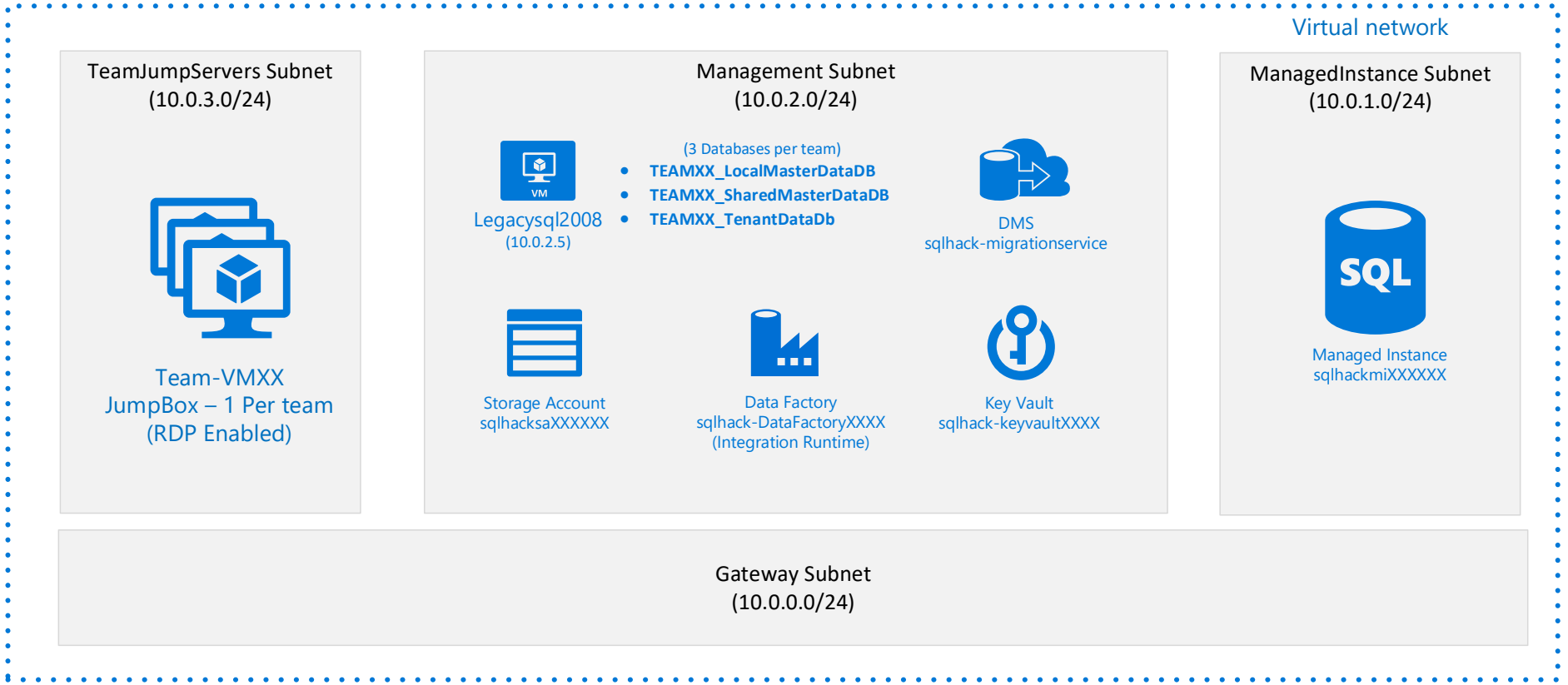
SQL Moderation Hack Database Migration Lab Step-by-step

Contents

Migration architecture and Azure components

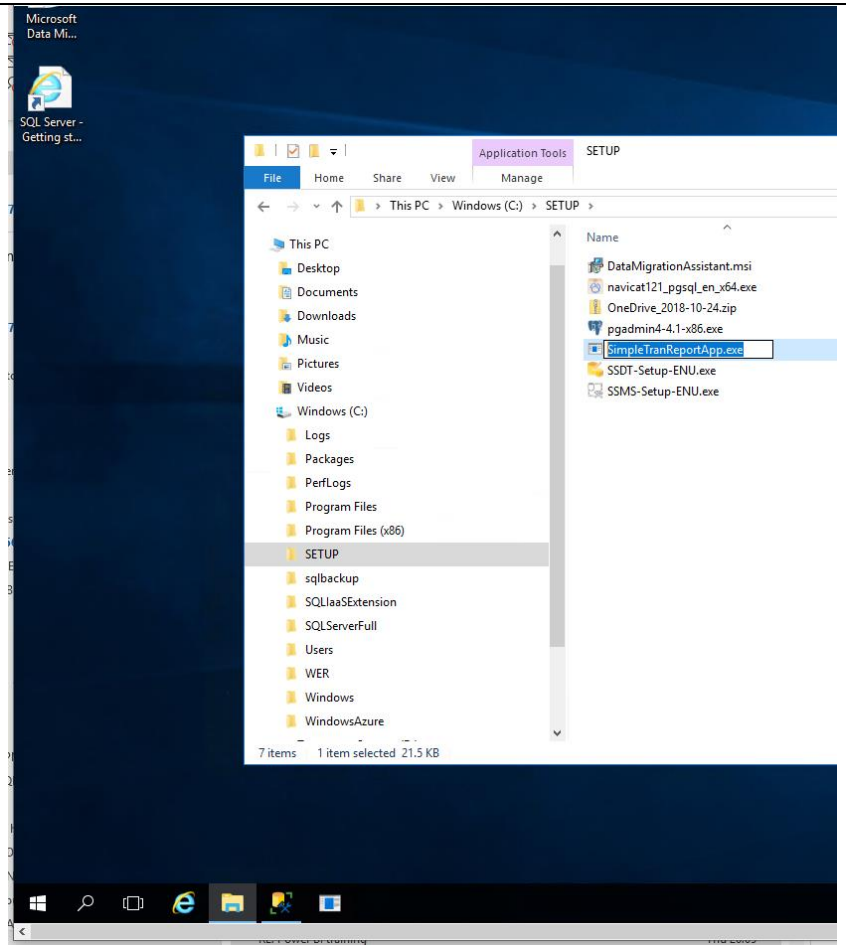


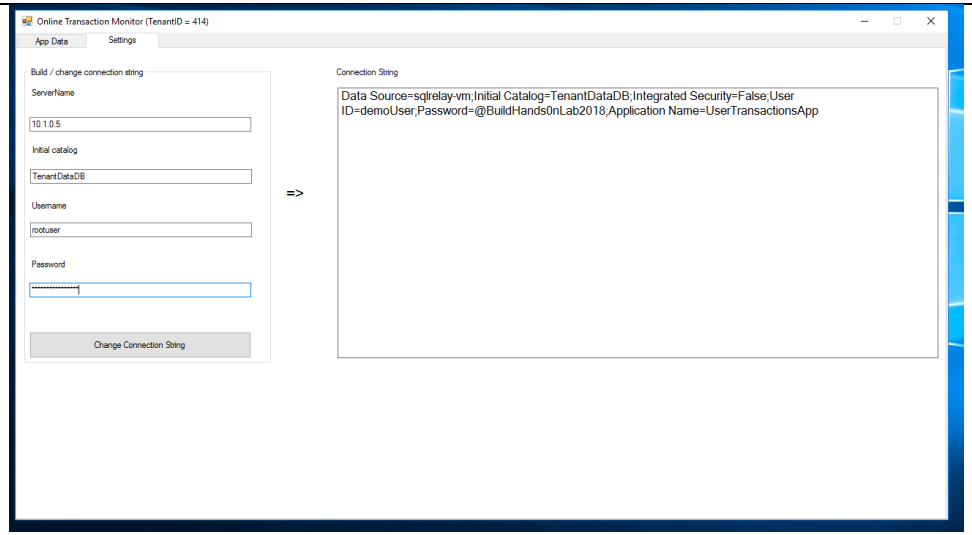
Generic Migration Content

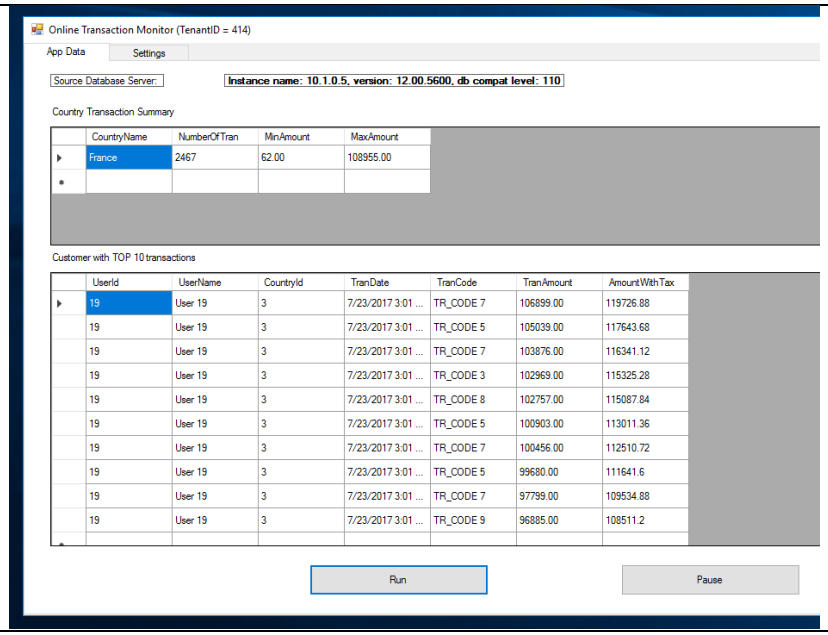
Narrative	Notes
Notes for outside of the workshop: *Familiarise yourself with Microsoft migration tools and the Azure Database Migration Guide*	Azure Database Migration Guide: https://www.microsoft.com/en-us/download/default.aspx DMA & download link: https://docs.microsoft.com/en-us/sql/dma/dma-overview?view=sql-server-2017 Microsoft Migration Portal: https://datamigration.microsoft.com/

1. Start the 'Online Transaction Monitor' application

In this section we'll connect the legacy Online Transaction Monitor application to the legacy SQL2008 databases and see it running.

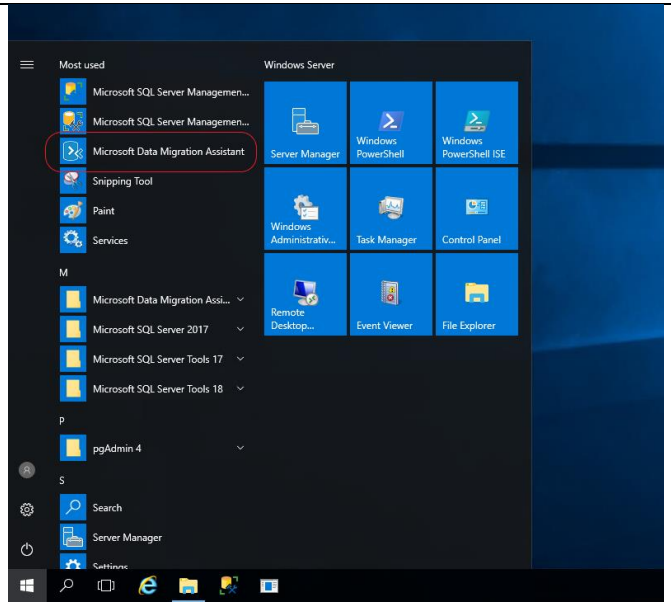
Narrative	Screenshot	Notes
We will set the sample application running to demonstrate how Azure Database Migration Services can be used to perform a an "on-line" migration of a running database with minimal downtime. RDP onto the Win10 management VM using the **IP Address from Labs and Parameters" doc** On the Task Bar– select 'SimpleTranReportApp' Or Windows Explorer - 'SimpleTranReportApp.exe'		In this scenario the legacy app has lost its source code, so only exists as an executable. We are not, however, blocked from migrating to Azure.

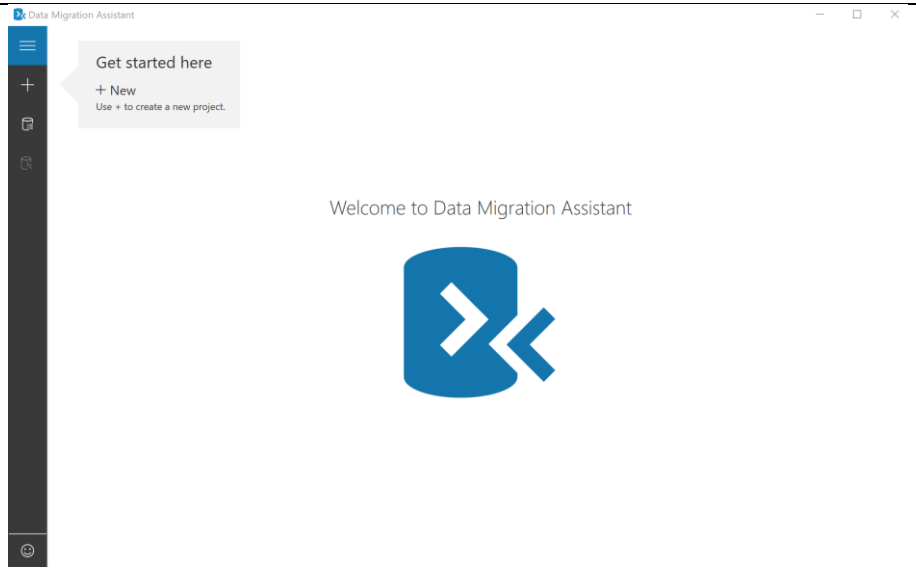
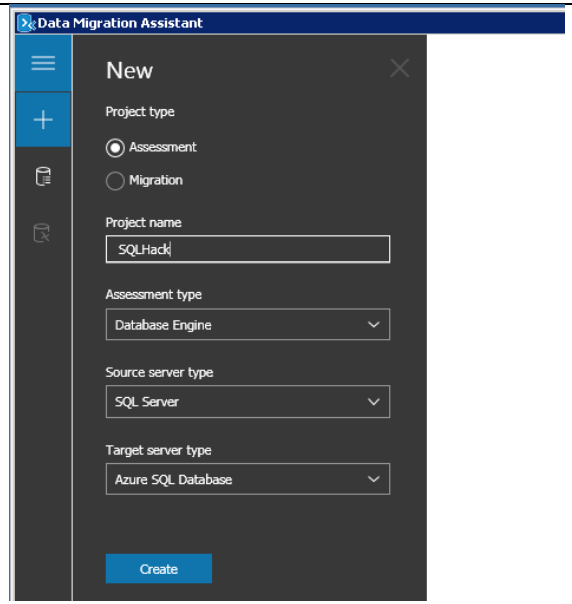
Once running, select the '**Settings**' tab and enter the following parameters into the fields identified: **ServerName:** **LEGACYSQL2008** **Initial Catalog:** **TEAMXX_TenantDataDb** **Username:** **TEAMXX** **Password:** **TEAMXX** Click the "**Change Connection String**" button to apply the connection string modifications		Use the parameters from the Appendix in the "Hands-on Lab - Data Migration" document. The connection string will now have been set to connect to the legacy SQL host – **LEGACYSQL2008** with appropriate Team database and login details.

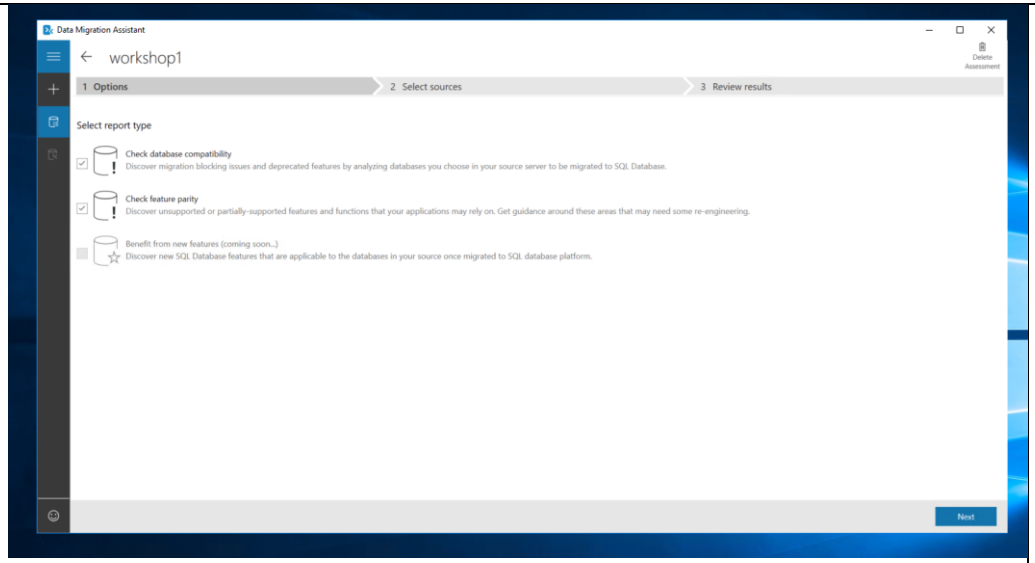
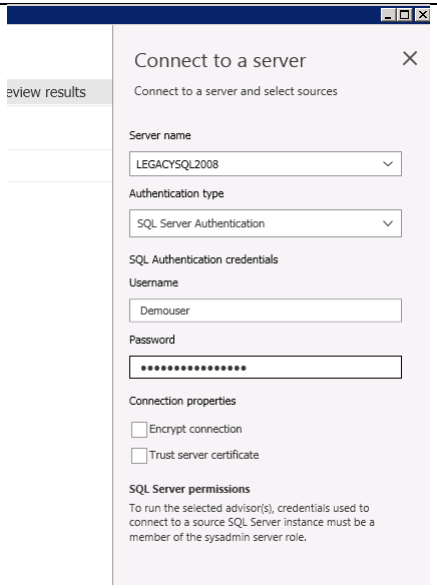
Select '**App Data**' tab and click the "**Run**" button.		The application will generate simulated transactional data. Notice how the 'Source Database Server' information at the top of the app reflects the parameters given in the previous step.

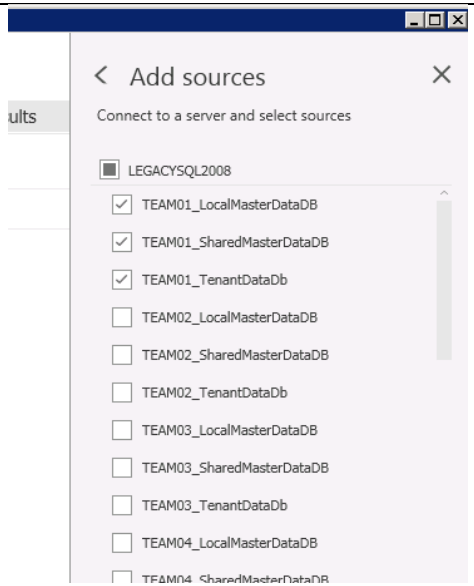
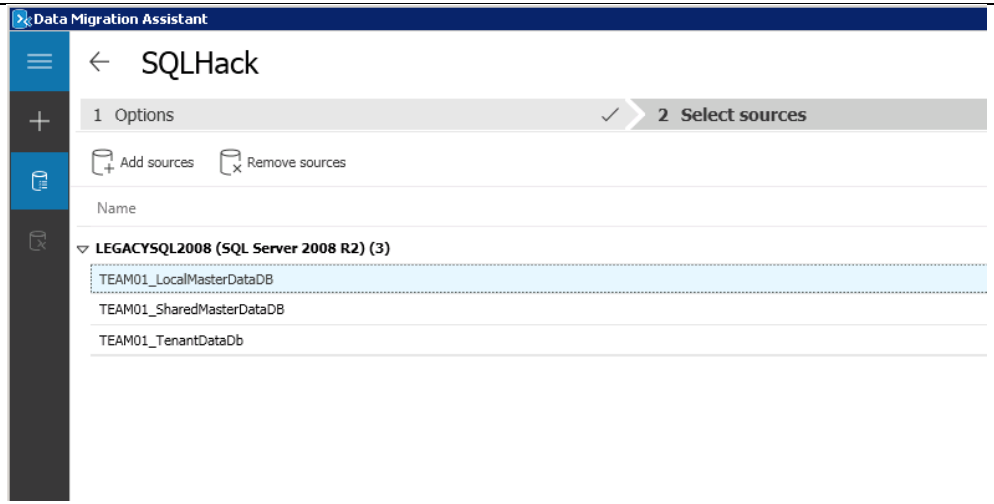
2. Assess the application databases for Azure SQL Database suitability

In this section we will use the Data Migration Assistant (DMA) to assess the applications database for suitability for migration to Azure Cloud.

Narrative	Screenshot	Notes
We need to determine the suitability of the database(s) for migration to Azure. This includes checking for compatibility and feature support with Azure Database. RDP onto **your teams Win10 Management VM (TEAM-VMXX)** and run DMA from the Start or Desktop icon.		See link above if you need to download DMA.

You should see this screenshot to the right. Select the **"+"** to create a new **assessment** project		
Select/Enter the following details: **Project name:** **Workshop1** **Assessment type:** **Database Engine** **Source server type:** **SQL Server** **Target server type:** **Azure SQL Database** Click '**Create**'		Our first project assessment assumes we will be migrating to Azure SQL DB, so the selections shown in the screenshot need to be selected.

Select the assessment checks (Report Type) to be made: **Check database compatibility** **Check feature parity** Click '**Next**'		DMA can test for both database compatibility and feature parity compliance against the Azure target. As this is the initial evaluation, we are assessing a database(s) we will perform all of these tests.
Enter the source/legacy SQL details: **Server Name:** **10.0.2.4** **Authentication Type:** **SQL Server Authentication** **Username:** **Demouser** **Password:** **Demo@pass1234567** **Untick "Encrypt connection"** Click '**Connect**'		When performing this within your own subscription you will enter the host, authentication and connection types according to your company guidelines and practices. *Bear in mind that DMA needs to connect to a source SQL Server using an account that belongs to the sysadmin role.* As this document is produced within a workshop environment Active Directory, Certificates and encryption has not been setup.

Select **only** the 3 database used by your 'Online Transaction Monitor' app. These will have a TEAMXX prefix where XX should be replaced by your team number. **TEAMXX_LocalMasterDataDb** **TEAMXX_SharedMasterDb** **TEAMXX_TenantDataDb** Click '**Add**' to add them to the assessment.		DMA will show all databases located on the Source host and display them so you can decide which ones to include in this assessment project. Note that you can assess multiple databases a t the same time.
You should now see the screen on the right with the relevant TEAMXX databases listed. Select '**Start Assessment**'		Note: DMA allows you to either 'Add' or 'Remove' additional data sources as needed at this point. Also note that DMA has identified what compatibility level each source database is running under.

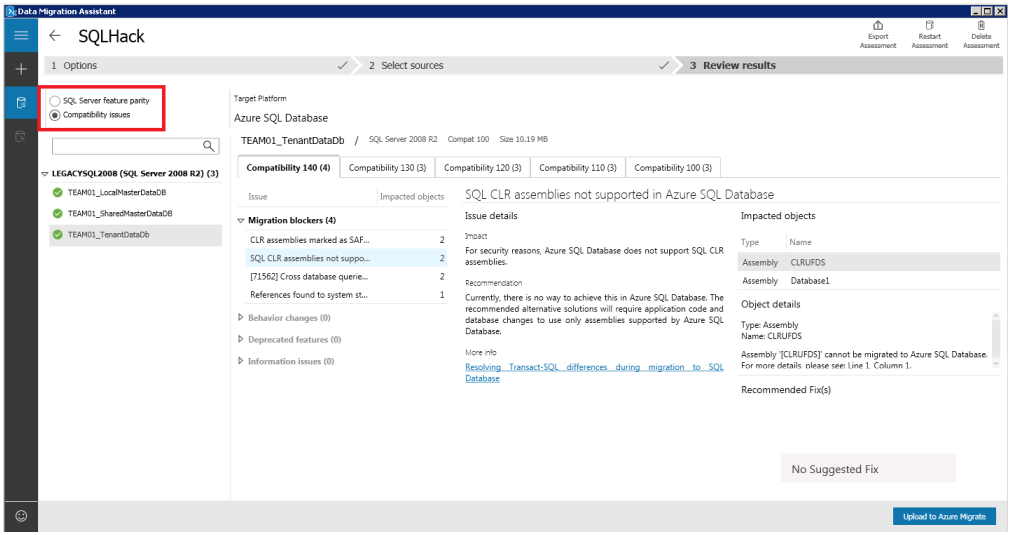
DMA will now show the results of the assessment using 2 separate reports:

'**SQL Server feature parity**' which is a server level report highlighting any server functionality (e.g. MSDTC) that the source DBs are using that isn't fully supported on the target – in this case Azure SQL Database. In our assessment here is one 'Unsupported feature' here (cross database queries).

'**Compatibility Issues**' which is a database level report detailing individual objects that have compatibility issues.

Select '**TEAMXX_TenantDataDb**' Note the 4 'Migration blockers' including CLR which the database uses.

CLR is not supported on Azure SQL DB but is supported by Azure SQL Database Managed Instance (SQLMI).

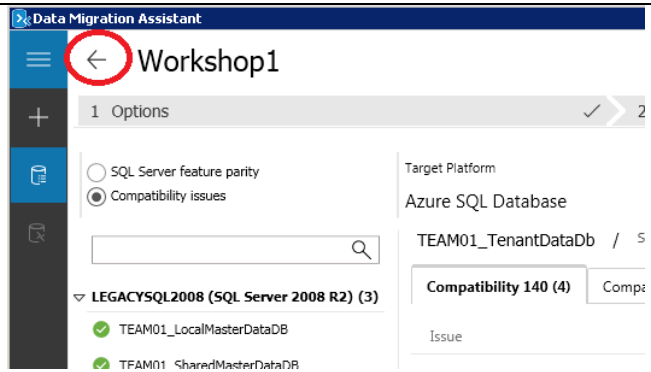
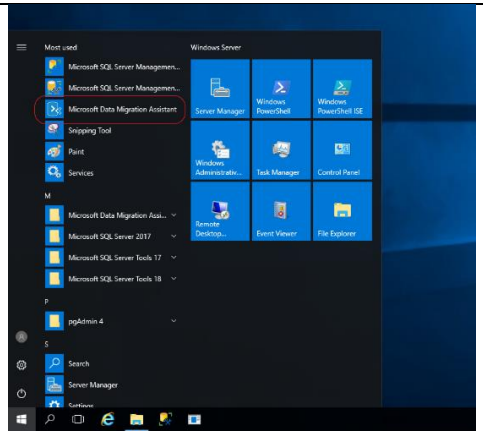


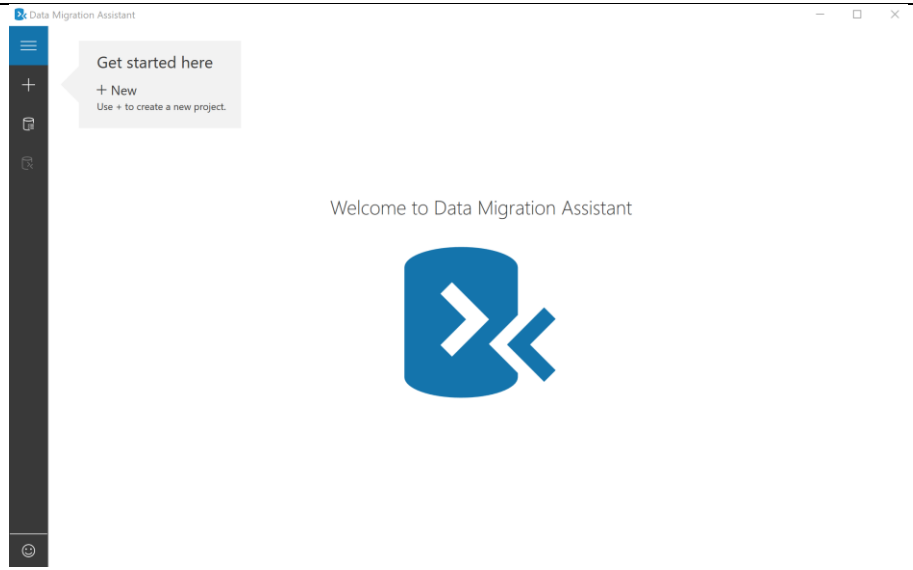
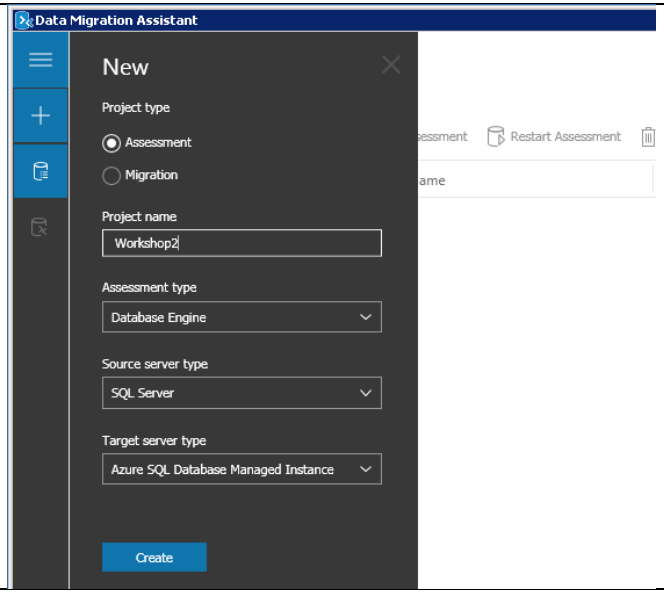
Note: Toggle the parity and compatibility issues radio button (top left) to switch between the 2 reports.

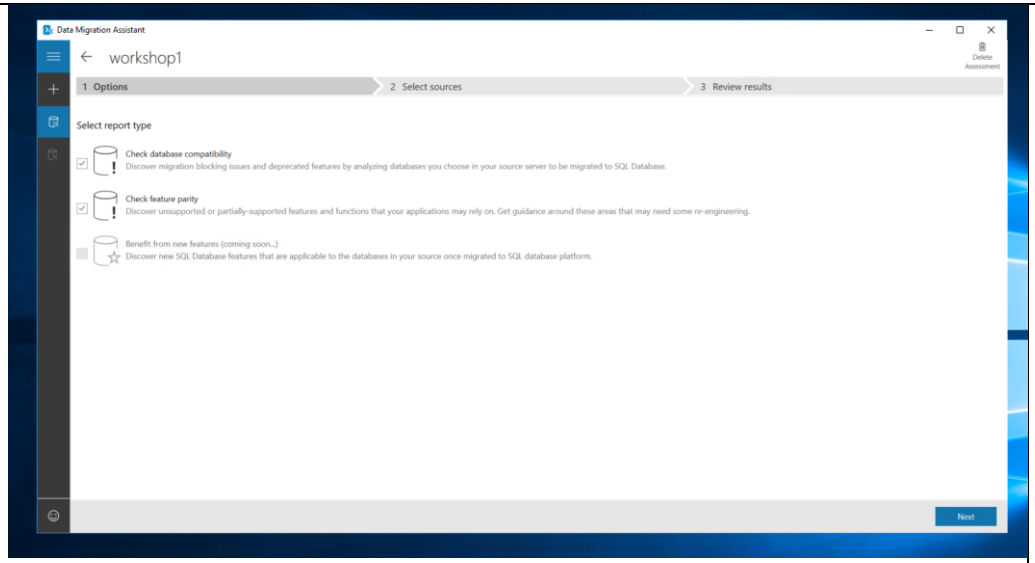
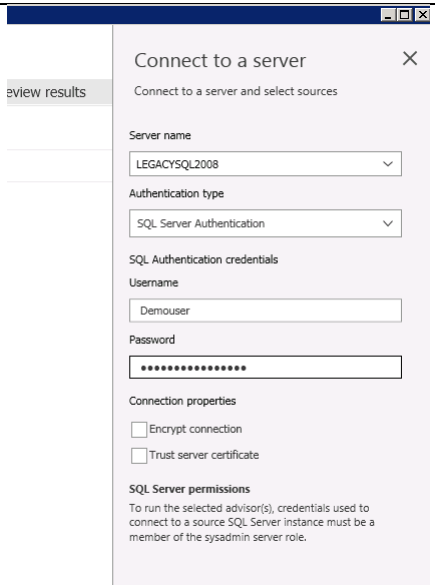
'SQL Server feature parity' shows what features are not supported in the target data source. Under 'Details' and 'Databases' you will find remedial action that are required and the databases impacted.

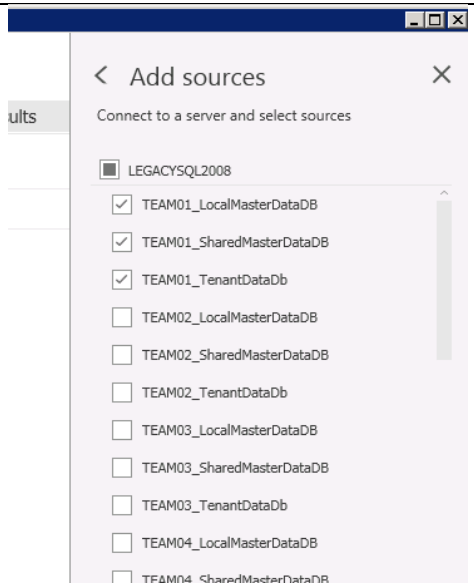
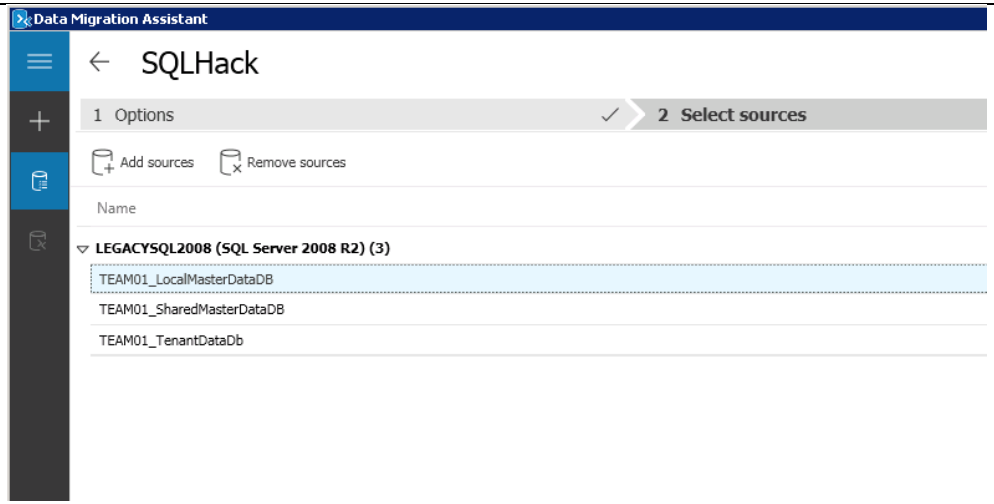
'Compatibility Issues' shows, over the compatibility tabs, issues that need to be addressed to permit the database(s) to run, in the chosen compatibility level (e.g. 140, 130, 120, 110).

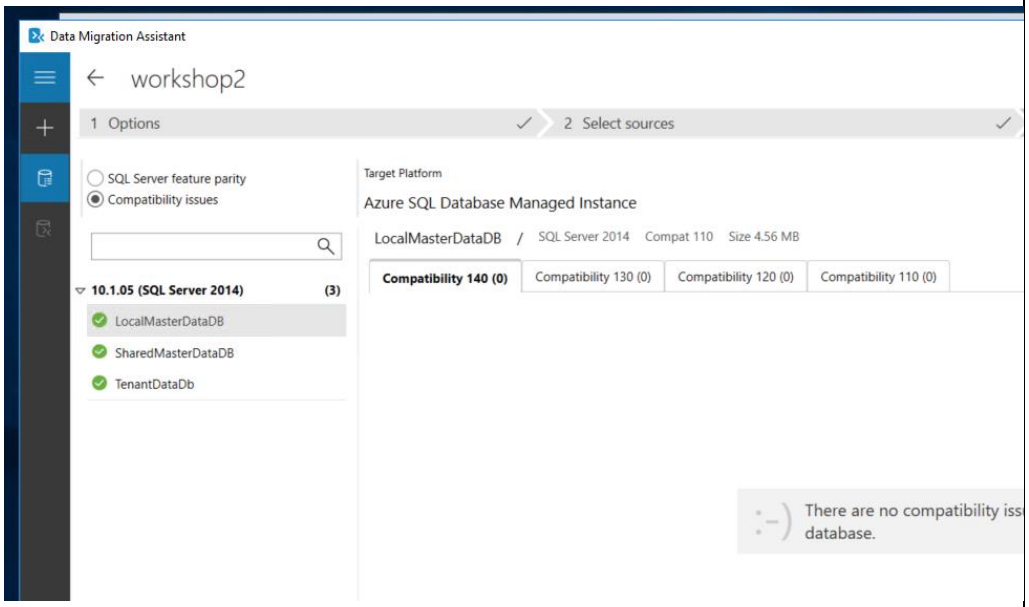
If you have multiple databases, as with the example screenshot, you need to highlight EACH database to see the compatibility issues.

Once you're reviewed the assessment click the back arrow to see a list of current DMA projects.		
	Due to the need for CLR, we need to repeat the assessment with Azure SQL DB Managed Instance as the target.	
We need to determine the suitability of the database(s) for migration to Azure. This includes checking for compatibility and feature support with Azure Database. Logon onto your teams Win10 management VM and run DMA from the Start or Desktop icon.		See link above if you need to download DMA.

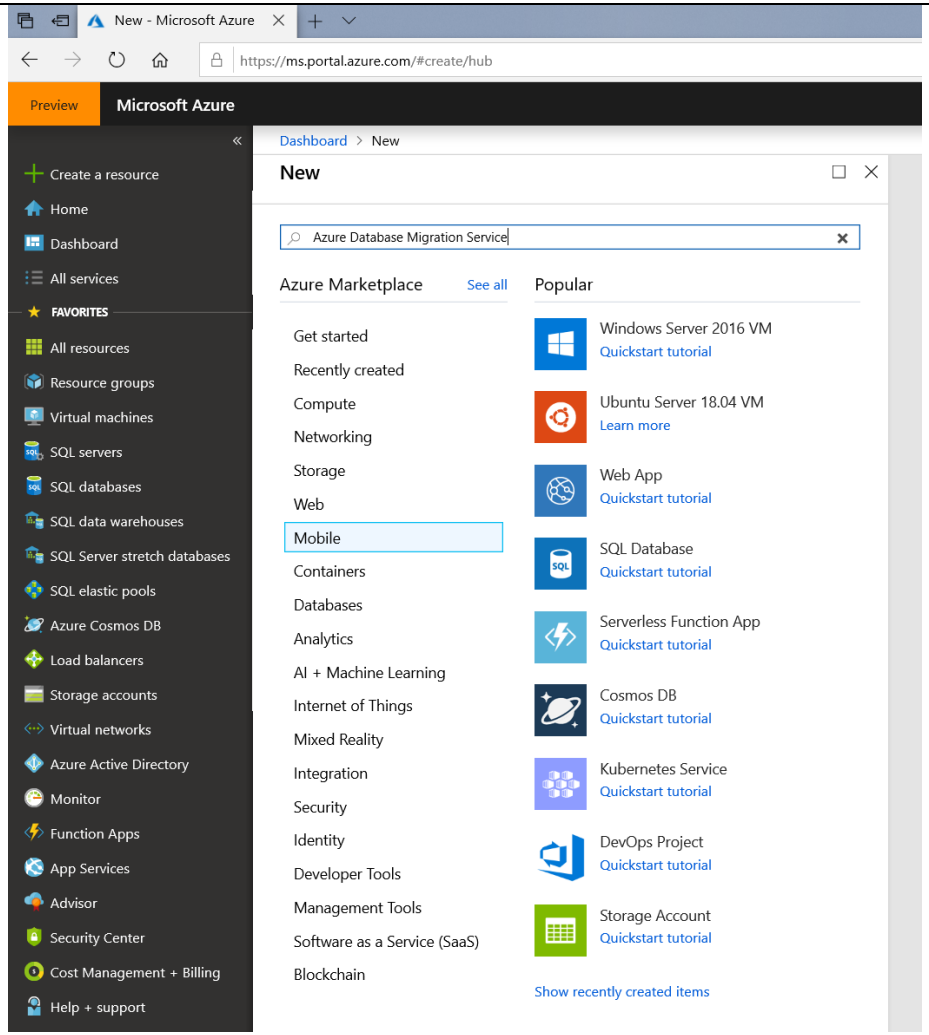
You should see this screenshot to the right. Select the **"+"** to create a new **assessment** project.		
Select/Enter the following details: **Project name:** **Workshop2** **Assessment type:** **Database Engine** **Source server type:** **SQL Server** **Target server type:** **Azure SQL Database Managed Instance** Click '**Create**'		Our 2nd assessment project assumes we will be migrating to Azure SQL DB Managed Instance, so the selections shown in the screenshot need to be selected.

Select the assessment checks (Report Type) to be made: **Check database compatibility** **Check feature parity** Click '**Next**'		As we saw previously DMA can test for both database compatibility and feature parity compliance against the chosen target. As before we will assess all the databases against all of the tests.
Enter the source/legacy SQL details: **Server Name:** **LEGACYSQL2008** **Authentication Type:** **SQL Server Authentication** **Username:** **Demouser** **Password:** **Demo@pass1234567** **Untick "Encrypt connection"** Click '**Connect**'		When performing this within your own subscription you will enter the host, authentication and connection types according to your company guidelines and practices. *Bear in mind that DMA needs to connect to a source SQL Server using an account that belongs to the sysadmin role.* As this document is produced within a workshop environment Active Directory, Certificates and encryption has not been setup.

Select **only** the 3 database used by your 'Online Transaction Monitor' app. These will have a TEAMXX prefix where XX should be replaced by your team number. **TEAMXX_LocalMasterDataDb** **TEAMXX_SharedMasterDb** **TEAMXX_TenantDataDb** Click '**Add**' to add them to the assessment.		DMA will show all databases located on the Source host and display them so you can decide which ones to include in this assessment project. Note that you can assess multiple databases a t the same time.
You should now see the screen on the right with the relevant TEAMXX databases listed. Select '**Start Assessment**'		Note: DMA allows you to either 'Add' or 'Remove' additional data sources as needed at this point. Also note that DMA has identified what compatibility level each source database is running under.

As before DMA will now show the results from the assessment as the 2 reports. Note the '**SQL Server feature parity**' report will either be clean or it will show a single issue for the system SQL Agent Job 'syspolicy_purge_history' which is not applicable to Azure SQL DB Managed Instance & can be ignored. The '**Compatibility Issues**' report should be clear for all 3 databases showing that they can be migrated to Azure SQLDB Managed Instance without changes.		Note: Toggle the parity and compatibility Issues radio button (top left) to see how DMA. 'SQL Server feature parity' shows what features are not supported in the target datasource. Under 'Details' and 'Databases' you will find remedial action that are required and the databases impacted. 'Compatibility Issues' shows, over the compatibility tabs, issues that need to be addressed to permit the database(s) to run, in the chosen compatibility level (e.g. 140, 130, 120, 110). If you have multiple databases, as with the example screenshot, you need to highlight EACH database to see the compatibility issues.
	We are now ready to migrate the application databases to Azure SQL Database Managed Instance	

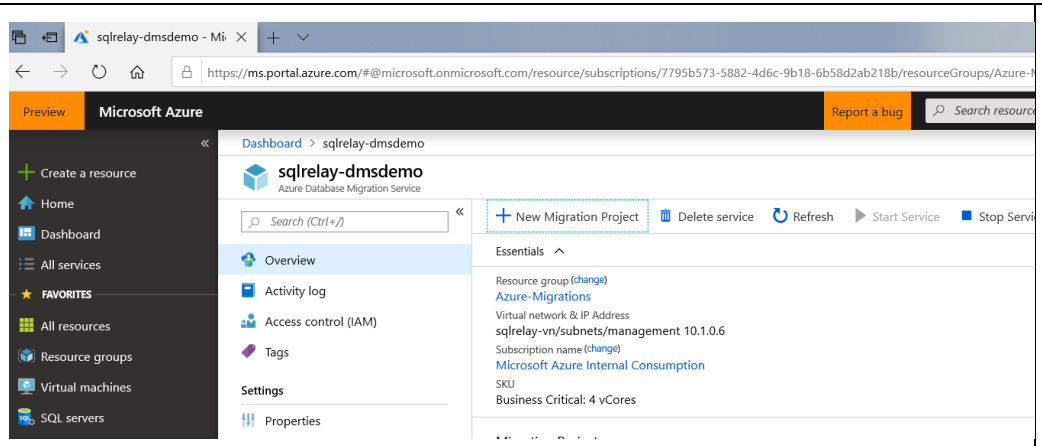
3. Use Azure Database Migration Service (DMS) to migrate the 3 application databases

Narrative	Screenshot	Notes
We will be using Azure Database Migration Service (DMS) to migrate the legacy databases to Azure. For the workshop DMS will already been provisioned. In your own subscription if you do not have DMS: • select **'+ Create a Resource'** • enter and search for 'Database Migration' • select **'Azure Database Migration Service'** Follow the setup blades according to your organisational guidelines.		DMS is provisioned as a service which hosts & runs multiple migration Projects. Each Project is responsible for migrating one or more databases. Although a Project can migrate multiple databases each Project can only migrate databases from a single source host to a single target destination. In this lab we will use a single Project to migrate 3 databases from the same legacy SQL2008 host to SQL Managed Instance. DMS can host and run different types of database migration Projects under the same e.g. separate Project for separate source servers. Overview of DMS: Azure Database Migration Service DMS tutorials: https://docs.microsoft.com/en-us/azure/dms/

For this workshop:

In the Azure portal, open the workshop Resource Group and locate the Azure Database Migration Service and open it.

On the DMS Overview blade click '**New Migration Project**'



On the "New migration project" blade use the following settings:

Project name:
TEAMXX_migration
(replace XX with your team number)

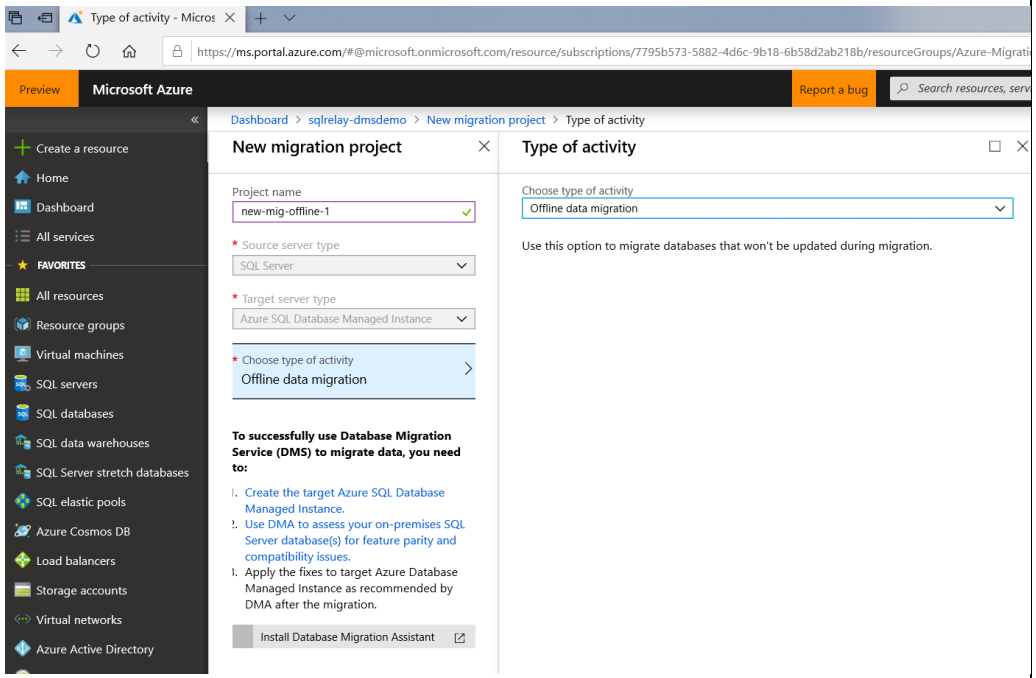
Source server type:
SQL Server

Target Server type:
Azure SQL Database Managed Instance

Choose type of activity:
Offline data migration

Click '**Save**'

Select '**Create and run activity**'

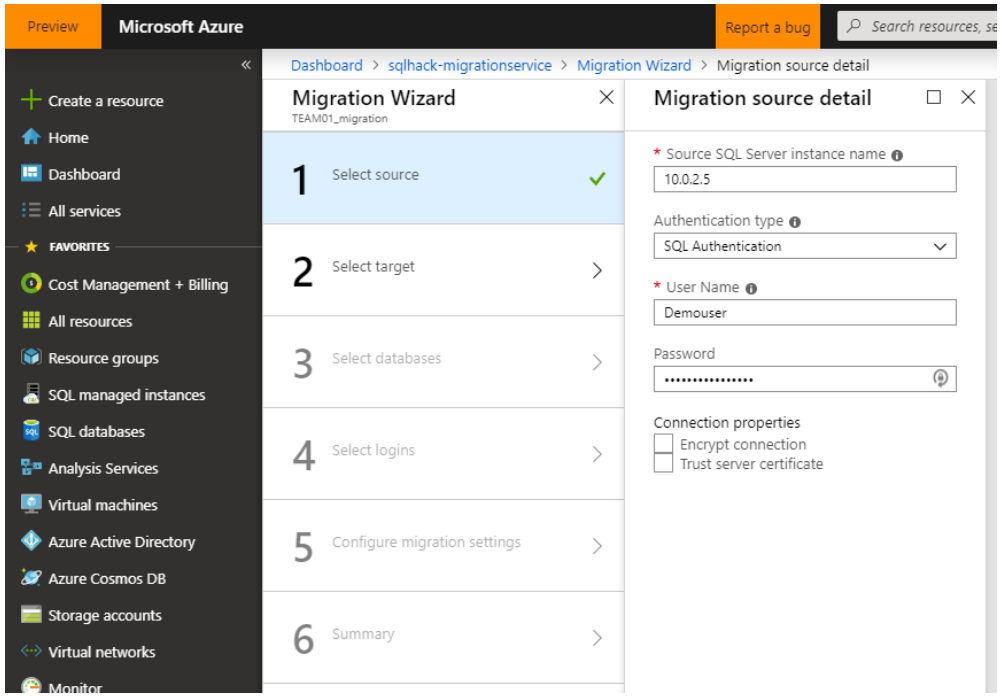


DMS can perform two types of database migrations:

- **Offline**
- **Online**

Offline migrations use backup files. The backups can be provided to DMS or DMS can create the backup as part of a project.
Whilst the simplest to perform taking the backup, moving it to Azure and restoring it can cause significant downtime.

Online migrations use a replication or log shipping approach to keep the source & target in sync. Whilst more complex it significantly reduces database downtime.

DMS will now launch the migration configuration blades. Use the following values for each of the configuration steps: **STEP 1: Select Source** *This uses the source database host VM LEGACYSQL2008 details from the "Lab and parameters" doc.* **Source SQL Server:** **10.0.2.4** **Authentication Type:** **SQL Authentication** **User Name:** **Demouser** **Password** **Demo@pass1234567** For this lab ***only*** uncheck both "Connection Properties" options as per the screenshot. Click '**Save**'. This will perform a connection test.		The account that DMS uses to connect to the source instance must be a member of sysadmin.

STEP 2: Select Target
This uses the target Azure SQL Managed Instance details from the "Lab and parameters" doc.

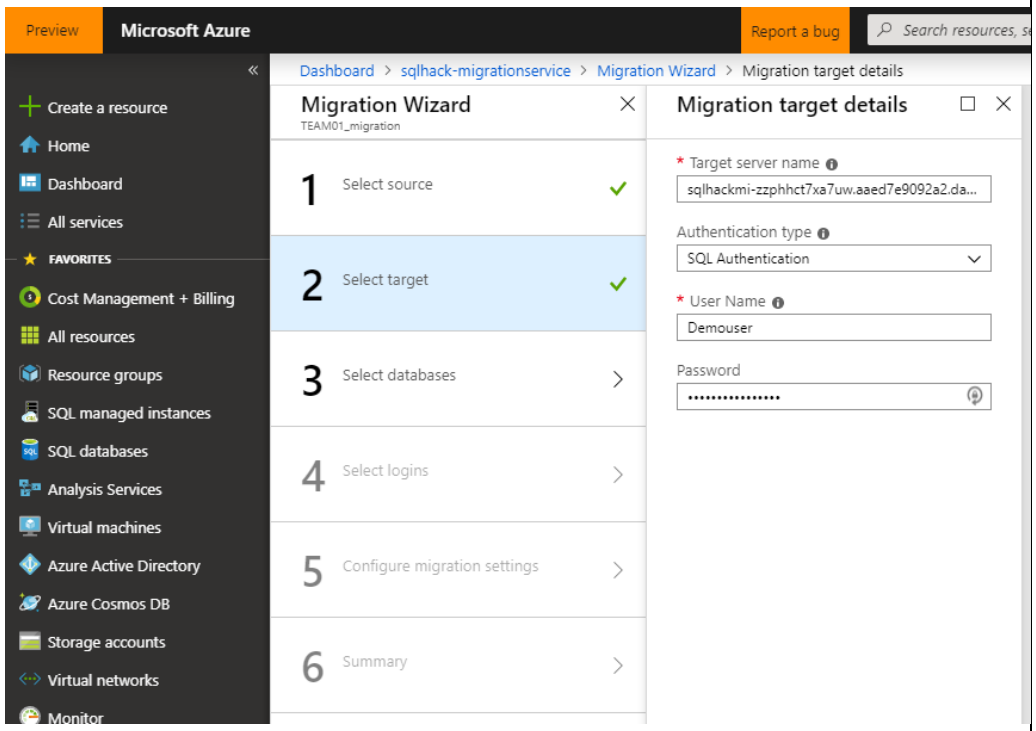
Target SQL Server:
(Use the Fully Qualified Domain Name for the SQL Managed Instance)
SqlhackmiXXXXXX
See C:_SQLHACK_\LABS\01-Data_Migration\ManagedInstanceFDQN.txt

Authentication Type:
SQL Authentication

User Name:
Demouser

Password:
Demo@pass1234567

Click '**Save**'.
This will perform a connection test.



STEP 3: Select Databases

The application has 3 databases supporting it. Select the 3 database for you team.

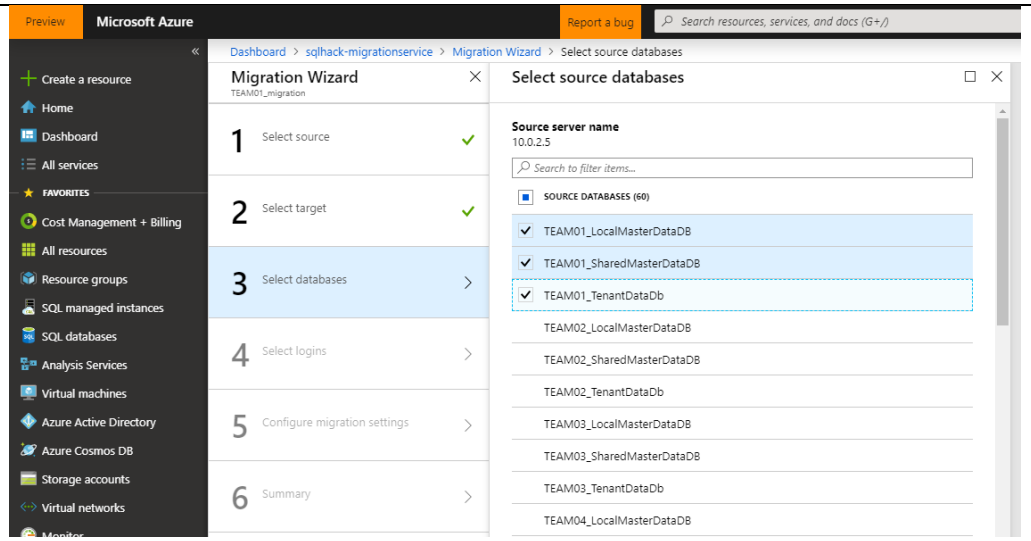
TEAMXX_LocalMasterDataDb

TEAMXX_SharedMasterDb

TEAMXX_TenantDataDb

(replace XX with your team number)

Select '**Save**'

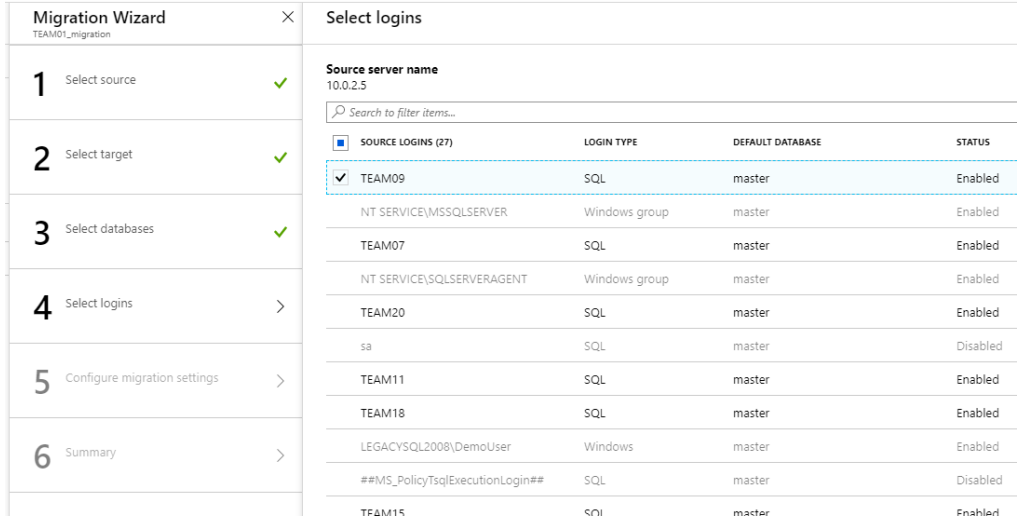


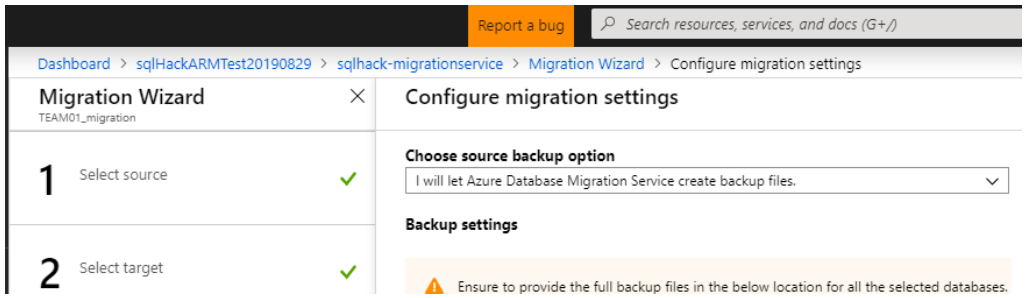
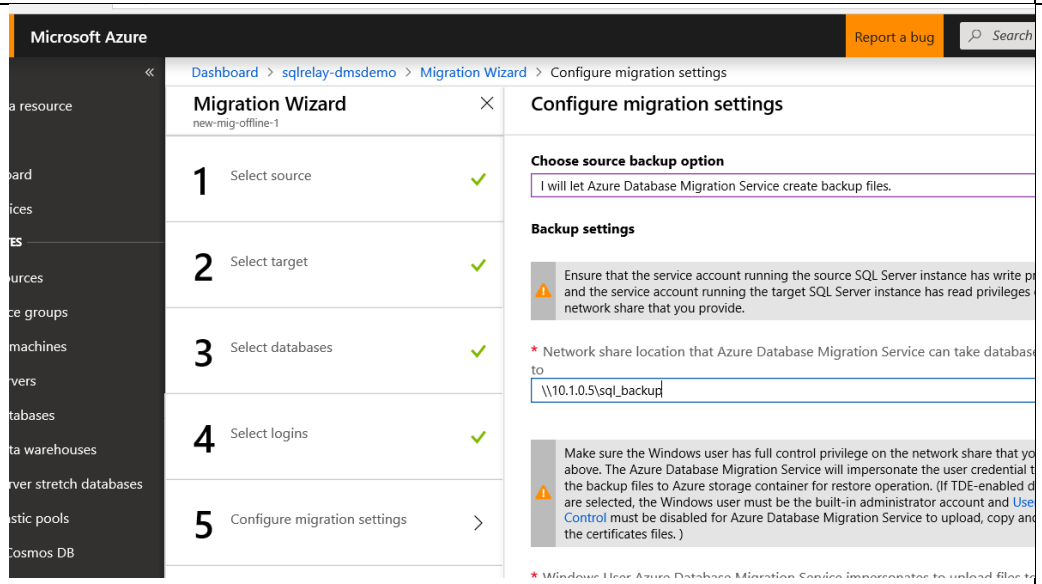
STEP 4: Select Logins

As with a traditional on-premise migration the SQL Server level logins must be migrated alongside the database. Select the database logins, from the list, that are required for the application.

Select ***only*** your '**TEAMXX**' account.

Select '**Save**'

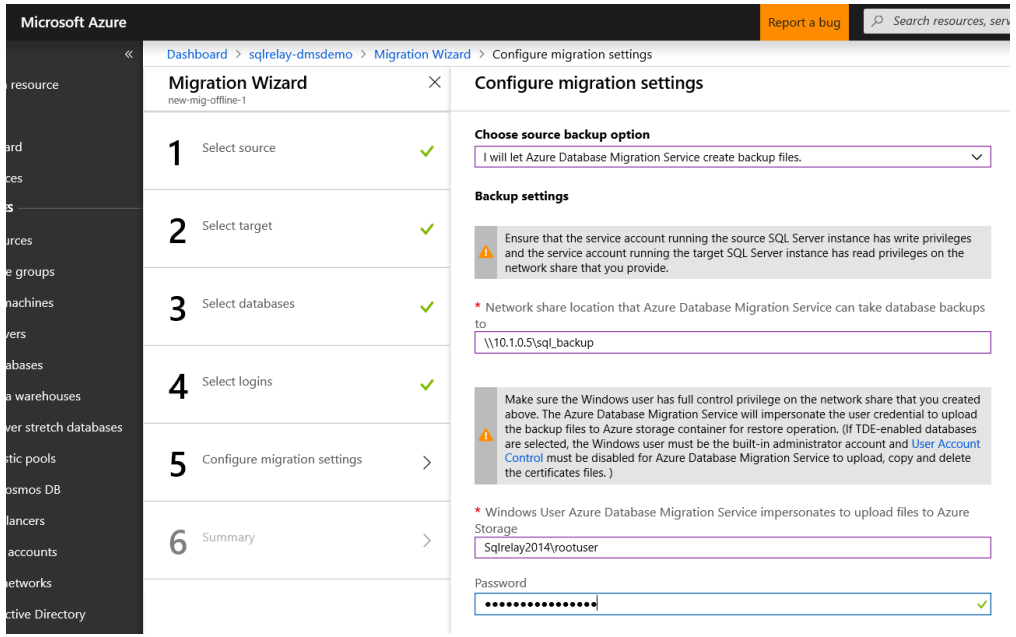


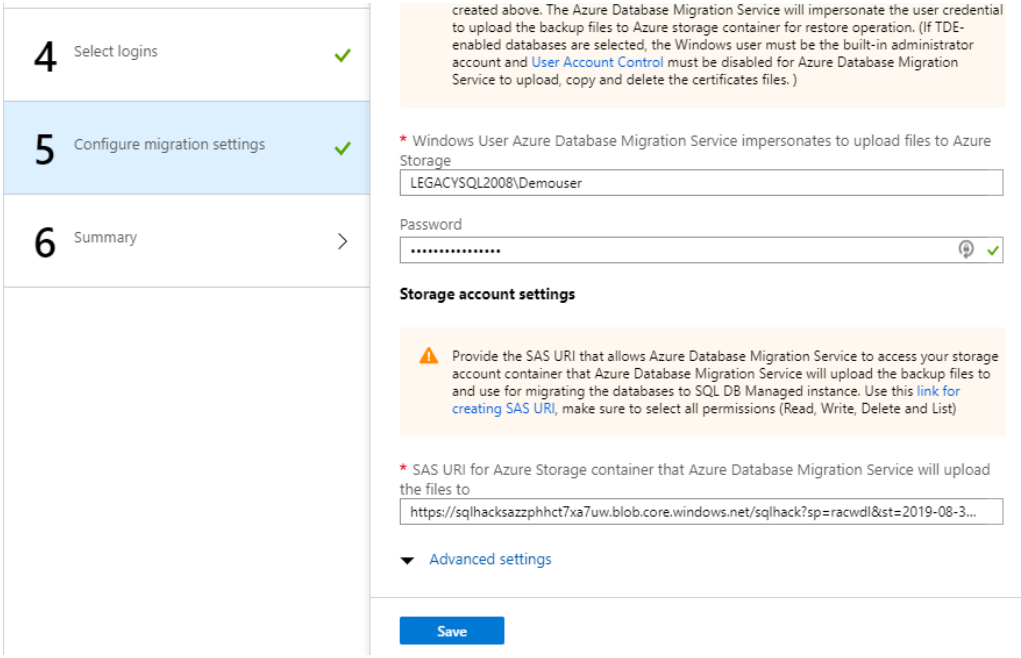
Step 5a: Configure migration Settings ***(Source Backup Option)*** We are running an offline migration which will use backups of the DBs to be migrated. We want DMS to perform the backup, so select this option from the "Choose source backup option" (as shown).		DMS can either use backup files provided or take the backups as part of the migration activity.
Step 5b: Configure migration Settings ***(backup location)*** We can now enter the Windows share that the source will write the database backups to. This is a Windows share which we'll enter in the format: **\\IPAddress\FILESHARE**		

Step 5c: Configure migration Settings
(source database access)

We now provide the username and password of the windows account that will permit the DMS service to run the backups on the source host and save them to the share on the legacy server.

Storage (Username):
LEGACYSQL2008\Demouser
Password:
Demo@pass1234567



Step 5d: Configure migration Settings ***(Azure storage account settings)*** DMS is an Azure Cloud Service. We have to provide the Shared Access Signature URI (SAS URI for short) to permit DMS to upload the backup files, from the share on the LEGACYSQL2008 host to Azure blob storage. The SAS URI is provided in the "Lab and Parameters" doc. Enter the SAS URI key located in: **C:_SQLHACK_\LABS\01-Data Migration\SASKey.txt** and click '**Save**'. This will perform a connection test and if successful will display the Migration Summary blade.		Once DMS has taken backups of the databases to be migrated it needs to move these backups to Azure storage. This is so the target SQL Managed Instance can access them to restore them.

STEP 5e: Configure migration settings
(Migration Summary)

DMS displays the migration configuration settings.

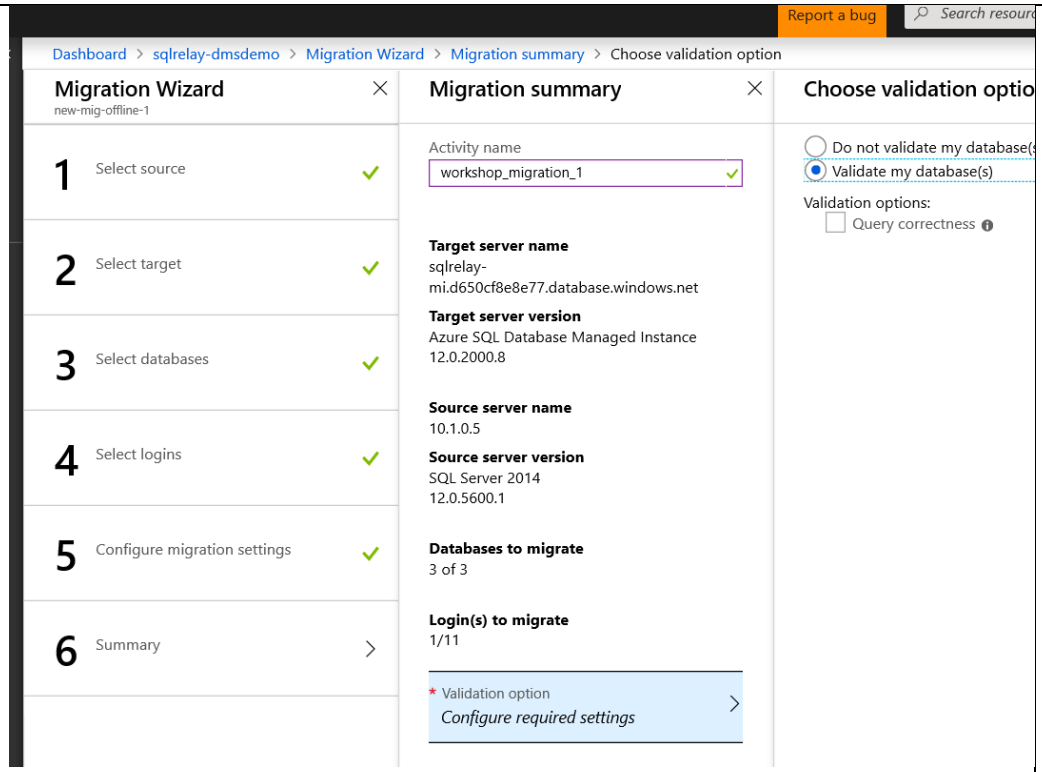
Now we need to use these settings to actually perform a migration. To do this we create an “Activity”.

Use the following values:

Activity Name
workshop_migration_1

Validation option:
Validate my database(s)’

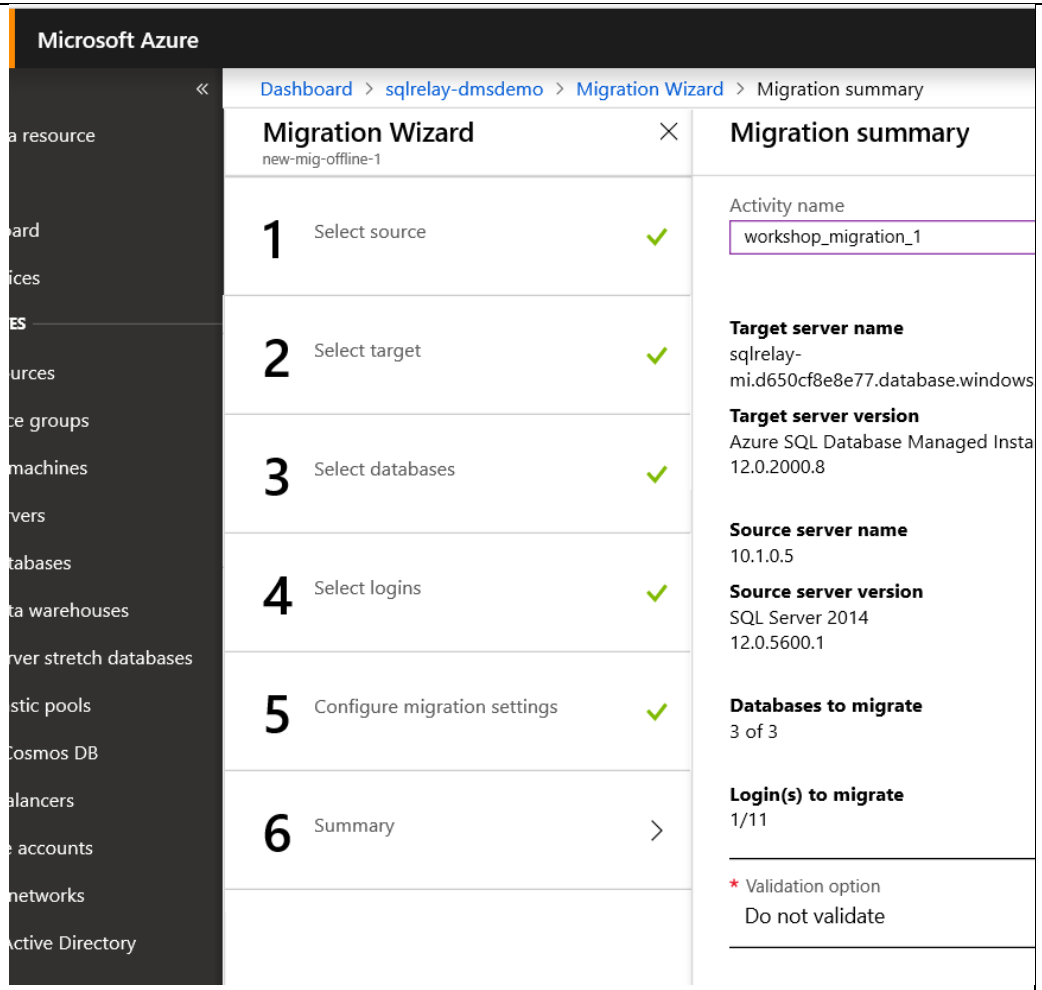
Select ‘**Save**’



STEP 5f: Migration Summary ***(run the migration)***

The migration activity is now configured and saved and is ready to run.

Select '**Run migration**'

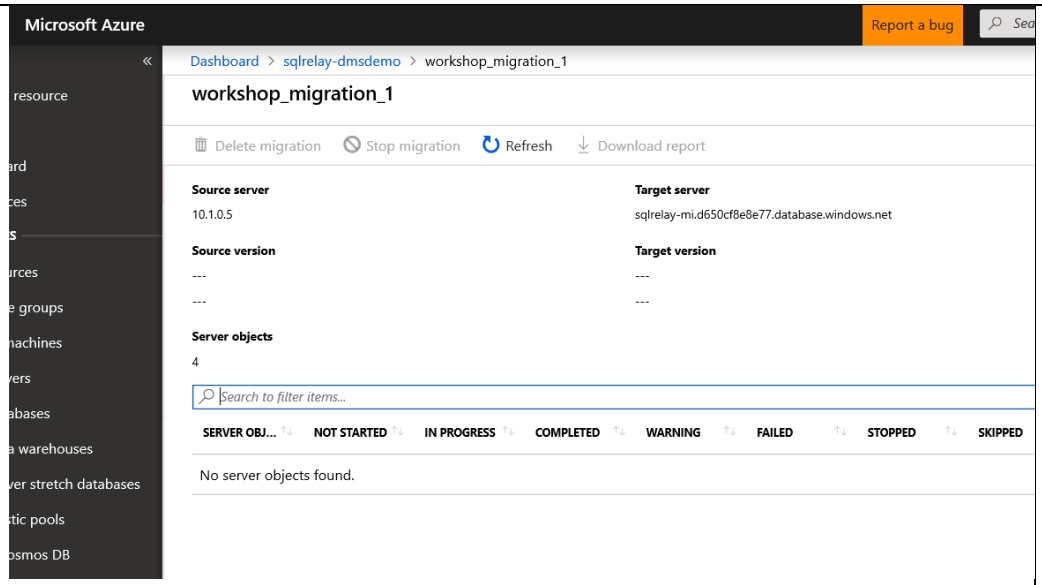


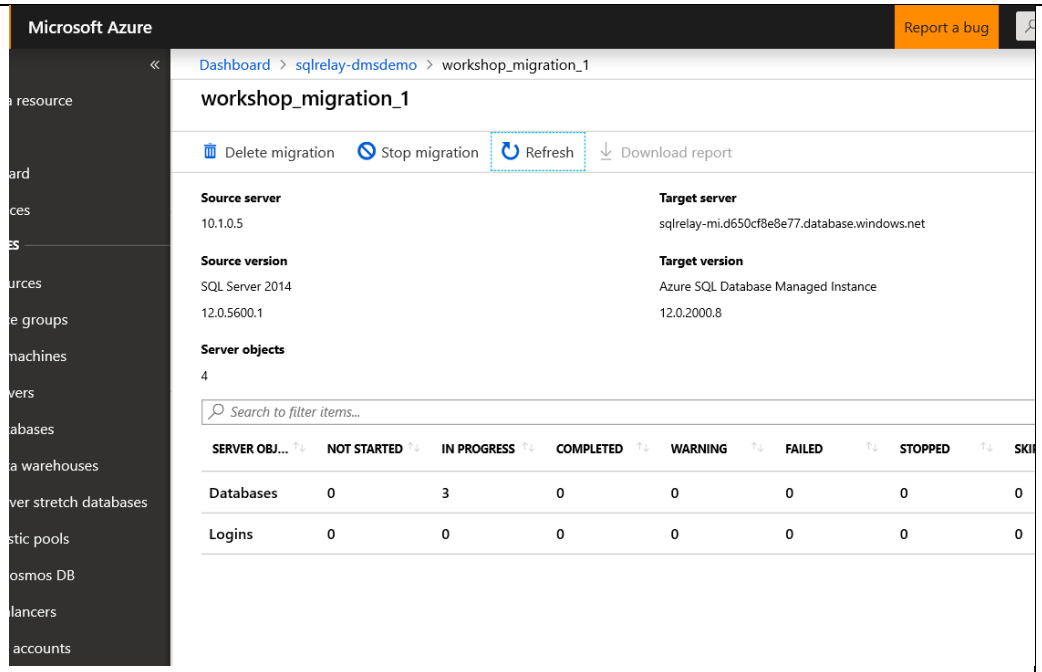
DMS will now run the migration activity.

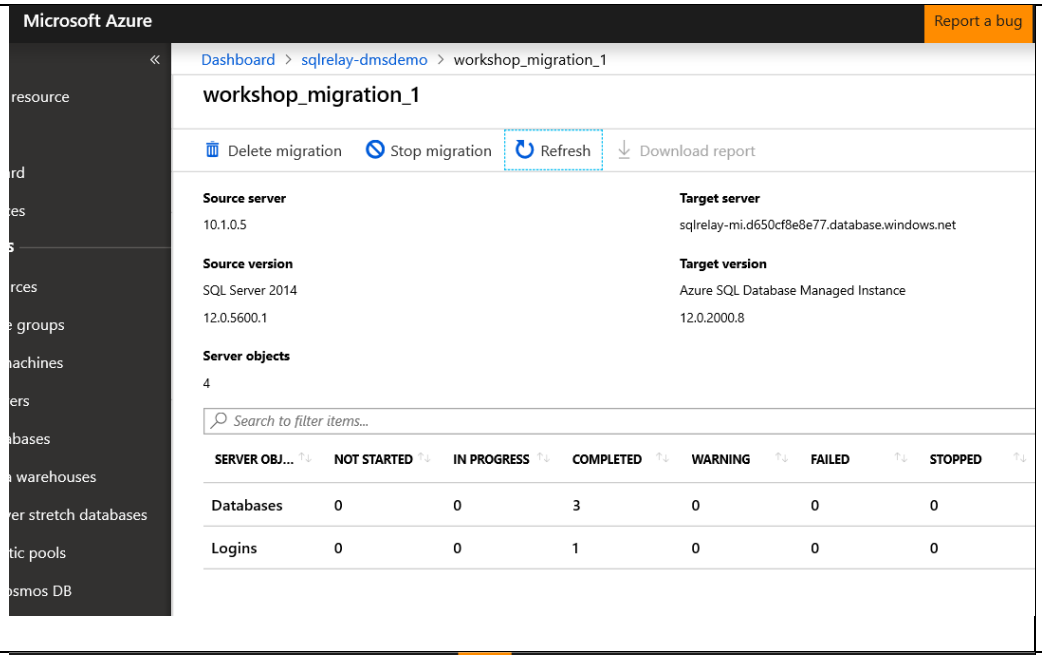
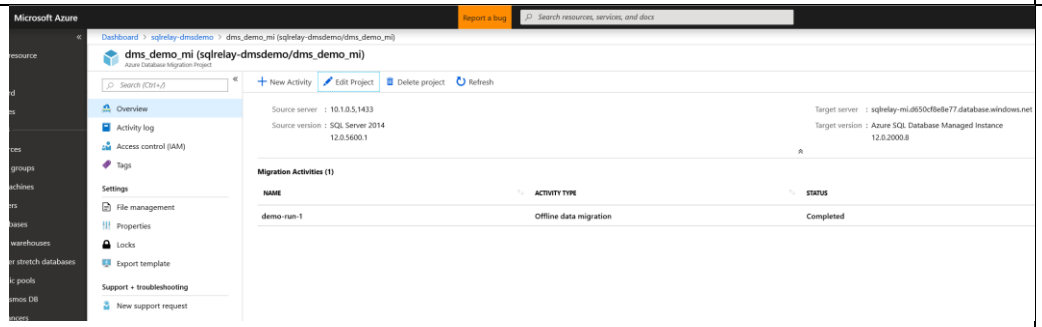
Initially this screen will be displayed.

Select '**Refresh**' to monitor the progress of your migration.

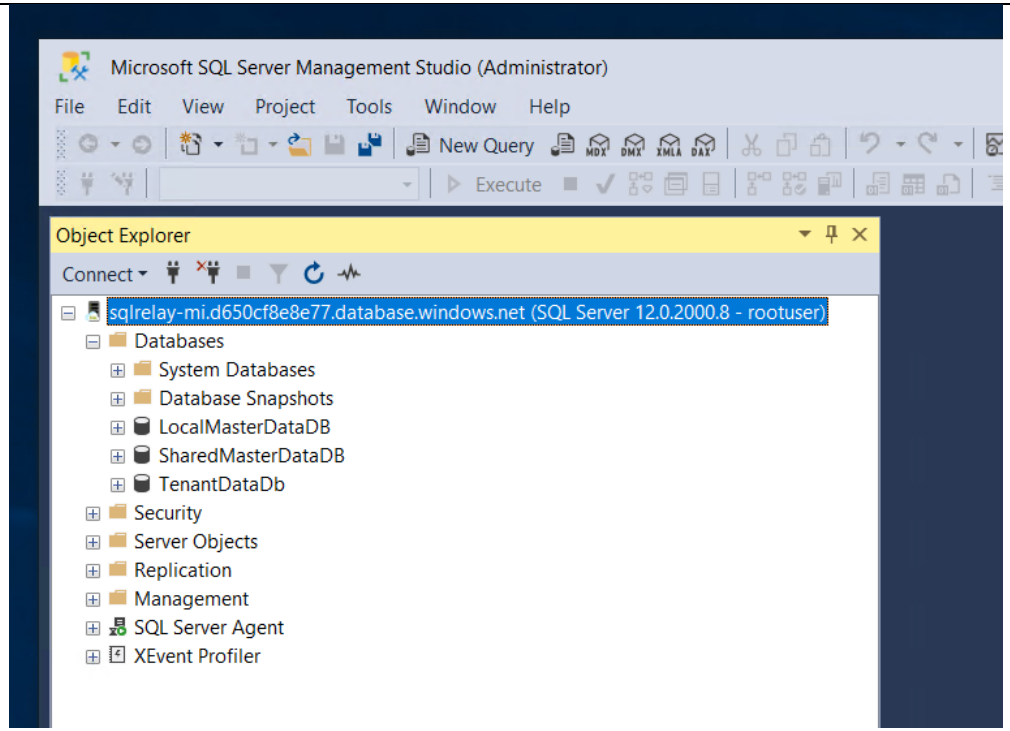
Notice the database counts under the following columns as you keep selecting '**Refresh**':
- "IN PROGRESS"
- "COMPLETED"
- "FAILED"

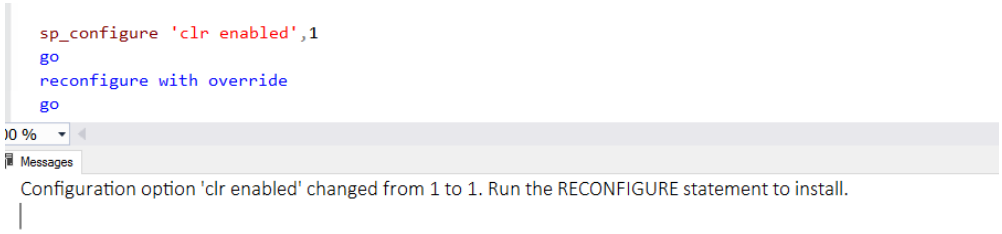



Microsoft Azure
Report a bug
Dashboard > sqlrelay-dmsdemo > workshop_migration_1
workshop_migration_1
Delete migration
Stop migration
Refresh
Download report
Source server
10.1.0.5
Target server
sqlrelay-mi.d650cf8e8e77.database.windows.net
Source version
SQL Server 2014
12.0.5600.1
Target version
Azure SQL Database Managed Instance
12.0.2000.8
Server objects
4
Search to filter items...
SERVER OBJ...
NOT STARTED
IN PROGRESS
COMPLETED
WARNING
FAILED
STOPPED
Databases 0 3 0 0 0 0 0
Logins 0 0 0 0 0 0 0

Under “COMPLETED”, when the number of databases says “3” the migration activity has completed.		If there any warnings, errors or skipped databases they will have a database count under the corresponding headings in the status page. It is assumed, for the workshop, that all three databases have migrated successfully.
Close the migration activity. On the migration project blade notice that your migration activity is displayed.		DMS keeps a history of activity runs for migration projects. A migration activity can be edited and ran again.

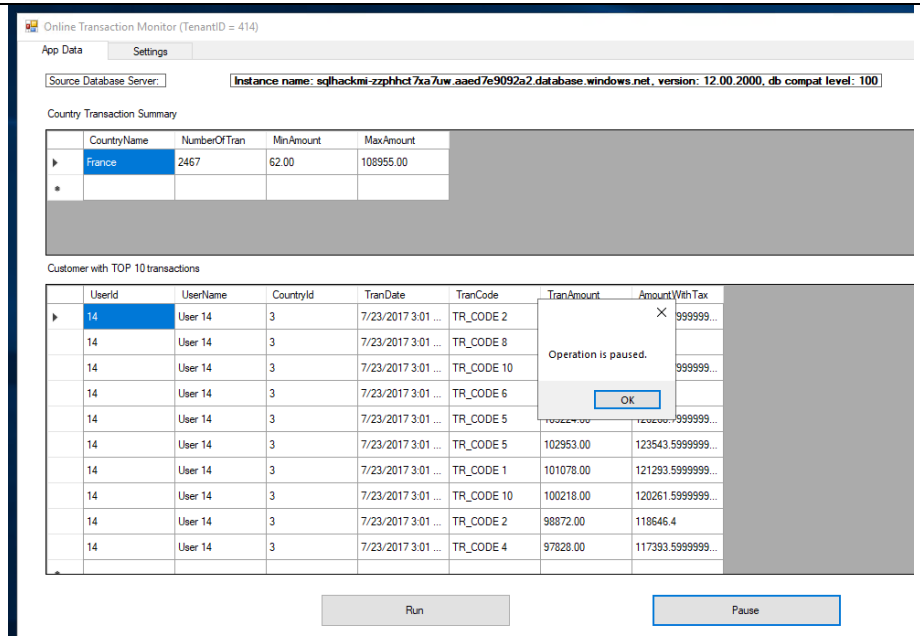
4. Confirm application databases have been migrated & configure

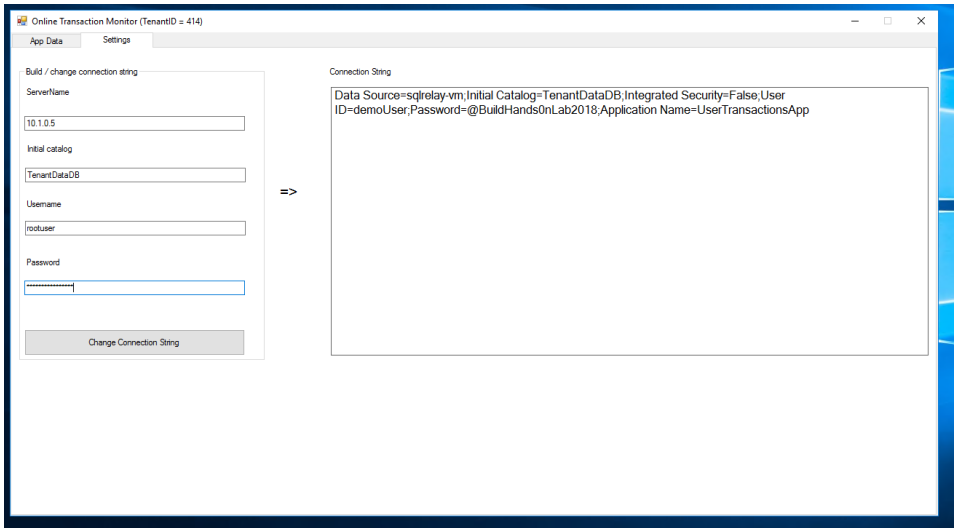
We can also validate the migration by using SSMS. Log into your teams Windows 10 management VM . Open SQL Management Studio and connect to the target Azure SQL Database Managed Instance using these details: **Server:** *(Use the Fully Qualified Domain Name for the SQL Managed Instance)* **SQL Authentication** **Username:** **TEAMXX** **Password:** **TEAMXX** Open the 'Databases' folder and verify the three databases have been migrated and are online. **TEAMXX_LocalMasterDataDb** **TEAMXX_SharedMasterDb** **TEAMXX_TenantDataDb**		SSMS should be installed, for this workshop, on the Windows VM.

Before we can test the application will run with the newly migrated databases we need enable CLR on the SQL Managed Instance. In SQL Server Management Studio open a new query window. *Make sure that you use the sysadmin account (below) for this new query window* **Sysadmin account:** **Username:** **Demouser** **Password:** **Demo@pass1234567** Once the query window has opened enter and run the TSQL statements opposite to enable CLR in the SQL Managed Instance.	``` sp_configure 'clr enabled',1 go reconfigure with override go ```  	Remember that this application must migrate to Azure SQL Database Managed Instance (SQL instance as a service) as it uses CLR. CLR is NOT available on Azure SQL DB (database as a service).

5. Connect 'Online Transaction Monitor' to Azure SQL DB Managed Instance

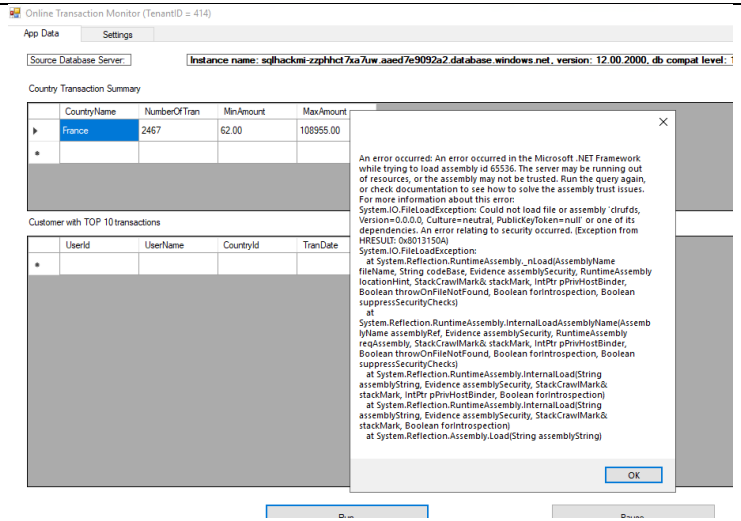
Now that we have migrated the databases to Azure we need to restart the application to use the new database.

Narrative	Screenshot	Notes
On your team Win10 management VM run the SImpleTranReportApp application. Note: You will likely already have the app loaded from the earlier stage in this workshop. IF it is still running simulated transactions, click '**Pause**'		

Reconfigure the applications connection string so it's connects to the newly migrated databases on the SQL Managed Instance. Once running, select the '**Settings**' tab Enter the following parameters into the fields identified: **ServerName:** **Fully qualified Managed Instance Name: SqlhackmiXXXXXXX** **Initial Catalog:** **TEAM01_TenantDataDb** **UserName:** **TEAMXX** **Password:** **TEAMXX** Click '**Change Connection String**' to apply these new settings.		Use the parameters from your "Workshop Sheet – Parameters".

Select the '**App Data**' tab
Click '**Run**'

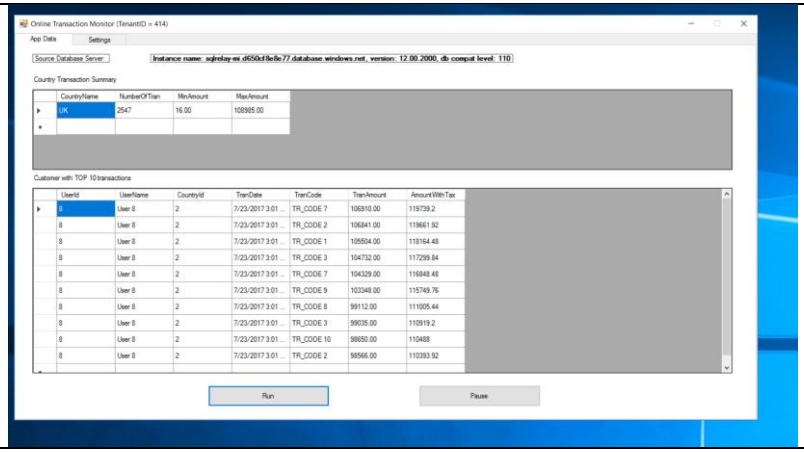
GOTCHA
If you get a long-winded error when you run the application it's because the CLR databases don't have the correct trust settings.
Run the 3 ALTER DATABASE statements below and try starting the application again.




```
-----------------------------------------------------
-- CHANGE BELOW TO YOUR TEAM NUMBER (REPLACE XX)
USE [TEAMXX_TenantDataDb]
GO
-----------------------------------------------------
EXEC dbo.sp_changedbowner 'sa'

alter database [LocalMasterDataDB] set trustworthy on
go
alter database [SharedMasterDataDB] set trustworthy on
go
alter database [TenantDataDb] set trustworthy on
go
```

The application will generate simulated transactional data. Notice how the 'Source Database Server' connection reflects the SQL Managed Instance proving that the database migration has been completed successfully.

Online Transaction Monitor (TenantID = 414)
App Data
Settings
Source Database Server:
Instance name: sqlrelay-mi.d650cf8e8e77.database.windows.net, version: 12.00.2000, db compat level: 110
Country Transaction Summary
CountryName
NumberOfTran
MinAmount
MaxAmount
UK
2547
16.00
108905.00
Customer with TOP 10 transactions
UserId
UserName
CountryId
TranDate
TranCode
TranAmount
AmountWithTax
Run
Pause