

SQL Server Modernisation Hack Lab Environment BUILD & RESET GUIDE

V3.1

Contents

[Revision History 2](#_Toc162384621)

[Pre-Requisites 3](#_Toc162384622)

[READ ME FIRST! Environment Setup - Update March 2024 4](#_Toc162384623)

[Hack Environment Architecture and Azure Components 5](#_Toc162384624)

[Environment Setup Overview – For Information Only 6](#_Toc162384625)

[Appendix 1 – Automated Environment Build Guide 7](#_Toc162384626)

[PREREQUISITE: Lower PowerShell’s Security Policy 7](#_Toc162384627)

[Download & run the PowerShell script & ARM templates 7](#_Toc162384628)

[Monitoring the ARM template deployments 12](#_Toc162384629)

[Confirming the environment automated build 13](#_Toc162384630)

[Confirming the database & software automated build 13](#_Toc162384631)

[Appendix 2 – Post Environment Build Tasks 18](#_Toc162384632)

[Appendix 2.1 - Post Environment Build Tasks: Database Migration Environment 19](#_Toc162384633)

[2.1.1 LEGACYSQL2012 Set CLR Permissions 21](#_Toc162384634)

[2.1.2 Managed Instance Enable CLR TSQL 25](#_Toc162384635)

[2.1.3 Managed Instance Configuration Script 25](#_Toc162384636)

[Appendix 2.2 – Post Environment Build: SSIS Migration Setup Tasks 26](#_Toc162384637)

[Appendix 2.2.1 - SSIS Migration Lab: Automated Build 26](#_Toc162384638)

[Appendix 2.2.2 - SSIS Migration Lab: Manual Build 27](#_Toc162384639)

[Appendix 2.3 – Security Lab Build 33](#_Toc162384640)

[Appendix 2.3.1 - Auditing 33](#_Toc162384641)

[Appendix 2.3.2 – Azure Defender for SQL 34](#_Toc162384642)

[Appendix 2.4 – Monitoring Lab Build 35](#_Toc162384643)

[Appendix 2.3.1 – Deploy TenantCRM database 35](#_Toc162384644)

[Appendix 2.3.2 – Run simulated workload against the TenantCRM database 35](#_Toc162384645)

[Appendix 3 – Daily Environment Reset Tasks 36](#_Toc162384646)

[SQL Managed Instance Environment Reset TSQL Script 37](#_Toc162384647)

[Appendix 4 – Repairing Core Components 39](#_Toc162384648)

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Comments |
| 2.0 | 2/11/19 | steve.morgan@microsoft.com | Major rewrite to include instructions for the automated environment build |
| 2.1 | 18/5/2020 | steve.morgan@microsoft.com | Added updates in prep for sharing with partners |
| 2.2 | 20/5/2020 | steve.morgan@microsoft.com | Removed Appendix 2: Manual Environment Build Guide |
| 2.2.1 | 22/5/2020 | steve.morgan@microsoft.com | Embedded post build TSQL scripts. |
| 2.2.2 | 26/5/2020 | steve.morgan@microsoft.com | Added SSIS post build tasks |
| 2.3 | 3/6/2021 | steve.morgan@microsoft.com | Amended SSIS lab build appendix task to reflect new PowerShell based build process |
| 3.0 | 3/21/2024 | mert.senguner@microsoft.com | Multiple corrections on various sections based on recent changes |
| 3.1 | 3/26/2024 | mert.senguner@microsoft.com | Architecture pictures adjusted to the new setup |
|  |  |  |  |

# Pre-Requisites

1. Azure Subscription

An existing Azure Subscription that you have Contributor rights in is needed to deploy the hack environment. You can use a One-Time PASS subscription for this. Please check the Github Repo for instructions to apply for a One-Time PASS subscription, create a tenant and associate PASS subscription with this tenant.

# READ ME FIRST! Environment Setup - Update March 2024

The set-up of the hack environment (as shown in the [architecture diagram](#_Hack_Environment_Architecture)) has been simplified and largely automated using PowerShell & ARM templates.

Instructions for the automated build process can be found here:

[Appendix 1 – Automated Environment Build Guide](#_Download_&_Run)

|  |
| --- |
| **There are some manual tasks that need to be completed after the build process.**  **See** [**Appendix 2 – Post Environment Build Tasks**](#_Appendix_3_–) **for details** |

|  |
| --- |
| **If reusing the environment for subsequent runs of the hack there are some tasks that need to be performed to rest the environment.**  **See** [**Appendix 3 – Daily Environment Reset Tasks**](#_Appendix_4_–) **for details** |

# Hack Environment Architecture and Azure Components

A computer screen shot of a computer

Description automatically generated

# Environment Setup Overview – For Information Only

This is the high-level build process performed by the PowerShell script and ARM templates.

|  |
| --- |
| *This section is for information only and you can skip straight to Appendix 1 – Automated Environment Build Guide to begin the environment build.* |

**NOTE:** All resources should be created in the same region (E.g. North Europe)

1. Create Resource Groups and Vnet
   * 1 x Resource Group for shared services (source SQL2012 VM, source SQL2016 VM, , SQL Managed Instance, Storage Account, ADF SSIS IR, DMS)
   * 1 x Resource Group for user VM’s (Jump boxes)
   * Create single Vnet
   * Create 3 subnet inside the Vnet as per the [architecture](#_Migration_architecture_and)
2. Create Azure SQL Managed Instance
   * The Managed Instance could take several hours to provision.
3. Create SQL2012 and SQL2016 legacy environments
   * This VM’s mimic an on-premises legacy SQL Server 2012 and 2016
   * This VM’s will be in the SQLHACK-SHARED resource Group.
   * Restore application databases to the LEGACYSQL2012 instance
   * Create file share for DMS (Azure Database Migration Service) to store local backup files in as part of the migration process.
4. Create a BLOB storage for DMS to move database backup files to
5. Create team Win11 VM used by workshop attendees for labs
   * Download all lab guides & the dummy application SimpleTranReportApp.exe
   * Install SQL Server Management Studio on each Win11 VM
   * Install DMS on each Win11 VM
   * Install DMA (Data Migration Assistance) on each Win11 VM

# Appendix 1 – Automated Environment Build Guide

This method makes use of a pre-built PowerShell script that submits an ARM template to Azure for execution.

## PREREQUISITE: Lower PowerShell’s Security Policy

Because the PowerShell script is unsigned you need to lower PowerShell’s security policy using the below statements. This only needs to be done once on your machine unless you modify the policy afterwards.

**#Set script execution policy to Unrestricted so unsigned script can be run:**

**Set-ExecutionPolicy -ExecutionPolicy Unrestricted**

**#Check ExecutionPolicy has been lowered to Unrestricted**

**Get-ExecutionPolicy**

## Download & run the PowerShell script & ARM templates

1. Download the complete environment from this GIT repository from Master branch:

[Azure/SQL-Modernisation-Hack (github.com)](https://github.com/Azure/SQL-Modernisation-Hack)

**Repository is public for all Microsoft employees under Microsoft Tenant in the “Azure” Organisation. (You need a MSFT alias and appropriate access rights to “Azure” organization to access it.)**

A screenshot of a computer

Description automatically generated

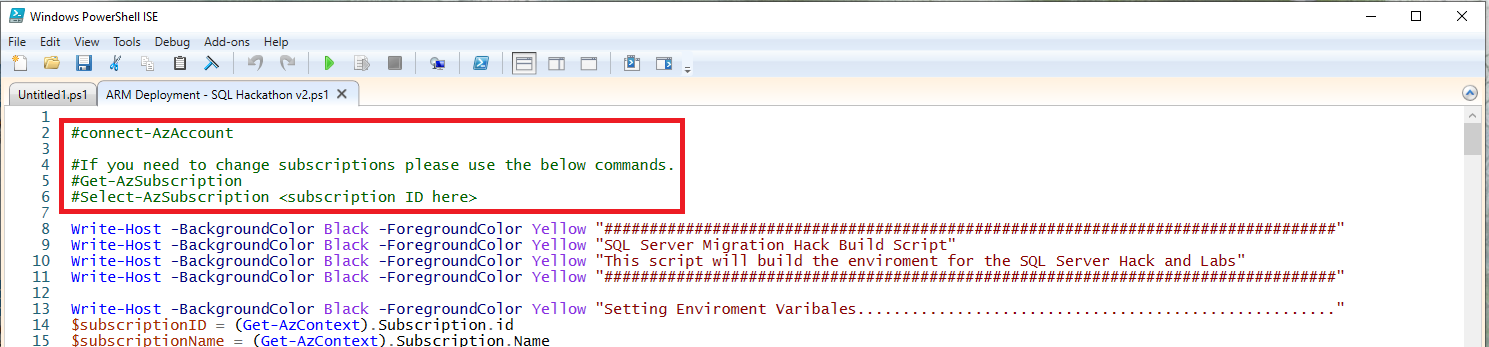
1. Open the “Build” folder in the unzipped/downloaded repository

Graphical user interface, application

Description automatically generated

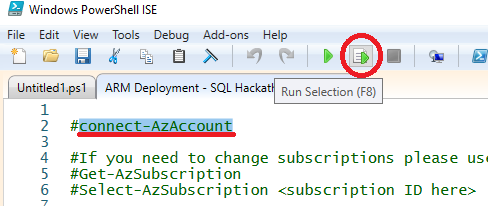
1. Open PowerShell ISE ***as Administrator*** pr VSCode and then open the PowerShell script “ARM Deployment - SQL Hackathon v2.ps1”
2. Before the script can be run the PowerShell session must be connected to the required Azure subscription.

To do this ***3 commented out Azure PS commands*** have been included at the top of the script that need to be run separately.



1. Highlight the text **connect-AzAccount**

|  |
| --- |
| ***Make sure you don’t highlight the preceding hash (“#”) character***  ***Make sure you press the “Run Selection” button and NOT the “Run” button*** |



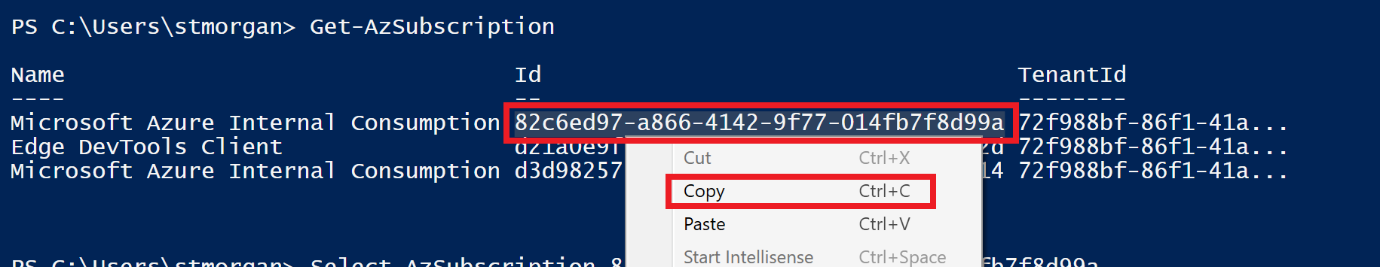
1. This will cause a standard Azure Login window to open. Login to Azure with the required account for the Subscription you want to deploy the hack environment to.

***REMEMBER: The login used maybe the login for a completely different Azure Tenant that has been created specifically to host the hack.***

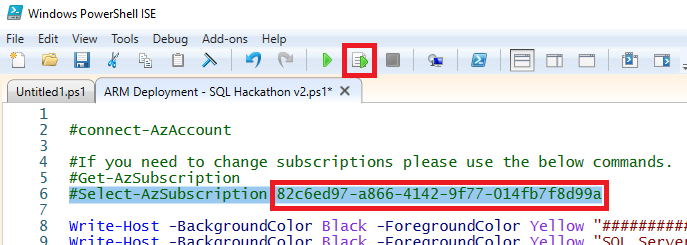
1. Once the PowerShell session has logged into the Azure account high-light & run the second commented out command **Get-AzSubscription**

Again use the “Run selection” button to run just this command. This will return the subscriptions associated with that Azure login.

High-light & copy the ID of the Subscription you want to deploy the hack environment to.

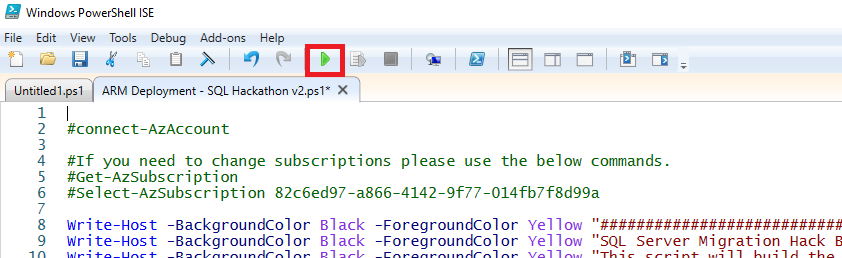


1. Run the 3rd commented command **Select-AzSubscription** replacing the holding text “**<subscription ID here>**” with the Subscription ID copied in step 7 above.



The PowerShell session is now connected to the target Subscription in the required Azure account.

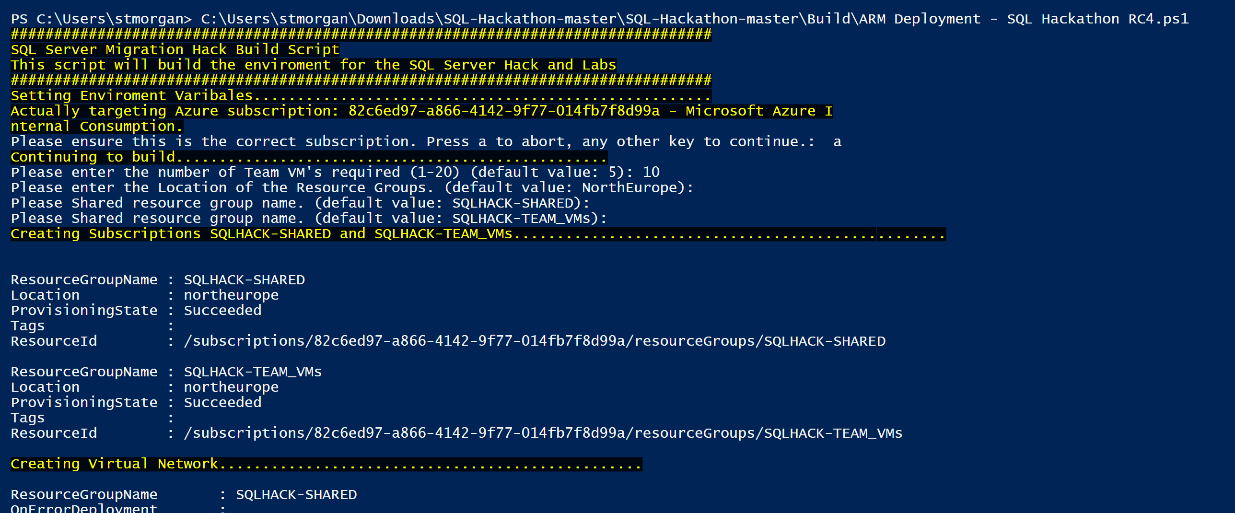
1. Now press the “Run script” button to execute the entire PowerShell script which uses the authentication & subscription defined using the 3 statements just executed.



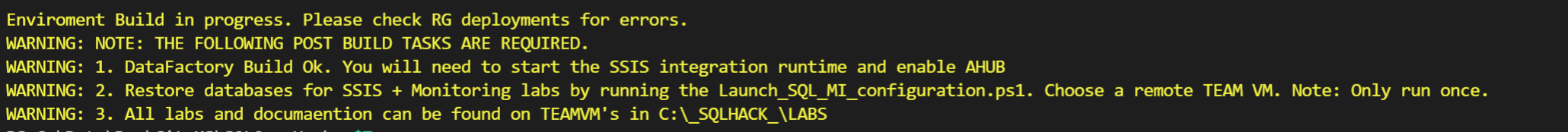
1. As the script executes it will pause several times prompting you to provide various parameters.

|  |  |
| --- | --- |
| ***For the Admin username and password we strongly recommend you use the following account details as these are reflected in the Labs Guides that users follow.*** | |
| ***Username*** | **DemoUser** |
| ***Password*** | **Demo@pass1234567** |

For all other parameters you may enter your own values or accept the defaults.



1. The script will run for 1 or 2 minutes when complete it will display some informational messages.



|  |
| --- |
| **NOTE: At this point the environment has NOT been built!** |

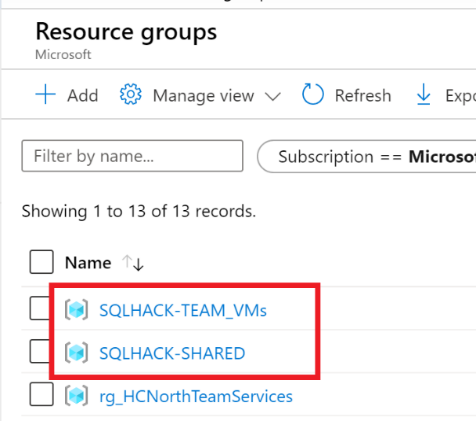
The PowerShell script has now successfully passed the ARM templates & required parameters to Azure and they will take several hours to run & complete the actual build (mainly due to the SQL Managed Instance creation).

The progress of the ARM templates can be monitored through the Azure portal (see the next section)

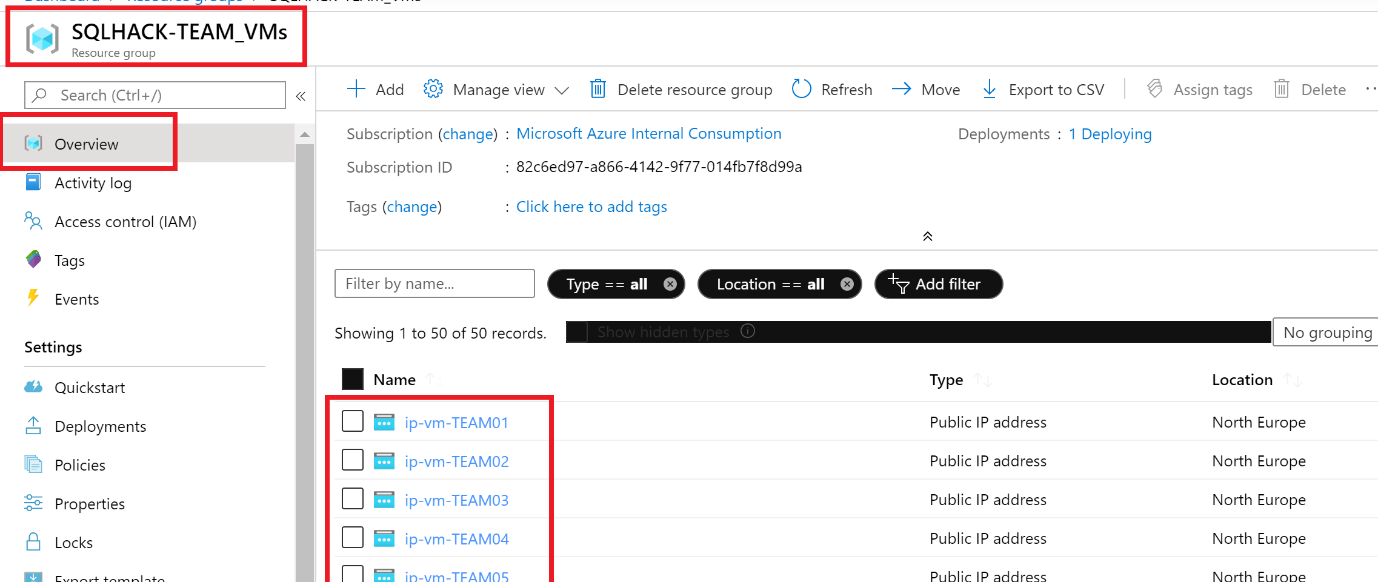
*PTO*

## Monitoring the ARM template deployments

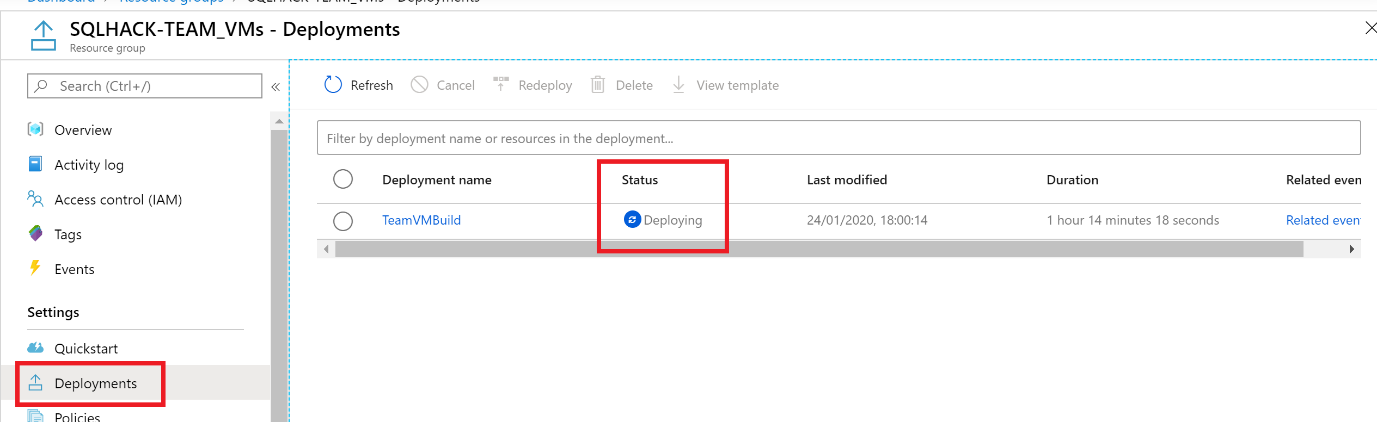
1. Open the Azure portal.
2. Under the Resource Groups you will find the 2 RGs created for the hack environment



1. Opening each RG will show you resources that have already been deployed.



1. Click the “Deployments” option to see how many ARM templates are still being executed in this RG.



Click the “Related events” link on the Deployment Name or the Related Events on link on the far right to see details of the currently executing steps.

## Confirming the environment automated build

Using the default parameters, the following high-level deployments should be listed under each Resource Groups’ “Deployments” blade:

|  |  |  |
| --- | --- | --- |
| **Resource Group** | **Deployment Name** | **Description/notes** |
| SQLHACK-SHARED | updateSubnetTemplate | Updates the required shared networking components of the environment |
| ManagedInstanceBuild | Builds the SQL Managed Instance used as the DB migration target |
| nic\_<GUID> | Builds network interface for the VNet |
| SharedServicesBuild | Builds the required ADF V2 and DMS instance |
| LegacySQLBuild(s) | Create the SQL2012-16 VM (1 for each)& auto installs a SQL2012-16 instance (1 on each VM) |
| NetworkBuild | Builds all shared networking components |
| SQLHACK-TEAMVMS | TeamVMBuild | Build the Win11 VMs used by the delegates |

If any of these core components are missing or the Deployment is showing as failed see [Appendix 4 - Repairing Core Components](#_Appendix_4:_Repair) for guidance.

## Confirming the database & software automated build

1. Make sure one of the Win11 VMs is started & RDP onto it using either your username & password or the default account “DemoUser”, password “Demo@pass1234567”
2. On one of the Win11VMs check that the following software & artefacts are installed:

|  |  |  |
| --- | --- | --- |
| **Artefact** | **Location** | **Notes** |
| DMA | Desktop shortcut |  |
| SQLHACK folder shortcut | Desktop shortcut |  |
| SSMS | (Start Menu) | Make sure SSMS launches – may need to reboot Win11 VMs |
| Azure Data Studio | (Start Menu) |  |

1. Launch SSMS and connect to the LEGACYSQL2012/16 server using windows authentication. The server VM and SQL instance may need to be started.

Check that 60 databases (3 each for teams 1 to 20) have been created.

(PS: On LEGACYSQL2016, each team/user will have 1 DB for online migration)

This confirms the LEGACYSQL VM’s has been built and the databases deployed.

***If the databases have not been restored follow below instructions***

|  |
| --- |
| ***If the databases have not been restored follow these instructions:***   * + *RDP or Bastion on to the LEGACYSQL2012 VM*   + *Check directory C:\Backups exists and that it contains the 60 .BAK files (3 for each Team)*   + *On the machine where you downloaded the hack materials from GitHub to locate the “Build\Database Build” directory. Copy the 4 SQL scripts to the LEGACYSQL2012 VM.*   + *On LEGACYSQL2012* ***run SQL scripts 1 to 3*** *to create the required SQL logins and restore the databases*   + *Check that the 60 databases have been restored.* |

1. Check the [TenantDataDb] has [SalesLT].[Customer] and [SalesLT].[Product] table and they are populated
   1. Run this TSQL to check tables exists that they have data (check the Messages tab for errors)

DECLARE @cmd varchar(500)

SET @cmd='

IF "?" LIKE "%TenantDataDb"

BEGIN

USE ?

select DB\_Name(), ''SalesLT.Customer'', count(\*) from SalesLT.Customer;

select DB\_Name(), ''SalesLT.Product'', count(\*) from SalesLT.Product;

END'

EXEC sp\_MSforeachdb @cmd;

|  |
| --- |
| ***If the databases have SalesLT.Customer or SalesLT.Product tables missing follow these instructions:***   * + *RDP or Bastion on to the LEGACYSQL2012 VM*   + *Copy the 3 files in this folder:* [*https://microsofteur.sharepoint.com/:f:/t/UKCSUDataPlatformMigrationTeam/EpThoy2kHe1EmSrLfhkTVEUBelE8\_xHwjazj5fWtSQrQXg?e=h0Wdai*](https://microsofteur.sharepoint.com/:f:/t/UKCSUDataPlatformMigrationTeam/EpThoy2kHe1EmSrLfhkTVEUBelE8_xHwjazj5fWtSQrQXg?e=h0Wdai)   *to \\LEGACYSQL2012\FILESHARE*   * + *On LEGACYSQL2012* ***run SQL script C:\FILESHARE\CREATECust+ProdTables.sql*** *to create the required tables and load them with data* |

*PTO*

1. Run the dummy SimpleTransReportApp application located at:

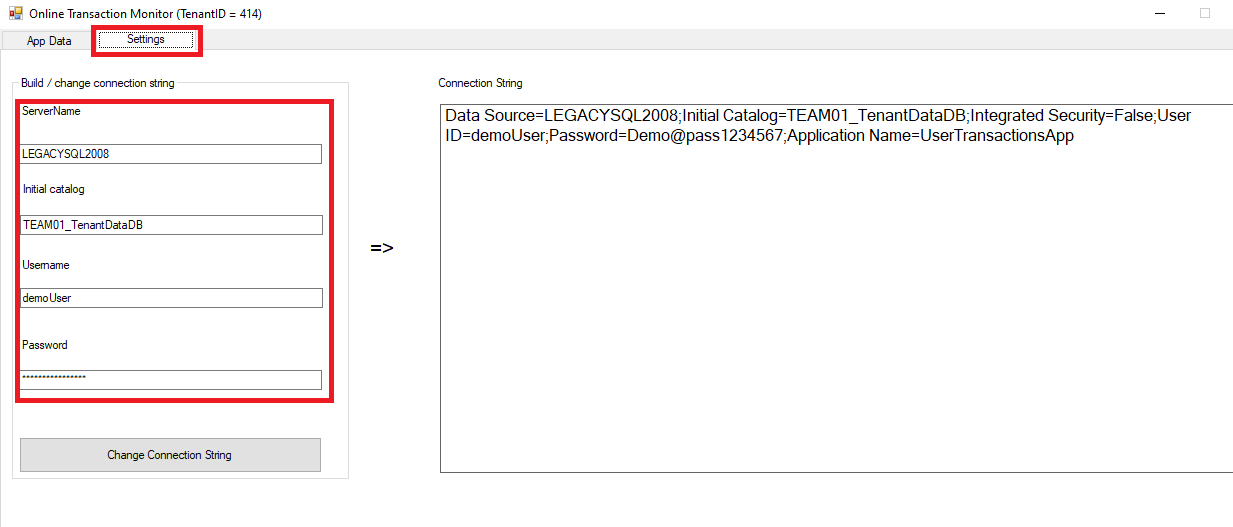
C:\\_SQLHACK\_\LABS\01-Data\_Migration\SimpleTranReportApp.exe

On the Settings tab change the connection string details on the left using the following:

|  |  |  |
| --- | --- | --- |
| **Server Name** | LEGACYSQL2008 |  |
| **Initial Catalog** | TEAMXX\_TenantDataDB | Where XX = a Team number between 01 and 20 |
| **Username** | DemoUser |  |
| **Password** | Demo@pass1234567 |  |

Then click the “Change Connection String” button. The changes should be reflected in the window on the right.

(PS: In the latest Lab versions, using the SimpleTransReportApp is obsolete. You just have 2 Jobs on SQL Server Agent which should be run)



Switch to the “App Data” tab and click the Run button. You should see transactions being returned & auto refreshing every second or so.

|  |
| --- |
| ***Now go to*** [***Appendix 2 – Post Environment Build Tasks***](#_Appendix_3_–) ***to complete a number of minor manual configuration tasks.*** |

*This page intentionally blank*

# Appendix 2 – Post Environment Build Tasks

Once the automated build has completed there still a number of tasks that need to be performed manually.

This appendix details those tasks.

*PTO*

## Appendix 2.1 - Post Environment Build Tasks: Database Migration Environment

Some tasks get missed during the environment current build process. To make sure the hack runs correctly check the following tasks are completed:

|  |  |  |
| --- | --- | --- |
| **LEGACYSQL 2012 VM** | Enable CLR on the SQL 2012 Server and set permissions on system CLRs to "unrestricted" | CLR assemblies in [TEAMXX\_TenantDataDB] are reported as blockers in DMA due to CLR permission settings. This needs to be fixed in all 60 databases.  Connect to the LEGACYSQL2012 using the DemoUser account (which is sysadmin) and run the TSQL in [section 2.1.1](#_2.1.1_LEGACYSQL2008_Environment) over the page.  (For SQL 2016 Server nothing has to be done, since the assessment is always against SQL 2012 on the lab1) |
| **SQL Managed Instance** | RUN “Launch\_SQL\_MI\_configuration.ps1” from the Build folder  Enable CLR on the Managed Instance | CLRs need to be enabled on the SQL Managed Instance for the dummy application to work after database migration.  See sample TSQL code in [section 2.1.2](#_2.1.2_Managed_Instance_1) below. |
| **Win11**  **Jump-boxes** | Check keyboard is set to “English (United Kingdom)” ***NOT*** “English (United States)” | The password for the DemoUser user account has an “@” character in it. If the VM is set to US English the “@” key produce a different character. |
| Generate SAS URI key for the SQLHack blob container and paste into C:\LABS\01...\SASKEY.txt on each Win11 VM | SAS key generated by template is incorrect. Easiest way to fix is to generate a new SSAS Key through Storage Explorer (either in the portal or desktop app) on the “migration” BLOB container. **The SAS key must have Read/Write/List/Delete permissions.**  ***\*\*\* DON'T FORGET TO PUT THE KEY'S EXPIREY DATE \*\*\****  ***\*\*\* FAR ENOUGH IN THE FUTURE \*\*\**** |
| Restart all VMs | Sometime have a build issue where SSMS doesn’t install correctly until the machine is shut down & restarted. |
| Corrected "ManagedInstanceFQDN.txt" and place in  "C:\\_SQLHACK\_\LABS\01-Data\_Migration" folder on each Win11 VM | Paste the SQL Managed Instance FQDN into a text document & place the text document in the C:\...LABS\01… directory |

### 

### 2.1.1 LEGACYSQL2012 Set CLR Permissions

|  |
| --- |
| **If you copying & paste the below TSQL be careful that no hidden characters are also copied** |

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**SQLHACK DB ENVIORNMENT RESET SCRIPT - LEGACY SQL2012 INSTANCE**

**=================================================**

**RUN AGAINST: [LEGACYSQL2012]**

**PURPOSE:**

**1. checks it's being run against the legacy SQL Server instance**

**2. Makes sure CLR is enabled on the legacy environment**

**3. Lowers the trust settings for the CLR SPs so they don't throw warnings/errors in DMA**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**IF @@SERVERNAME <> 'LEGACYSQL2012'**

**RAISERROR('\*\*\* NOT CONNECTED TO [LEGACYSQL2012] INSTANCE \*\*\*', 20, 1) WITH LOG;**

**USE master;**

**GO**

**EXEC sp\_configure 'clr enabled', 1;**

**GO**

**RECONFIGURE;**

**GO**

**USE TEAM01\_TenantDataDb;**

**GO**

**ALTER ASSEMBLY CLRUFDS WITH PERMISSION\_SET = UNSAFE;**

**ALTER ASSEMBLY Database1 WITH PERMISSION\_SET = UNSAFE;**

**GO**

**USE TEAM02\_TenantDataDb;**

**GO**

**ALTER ASSEMBLY CLRUFDS WITH PERMISSION\_SET = UNSAFE;**

**ALTER ASSEMBLY Database1 WITH PERMISSION\_SET = UNSAFE;**

**GO**

**USE TEAM03\_TenantDataDb;**

**GO**

**ALTER ASSEMBLY CLRUFDS WITH PERMISSION\_SET = UNSAFE;**

**ALTER ASSEMBLY Database1 WITH PERMISSION\_SET = UNSAFE;**

**GO**

**USE TEAM04\_TenantDataDb;**

**GO**

**ALTER ASSEMBLY CLRUFDS WITH PERMISSION\_SET = UNSAFE;**

**ALTER ASSEMBLY Database1 WITH PERMISSION\_SET = UNSAFE;**

**GO**

**USE TEAM05\_TenantDataDb;**

**GO**

**ALTER ASSEMBLY CLRUFDS WITH PERMISSION\_SET = UNSAFE;**

**ALTER ASSEMBLY Database1 WITH PERMISSION\_SET = UNSAFE;**

**GO**

**USE TEAM06\_TenantDataDb;**

**GO**

**ALTER ASSEMBLY CLRUFDS WITH PERMISSION\_SET = UNSAFE;**

**ALTER ASSEMBLY Database1 WITH PERMISSION\_SET = UNSAFE;**

**GO**

**USE TEAM07\_TenantDataDb;**

**GO**

**ALTER ASSEMBLY CLRUFDS WITH PERMISSION\_SET = UNSAFE;**

**ALTER ASSEMBLY Database1 WITH PERMISSION\_SET = UNSAFE;**

**GO**

**USE TEAM08\_TenantDataDb;**

**GO**

**ALTER ASSEMBLY CLRUFDS WITH PERMISSION\_SET = UNSAFE;**

**ALTER ASSEMBLY Database1 WITH PERMISSION\_SET = UNSAFE;**

**GO**

**USE TEAM09\_TenantDataDb**

**GO**

**ALTER ASSEMBLY CLRUFDS WITH PERMISSION\_SET = UNSAFE;**

**ALTER ASSEMBLY Database1 WITH PERMISSION\_SET = UNSAFE;**

**GO**

**USE TEAM10\_TenantDataDb**

**GO**

**ALTER ASSEMBLY CLRUFDS WITH PERMISSION\_SET = UNSAFE;**

**ALTER ASSEMBLY Database1 WITH PERMISSION\_SET = UNSAFE;**

**GO**

**USE TEAM11\_TenantDataDb**

**GO**

**ALTER ASSEMBLY CLRUFDS WITH PERMISSION\_SET = UNSAFE;**

**ALTER ASSEMBLY Database1 WITH PERMISSION\_SET = UNSAFE;**

**GO**

**USE TEAM12\_TenantDataDb**

**GO**

**ALTER ASSEMBLY CLRUFDS WITH PERMISSION\_SET = UNSAFE;**

**ALTER ASSEMBLY Database1 WITH PERMISSION\_SET = UNSAFE;**

**GO**

**USE TEAM13\_TenantDataDb**

**GO**

**ALTER ASSEMBLY CLRUFDS WITH PERMISSION\_SET = UNSAFE;**

**ALTER ASSEMBLY Database1 WITH PERMISSION\_SET = UNSAFE;**

**GO**

**USE TEAM14\_TenantDataDb**

**GO**

**ALTER ASSEMBLY CLRUFDS WITH PERMISSION\_SET = UNSAFE;**

**ALTER ASSEMBLY Database1 WITH PERMISSION\_SET = UNSAFE;**

**GO**

**USE TEAM15\_TenantDataDb**

**GO**

**ALTER ASSEMBLY CLRUFDS WITH PERMISSION\_SET = UNSAFE;**

**ALTER ASSEMBLY Database1 WITH PERMISSION\_SET = UNSAFE;**

**GO**

**USE TEAM16\_TenantDataDb**

**GO**

**ALTER ASSEMBLY CLRUFDS WITH PERMISSION\_SET = UNSAFE;**

**ALTER ASSEMBLY Database1 WITH PERMISSION\_SET = UNSAFE;**

**GO**

**USE TEAM17\_TenantDataDb**

**GO**

**ALTER ASSEMBLY CLRUFDS WITH PERMISSION\_SET = UNSAFE;**

**ALTER ASSEMBLY Database1 WITH PERMISSION\_SET = UNSAFE;**

**GO**

**USE TEAM18\_TenantDataDb**

**GO**

**ALTER ASSEMBLY CLRUFDS WITH PERMISSION\_SET = UNSAFE;**

**ALTER ASSEMBLY Database1 WITH PERMISSION\_SET = UNSAFE;**

**GO**

**USE TEAM19\_TenantDataDb**

**GO**

**ALTER ASSEMBLY CLRUFDS WITH PERMISSION\_SET = UNSAFE;**

**ALTER ASSEMBLY Database1 WITH PERMISSION\_SET = UNSAFE;**

**GO**

**USE TEAM20\_TenantDataDb**

**GO**

**ALTER ASSEMBLY CLRUFDS WITH PERMISSION\_SET = UNSAFE;**

**ALTER ASSEMBLY Database1 WITH PERMISSION\_SET = UNSAFE;**

**GO**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**END SCRIPT**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

*Click here to return to the list of* [*post build tasks*](#_Appendix_2.1_-)*.*

### 2.1.2 Managed Instance Enable CLR TSQL

*If you copying & paste the below TSQL be careful that no hidden characters are also copied*

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**SCRIPT START**

**RUN AGAINST: <<SQL MANAGED INSTANCE>>**

**PURPOSE:**

**1. Enables CLR execution**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**EXEC sp\_configure 'clr enabled', 1;**

**GO**

**RECONFIGURE;**

**GO**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**SCRIPT END**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

*Click here to return to the list of* [*post build tasks*](#_Appendix_2.1_-)*.*

### 2.1.3 Managed Instance Configuration Script

RUN “Launch\_SQL\_MI\_configuration.ps1” from the Build folder. This will restore the necessary Databases and prepares the MI.

## Appendix 2.2 – Post Environment Build: SSIS Migration Setup Tasks

|  |
| --- |
| **Jun-2021 UPDATE:** The build of the SSIS Migration to ADF lab has now  been automated using a PowerShell Script. |

Follow [Appendix 2.2.1](#_Appendix_2.2.1_-) for running the automated build or [Appendix 2.2.2](#_Appendix_2.2.2_-) if you want to build and configure the SSIS migration lab manually.

### Appendix 2.2.1 - SSIS Migration Lab: Automated Build

1. RDP or Bastion onto one of the Team Win11 VMs
2. Open PowerShell ISE and *run* *as Administrator*
3. Run the below command to lower PowerShells security policy to allow scripts to be executed

Set-ExecutionPolicy Unrestricted

1. In PowerShell open file **C:\Install\ folder as SSIS Build Script.ps1**
2. When prompted for NuGet provide access click **Yes**

It can take a few minutes to download, unzip and install the various AZ and SQL Server modules

1. When prompted login to Azure using a relevant account for tenant you have deployed the lab environment to.
2. Accept the default values for th other prompts including to use the DemoUser admin account.
3. Once completed check on the shared SQL Managed Instance that the [2008DW], [LocalMasterDataDb], [SharedMasterDataDB] and [TenantDataDb] databases have been restored

Graphical user interface, text, application

Description automatically generated

### Appendix 2.2.2 - SSIS Migration Lab: Manual Build

|  |
| --- |
| NOTE: When logging on VM’s check whether Keyboard is UK, as passwords will be incorrectly mapped to keys if US is selected |

|  |
| --- |
| A video walking through the SSIS environment deployment & configuration can be found here: <https://microsofteur.sharepoint.com/:v:/t/UKCSUDataPlatformMigrationTeam-SQLModernisationTTTforOCP/EXqq3mXdmYRJmxb55db8-0oBlPPTppQctdrGIcBYUfqqlw?e=4glZG8> |

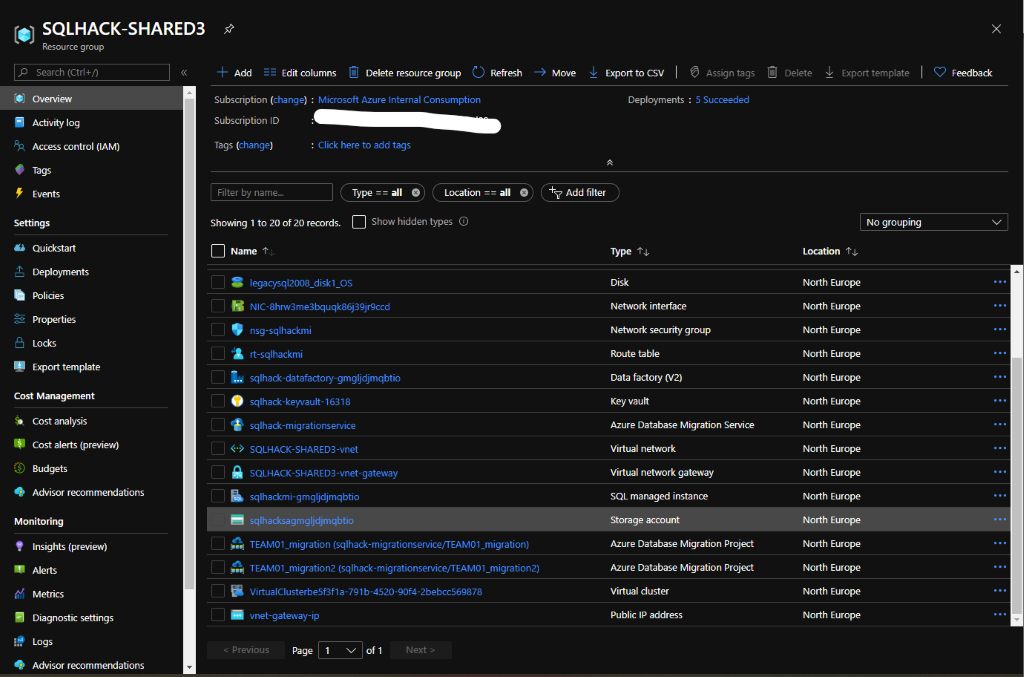
**Download demo database backup files from Github to a local folder**

* Download the SQL Database files (4 BAK files) and one SQL Script from

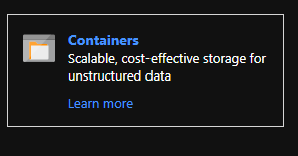
<https://github.com/markjones-msft/SQL-Hackathon/tree/master/Build/SQL%20SSIS%20Databases>

**Set-up folder in Azure**

* Set-up container in Azure Portal to upload SQL Database files
* In the Azure Portal, in the “SQLHACK-Shared” Resource Group
* Find the Storage account, for example below “sqlhacksagmgljdjmqbtio”



* Select Containers



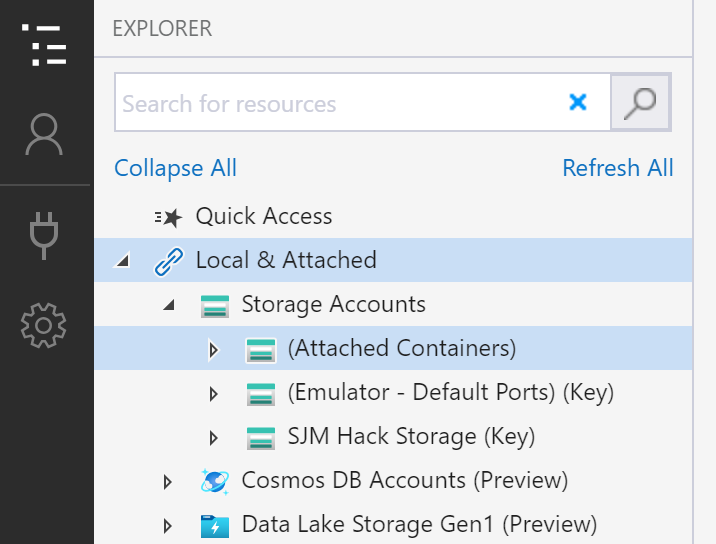
* Add a new Container



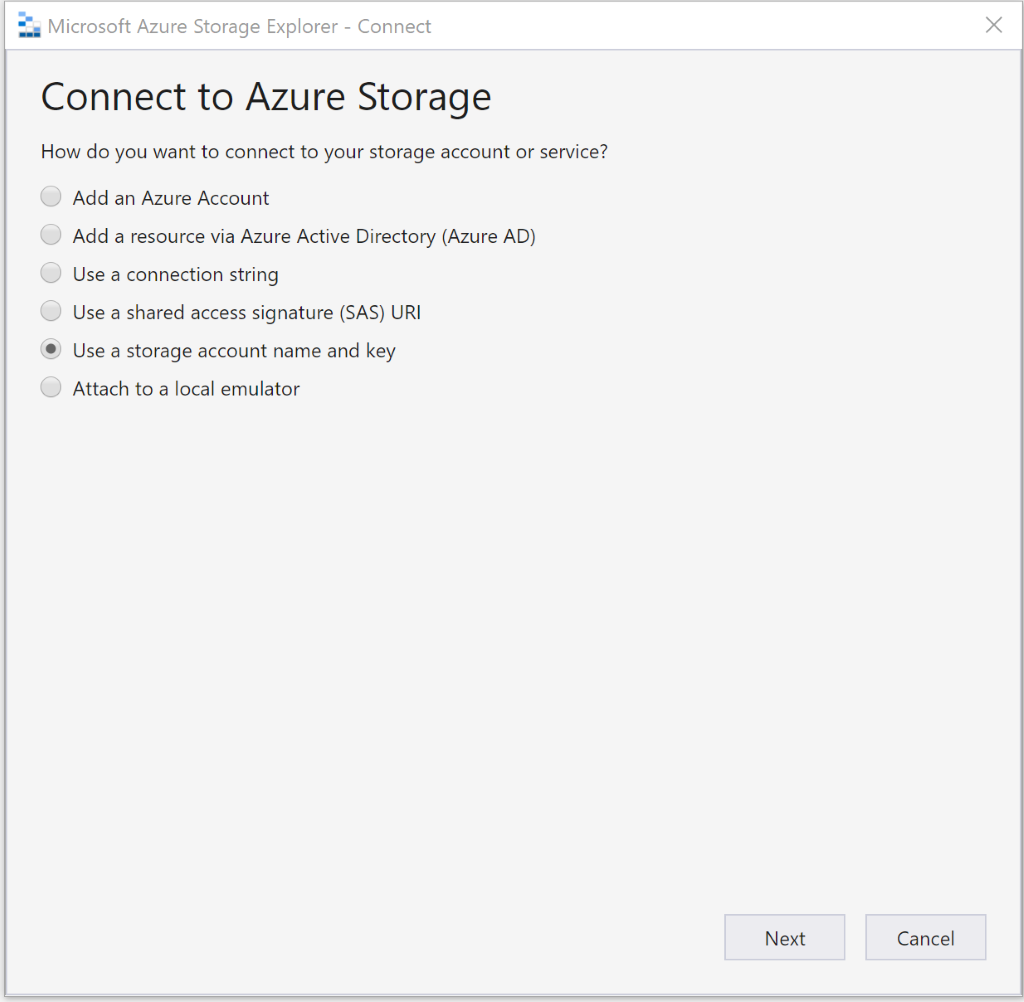
* Name : sqlbackups
* Public access level : Private

**Upload Local files to Azure**

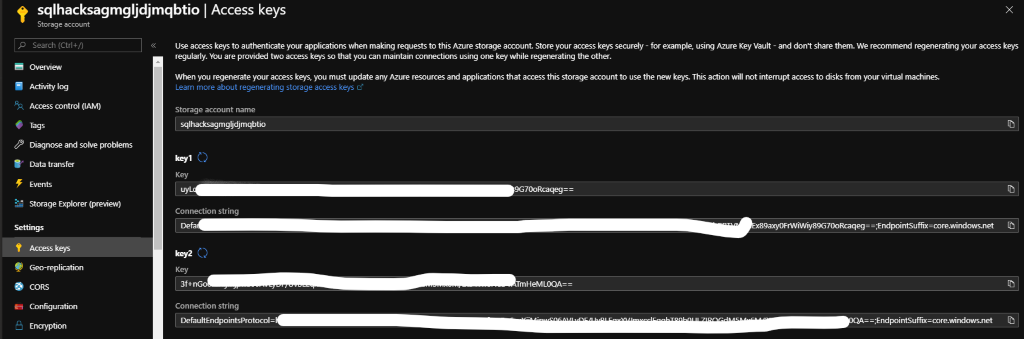
* Open Storage Explorer (Download :<https://azure.microsoft.com/en-us/features/storage-explorer/>)
* In Explorer click on the connect (Plug)



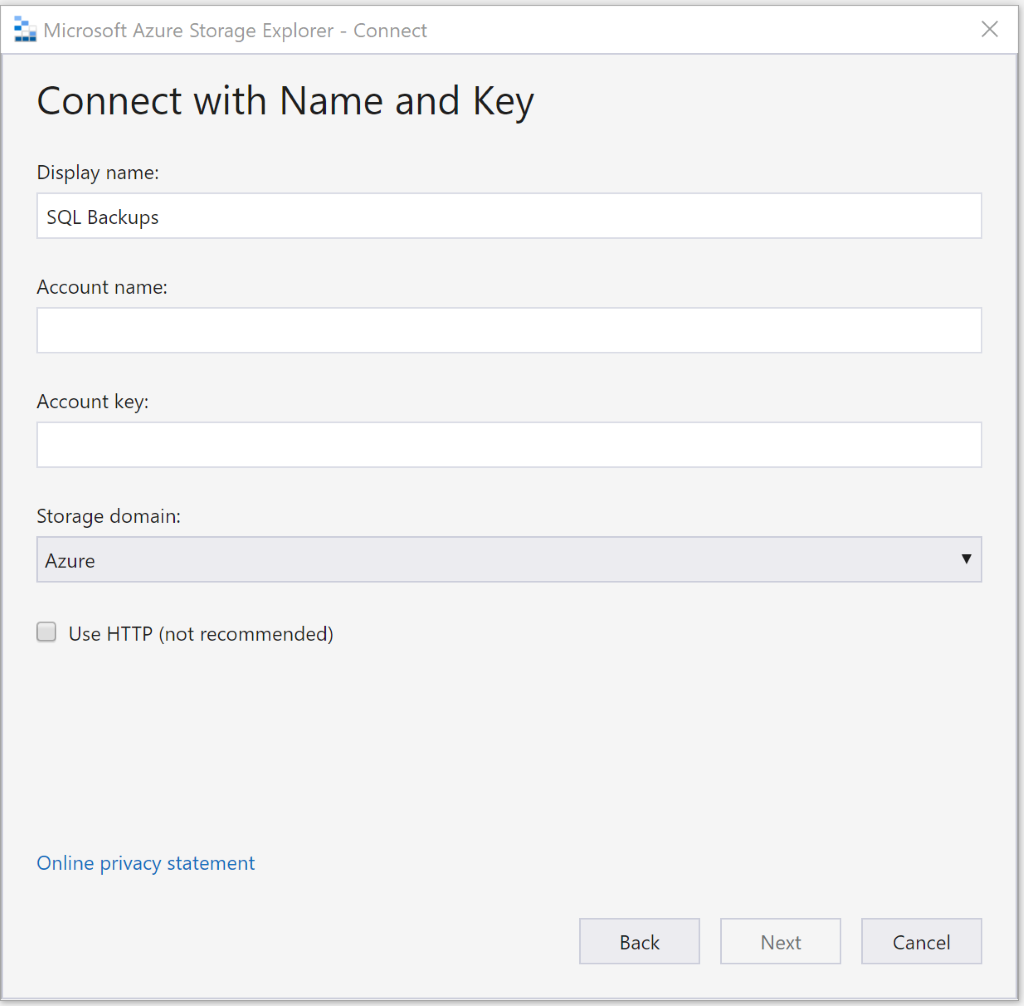
* Select the Storage account name and Key



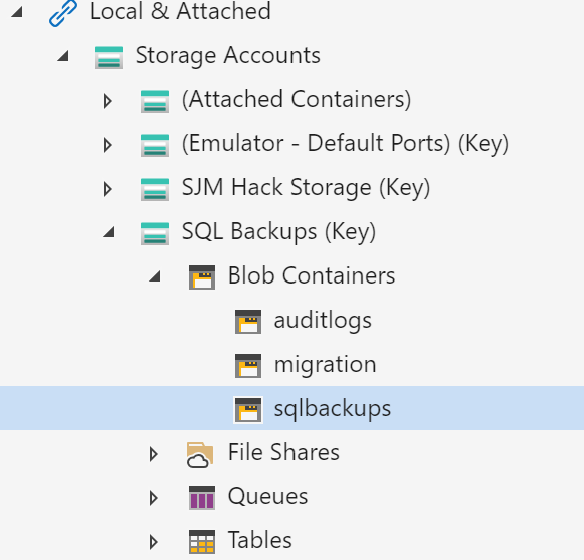
* From the Azure Portal for the storage account, click on Access Keys and copy one of the keys



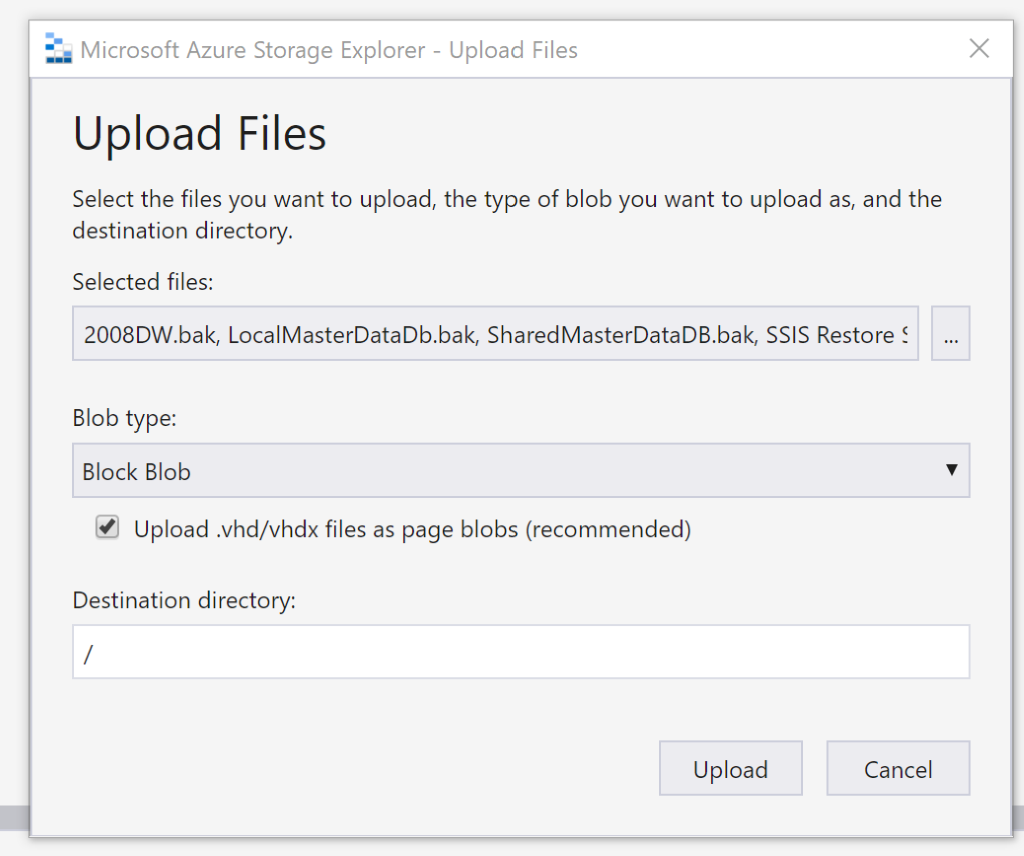
* Input the fields below



* You should be able to expand the container and see the SQL Backups folder created previously



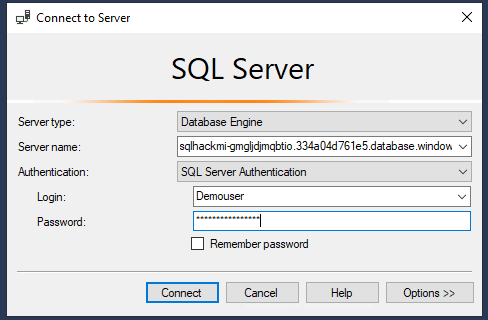
* Upload the 4 Backups and SQL Script downloaded earlier to the Storage account



* Files should then be uploaded to Container in Azure for the next steps

**Restoring databases to SQL Managed Instance**

* Log on to one of the Team VMs
* Open SQL Server Management Studio
* Login into your SQL Managed Instance (MI)



Once you are on the Managed Instance, run the below TSQL script (section [2.2.2.3](#_2.2.2.3_RESTORE_SSIS) over the page) to attach the demo databases from the Storage account.

The secret is the SAS URI key from previous set-up steps.

### 

#### 2.2.2.3 Restore SSIS Migration Demo Databases to SQL Managed Instance TSQL

|  |
| --- |
| **If you copy & paste the below TSQL be careful that no hidden characters are also copied** |

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

- Replace <ENTER STORAGE ACCOUNT HERE> with the Storage Account name

- Replace <ENTER SAS URI KEY HERE> with SAS URI key

- The backups to restore can be found in the repository folder SQL SSIS Databases, you

will need to upload these to a blob store

- For more information on how to restore databases from URL see: https://docs.microsoft.com/en-us/azure/sql-database/sql-database-managed-instance-get-started-restore

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

USE master

GO

EXEC sp\_configure "CLR Enabled", 1

RECONFIGURE WITH OVERRIDE

GO

-- DROP CREDENTIAL [https://<Enter storage account name here>.blob.core.windows.net/sqlbackups];

CREATE CREDENTIAL [https://<Enter storage account name here>.blob.core.windows.net/sqlbackups]

WITH IDENTITY='Shared Access Signature',

SECRET='<ENTER SAS URI KEY HERE>';

GO

RESTORE DATABASE [LocalMasterDataDb]

FROM URL = 'https://<ENTER STORAGE ACCOUNT HERE>.blob.core.windows.net/sqlbackups/LocalMasterDataDb.bak';

GO

RESTORE DATABASE [SharedMasterDataDb]

FROM URL = 'https://<ENTER STORAGE ACCOUNT HERE>.blob.core.windows.net/sqlbackups/sharedMasterDataDb.bak';

GO

RESTORE DATABASE [TenantDataDb]

FROM URL = 'https://<ENTER STORAGE ACCOUNT HERE>.blob.core.windows.net/sqlbackups/TenantDataDb.bak';

GO

RESTORE DATABASE [2008DW] FROM URL = 'https://<ENTER STORAGE ACCOUNT HERE>.blob.core.windows.net/sqlbackups/2008DW.bak';

GO

# Appendix 2.3 – Security Lab Build

## Appendix 2.3.1 - Auditing

For part 1 of the security lab we must enable auditing on the shared SQLMI. This requires:

* Creating a credential that holds the SAS URI key for writing the audit log to Azure Storage
* Creating an Audit Log using the credentials
* Switch auditing on

1. Open SSMS, connect to the shared SQLMI and run the below SQL:
   1. Replace the credential name with the URL of the shared storage account Auditlogs container
   2. Replace the SECRET parameter with the SAS key (labelled the “Query string” in the Azure SAS Key creation blade) for the shared storage account container

Remember to remove the “?” at the beginning of the Query String

USE master;

CREATE CREDENTIAL [https://<<HACK SHARED STORAGE ACCOUNT HERE>>/auditlogs]

WITH IDENTITY='SHARED ACCESS SIGNATURE',

SECRET = '<<QUERY STRING FOR THE SHARED STORAGE ACCOUNT CONTAINER HERE>>'

GO

CREATE SERVER AUDIT [sqlmi\_auditlog]

TO URL ( PATH ='<<CREDENTIAL NAME HERE i.e. the URL USED ABOVE>>'

, RETENTION\_DAYS = 30 )

GO

ALTER SERVER AUDIT [sqlmi\_auditlog]

WITH (STATE = ON)

GO

1. In SSMS you should now be able to see the Audit created

Graphical user interface, text, application, email

Description automatically generated

## Appendix 2.3.2 – Azure Defender for SQL

For part 4 of the Security labs Azure Defender for SQL must be enabled on the shared SQL Managed Instance through the SQLMI Security Centre screen.

# Appendix 2.4 – Monitoring Lab Build

## Appendix 2.3.1 – Deploy TenantCRM database

* TenantCRM is just a renamed copy of AdventureWorks with some tables removed to keep it under 100MB

This is already handled by the provisioning script. However if you don’t see TenantCRM database restored on the Managed instance you can restore it with a .bak file using this SQL script:

USE master;

CREATE CREDENTIAL [<<URL STORAGE ACCOUNT CONTAINER HERE>>]

WITH

IDENTITY = 'SHARED ACCESS SIGNATURE', --DO NOT CHANGE

SECRET = '<<QUERY STRING FOR THE SHARED STORAGE ACCOUNT CONTAINER HERE>>'

--REMEMBER TO REMOVE LEADING “?” AT BEGINNING OF QUERY STRING

GO

/\*

PROVE BACKUPS CAN BE ACCESSED

\*/

RESTORE FILELISTONLY FROM URL = '<<URL FOR BACKUP FILE HERE>>'

--This is under DB\_Perf Folder under the build folder.

GO

/\*

RESTORE DATABASES

\*/

RESTORE DATABASE TenantCRM

FROM URL = '<<URL FOR BACKUP FILE HERE>>’

GO

## Appendix 2.3.2 – Run simulated workload against the TenantCRM database

1. Drop existing index so the required performance telemetry can be generated (already done).
2. Run the 2 jobs already under SQL Server Agent to get the MI CPU’s loaded.

# Appendix 3 – Daily Environment Reset Tasks

If the hack environment is to be reused, a number of quick tasks need to be performed to reset the environment for another run:

|  |  |
| --- | --- |
| **Check DMS is running** | Do this manually through the Azure Portal |
| **Start Win11 VMs**  (set to shut down at 8pm CET) | Do this manually through the Azure Portal |
| **Start LEGACYSQL2012 VMs**  (set to shut down at 8pm CET) | Do this manually through the Azure Portal |
| **Delete any existing DMS projects** | Do this manually through the Azure Portal |
| **Delete migrated DBs & migrated logins from the SQL MI** | Run [SQL Managed Instance Environment Reset](#_SQL_Managed_Instance) TSQL script (below) to drop all previously migrated DBs and logins. |
| **Delete any DB backups in \\SQL2012\FILESHARE** | *This might be done automatically by DMS* |
| **Delete backups from BLOB Storage** | *This might be done automatically by DMS* |
| **Check location Wi-Fi allows RDP** | RDP outbound may not be enabled if doing a closed/private hack. With Bastion you don’t need to take an action. |

## SQL Managed Instance Environment Reset TSQL Script

|  |
| --- |
| **If you copy & paste the below TSQL be careful that no hidden characters are also copied** |

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SQLHACK DB ENVIORNMENT RESET - MANAGED INSTANCE SCRIPT

======================================================

RUN AGAINST: SQL MI

PURPOSE:

1. Checks it's being run against the SQL Managed Instance

2. Runs a test version of the DROP statement to make sure it drops the correct DBs

3. Commented out version of the DROP DBs statement that does actually drop all the DBs

4. Drop all TEAMXX sql logins so DMS doesn’t moan when you re-migrate DBs & logins

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

1. Check connected to the SQL Managed Instance

\*/

IF @@SERVERNAME NOT LIKE 'sqlhackmi-%.database.windows.net'

RAISERROR('\*\*\* NOT CONNECTED TO SQL MANAGED INSTANCE \*\*\*', 20, 1) WITH LOG;

/\*

2. Run a test version of the DROP statement to make sure it drops the correct DBs

RUN THIS TO MAKE SURE YOU'RE ONLY DROPPING THE CORRECT DATABASES:

\*/

DECLARE @command nvarchar(max)

SET @command = ''

SELECT @command = @command

+ 'DROP DATABASE [' + [name] +'];'+CHAR(13)+CHAR(10)

FROM [master].[sys].[databases]

WHERE [name] NOT IN ('master','model','msdb','tempdb', 'SSISDB');

PRINT @COMMAND

GO

/\*

3. Commented out version of the DROP DBs statement that drops all the DBs

ONLY RUN THIS ONCE YOU'VE TESTED USING ABOVE STATEMENT

\*/

/\*

DECLARE @command nvarchar(max)

SET @command = ''

SELECT @command = @command

+ 'DROP DATABASE [' + [name] +'];'+CHAR(13)+CHAR(10)

FROM [master].[sys].[databases]

WHERE [name] NOT IN ('master','model','msdb','tempdb', 'SSISDB');

EXECUTE sp\_executesql @command

\*/

/\*

4. Drop all TEAMXX sql logins so DMS doens't moan when you re-migrate DBs & logins

\*/

IF EXISTS (select loginname from master.dbo.syslogins where name = 'TEAM02')

DROP LOGIN TEAM02;

IF EXISTS (select loginname from master.dbo.syslogins where name = 'TEAM01')

DROP LOGIN TEAM01;

IF EXISTS (select loginname from master.dbo.syslogins where name = 'TEAM03')

DROP LOGIN TEAM03;

IF EXISTS (select loginname from master.dbo.syslogins where name = 'TEAM04')

DROP LOGIN TEAM04;

IF EXISTS (select loginname from master.dbo.syslogins where name = 'TEAM05')

DROP LOGIN TEAM05;

IF EXISTS (select loginname from master.dbo.syslogins where name = 'TEAM06')

DROP LOGIN TEAM06;

IF EXISTS (select loginname from master.dbo.syslogins where name = 'TEAM07')

DROP LOGIN TEAM07;

IF EXISTS (select loginname from master.dbo.syslogins where name = 'TEAM08')

DROP LOGIN TEAM08;

IF EXISTS (select loginname from master.dbo.syslogins where name = 'TEAM09')

DROP LOGIN TEAM09;

IF EXISTS (select loginname from master.dbo.syslogins where name = 'TEAM10')

DROP LOGIN TEAM10;

IF EXISTS (select loginname from master.dbo.syslogins where name = 'TEAM11')

DROP LOGIN TEAM11;

IF EXISTS (select loginname from master.dbo.syslogins where name = 'TEAM12')

DROP LOGIN TEAM12;

IF EXISTS (select loginname from master.dbo.syslogins where name = 'TEAM13')

DROP LOGIN TEAM13;

IF EXISTS (select loginname from master.dbo.syslogins where name = 'TEAM14')

DROP LOGIN TEAM14;

IF EXISTS (select loginname from master.dbo.syslogins where name = 'TEAM15')

DROP LOGIN TEAM15;

IF EXISTS (select loginname from master.dbo.syslogins where name = 'TEAM16')

DROP LOGIN TEAM16;

IF EXISTS (select loginname from master.dbo.syslogins where name = 'TEAM17')

DROP LOGIN TEAM17;

IF EXISTS (select loginname from master.dbo.syslogins where name = 'TEAM18')

DROP LOGIN TEAM18;

IF EXISTS (select loginname from master.dbo.syslogins where name = 'TEAM19')

DROP LOGIN TEAM19;

IF EXISTS (select loginname from master.dbo.syslogins where name = 'TEAM20')

DROP LOGIN TEAM20;

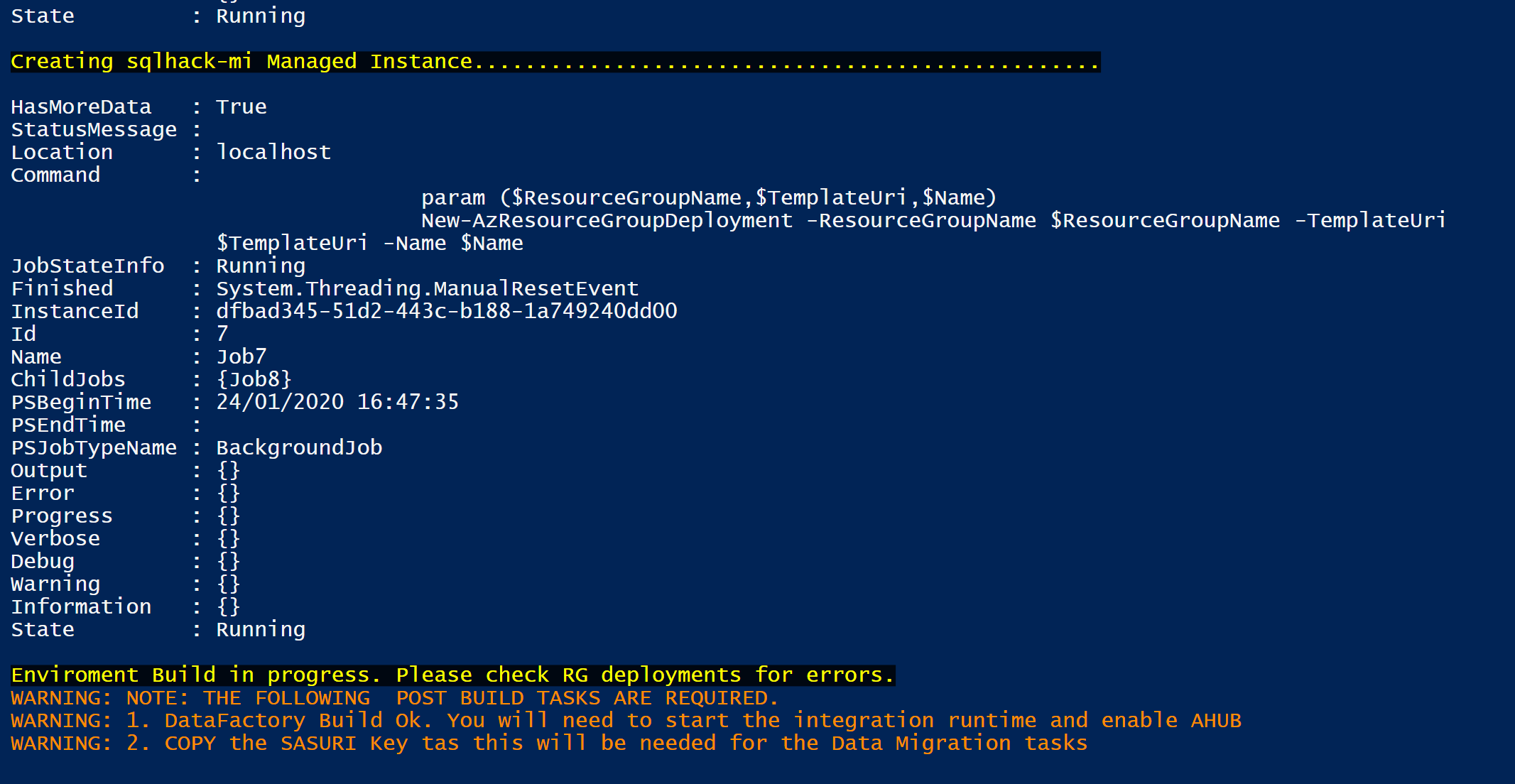
# 

# Appendix 4 – Repairing Core Components

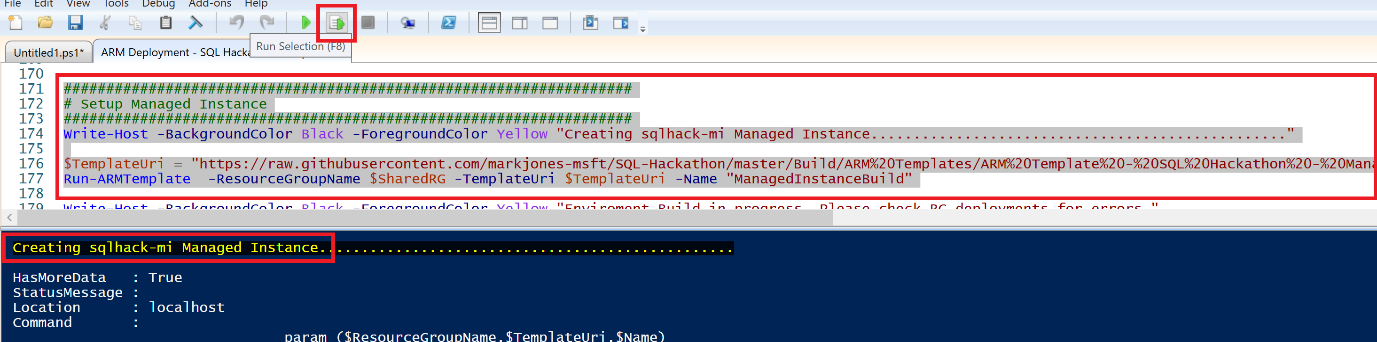
If any of these core components are missing or the Deployment is showing as failed, locate the relevant part in the PowerShell script using the yellow section separators/headings & re-run just that part.

E.G. Initially the ARM template for the SQL Managed Instance has failed to run so the Deployments blade of the SQLHACK-SHARED Resource Group is only showing 4 deployments rather than 5.

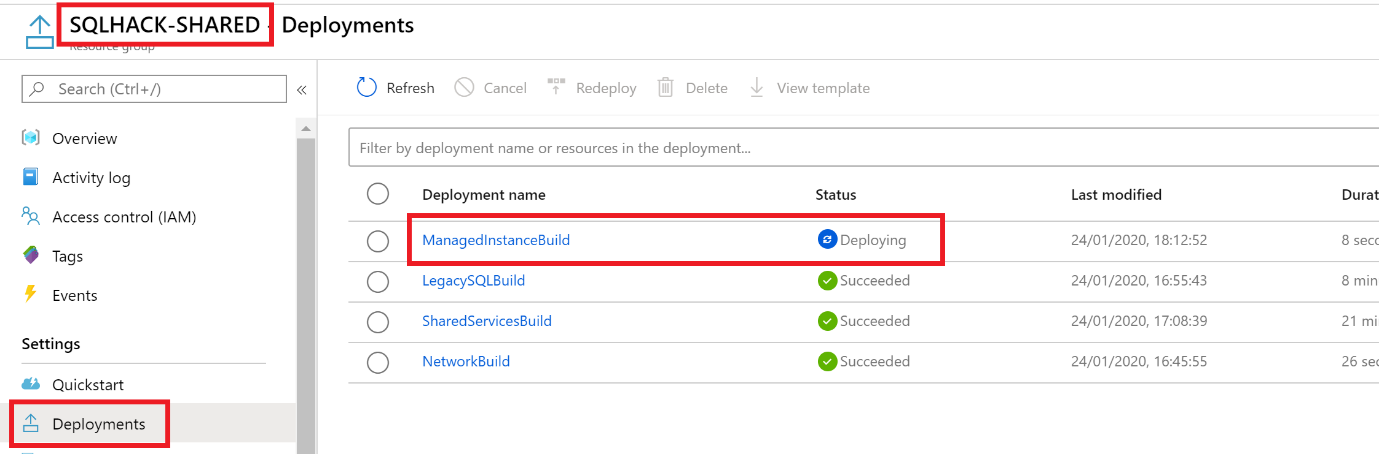
The PowerShell output below doesn’t show any errors but the lack of a SQL Managed Instance in the Resource Group means the ARM template failed for some reason.



Once you’ve established which component is missing, use the comments in the PowerShell output to locate the relevant part, select just the relevant part & re-run it using the “Run selection” button.



In the Azure portal check the Deployment blade of the relevant RG to see if the missing component is being deployed.



*This page intentionally blank*