Microsoft

SQL Server to Azure Synapse Migration Process, Modules and Scripts

Prepared by

Gaiye “Gail” Zhou

Contributors

Faisal Malik, Aruna Dadi

Azure Data & AI Americas

Version: Draft V1.0

July 2021

Contents

Disclaimer

Theis document was developed in consultation and collaboration with Microsoft Corporation technical architects. Because Microsoft must respond to changing market conditions, this document should not be interpreted as an invitation to contract or a commitment on the part of Microsoft. Microsoft has provided high-level guidance in this document with the understanding that MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THE INFORMATION CONTAINED HEREIN. This document is provided "as-is". Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. Some examples depicted herein are provided for illustration only and are fictitious. No real association or connection is intended or should be inferred. This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

1 Design Principles and Programming Styles

1.1 Overview

What does the Scripts do?

- ✓ Translate SQL Server Table DDLs into Azure Synapse DDLs
- ✓ Execute Translated Table DDLs in Azure Synapse– to migrate Tables into Azure Synapse
- ✓ Genertate Polybase Export T-SQL Scripts
- ✓ Execute Polybase Export T-SQL Scripts – to export data directly into Azure Storage from SQL Server
- ✓ Upload Exported Data into Azure Data Lake Store (or Blob Storge) if using BCP export
- ✓ Generate T-SQL Copy Scripts
- ✓ Execute T-SQL Copy Scripts - to Import Data into Azure Synapse from Azure Storage

Why do we need these Scripts when we already have Azure Synapse Pathway?

These Scripts are complementary to Azure Synapse Pathway (ASP). ASP does not perform data migration today. We designed and implemented 6 modules to complete the end-to-end tasks of tables migration and data migration (using BCP or Polybase Export). Please check the latest release of Azure Synapse Pathway for more advanced SQL Server code translation capabilities.

We recommend using Azure Synapse Pathway for all Code Translation Tasks.
You can use scripts described in this document (Module 5_RunSqlFilesInFolder) to execute all translated code by ASP or other methods. Please check the newest release of Azure Synapse Pathway so you can use the best available functions. sql-docs/azure-synapse-pathway-overview.md at live · MicrosoftDocs/sql-docs (github.com)

In addition, Module 3, 4, 5 are reusable for other types of migrations, for example, Netezza or Teradata or Exadata or Oracle to Azure Synapse migrations. After the code is translated, and data is exported out of source systems, the rest of the tasks are the same. Therefore module 3-5 can be utilized for any of those migrations.

1.2 Design Principles

We adhere to below design principles:

- ✓ **Modular**: Modules that run *independently* but can use output from other modules
- ✓ **Consistent**: All driver programs are written in PowerShell Scripts.
- ✓ **Simple**: Only one PowerShell Program (scripts) for each module.
- ✓ **Configurable**: Each module has easy way to config parameters.
- ✓ **User Friendly**: Users are prompted for config file name only. Well documented configuration parameters. Strong error handling mechanism to provide friendly messages. Sample config files are provided. Manual work is minimalized. Utilities are provided to generate config file(s) that involves list of tables.
- ✓ **Reusable**: Module 1 is for code translation; Module 2 is for SQL Server data export. Only these two modules are specific to SQL Server. *Modules 3-5 are reusable for any migration into Azure Synapse, the sources can be: Netezza, Teradata, Exadata, Oracle, DB2, Snowflake, Redshift, Google Big Query, etc., once the code is translated and data is exported out of source system.*
- ✓ **Extensible to Leverage Azure Synapse Pathway:** Current version only translates tables. When Azure Synapse Pathway releases the full version of code translation from SQL server to Azure Synapse, the translated code can be readily utilized by this process. You just need execute translated code using Module 5, "5_*RunSqlFilesInFolder*". You only need to specify the folder where the Translated Code is stored.

1.3 Best Practices in Programming Styles

We adopt below best practice as our programming style:

- ✓ Being Protective - No hardcoded security information anywhere. We will ask you to provide security such as username and password.
- ✓ Being Assertive – We will ask you to specify location of needed software such as **BCP** or **Azcopy**.
- ✓ Being Friendly - We prompt you for your information with sample values. We also provide utilities to generate configuration files.

2 Overview – Modules and Scripts

There are six modules that contain PowerShell Scripts and T-SQL Scripts designed to accomplish key task(s) that are relevant to SQL server to Azure Synapse migration.

The six modules are summarized as below:

1_TranslateTableDDLs: Translate SQL objects (DDLs) from source system format to Azure Synapse format. The output is stored as .sql files in specified file folder (configurable).

2_ExportSourceDataWithBCP: Export SQL Server Tables into data files stored in predefined structure and format (.csv or .txt).

2A_GeneragePolybaseExportScripts: Generate Polybase Export T-SQL Script for each table. Polybase export set up examples are provided in subfolder "Utilities" inside this module.

3_LoadDataIntoAzureStorage: Load exported data files into specified container in Azure Storage (Blob Storage or Azure Data Lake Store).

4_GenerateCopyIntoScripts: Generate "COPY Into" T-SQL Scripts that will move data from Azure Storage into Azure Synapse SQL Pool tables, once executed.

5_RunSqlFilesInFolder: Run all T-SQL Scripts defined in .sql files stored in a specified file folder. The T-SQL Scripts can be DDL, DML, Data Movement Scripts (such as Copy Into scripts or Polybase Export Scripts), or any other scripts such as create/update statistics or indexes. In fact, this module is designed to run any SQL scripts in a folder.

The organization of the modules, output folder, and documentations is illustrated in Figure 1. Modules are stored in the "modules" directory. "output" is designed to store output of the modules. You can use different folder as you wish. It is configurable.

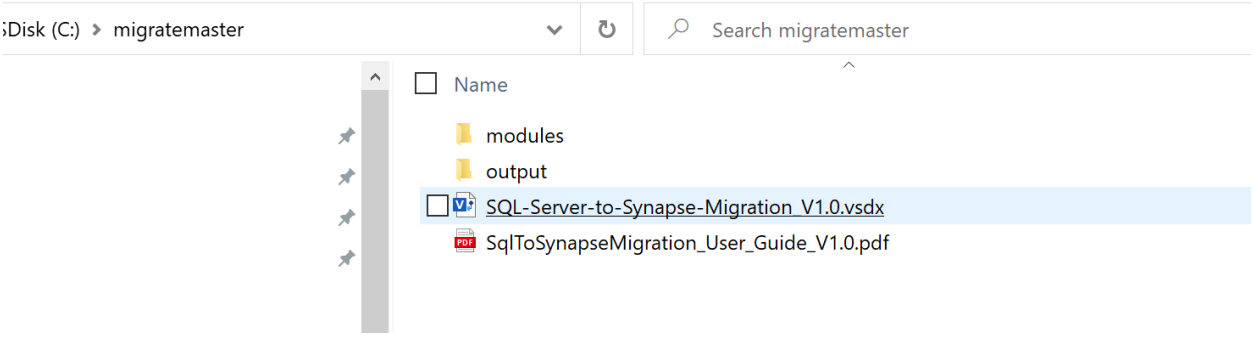

Figure 1 File Structures and Documentations

Inside the "modules" directory, there are 6 modules, as illustrated in Figure 2.

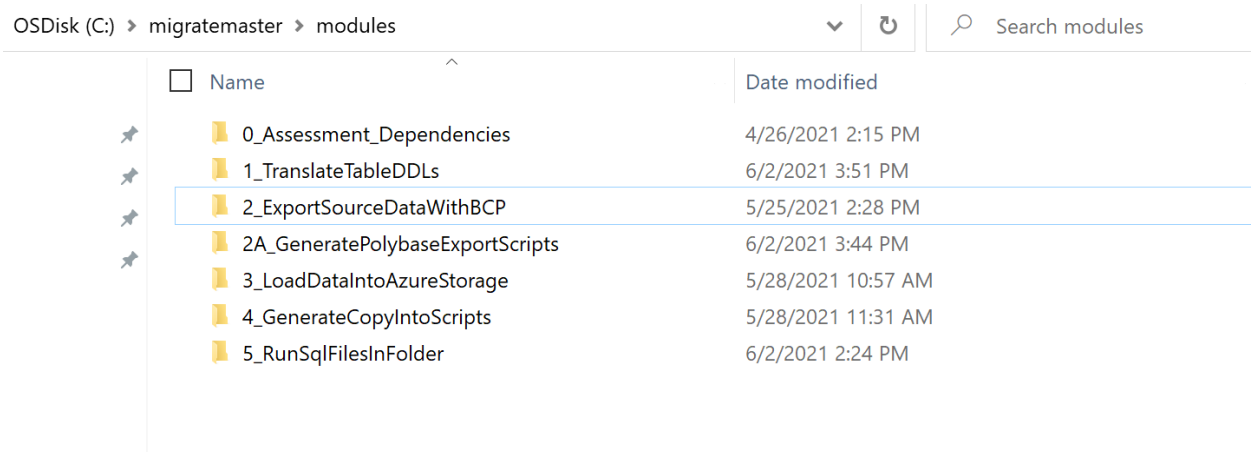

Figure 2 Organization of Modules, output folder and documentations

3 What Do I Need to Do to Run the PowerShell Scripts?

This section provides instructions to download and set up the Scripts.

3.1 Download the Code from GitHub

Go to the public repository site:

[microsoft/AzureSynapseScriptsAndAccelerators (github.com)]

Download the repository, as illustrated in Figure 3, you will receive a zip file in your own download directory.

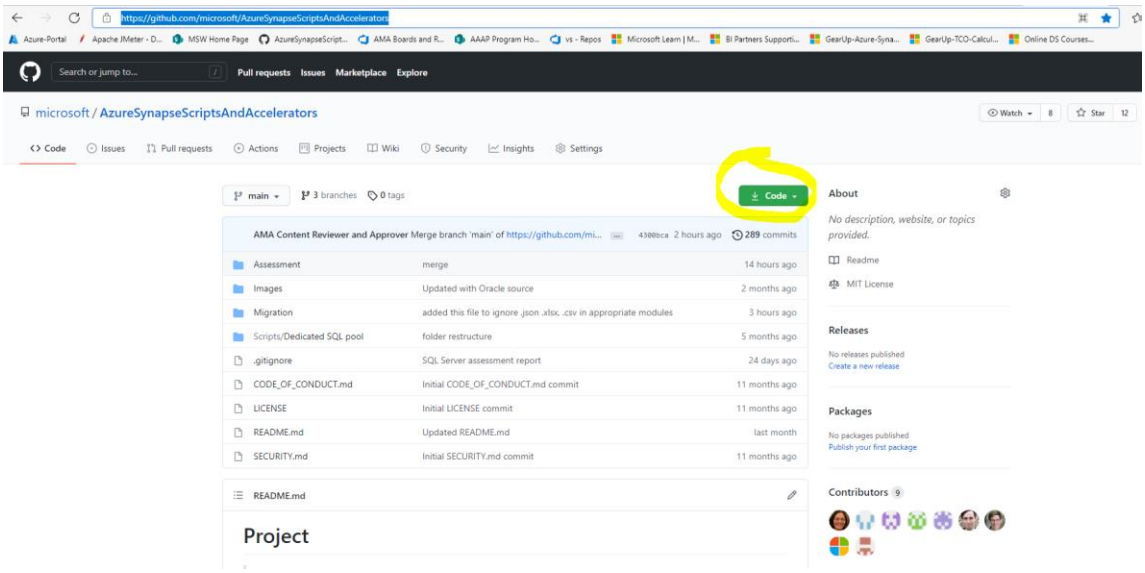

Figure 3 Download the Code from the Public Repository

Unzip the file, and you will find this folder below. Inside the SQLServer Folder, you will have all the code, including this document, as illustrated in Figure 4.

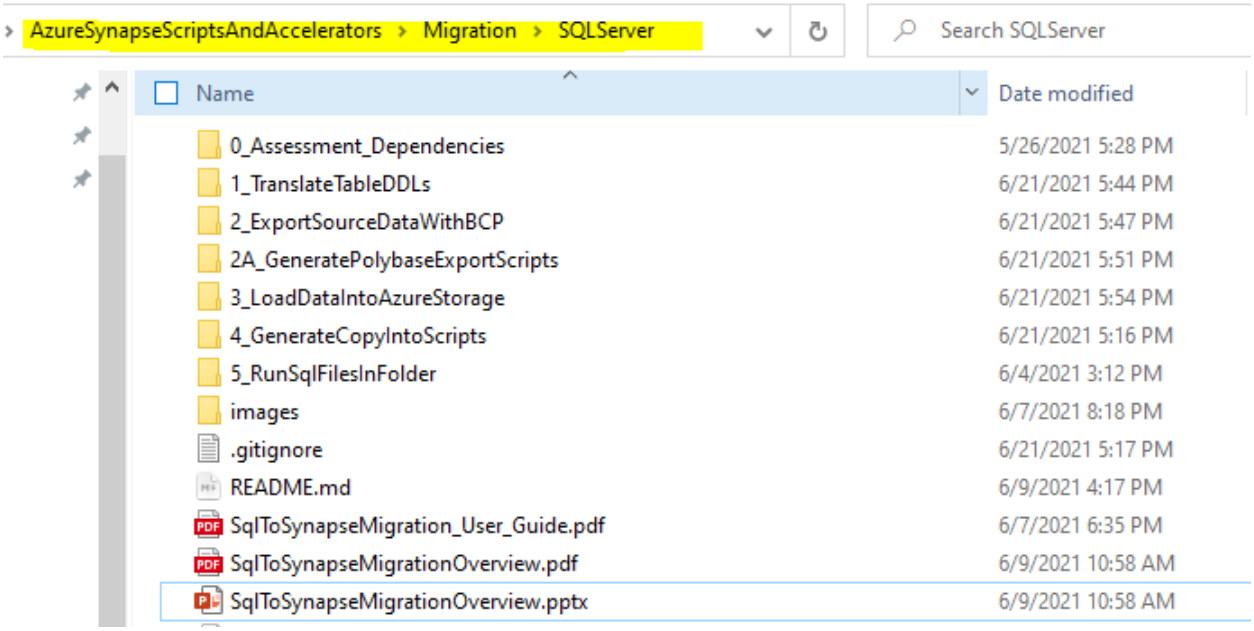

Figure 4 SQL Server to Synapse Migration Code Location

3.2 Set up the Necessary Tools/Utilities

Choose One of the Environments:

1. Windows PowerShell ISE (preferred)
2. Visual Studio Code (with PowerShell Extension Installed)

PowerShell Modules required:

(1) Install-Module (Import-module) -name SqlServer

(2) Install-Module (Import-module) -name ImportExcel

(3) Install-Module (Import-module) -Name Az –AllowClobber

PowerShell Permissions (Permissions may be denied if the Scripts are from GitHub or Email):

Use one of the options to set Powershell permissions (examples):

Set-ExecutionPolicy Unrestricted -Scope CurrentUser

Unblock-File -Path C:\migratemaster\modules\1_TranslateTableDDLs\TranslateTables.ps1

Download and Install AzCopy (Only if you will be using BCP Export Method). This task can be skipped if you will be using other methods to upload data into Azure Storge such as Azure Data Box Gateway or Azure Data Explore).

Copy or move data to Azure Storage by using AzCopy v10 | Microsoft Docs

4 How Migration Tasks are Modularized

Migration process is illustrated in Figure 5 for BCP export option, and Figure 6 for Polybase Export Option.

The numbers specify the module numbers, not the execution sequence.

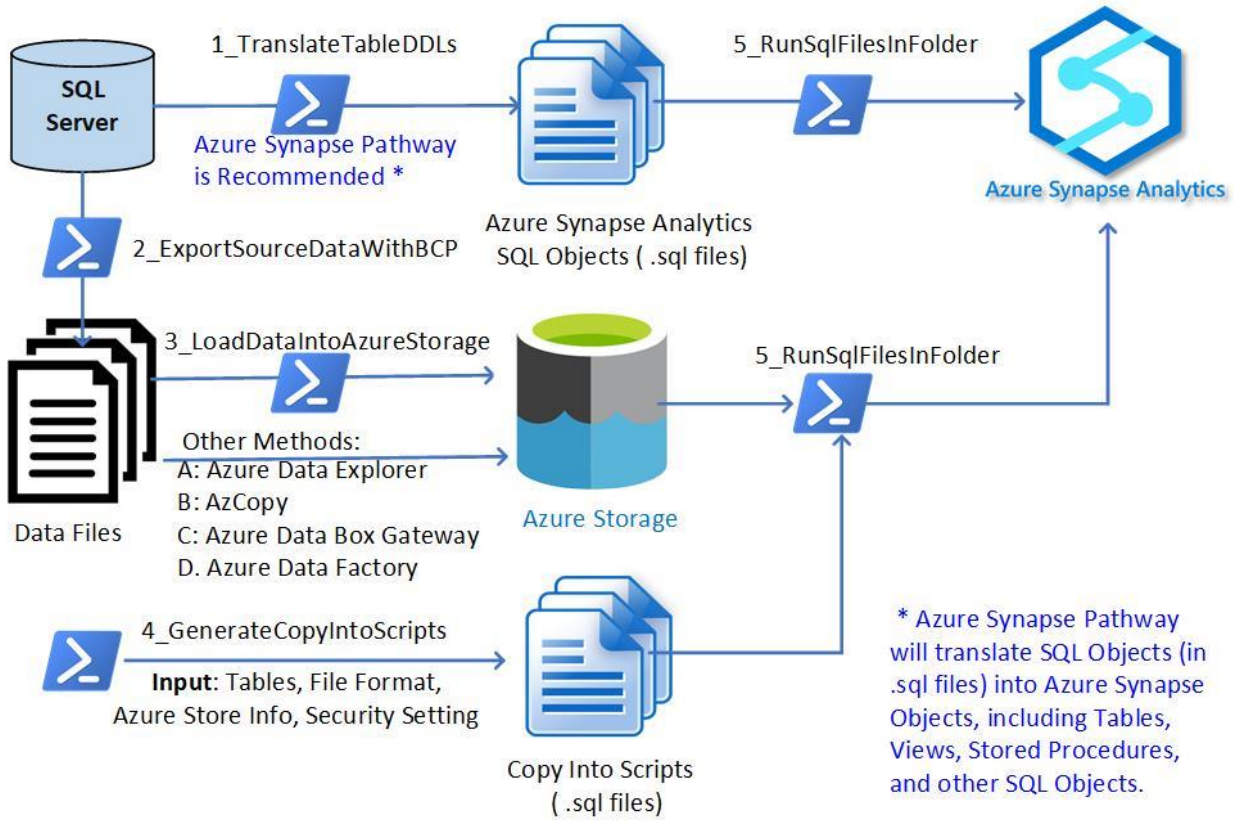

Figure 5 Migration Modules are Applied for the Migration Process (BCP Export Option)

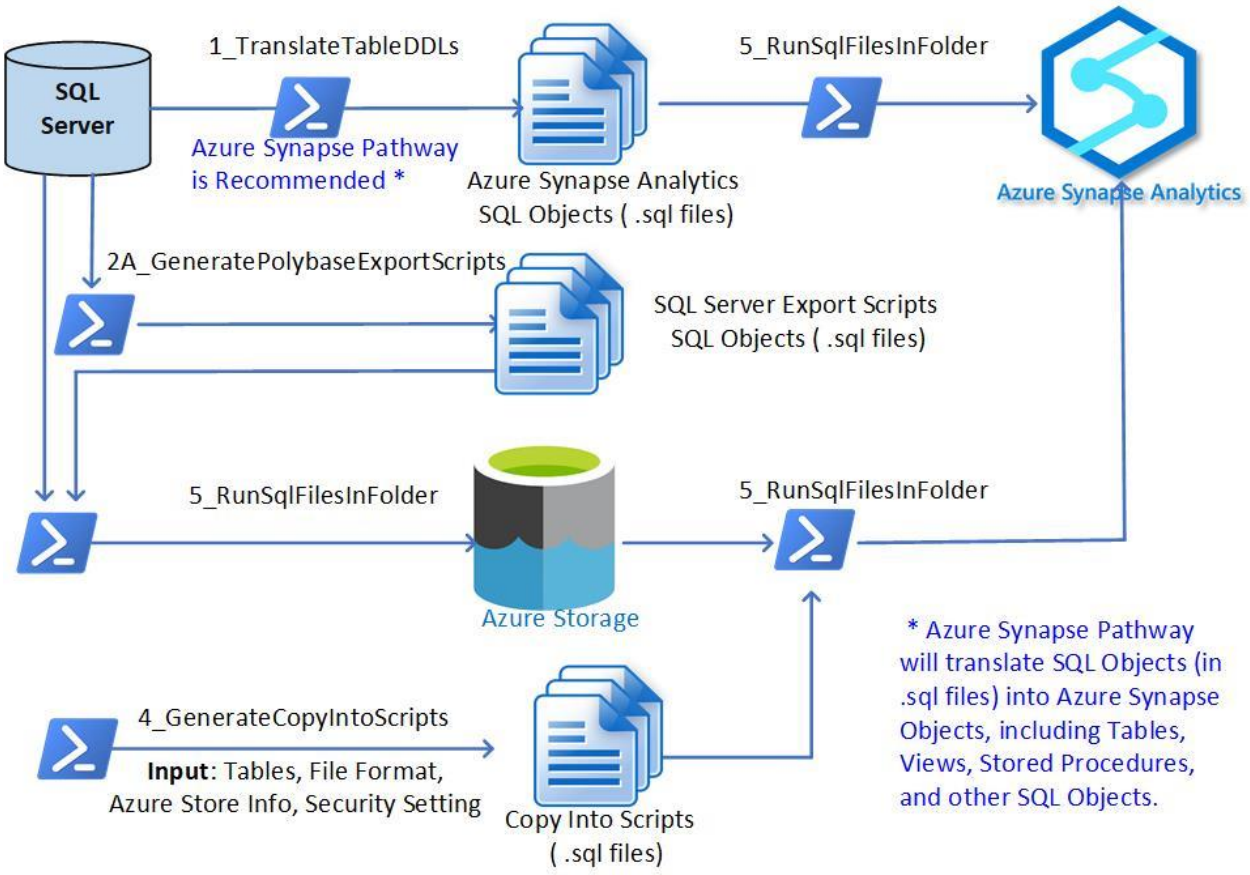

Figure 6 Migration Modules are Applied for the Migration Process (Polybase Export Option)

Source System Tasks (SQL Server):

- Use **1_TranslateTableDDLs** to translate and save results into .sql files
- Use **2_ExportSourceDataWithBCP** to export source data and save results into local storage using BCP.
- Use **2A_GeneratePolybaseExportScripts** to generate SQL Server Polybase Export T-SQL Scripts.

Data Movement Tasks:

- Use **3_LoadDataIntoAzureStorage** to load exported data into Azure Storage. Optionally you can use AzCopy or Azure Data Box Gateway to complete the task. Azure Data Box Gateway is a good choice if the volume of data is very large, for example, 50TB+.
- Use **4_GenerateCopyIntoScripts** to prepare "Copy Into" T-SQL Scripts
- Use **5_RunSqlFilesInFolder** to execute "Copy Into" T-SQL Scripts or Polybase Export T-SQL Scripts

Target System Tasks (Azure Synapse):

- Use **5_RunSqlFilesInFolder** to execute Azure Synapse T-SQL Scripts prepared by **1_TranslateMetadata,** to create meta data (tables) in Azure Synapse.
- Use **5_RunSqlFilesInFolder** to execute "Copy Into" T-SQL Scripts prepared by 4**_GenerateCopyIntoScripts,** to move data from Azure Storage into Synapse

5 Step-by-Step Migration Guide (with BCP Export)

Although some of the module can be run in parallel, we have design a simple sequence for your Migration Journey.

5.1 Step 1 - Code (DDLs) Migration

Tables migration process is illustrated in Figure 7.

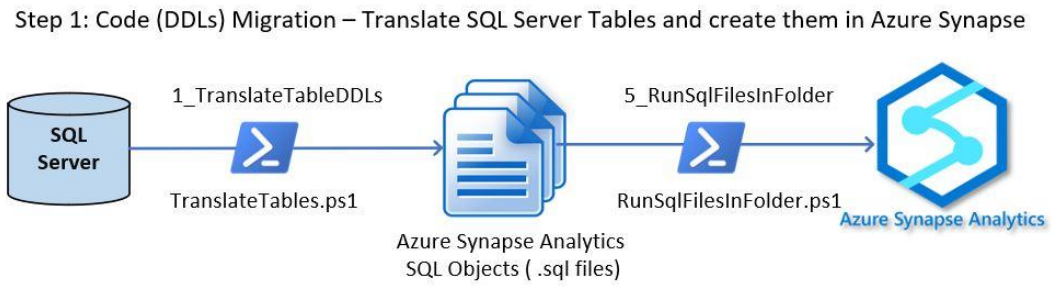

Figure 7 Step 1: Code (DDLs) Migration

Task 1
Execute PowerShell Scripts "**TranslateTables.ps1**" (Inside folder 1_TranslateTableDDLs folder)
Output: Azure Synapse Create Table Statement (DDLs) stored in .sql format.
Config Files needed (Samples are provided):
SourceToTargetTablesConfig.xlsx
translate_config.json
Note 1: Need to access SQL Server for this. "db_datareader" role permission is needed

Note 2: Look for T-SQL Scripts "**GenerateSourceToTargetConfig.sql**" in the Utilities Subfolder to create starter SourceToTargetTablesConfig.xlsx.

Task 2
Execute PowerShell Scripts"**RunSqlFilesInFolder.ps1**" (Inside folder 5_RunSqlFilsInFolder)
Input: Azure Synapse Table DDL files (.sql) stored in one file folder, which were generated by Task 1.
Output: Timestamped log files in the "Log" subfolder where you run this PowerShell Scripts.
Results: Tables will be created in Azure Synapse Dedicated SQL Pool.
Config File(s) needed (Samples are provided):
sql_synapse.json

Note: Need "Create Schema" and "Create Table Permission" in Azure Synapse SQL Pool.

5.2 Step 2 - Export Data out of SQL Server Using BCP

Exporting SQL Server data process is illustrated in Figure 8.

Figure 8 Step 2: Export SQL Server Data using BCP

Task
Execute PowerShell Scripts "**ExportSourceData.ps1**" (Inside 2_ExportSourceDataWithBCP).
Output: Data files in .csv format are produced and saved into local storage.
Config Files needed (Sample(s) are provided):
ExportTablesConfig.csv
sql_bcp.json
Note 1: Need to access SQL Server for this. "db_datareader" role permission is needed
Note 2: You need **bcp utility (bcp.exe)** for this. If you have SQL server installed, you may find it in this location:
C:\Program Files\Microsoft SQL Server\Client SDK\ODBC\130\Tools\Binn
If you are not able to find bcp.exe, you can download a copy from

https://docs.microsoft.com/en-us/sql/tools/bcp-utility?view=sql-server-ver15

5.3 Step 3 – Upload Data into Azure Data Lake Store or Blob Storage

Uploading data into Azure Storage process is illustrated in Figure 9.

Figure 9 Step 3: Upload Data into Azure Storage

Task (If using 3_LoadDataIntoAzureStorage)
Execute PowerShell Scripts "**LoadDataIntoAzureStorage.ps1**" (inside folder 3_LoadDataIntoAzureStorage)
Output: None
Results: Data in Data Files are uploaded to Azure Storage (Data Lake Store or Blob Storage)
Config File(s) needed (Samples are provided):
sql_bcp.json

Note: The user needs to have the permission of "Storage Blob Data Contributor"

5.4 Step 4: Generate COPY T-SQL Scripts

The process of generating COPY T-SQL Scripts is illustrated in Figure 10.

After executing each of the generated T-SQL Copy Script in the format of .sql files, the corresponding data will be imported into Azure Synapse. The execution step is carried out in next step, Step 5.

This step only generates T-SQL Scripts, it does not execute any T-SQL Scripts.

Figure 10 Step 4: Generate T-SQL COPY Into T-SQL Scripts

Task
Execute PowerShell Scripts "**GenerateCopyIntoScripts.ps1**" (inside folder 4_GenerateCopyIntoScripts)
Output: T-SQL COPY Script for Each Table (in the form of .sql files)
Config Files needed (samples are provided)

(1) csv_mi.json (recommended) or csv_key.json (this requires Storage Account Key, not recommended but can be used for quick testing)
(2) TablesConfig.csv: This is a table list with information needed for each T-SQL Copy script.

Note 1: If using csv_mi.json file, and the Azure Storage was created independently (not as part of the Azure Synapse Workspace Creation), you will need to set up Azure Synapse Workspace as a managed instance, for the Scripts to work. See instruction in top of the "GenerateCopyIntoScripts.ps1". In addition, you can find sample PowerShell script "**SetManagedIdentity.ps1**" (inside subfolder Utilities) to set up Managed Instance.

Note 2: Sample configuration files are provided for data file formats other than CSV: parquet and orc. However, these file formats are not tested.

5.5 Step 5: Import Data into Azure Synapse (SQL Pool)

The process of generating COPY T-SQL Scripts is illustrated in Figure 11. You will utilize the 5_RunSqlFilesInfolder to execute all the COPY T-SQL Scripts generated.

After this step, the data stored in Azure Storage will be imported into Azure Synapse SQL pool. The program also produces a log file, itemizing the results of each T-SQL Scripts. If there is errors, the errors will be found in the log file.

Figure 11 Step 5: Import Data into Azure Synapse SQL Pool

Task: Execute PowerShell Scripts"**RunSqlFilesInFolder.ps1**" (Inside folder 5_RunSqlFilsInFolder)
Input: T-SQL script (Copy Into) generated by **GenerateCopyIntoScripts.ps1** (Inside Module 4_GenerateCopyIntoScripts) stored in one file folder.
Output: Timestamped log files in the "Log" subfolder where you run this PowerShell Scripts.
Results: Data is imported to Azure Synapse Tables (Dedicated SQL Pool).
Config File(s) needed: sql_synapse.json

Note: Need "Write Data" Permission" in Azure Synapse SQL Pool.

6 Step-by-Step Migration Guide (with Polybase Export)

6.1 Step 1 - Code (DDLs) Migration

The work to be done in this step is exactly the same as Step 1 – Code (DDLS) Migration with BCP Export. Please refer to this Section: 5.1, Step 1 - Code (DDLs) Migration.

6.2 Step 2A – Generate Polybase Export T-SQL Scripts

In this step, you will generate the Polybase Export T-SQL Scripts. Please note this step just generates the code, not execute it. The process of generating Polybase Export T-SQL Scripts is illustrated in Figure 12.

Figure 12 Step 2A Generate Polybase Export T-SQL Scripts

Task
Execute PowerShell Scripts "**GenerateExportTablesScripts.ps1**" (Inside 2A_GeneratePolybaseExportScripts).
Output: Polybase Export T-SQL Scripts (Create External Table) for each table in the format of .sql.
Config Files needed (Sample(s) are provided):
ExportTablesConfig.csv
export_tables_config.json
Note 1: Need SQL Server Permissions for create schema, create/drop table, read data. **Note 2:** You will need to set up Polybase export functions in SQL Server. Sample T-SQL Scripts are provided in the subfolder Utilities.

6.3 Step 3A – Export SQL Server Tables Data to Azure Storage

The process of Exporting SQL Server Tables Data into Azure Storage is illustrated in Figure 13. You will utilize the 5_RunSqlFilesInfolder to execute all the Polybase Export T-SQL Scripts generated in Step 2A.

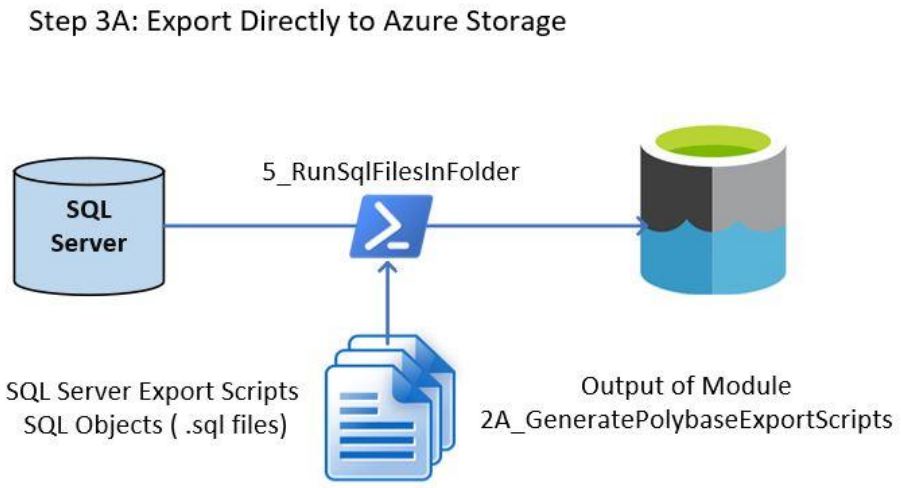

Figure 13 Step 3 (3A) Export SQL Server Tables Data into Azure Storage

Task: Execute PowerShell Scripts"**RunSqlFilesInFolder.ps1**" (Inside folder 5_RunSqlFilsInFolder)
Input: T-SQL script (Polybase Export) generated by **GenerateExportTablesScripts.ps1** (Inside Module 2A_GeneratePolybaseExportScripts), stored in one file folder.
Output: Timestamped log files in the "Log" subfolder where you run this PowerShell Scripts.
Results: Data is exported to Azure Storage from SQL Server Tables
Config File(s) needed: sql_sql.json
Note:

(1) Need Blob Storage Contributor Role in Azure Storage
(2) Permissions from SQL Server for Create Schema, Create Table, Read Data.
(3) Polybase Export is set up in SQL Server. External Data Source and File Format are created. See samples inside subfolder Utilities.

6.4 Step 4: Generate COPY T-SQL Scripts

The work to be done in this step is exactly the same as Step 4 using BCP export method. Please refer to Section 5.4, Step 4: Generate COPY T-SQL Scripts.

6.5 Step 5: Import Data into Azure Synapse (SQL Pool)

The work to be done in this step is exactly the same as Step 5 using BCP export method. Please refer Section: 5.5, Step 5: Import Data into Azure Synapse (SQL Pool).