**PowerShell Script to Move SQL Server Logins/Users/Roles to Azure**

Version 1.94 – July 17, 2019

The motivation behind writing this script was to fill a temporary gap in migration tooling and enable organizations looking to move SQL Servers from on premises to SQL Server running as Platform as a Service (PaaS) in Azure to complete the migration. The script is written in PowerShell and generates a SQL script, it does not apply the commands on the target SQL Server environment. We recommend that you use shipping products, if possible, to migrate security information, since permissions in SQL Server can be incredibly complex. DMA does have the ability to transfer security information and does a full dependency graph to ensure permissions are transferred properly – which this script does not do.

Migrating Schema and Data

The script does not perform schema or data movement, for that we assume that you will be using Data Migration Assistant (DMA), Data Migration Services (DMS), Azure Data Factory (ADF), native backup/restore (Azure SQL MI only), bacpac files, or some other ETL tool (i.e. SSIS) or custom scripts.

This script allows you to generate a TSQL script that can be applied to the target SQL Server environment to transfer logins, database users, roles and permissions. Permissions in SQL Server can be very complex, and this script is designed to only deal with relatively simple security models. You need to carefully review the generated script output before applying it to the target system. Note that the script may script roles and permissions that are not available on the target platform and **it is** **required that the script user carefully review the output before applying the generated script**.

Logins and Users in Azure

In an on premises SQL Server environment, you have the choice of either SQL Server native logins or Windows Active Directory (AD) logins. When moving to the Azure cloud, you continue to have the choice of using SQL native logins, but Windows AD logins are not available. To get the single sign-on, integrated security experience, you must use Azure Active Directory (AAD) accounts. The issue that arises is that the format of AD and AAD accounts is different. AD accounts have the form; domain\user, while AAD accounts have the form; [user@domain.com](mailto:user@domain.com), where the domains may not be the same, depending on the company DNS name and AD forest complexity. Windows AD has [user@domain.com](mailto:user@domain.com) available in the User Principal Name (UPN) property, but if the domains match or there is a simple mapping from the AD domain to the AAD domain, there is no reason to do user lookups in AD (which can be quite slow). Note that if you are going to use the AD lookup feature, you will need the [Remote Server Administrator Tool](https://www.microsoft.com/en-us/download/details.aspx?id=45520) enabled or installed.

Using the Script

The script has a block of variables at the beginning that the user must set or review before running the script. Connection strings to both the source and the target database(s) are required and the user provides these by setting the **$SrcConStr** and **$TargConStr** variables. The **$TargConStr** has a {0} substring that is used to substitute in the master or the Azure SQL DB/DW database name to connect to that you specify in the **$TargDatabase** variable. For Azure SQL MI, the **$TargDatabase** can be set to master. The **$UseADLookup** variable controls whether the script will do the AD UPN lookup as described above or just a simple substitution using the **$Domain** variable to generate the AAD login. The **$DoSQLLogins** variable controls whether SQL Server logins are included in the output or ignored.

Script Output

Depending on the target type (i.e. Azure SQL DB, Azure SQL MI or Azure SQL DW), the script can generate different output. In the case of Azure SQL DB and DW, creating AAD logins and associated database users is not supported. Instead, AAD users are created at the database level. For Azure SQL MI, the security model is very similar to the on-premises SQL Server, with server level logins and database users.

When scripting for Azure SQL MI, if the target database was restored from a backup from the on premises SQL Server, the database may contain existing users mapped to the old AD logins, so the script generates statements to drop the existing database user before creating the user linked to the new AAD (or SQL) login. Note that this drop will fail if the user owns any schema object, so there is now code in the latest version to change the ownership for a few of the typical types of objects owned by a user (note that only role, schemas, types and objects in sys.objects (i.e. tables, view, procs, functions) are handled). The script moves the permissions to either dbo or sa (specifically for dbo user ownership).

Note that both server custom roles and database roles are scripted in the current version, but server roles are not available on Azure SQL DB/DW and the fixed roles and permissions might be different or not available on all target platforms. The roles, role membership, role nesting and permissions must be carefully reviewed to ensure permissions granted are what is expected and required.

The script also generates statements for moving database permissions for both logins/users and database roles at the schema, object and column levels, to the target platform. Permissions on SQL Server can be very complicated, so the script has only a partial implementation of moving all possible permissions.

Limitations

Note that in the current version, Application roles are not scripted.

The script does not detect database users in source database(s) that are not properly linked to server logins and does not clean up such users in the target database. If these users own objects in the source database, the script may not be able to provide proper AAD user substitution for these objects (i.e. database role owned by mydomain\mitch, but there is no login for mydomain\mitch – the create role statement will be “create role myrole authorization [mydomain\mitch]”, because there is no associated AAD login).

Running the Script

To run the script, use the PowerShell command line or PowerShell ISE. The statements generated will always appear as console output, whether or not they are executed on the target system.

Feedback and Support

Please send any issues or feedback to the author at; [mitch.van.huuksloot@microsoft.com](mailto:mitch.van.huuksloot@microsoft.com)