SQL Modernisation Open Hack

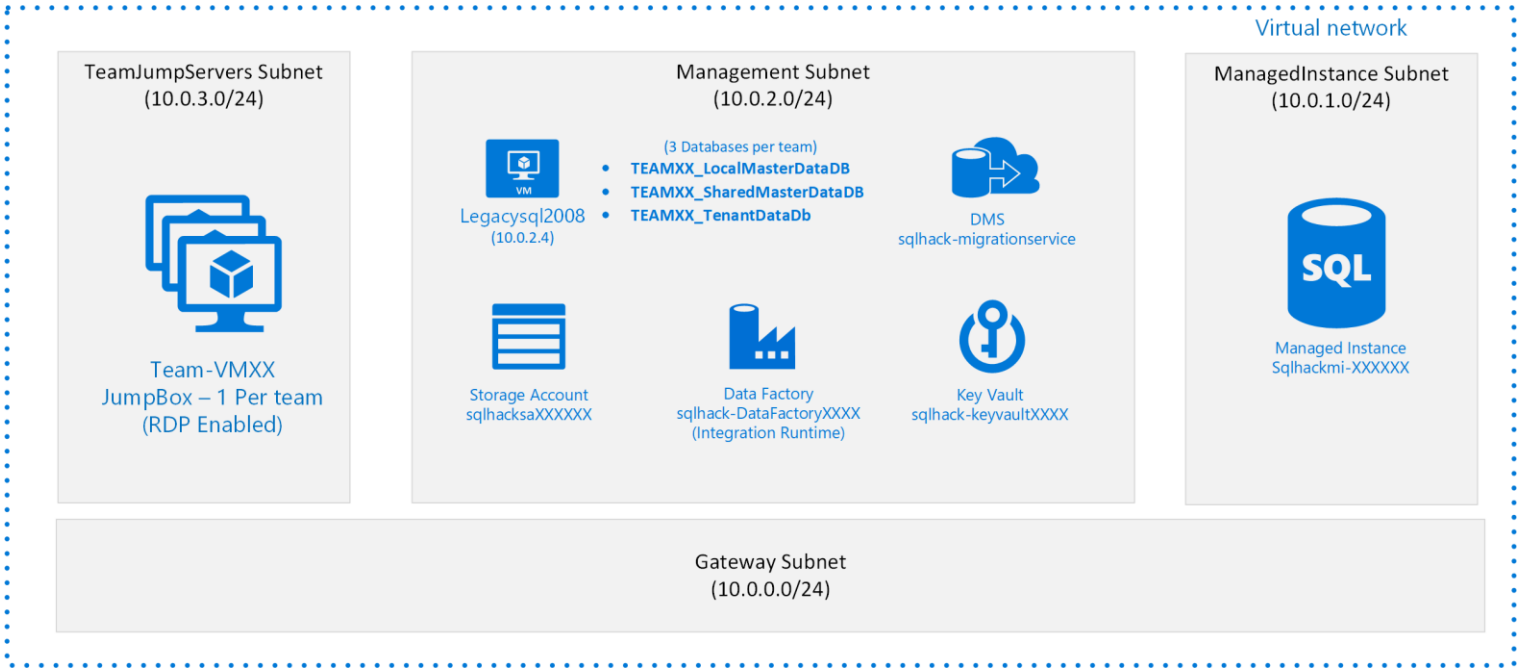
SQL Moderation Hack Database Migration Lab Step-by-step

V2.4

Contents

Microsoft

Migration architecture and Azure components



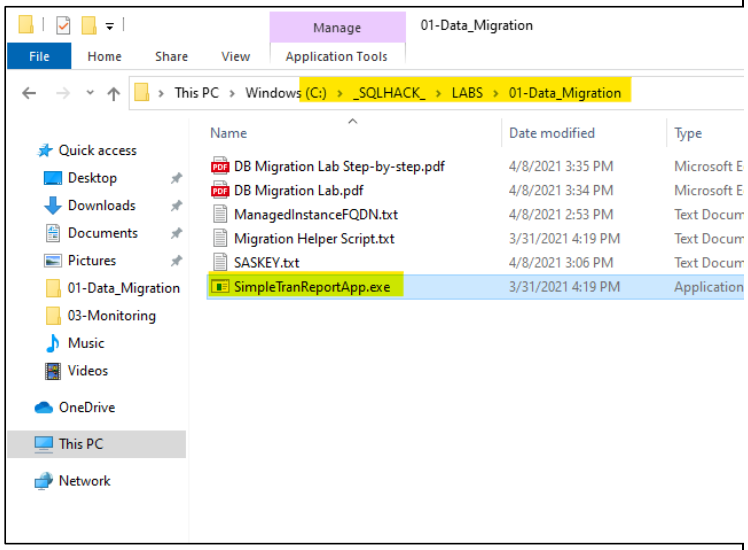
SQLHACK-SHARED-VNET Single Virtual Network containing all workshop resources		
TeamJumpServers Subnet Each team is assigned a Win10 VM that mimics their company desktop	**Management Subnet** Several machines and services are already deployed within a dedicated subnet within the Virtual Network	**ManagedInstance Subnet** The Azure SQL Managed Instance has been deployed into a dedicated Subnet

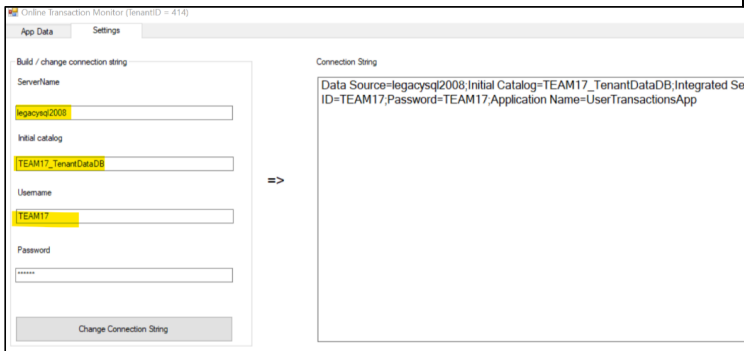
Generic Migration Content

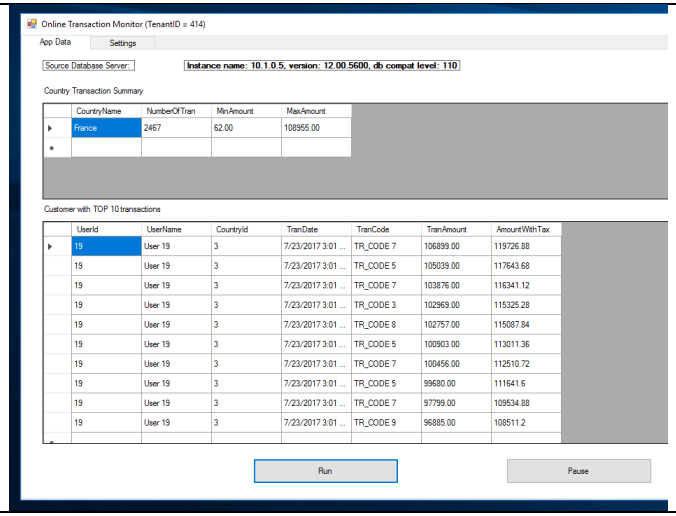
Narrative	Notes
Notes for outside of the workshop: *Familiarise yourself with Microsoft migration tools and the Azure Database Migration Guide*	Azure Database Migration Guide: https://www.microsoft.com/en-us/download/default.aspx DMA & download link: https://docs.microsoft.com/en-us/sql/dma/dma-overview?view=sql-server-2017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

1. Investigate the 'Online Transaction Monitor' legacy application

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

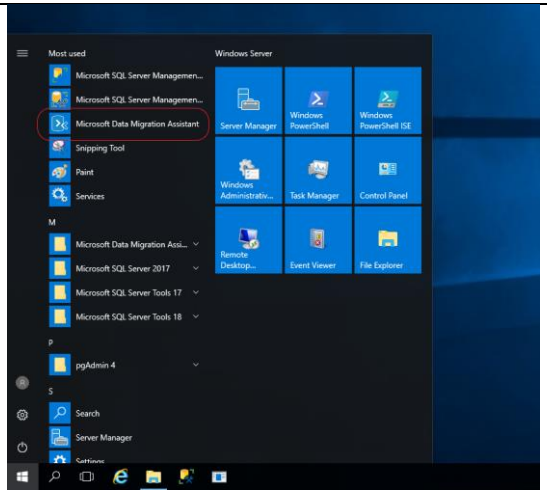
Narrative	Screenshot	Notes
We will set the sample application running to demonstrate how Azure Database Migration Services can be used to perform a migration of a database. RDP onto the Win10 management VM using the **details from the "DB Migration Lab and Parameters.pdf"** Run the 'SimpleTranReportApp' app which an be found in: **C:_SQLHACK_\LABS\01-Data_Migration** Or In Windows Explorer search for '**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.

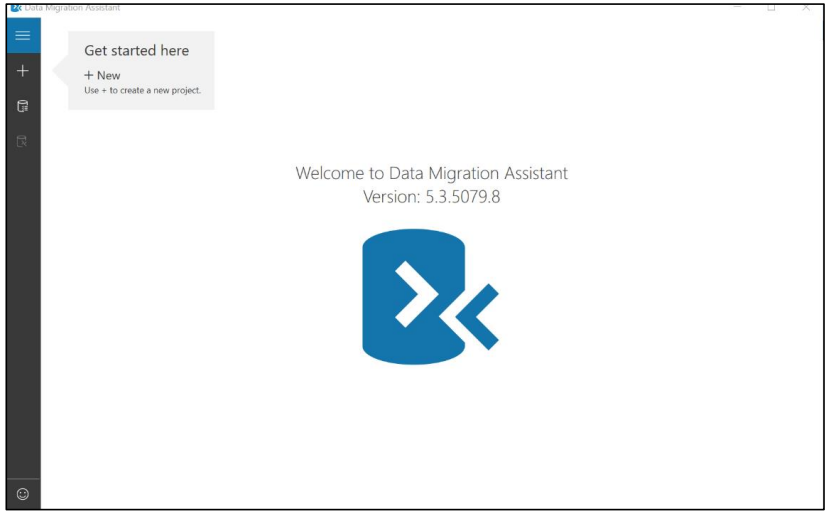
Once running, select the '**Settings**' tab and enter the following parameters into the fields identified: **ServerName:** **LEGACYSQL2008** **Initial Catalog:** **TEAMxx_TenantDataDb** **Username:** **TEAMxx** **Password:** **TEAMxx** Click the "**Change Connection String**" button to apply the connection 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.

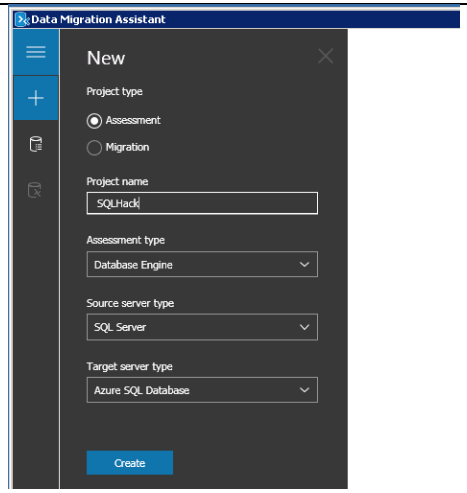
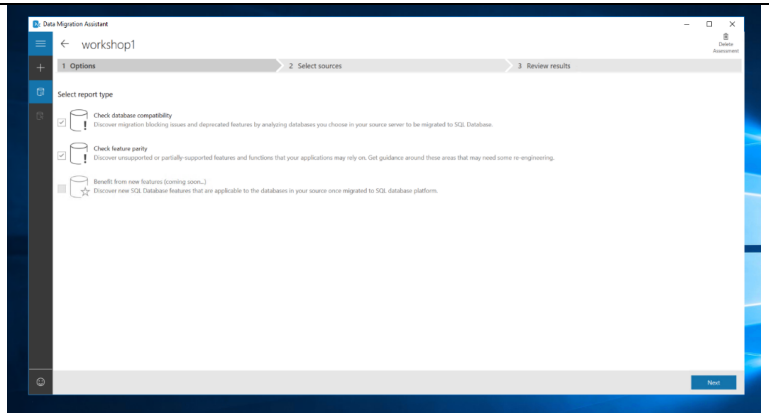
Select '**App Data**' tab and click the "**Run**" button. After a few seconds transaction will start to appear in the application.		The application will generate simulated transactional data. Notice how the 'Source Database Server' information at the top of the app reflects the parameters given in the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

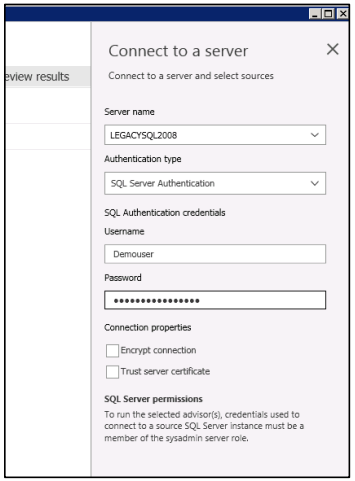
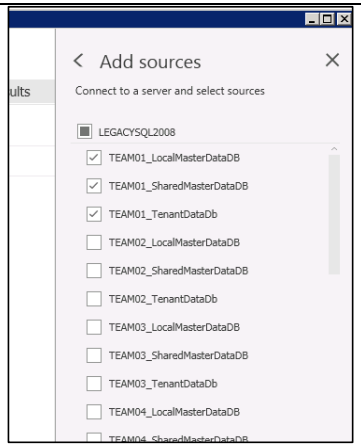
2. Assess the application databases for Azure SQL Database suitability using the Database Migration Assistant (DMA)

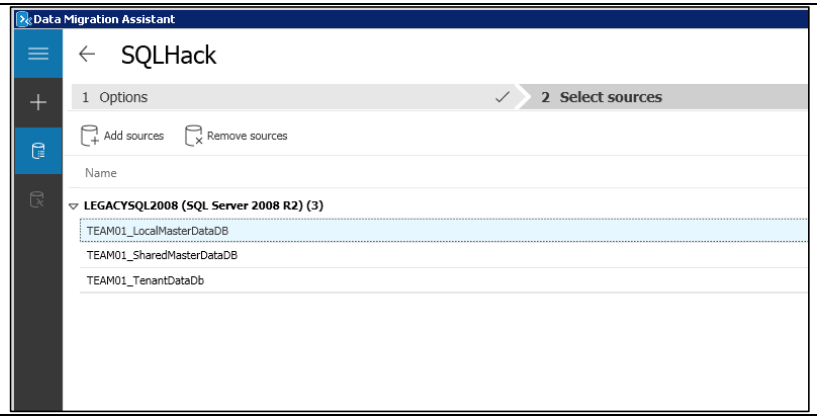
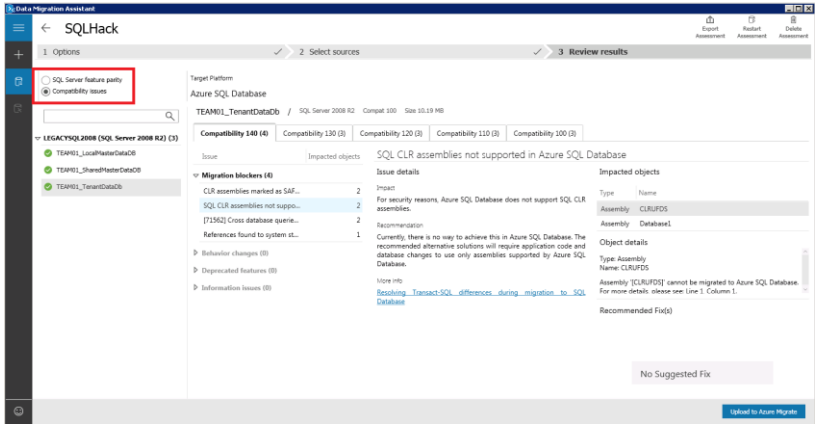
In this section we will use the Data Migration Assistant (DMA) to assess the applications database for suitability for migration to Azure Cloud.

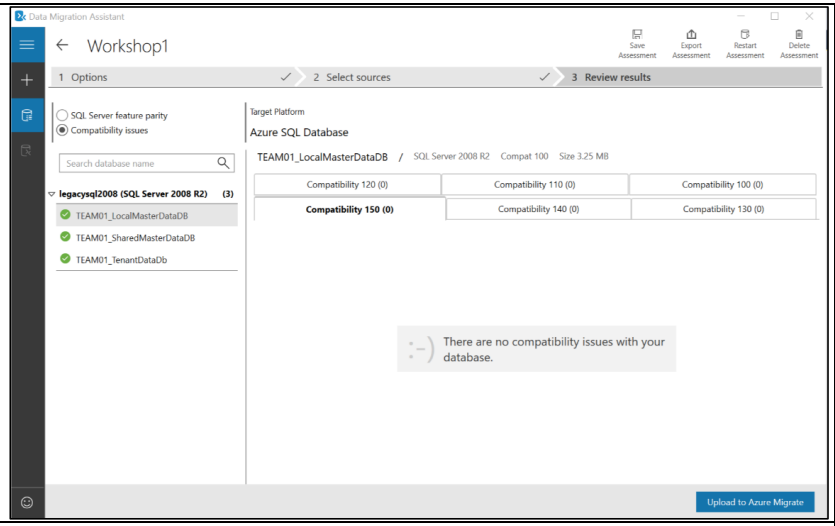
Narrative	Screenshot	Notes
We need to determine the suitability of the database(s) for migration to Azure. This includes checking for compatibility and feature support with Azure Database. You should already have an RDP session open to your teams Win10 Management VM, if so run DMA from the Start menus or Desktop icon.		Database Migration Assistant (DMA) is a free download from Microsoft. It can be used to assess a number of database migration & upgrade scenarios not just SQL Server to Azure SQL Database.

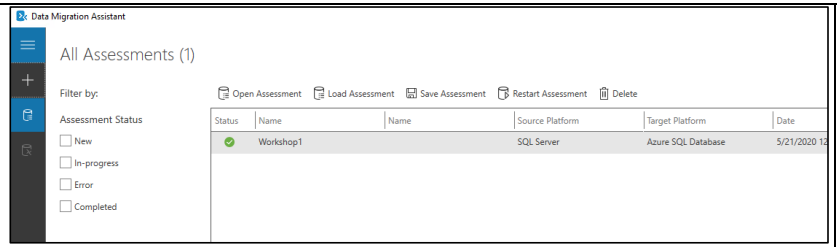
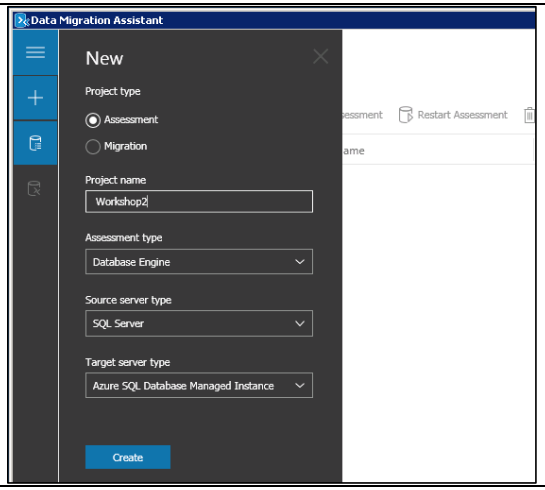
You should see this screenshot to the right. Select the **"+"** to create a new **assessment** project		

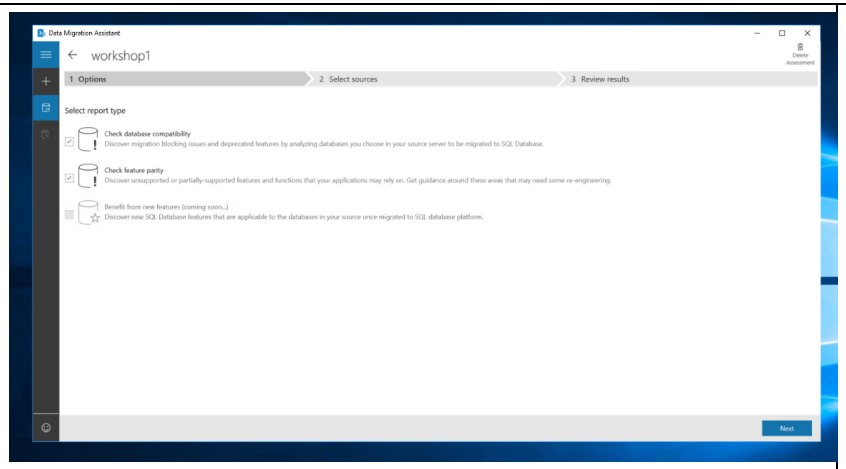
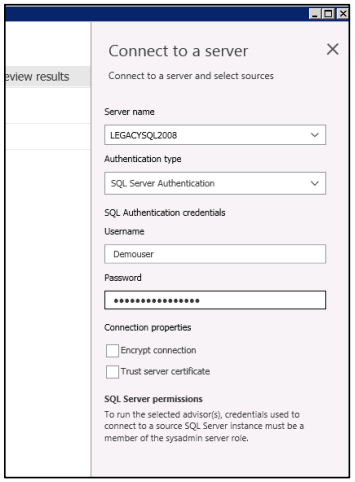
Select/Enter the following details: **Project name:** **Workshop1** **Assessment type:** **Database Engine** **Source server type:** **SQL Server** **Target server type:** **Azure SQL Database** Click '**Create**'		Our first project assessment assumes we will be migrating to Azure SQL DB, so the options 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.

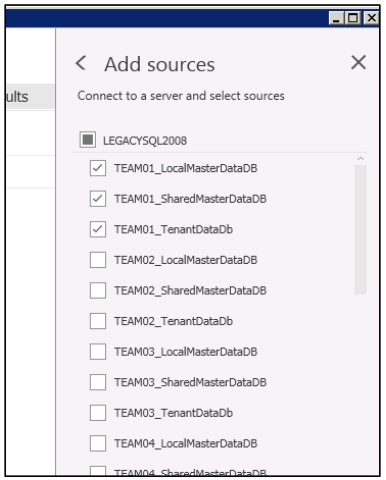
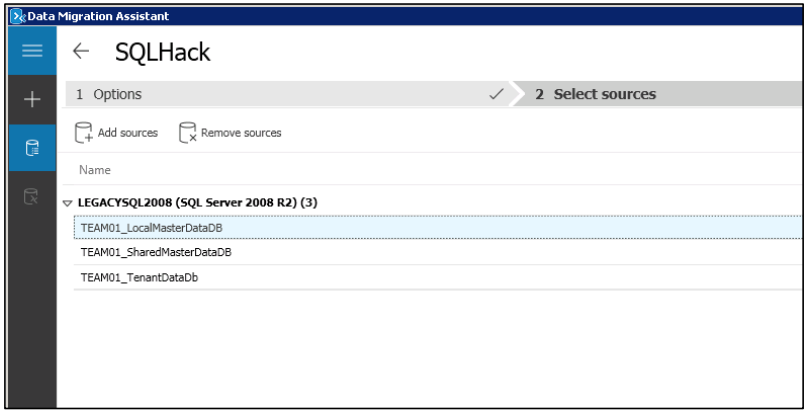
Enter the source/legacy SQL details: **Server Name:** **LEGACYSQL2008** **Authentication Type:** **SQL Server Authentication** **Username:** **Demouser** **Password:** **Demo@pass1234567** **Untick "Encrypt connection"** Click '**Connect**' ***If you get an error logging in check that the Win10 keyboard is set to UK English not US***		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 databases 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 at 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 provides some high-level metadata about the databases including their compatibility level the total size of each database.
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 settings or components (e.g. MSDTC) that the source DBs are using that isn't supported on the target – in this case Azure SQL Database. In our assessment there is one 'Unsupported feature' reported (cross database queries). '**Compatibility Issues**' which is a database level report detailing individual objects that have compatibility issues.		**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 the 'Details' and 'Databases'sections on the right 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

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).		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've reviewed the assessment click the back arrow to see a list of current DMA projects.		
	Because we need to migrate CLR Stored Procs, we need to repeat the assessment with Azure SQL DB Managed Instance as the target to see if it's compatible	
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 options shown in the screenshot need to be selected.

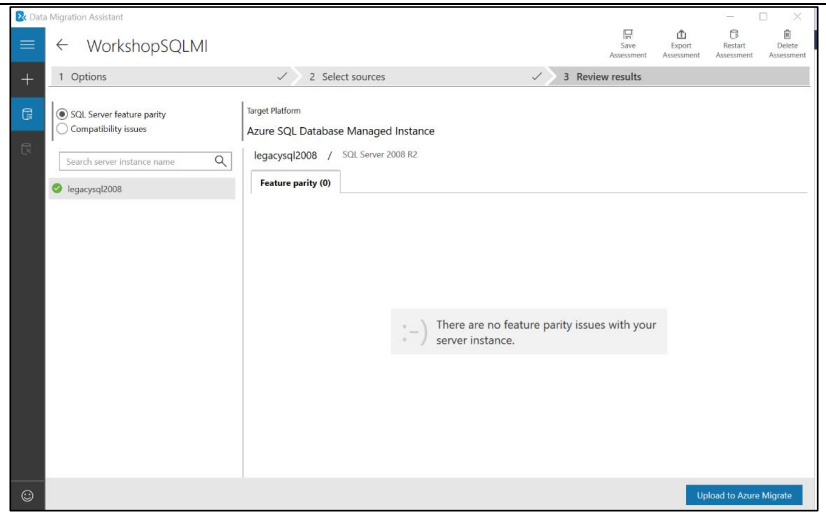
Select the assessment checks (Report Type) to be made: **Check database compatibility** **Check feature parity** Click '**Next**'		As we saw previously DMA can test for both database compatibility and feature parity compliance against the chosen target. As before we will assess all the databases against 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.

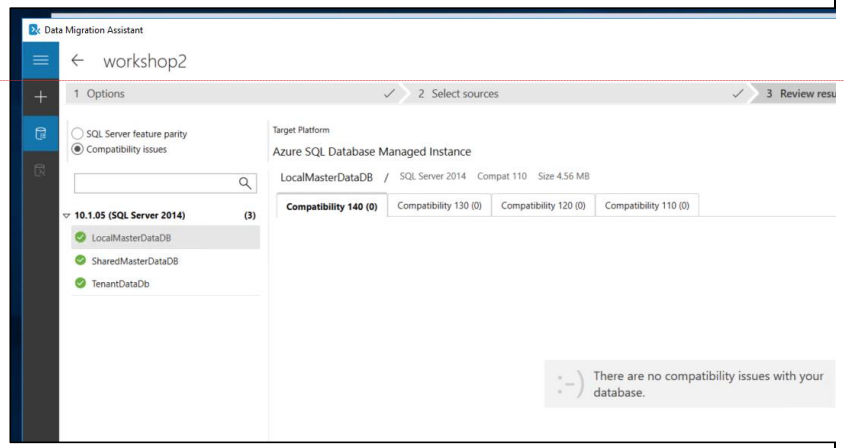
Select **only** the 3 database used by your 'Online Transaction Monitor' app. These will have a TEAMXX prefix where XX should be replaced by your team number. **TEAMxx_LocalMasterDataDb** **TEAMxx_SharedMasterDb** **TEAMxx_TenantDataDb** Click '**Add**' to add them to the assessment.		DMA will show all 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 separate 2 reports.

Note the **'SQL Server feature parity'** report will either be clean or it will show a single PowerShell 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.

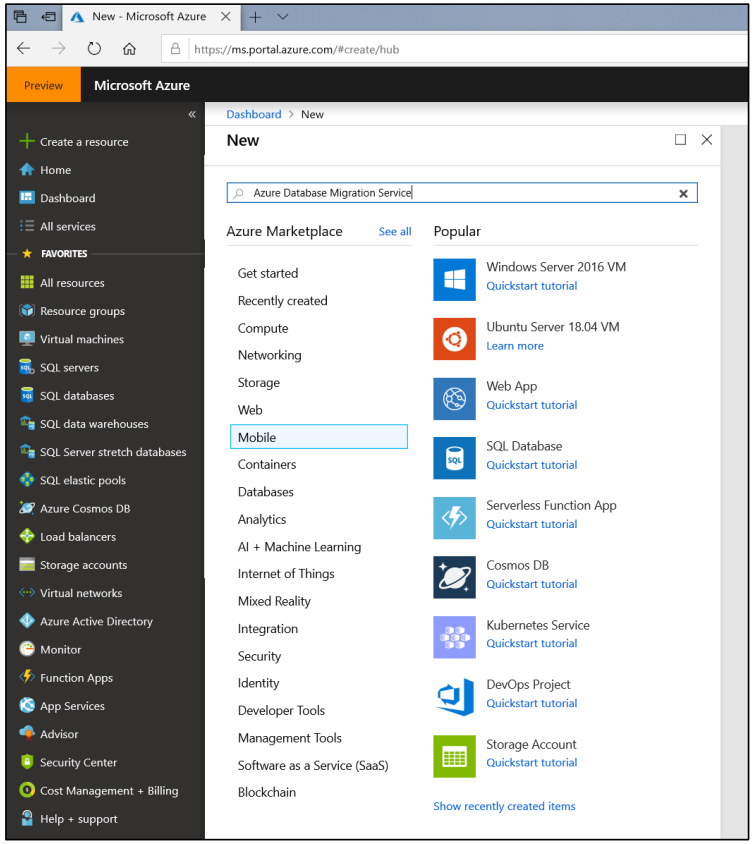
'Compatibility Issues' shows, over the compatibility tabs, issues that need to be addressed to permit the database(s) to run, in the chosen compatibility level (e.g. 140, 130, 120, 110).

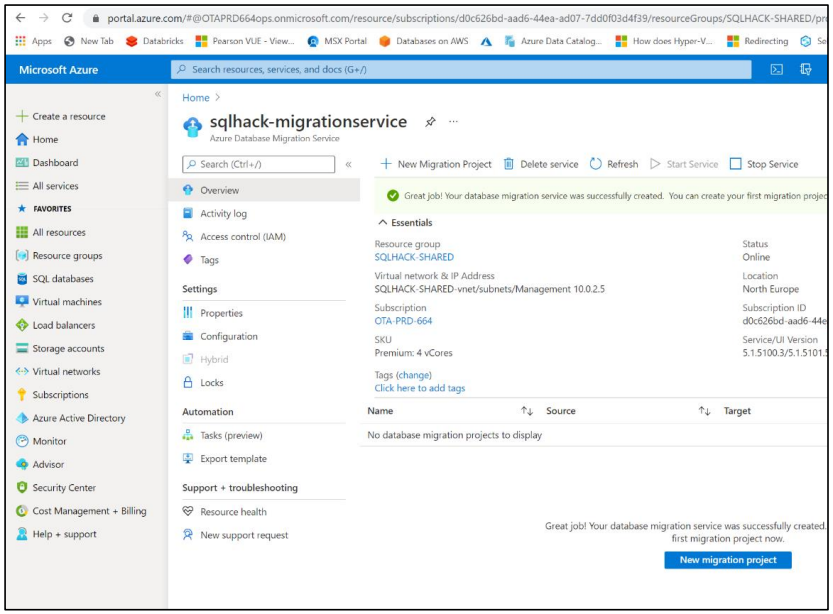
If you have multiple databases, as with the example screenshot, you need to highlight EACH database to see the compatibility issues.

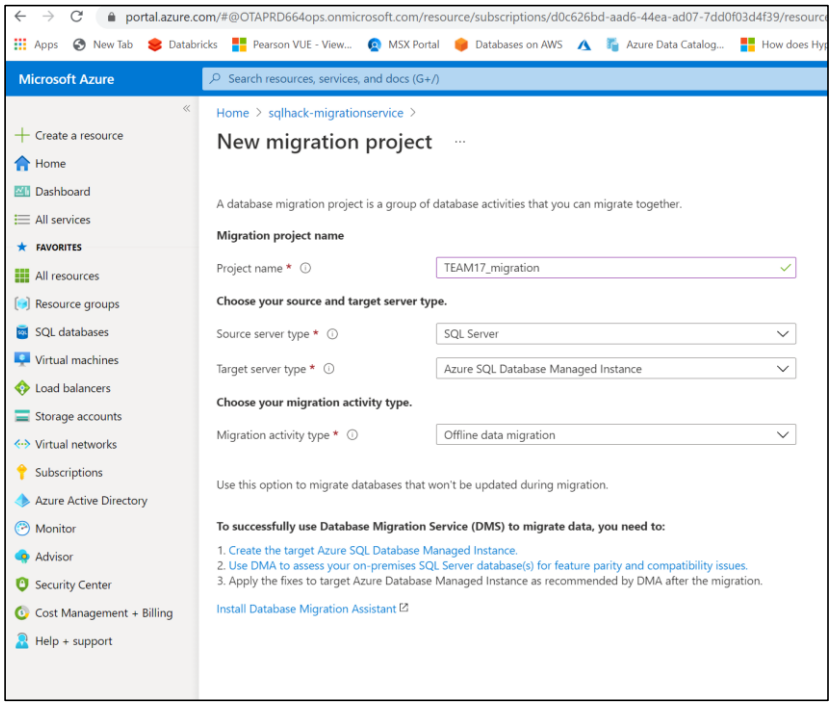
Commented [SM(C1]: Issue about SAFE CLRs not being migrated...

	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 SQL2008 databases to Azure. For the workshop DMS will already been provisioned as it can take 20-30mins to be deployed. If you were doing this yourself you would need to provision DMS before you could begin the migration process and would need to follow the DMS 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 Azure SQL Managed Instance. DMS can host and run different types of database migration Projects under the same instance 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: On your Win10 VM open Edge browser and got to: **HTTPS://portal.azure.com** **Username and Password:** *(see C:_SQLHACK_\Azure Portal Domain Accounts.pdf)* In the Azure portal, open the **SQLHACK-SHARED 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 '**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

The source is the legacy SQL VM host:

Source SQL Server:
LEGACYSQL2008

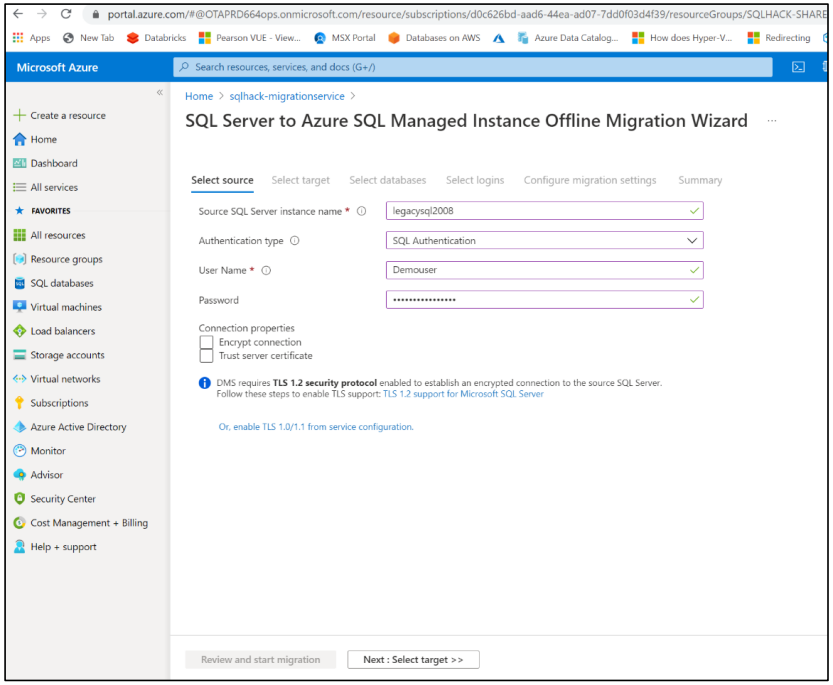
Authentication Type:
SQL Authentication

User Name:
Demouser

Password
Demo@pass1234567

Click '**Next: Select target >>**'

DMS will now perform a connection test to source environment.



The account that DMS uses to connect to the source instance must be a member of sysadmin.

STEP 2: Select Target

Target SQL Server:
(see C:_SQLHACK_\LABS\01-Data_Migration\ManagedInstanceFQDN.txt for the SQL Managed Instance name)

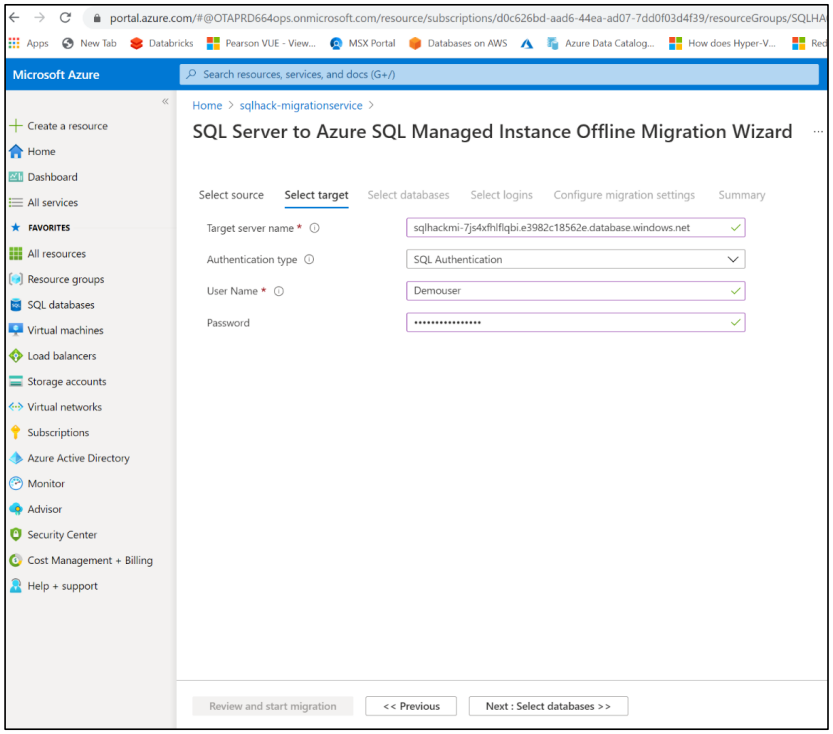
Authentication Type:
SQL Authentication

User Name:
Demouser

Password:
Demo@pass1234567

Click '**Next:Select databases>>**'

DMS will now perform a connection test to target environment.



STEP 3: Select Databases

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

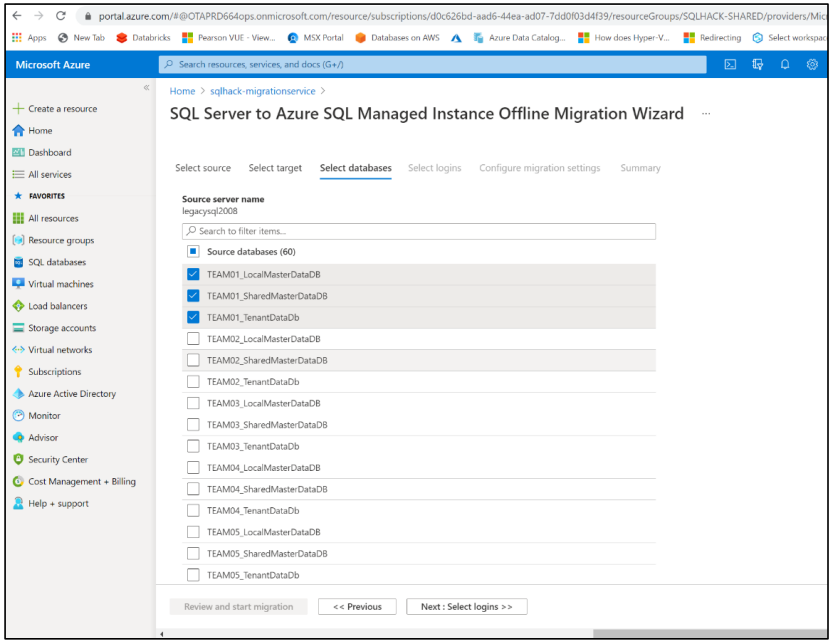
TEAMxx_LocalMasterDataDb

TEAMxx_SharedMasterDb

TEAMxx_TenantDataDb

(replace XX with your team number)

Select '**Next: Select logins>>**

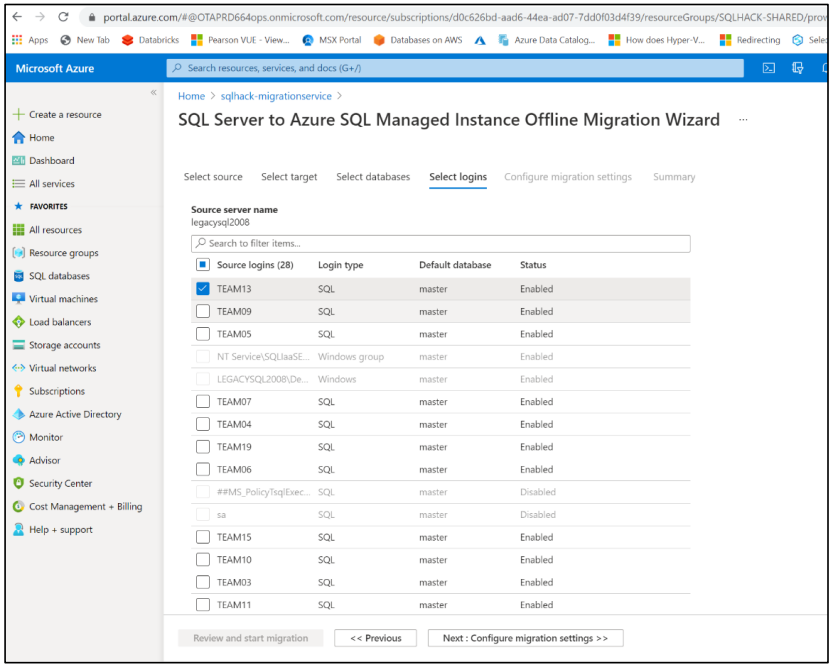


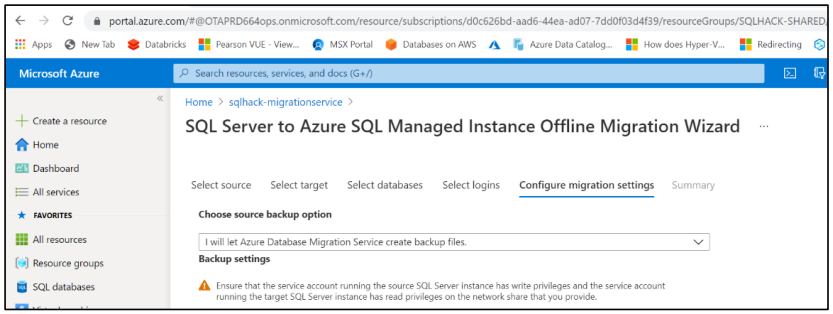
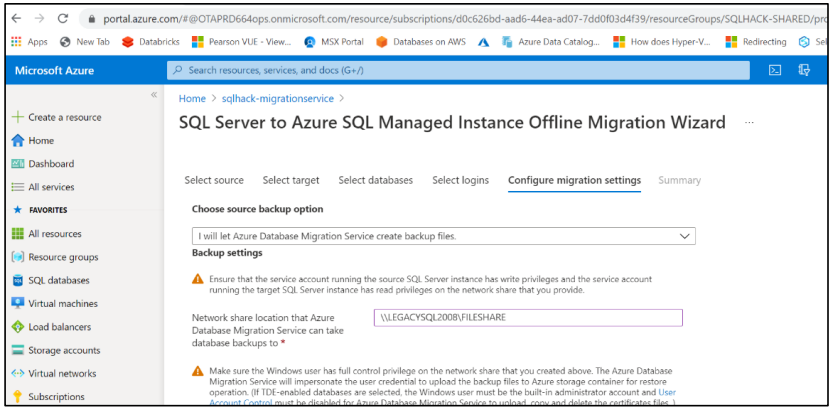
STEP 4: Select Logins

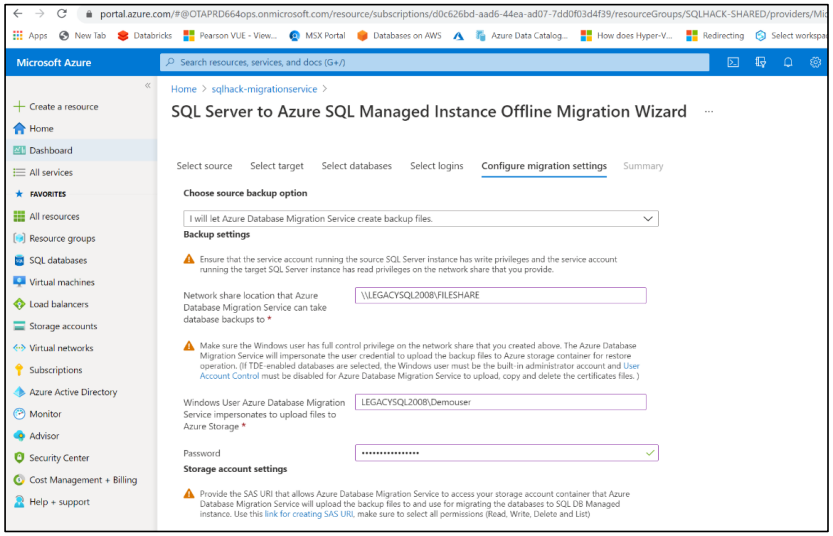
As with a traditional on-premise migrations the SQL Server level logins must be migrated alongside the database. Select the database logins, from the list, that are required for the application.

Select ***only*** your '**TEAMxx**' login.

Select
Next: Configure migration settings>>



Step 5: Configure migration Settings ***(Choose source backup option)*** We are running an offline migration which will crate and use backups of the DBs that are being migrated. We want DMS to perform the backups, so select this option from the "Choose source backup option" (as shown).		DMS can use backup files provided or take the backups as part of the migration activity.
Step 5: Configure migration Settings ***(Backup settings)*** We can now enter the Windows share that the source server will write the database backups to. **Network Share:** **\\LEGACYSQL2008\FILESHARE**		

Next 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. **Windows User Azure Database Migration Service impersonates to upload files to Azure Storage:** **LEGACYSQL2008\Demouser** **Password:** **Demo@pass1234567**		

Step 5: Configure migration Settings
(Storage account settings)

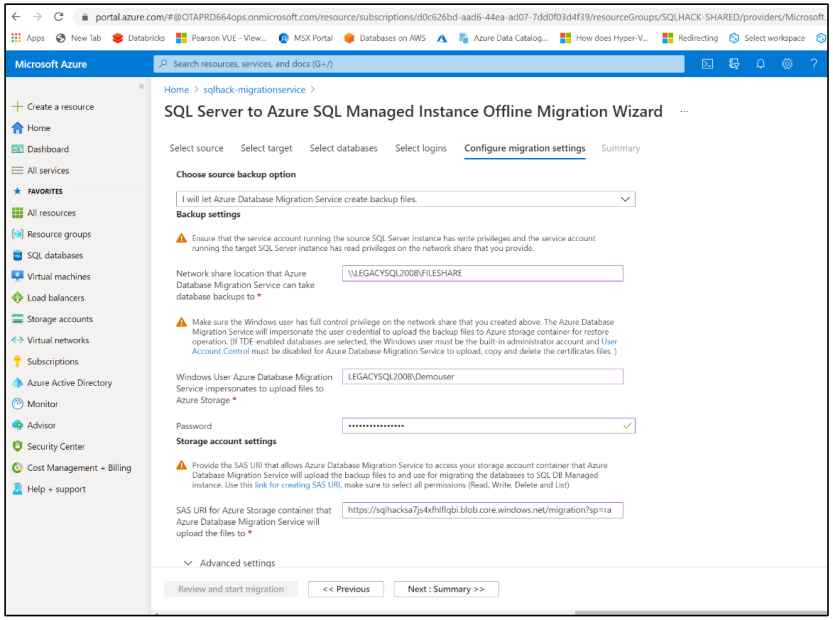
DMS is an Azure Service. We have to provide the Shared Access Signature token (or "SAS URI" for short) to permit DMS to upload the backup files from the share on the LEGACYSQL2008 host to Azure blob storage where the SQL Managed Instance can access them during the restore process.

The SAS URI is both the URL of a container (folder) in Azure Blob Storage and the key to access it.

The SAS URI can be found in: **C:_SQLHACK_\LABS\01-Data Migration\SASKey.txt**

Enter the SAS URI key and click '**Next : Summary>>**.

*This will perform a connection test and if successful will display the **Summary** screen.*



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 5: Configure migration Settings
(Summary)

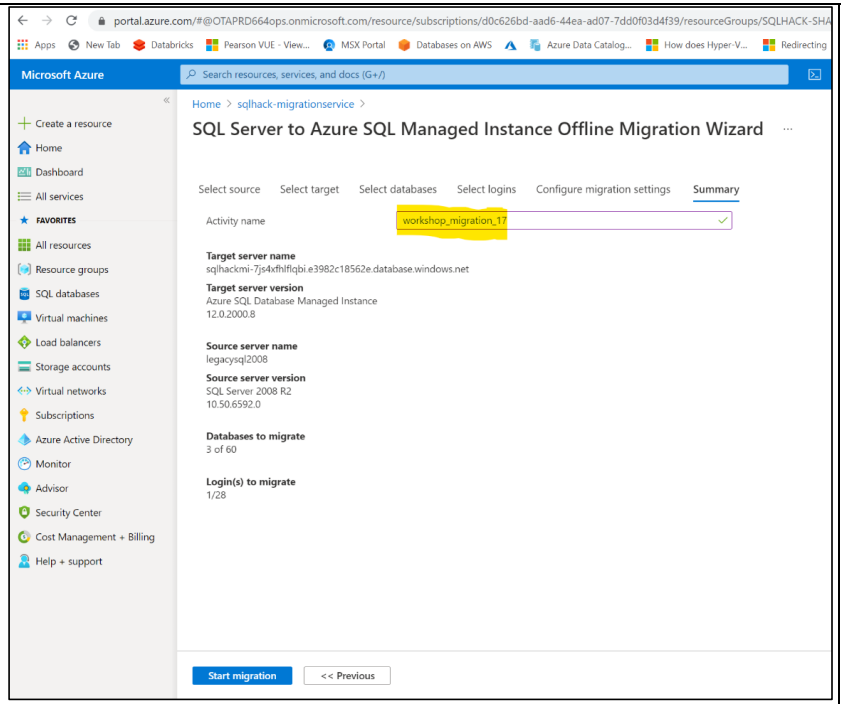
DMS displays a summary of the migration settings.

Now we need to use these settings to actually perform a migration. To do this we create and run an "Activity".

On the **Summary** settings enter an activity name :

Activity Name
TEAMxx_migration_activity
(replace XX with your team number)

Click '**Start migration**'

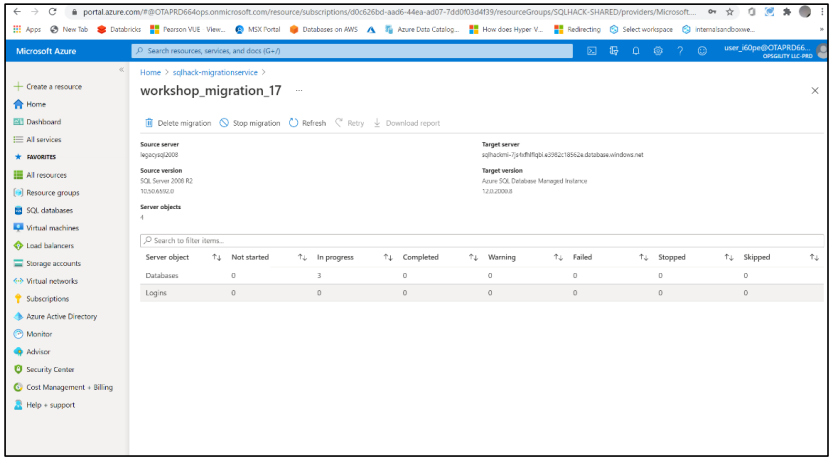


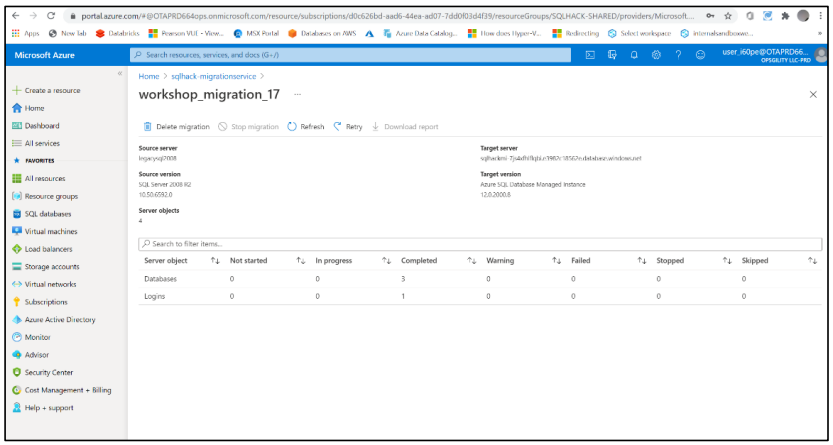
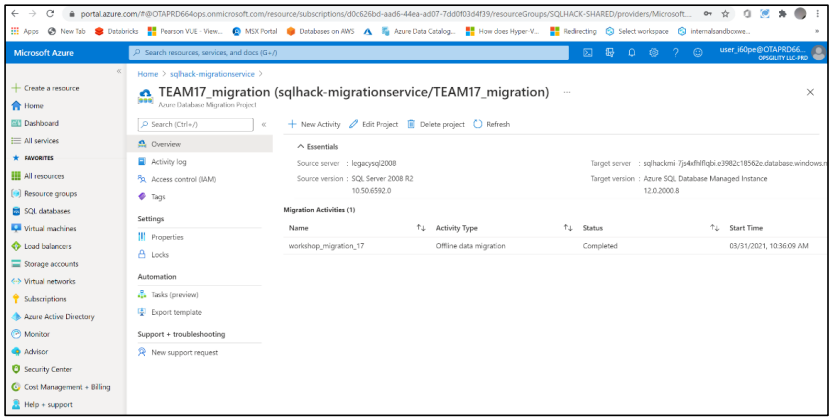
Validate my databases option

Selecting this validation option forces DMS to do the following tasks:

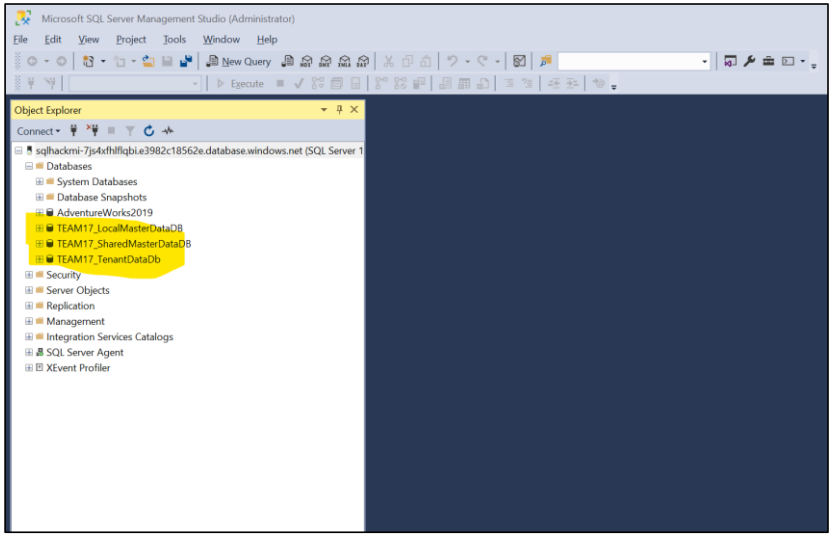
1. Takes top 100 resource intense queries and re-runs them against the target and reports the success/failure rate
2. Table checksum on all rows and report any differences

This can be an intensive process so best test it with a non-production migration to see how much extra time it adds to the migration.

DMS will now run the migration activity. Initially this screen will be displayed. Click '**Refresh**' to monitor the progress of your migration. Notice the database counts under the following columns as you keep pressing '**Refresh**': "IN PROGRESS" "COMPLETED" "FAILED"		During the migration process you can monitor the creation of the 3 backup files by opening the fileshare \\LEGACYSQL2008\FILESHARE Also note that DMS cleans-up after itself & deletes the backups from the fileshare once they have been copied up the Storage Account.

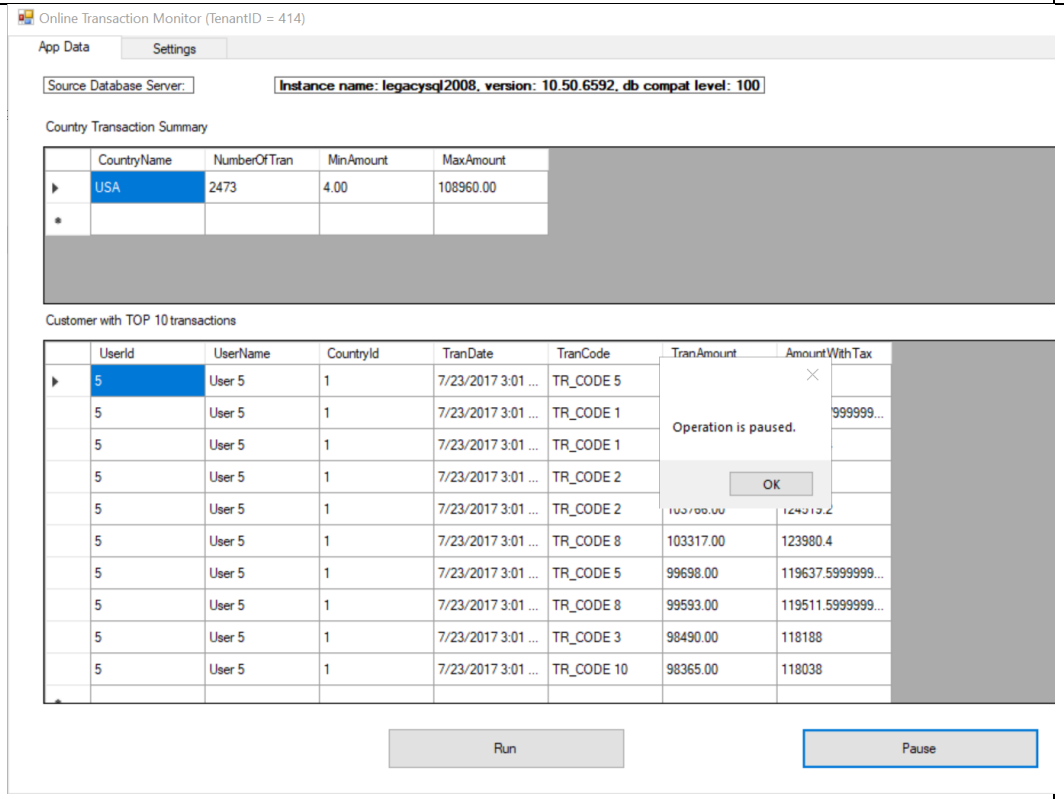
Under “COMPLETED”, when the number of databases says “3” the migration activity has completed.		If there any warnings, errors or skipped databases they will have a database count under the corresponding headings in the status page. It is assumed, for the workshop, that all three databases have migrated successfully.
Close the migration activity. On the migration project screen 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.

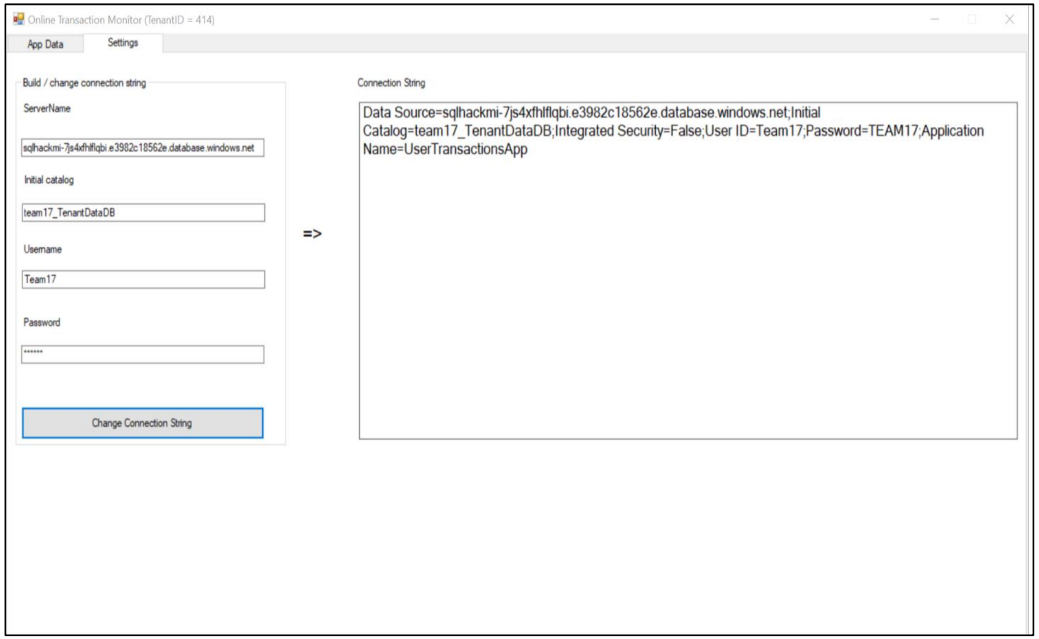
4. Confirm application databases have been migrated to Azure SQL Managed Instance

On your Win10 VM open SQL Management Studio and connect to the target Azure SQL Database Managed Instance using these details: **Server:** *(see C:_SQLHACK_\LABS\01-Data_Migration\ManagedInstanceFQDN.txt)* **SQL Authentication** **Username:** **Demouser** **Password:** **Demo@pass1234567** Open the 'Databases' folder and verify the three databases have been migrated and are online. **TEAMxx_LocalMasterDataDb** **TEAMxx_SharedMasterDb** **TEAMxx_TenantDataDb**		

5. Connect 'Online Transaction Monitor' App 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:** *(see* **C:_SQLHACK_\LABS\01-Data_Migration\ManagedInstanceFQDN.txt***)* **Initial Catalog:** **TEAMxx_TenantDataDb** **UserName:** **TEAMxx** **Password:** **TEAMxx** Click '**Change Connection String**' to apply these new settings.		

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

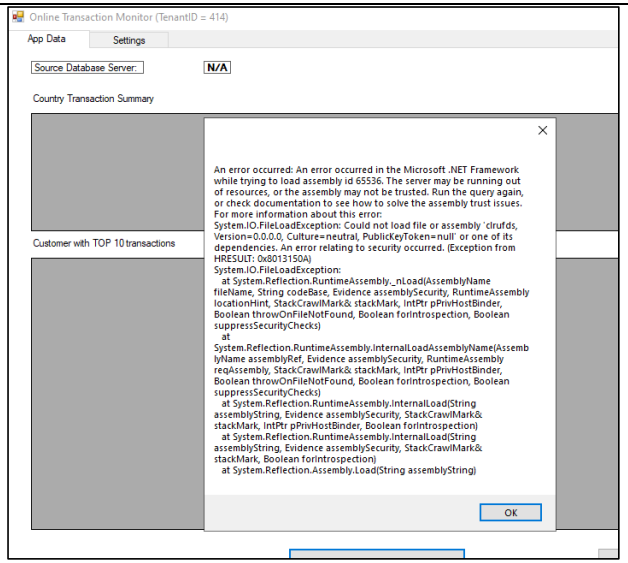
GOTCHA
If you get a long-winded error when you run the application it's because the CLR assemblies don't have the correct trust settings in the migrated DBs.

Run the 3 ALTER DATABASE statements below and try starting the application again.

You can also find this statements already prepared in:

C:_SQLHACK_\LABS
01-Data_Migration
Migration Helper Script.txt

Now go back to the application and try running it again. After a few seconds you should see transactions start to appear.




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

alter database [TEAMXX_LocalMasterDataDB] set trustworthy on
go
alter database [TEAMXX_SharedMasterDataDB] set trustworthy on
go
alter database [TEAMXX_TenantDataDb] set trustworthy on
go

EXEC sp_configure 'clr enabled', 1;
RECONFIGURE;
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.

Notice that the “Source Database Server” displayed at the top of the application shows the SQL Managed Instance FQDN.