
    a. MySQL community edition
    b. InnoDB engine
3. Same version migration. Migrate MySQL 5.6 to Azure Database for MySQL 5.7 is not supported.
4. Enable binary logging in my.ini (Windows) or my.cnf (Unix)
    a. Set Server_id to any number larger or equals to 1. E.g Server_id=1 (only for MySQL 5.6)
    b. Set log-bin = <path> (only for MySQL 5.6)
    c. Set binlog_format = row
    d. Expire_logs_days = 5 (recommended - only for MySQL 5.6)
5. User must have ReplicationAdmin role

Pre-migration Steps

1. To complete all the database objects like table schemas, indexes, triggers, and stored procedures, we need to extract schema from the source database and apply to the database. To extract schema, you can use mysqldump with - - no-data parameter. For example:-

    mysqldump -h [servername] -u [username] -p[password] --databases [db name] --no-data > [schema file path]

    eg.
    mysqldump -h 10.10.123.123 -u root -p123123 --databases customer --no-data > d:\customerschema.sql

2. Create an empty database on target, which is Azure Database for MySQL. You can refer to this doc on how to connect and create a database.

    https://docs.microsoft.com/en-us/azure/mysql/quickstart-create-mysql-server-database-using-azure-portal

https://docs.microsoft.com/en-us/azure/mysql/connect-workbench

3. Import the schema to target which is Azure Database for MySQL

   mysql.exe -h [servername] -u [username] -p[password] [database]< [schema file path]

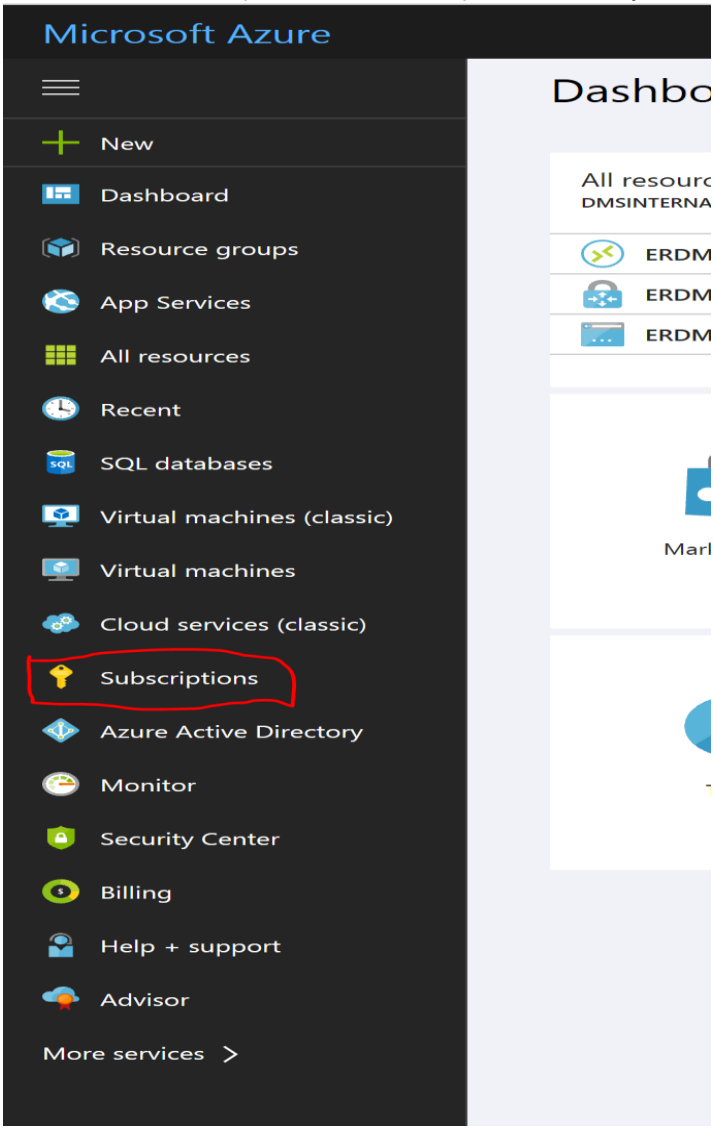
   e.g.
   mysql.exe -h shausample.mysql.database.azure.com -u dms@shausample -p123123 customer < d:\customerschema.sql
4. If you have foreign keys in your schema, the initial load and continuous sync of the migration will fail. Please execute the following script in MySQL workbench to extract the drop foreign key script and add foreign key script.

```
SET group_concat_max_len = 8192;
  SELECT SchemaName, GROUP_CONCAT(DropQuery SEPARATOR ';\n') as DropQuery,
GROUP_CONCAT(AddQuery SEPARATOR ';\n') as AddQuery
  FROM
  (SELECT
  KCU.REFERENCED_TABLE_SCHEMA as SchemaName,
    KCU.TABLE_NAME,
    KCU.COLUMN_NAME,
    CONCAT('ALTER TABLE ', KCU.TABLE_NAME, ' DROP FOREIGN KEY ', KCU.CONSTRAINT_NAME) AS
DropQuery,
  CONCAT('ALTER TABLE ', KCU.TABLE_NAME, ' ADD CONSTRAINT ', KCU.CONSTRAINT_NAME, ' FOREIGN KEY
(`', KCU.COLUMN_NAME, '`) REFERENCES `', KCU.REFERENCED_TABLE_NAME, '`(`',
KCU.REFERENCED_COLUMN_NAME, '`) ON UPDATE ',RC.UPDATE_RULE, ' ON DELETE ',RC.DELETE_RULE) AS
AddQuery
    FROM INFORMATION_SCHEMA.KEY_COLUMN_USAGE KCU,
information_schema.REFERENTIAL_CONSTRAINTS RC
    WHERE
     KCU.CONSTRAINT_NAME = RC.CONSTRAINT_NAME
     AND KCU.REFERENCED_TABLE_SCHEMA = RC.UNIQUE_CONSTRAINT_SCHEMA
 AND KCU.REFERENCED_TABLE_SCHEMA = 'sakila') Queries
 GROUP BY SchemaName;
```

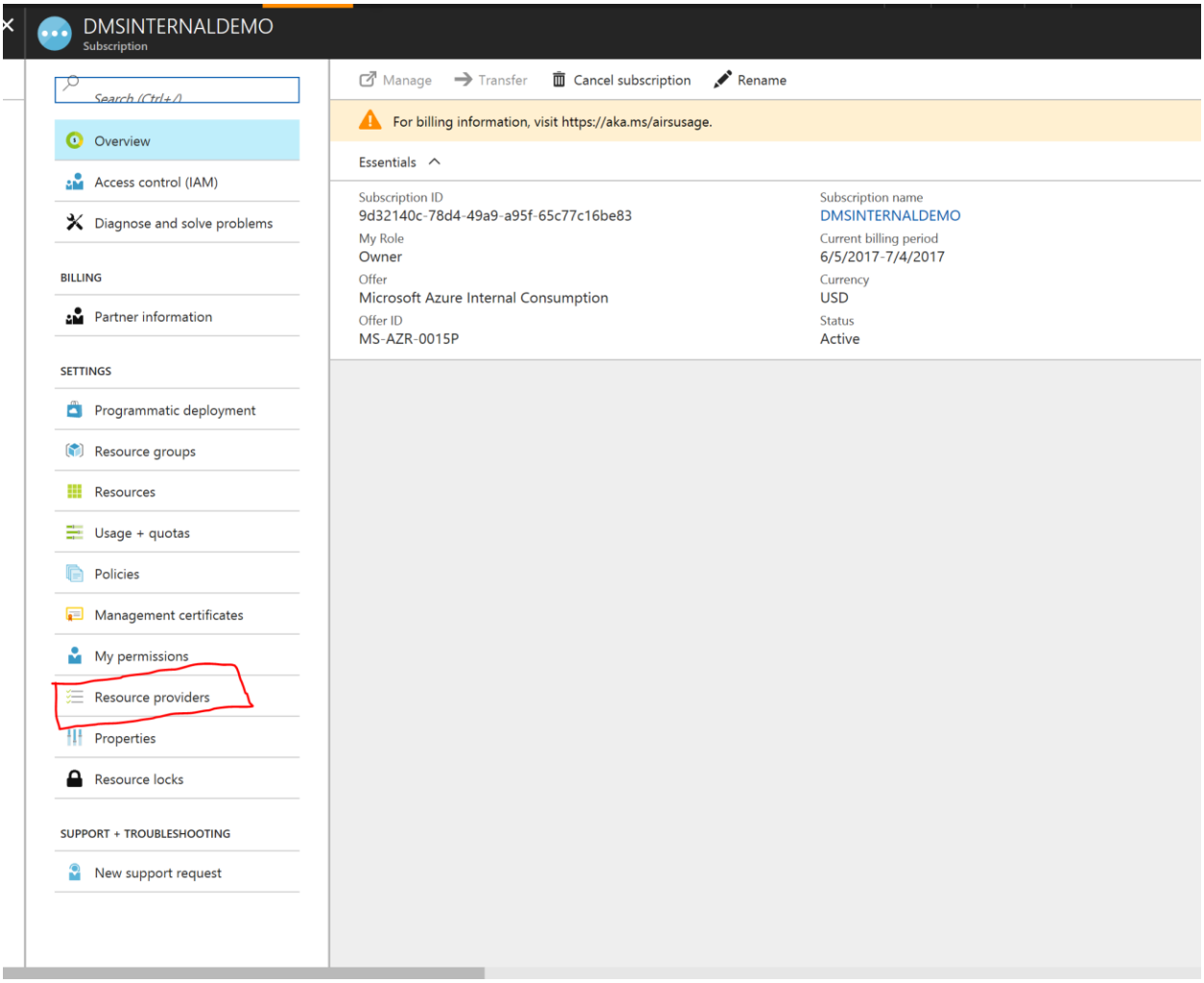
   Run the drop foreign key (which is the second column) in the query result.
5. Follow the DMS steps below to start the data migration.

Register Microsoft.DataMigration resource provider on your subscription

1. Logon to this URL with private preview flag using your subscription: https://ms.portal.azure.com/?Microsoft_Azure_DMS=privatepreview&feature.canmodifystamps=true&feature.canmodifyextensions=true&clientOptimizations=false
2. Click on the "Subscriptions" on the left pane. It takes you to the Subscription page.



3. Select the subscription that you chose to create DMS instance.
4. Click on "Resource providers"

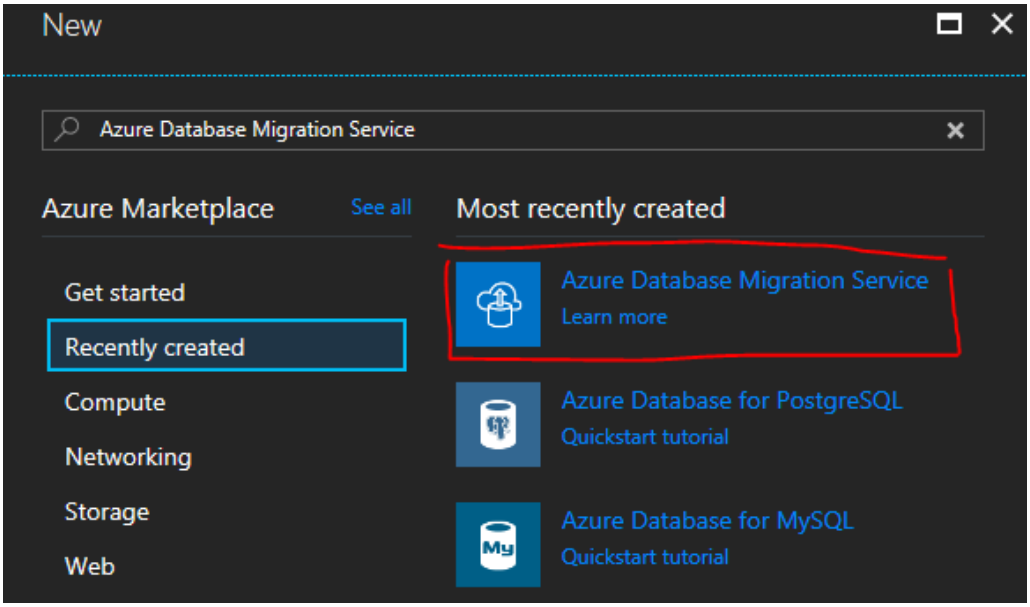


5. Search for "Microsoft.DataMigration" and click on "Register" if not registered already.
6. Once the registration completed, you can see the status below.

PROVIDER	STATUS		
Microsoft.DataMigration	Registered	Re-register	Unregister

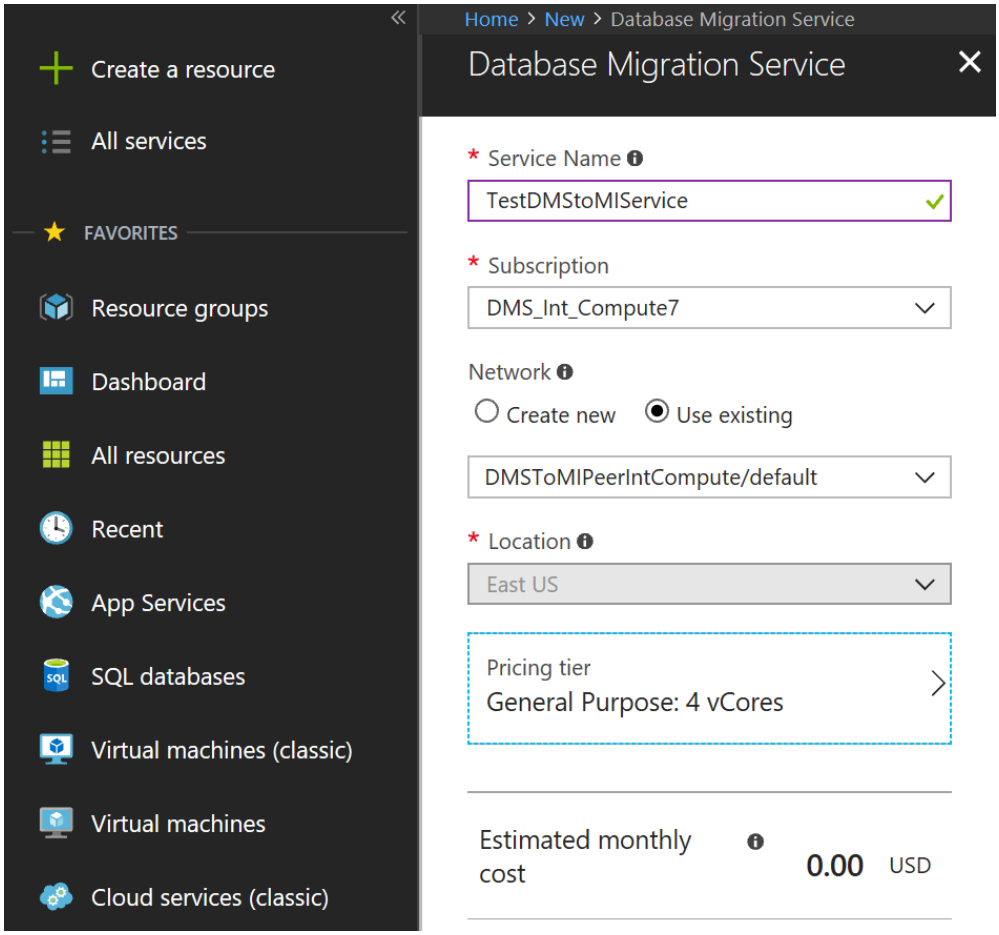
Provisioning DMS Service

1. Logon to this URL with private preview flag using your subscription: https://ms.portal.azure.com/?Microsoft_Azure_DMS=syncPrivatePreview&feature.canmodifystamps=true&Microsoft_Azure_DMS_feature=r&feature.canmodifyextensions=true
2. Select +Create a resource on the left hand panel
3. Search for Azure Database Migration Service in Marketplace. The logo should look like this:-



4. Next blade is about basic information about DMS service:-
    a. Service Name: Give it a name for this service
    b. Subscription: Your subscription
    c. Network: This should be the network that has connectivity to your on-premise MySQL Server:-
        i. Azure virtual network (use existing or creating new) will need to have access to the following ports (443, 53,93543, 445, 12000 for the network and 3306 for MySQL service). https://docs.microsoft.com/en-us/azure/dms/pre-reqs
        ii. If you are creating a new virtual network, we recommend creating the virtual network in the same region that is hosting the Azure Database for MySQL. https://docs.microsoft.com/en-us/azure/virtual-network/quick-create-portal

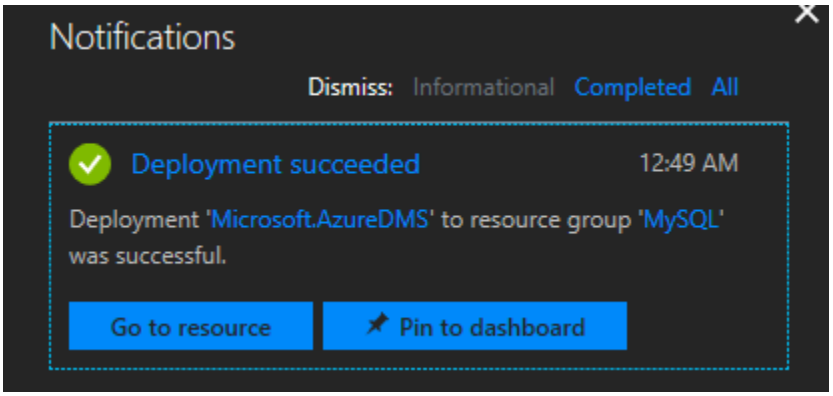
    d. Location: South Central US (it will default to the location of the VNET)
    e. Pricing Tier: I will recommend selecting 4vCores to start with
    f. Click Create



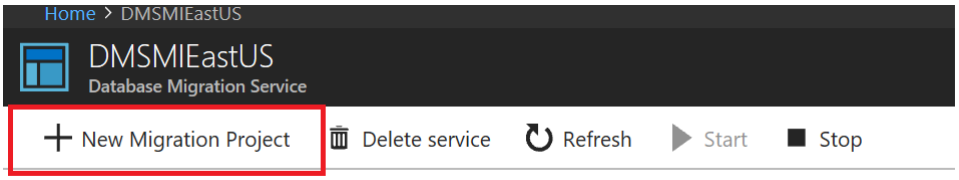
5. It should take about 10 minutes for DMS to be created.

Setting up Migration

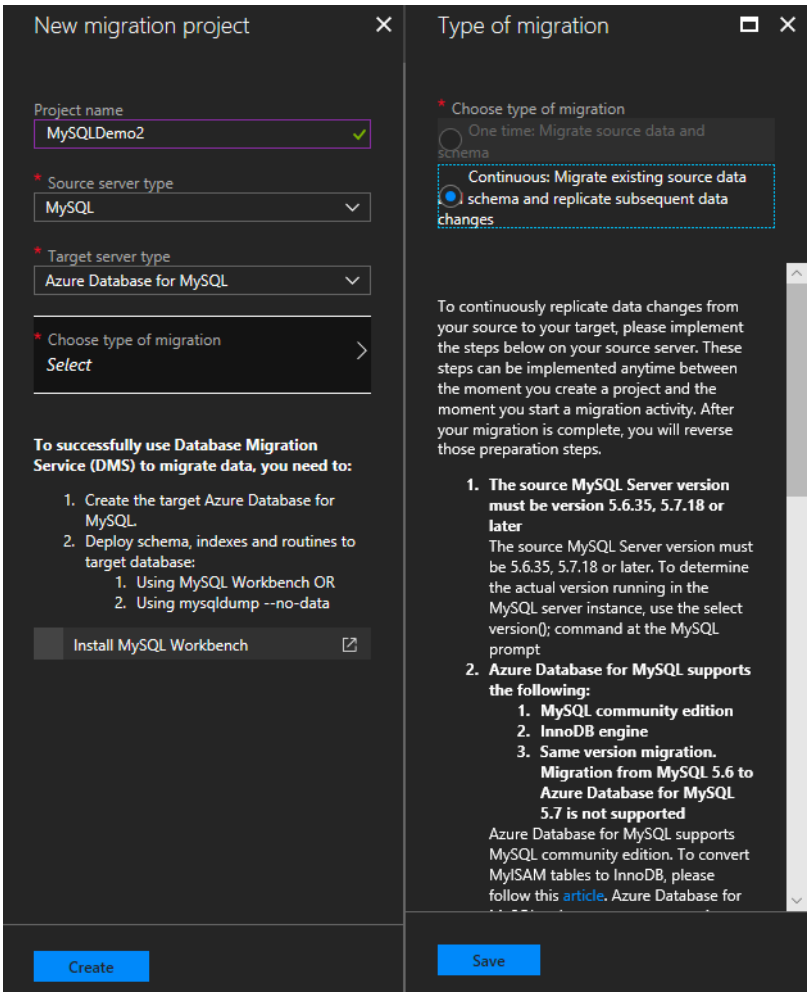
7. After DMS has completed in provisioning, you will get a notification in the notification tab (upper right corner of Azure portal). Click on “Go to resource” button.



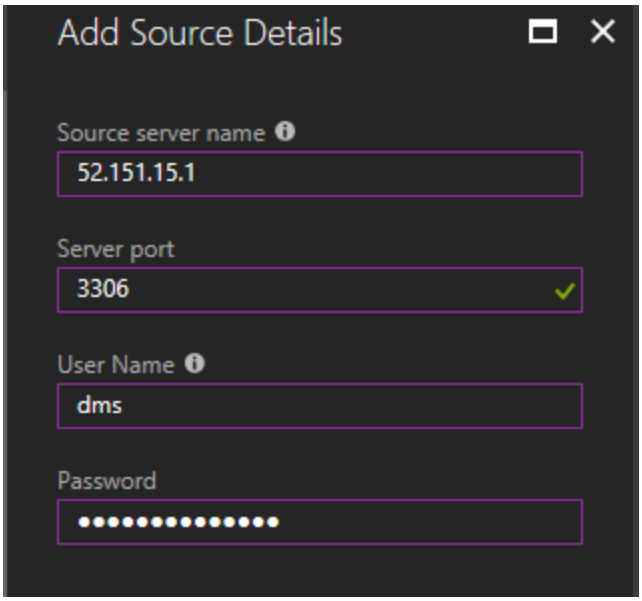
8. Click on +New Migration Project



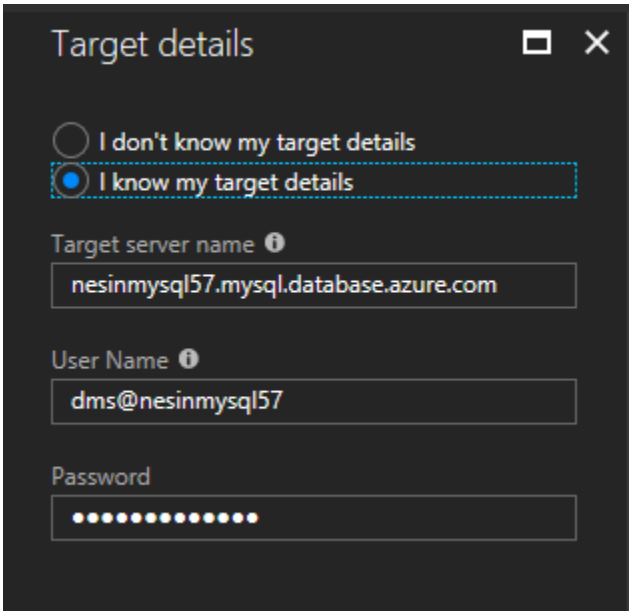
9. Fill in the information about the migration project:-
    a. Fill in a project name
    b. Source server type: MySQL
    c. Target server type: Azure Database for MySQL
    d. Type of migration: Continuous (please read for the pre-requisite)
10. Click Create



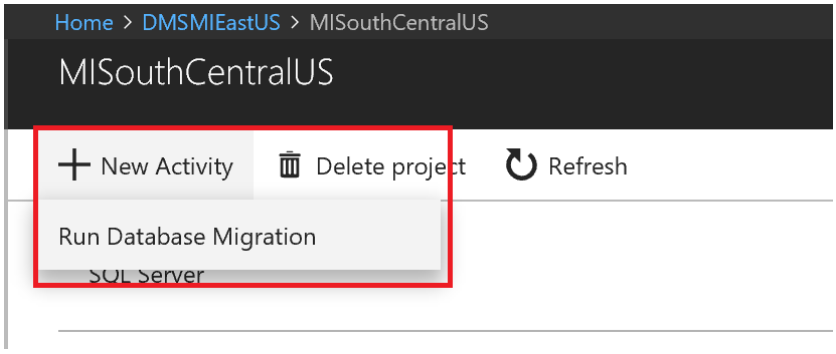
11. Next blade is about the source server and target server information:-
    a. Source: Use IP or DNS name to connect to on premise SQL Server
    b. Save



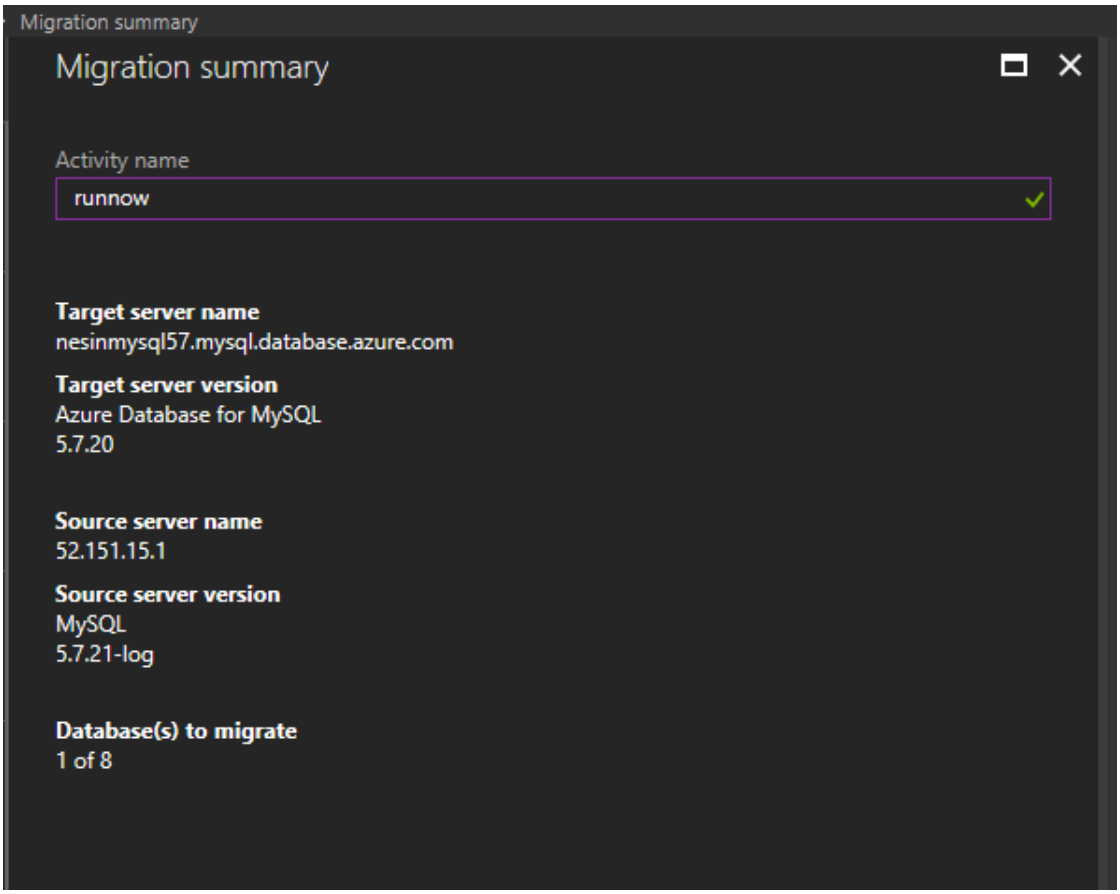
12. Select the database(s) that you like to migrate
13. Fill in the target which is Azure Database for MySQL information. If don't have an instance, create the instance here - https://ms.portal.azure.com/#create/hub



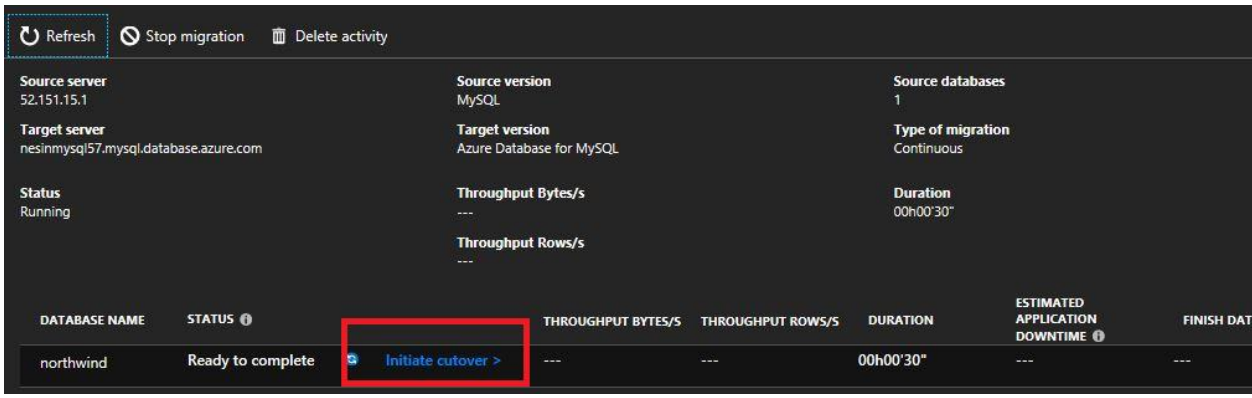
14. Click save on the summary page.
15. Click on +New Activity. Over the two screens, enter the password to connect to both source and target again.



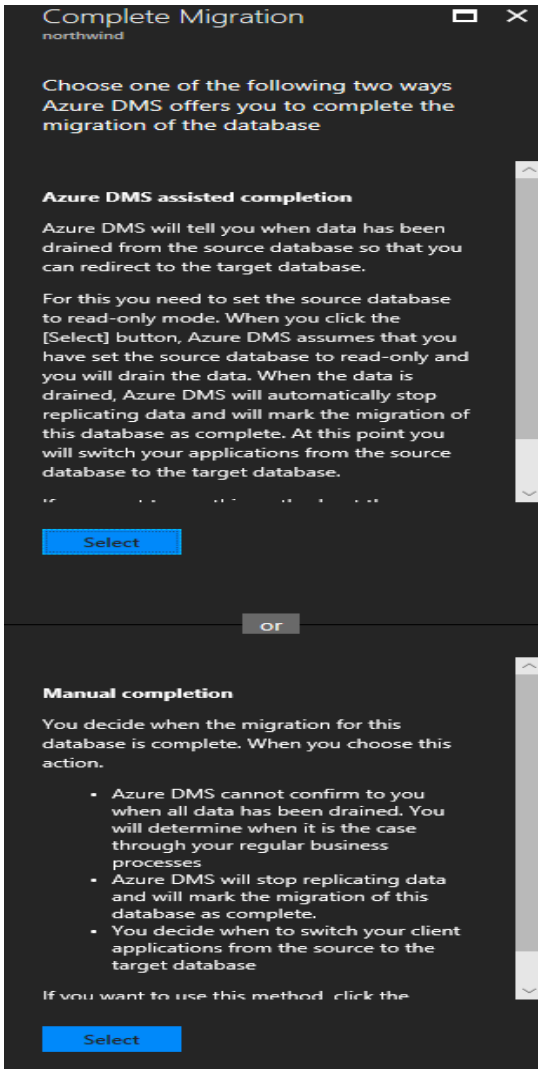
16. Confirm the database you want to migrate
17. Final step is to run this activity (migration). Give an activity name, and verify all the information in the summary blade:-

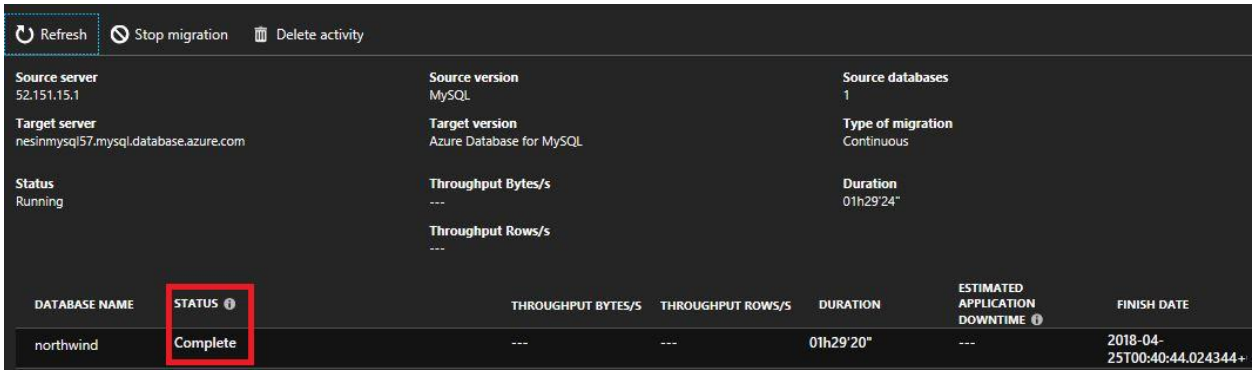


18. Click on Run migration to start migration.
19. The next blade will show the progress of the of migration in the following status:-
    a. Initializing – the migration is being setup. Immediate after setup is done, DMS will do an initial load and continue to replicate the new transactions to target. If you run select count(*) from tables at the target, you will start seeing rows get inserted.
    b. At this step, if you have foreign keys that you dropped during step #4 in section pre-migration step, you should add the foreign key here at this step.
    c. Ready to complete– Initiate cutover. This means the initial load and replication has complete. The database is ready for you take the next step.



d. If you click on Initiate cutover – you will see two options on how to cutover (complete) the migration.
    i. Option 1 DMS assist: Meaning DMS will drain the last transaction log from the source and write to the target. It will also mark the status of migration in DMS as complete.
    ii. Option 2 Manual completion: That means you decide to stop the migration (including continuous sync). There might be new transactions that remain in the source and haven't been replicated over to the target. DMS will mark the status of migration as complete.





Congratulations! You have completed a migration from MySQL to Azure Database for MySQL.