Microsoft

Backup & Restore SQL Server to Azure VM

Prepared by

DM Jumpstart Engineering Team (askdmjfordmtools@microsoft.com)

Disclaimer

The High-Level Architecture, Migration Dispositions and guidelines in this document is 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 generic 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.

Note: The detail provided in this document has been harvested as part of a customer engagement sponsored through the Azure Data Services Jumpstart Program.

Table of Contents

1 Executive Summary

1.1 Objective

Azure Database Migration Service (DMS) is the recommended Azure service to migrate heterogenous database workload from On-Premise or 3rd party cloud providers such as Google cloud and AWS to Azure. Azure Database Migration Service enables seamless migrations from multiple database sources to Azure Data platforms with minimal downtime. However, if customer wants to migrate to Azure VM, then Azure DMS isn't a viable solution. Alternatives are either Azure Site Recovery (ASR) or backup & restore.

Azure Site Recovery (ASR) replicates workloads running on physical and virtual machines (VMs) from a primary **site** to a secondary location. ASR can protect SQL Server as summarized in the below table:

Scenario	To a Secondary Site	To Azure
Hyper-V	Yes	Yes
VMWare	Yes	Yes
Physical Server	Yes	Yes
Azure	NA	Yes

Supported SQL Server versions are:

- SQL Server 2016 Enterprise and Standard
- SQL Server 2014 Enterprise and Standard
- SQL Server 2012 Enterprise and Standard
- SQL Server 2008 R2 Enterprise and Standard

ARS is a great solution to lift & shift on-premise SQL server workload to Azure. However, if you are planning to upgrade your sql server or consolidate your databases on a larger VM during the migration, then ASR isn't the right solution.

Other option to move your SQL Server databases from on-premise to Azure VM is backup & recovery which will allow you to upgrade your databases. The challenge with backup & recovery is scale. If you have 100s or 1000s of databases to migrate, then backup & recovery can be labor intensive.

1.2 Migrate SQL Server to Azure SQL VM

This document outlines steps requited to automate the backup & restore on-premise databases to Azure SQL VM by using custom t-sql scripts. The process outlined here is semi-automated but it provides flexibility consolidate on a larger VM or migrate source databases across multiple VMs.

Step-1 – Identify database(s) to Restore – Identify list of databases you would like restore. Either you can hardcord these in the vmRestoreFull.sql script which is the current process. However, if you are planning to use this process to restore large number of databases, then hardcoding these values isn't realistic.

Step-2. Copy backup files to Azure Blob Storage using AzCopy – You can either use either command line option or configure AzCopy in Azure Storage Explorer which is a Preview Feature. If your source SQL Server is 2012 or above, then configure backups directly in Azure Blob storage using "SQL Server Backup to URL".

Option-1 – AzCopy from Command Prompt

- Generate SAS (Shared Access Signature) Token from Azure Portal
- Run AzCopy from source location command prompt

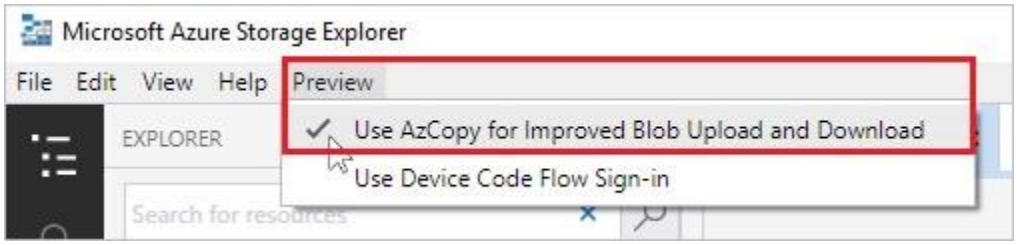
```
AzCopy cp

"C:\local\path\*.ext"

https://sqlbackups.blob.core.windows.net/sql/<<SAS Token>>"
```

Option-2 – AzCopy in Storage Explorer

- Enable AzCopy in Azure Storage Explorer (Preview Feature)



Step-3. Generate Database Restore Command – Next step in the process is to generate database restore command against the database(s) you want to restore on your Azure SQL VM(s) which you have already provisioned with SQL Server. High-level automation restore logic includes ….

- Input list of databases – Currently dbname is hardcoded in the script
- For each database …
  - ✓ Identify associated logical and physical file path & file name

```
SELECT d.name DatabaseName, f.name LogicalName,
    f.physical_name AS PhysicalName
FROM sys.master_files f, sys.databases
```

  - ✓ Identify full and transaction log backups

```
SELECT msdb.dbo.backupmediafamily.physical_device_name
FROM msdb.dbo.backupmediafamily, msdb.dbo.backupset
```

  - ✓ Build "restore database" and "move datafile" command
  - ✓ Write restore command to a variable hardcoded as "C:\temp\vmFullRestore.sql"

- Generate SAS (Shared Access Signature) Token from Azure Portal
- Create SQL Server Credential

  CREATE CREDENTIAL [https://vmsqlbackups.blob.core.windows.net/sql] – Name must match the container path

  WITH IDENTITY='SHARED ACCESS SIGNATURE', -- this is a mandatory string and do not change it.

  SECRET = 'SAS Token generated in the previous step'
- Copy vmFullRestore.sql to the target server or execute remotely from SQL Server Management Studio (SSMS)

The scripts documented here tested in a POC environment and will require further enhancement or changes to accommodate your specific needs.

Step-4 – Execute Database Restore – This is the last step in restore process. The vmRestoreFull.sql script we built in Step-3 includes restore database commands for all the databases identified as input in Step-2. You can either copy this script to the target server and execute it locally or remotely from the local server. The script will restore each database one by one. Alternatively, you can group these databases in separate files to execute in parallel on the same VM or different VMs.

1.3 Backup & Restore Architecture/Workflow

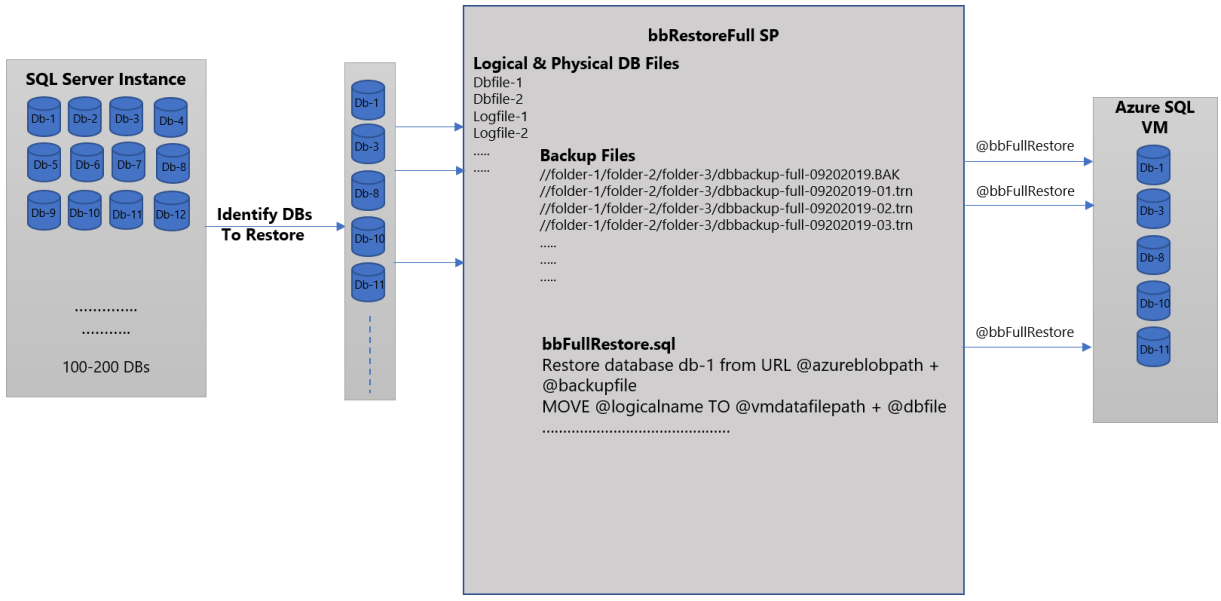


Fig-1 – High-level Backup & Restore Automation Architecture/Workflow

2 Appendix

AzCopy is command line utility which you can use to copy backups to Azure Blob Storage. Attached is the script which you can customize to your environment.

-- Update source location and blob url where you want to copy backupfiles.

-- Generate SAS token from Azure Portal

AzCopy cp C:\temp*.csv "https://sqlbackups.blob.core.windows.net/sql/<<SAS Token>>"

spBuildRestoreDB.sql is the primary script which will build restore command for each database you input. The script will generate vmFullRestore.sql and will be stored, by default, in C:\temp\

```
----------------------------------------------------------------------------------------
-- Disclaimer
-- Any Sample Code, High Level Architecture, Migration Dispositions and guidelines in this
script is developed in consultation and collaboration with Microsoft Corporation technical
architects.
-- This script was developed based on the information provided to Microsoft by customer and
represents current Microsoft high level architectural guidelines and practices.
-- Because Microsoft must respond to changing market conditions, this script 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 artifact with the understanding that
customer would undertake detailed design and comprehensive reviews of the overall solution
before the solution
-- would be implemented/delivered.  Microsoft is not responsible for the final implementation
undertaken by Customer.
-- MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, TO <<Customer>> OR <<Partner>> WITH
RESPECT TO THE INFORMATION CONTAINED HEREIN.
----------------------------------------------------------------------------------------
--
--
-- The spBuildRestoreDB SP will generate restore script to restore the database using the
lastest FULL backup in last 24 hrs.
-- Last update - 5/31/2019
----------------------------------------------------------------------------------------
-- Enable advanced options and is needed to configure xp_cmdshell

exec master..sp_configure 'xp_cmdshell', '1'
RECONFIGURE WITH OVERRIDE
----------------------------------------------------------------------------------------

Drop Procedure IF EXISTS spBuildRestoreDB -- Doesnt work in SQL 2018. Drop procedure manually
Go
Create Procedure spBuildRestoreDB
As

Set Quoted_Identifier off
Set NoCount On
```

```
-------------------------------------------------------------------------------------------
-- Variables defined in this section are used to build restore commend dynamically and write
in the Input file which
-- is defined in the variable @restorefullfile
DECLARE @cmd1 varchar(2048)
Declare @cmd2 varchar(2048)
DECLARE @restoreheadermsg VARCHAR(2048)
DECLARE @restoremsg VARCHAR(2048)
DECLARE @restorefullfile     VARCHAR(128) = 'C:\Temp\vmFullRestore.SQL'
-- exec master..xp_cmdshell 'del c:\Temp\vmFullRestore.sql'
-- exec master..xp_cmdshell 'type c:\Temp\vmFullRestore.sql'
-------------------------------------------------------------------------------------------

DECLARE @userdb                     VARCHAR(50)    -- Source Database Name
DECLARE @logicalname         VARCHAR(250)   -- Logical Datafile Name
DECLARE @physicalname        VARCHAR(250)   -- Physical Datafile Name
DECLARE @databasename        VARCHAR(50)          -- Source database name
DECLARE @dbfilename                 VARCHAR(250)   -- Source Server dbfile name
DECLARE @dbfilecount         INT    -- database file count to build the MOVE statement
DECLARE @dbfilerevcount             INT                           -- Reverse database file
count to identify the total number of files in the database
DECLARE @maxfilecount        INT                           -- Total number of datafiles in the
database. This variable will set by 1st value in Reverse file count varible.  DB Files count
DECLARE @server                           VARCHAR(50)          -- Source Server name
DECLARE @backuptype                 VARCHAR(15)          -- Backup type - Full, Transaction
Log
DECLARE @backupsize                 INT                  -- Size of the backup
DECLARE @backupdevice        VARCHAR(250)                 -- Full Backup Path and File Name
DECLARE @backupfile                 VARCHAR(250)         -- Backup filename to append with
the Azure Blob URL to build restore command
DECLARE @backupset                  VARCHAR(120)         -- backup set
-- -- Update this location with your blob storage where you are copy source database backups.
DECLARE @azureblobpath              VARCHAR(250) =
'https://d01sql08demotosql16rgdia.blob.core.windows.net/sqlserver08backups/'
DECLARE @vmdatafilepath             VARCHAR(250) = 'F:\Data\'    -- Folder where data files
will be copied during the database restore
DECLARE @vmlogfilepath              VARCHAR(250) = 'F:\Log\'     -- Folder where data files
will be copied during the database restore

DECLARE AllDBs_CURSOR CURSOR FOR

       -- Cursor to fetch physical and logical datafiles for each input database. Using
PARTITION and ROW_NUMBER to build the Move command and identify total number of data files.
       -- The database to be restored here is hardcoded, however, it can easily be supplied as
an input parameter or input from another CSV or database table for multiple databases.
       SELECT d.name DatabaseName, f.name LogicalName,
       f.physical_name AS PhysicalName,
       RIGHT(f.physical_name,CHARINDEX('\',REVERSE(f.physical_name))-1) as dbfile,
       ROW_NUMBER() OVER(PARTITION BY d.name ORDER BY f.name) AS dbfilecount,
       ROW_NUMBER() OVER(PARTITION BY d.name ORDER BY f.name DESC) AS dbfilerevcount
       FROM sys.master_files f
       INNER JOIN sys.databases d ON d.database_id = f.database_id
       WHERE d.name NOT IN ( 'model','tempdb', 'pubs','northwind')
       AND d.name in ('dbnameforrestore')
       ORDER BY databasename, dbfilecount
```

```
OPEN AllDBs_CURSOR
FETCH NEXT FROM AllDBs_CURSOR INTO @userdb, @logicalname, @physicalname, @dbfilename,
@dbfilecount, @dbfilerevcount
WHILE (@@FETCH_STATUS = 0) -- Outer Cursor
BEGIN
                        DECLARE RestoreFull_CURSOR CURSOR FOR
                                -- Cursor to fetch last few days backup files, currently its 10
days in the where clause.
                                SELECT
                                                RTRIM(CONVERT(CHAR(100),
SERVERPROPERTY('Servername'))) AS Server,
                                                msdb.dbo.backupset.database_name,
                                                CASE msdb..backupset.type
                                                        WHEN 'D' THEN 'Database'
                                                        WHEN 'L' THEN 'Log'
                                                END AS backup_type,
                                                msdb.dbo.backupset.backup_size,
                                                msdb.dbo.backupmediafamily.physical_device_name,

        RIGHT(msdb.dbo.backupmediafamily.physical_device_name,CHARINDEX('\',REVERSE(msdb.dbo.ba
ckupmediafamily.physical_device_name))-1) as backfile
                                        FROM msdb.dbo.backupmediafamily
                                        INNER JOIN msdb.dbo.backupset ON
msdb.dbo.backupmediafamily.media_set_id = msdb.dbo.backupset.media_set_id
                                        WHERE msdb.dbo.backupset.database_name = @userdb
                                        AND msdb.dbo.backupset.type = 'D'
                                        AND (CONVERT(datetime,
msdb.dbo.backupset.backup_start_date, 102) >= GETDATE() - 10)
                --                      AND msdb.dbo.backupmediafamily.physical_device_name like
'https://%blob.core.windows.net%'
                                        ORDER BY
                                        msdb.dbo.backupset.database_name,
                                        msdb.dbo.backupset.backup_finish_date
                        OPEN RestoreFull_CURSOR;
                        FETCH NEXT FROM RestoreFull_CURSOR INTO
@server,@databasename,@backuptype,@backupsize, @backupdevice, @backupfile;

                        WHILE @@FETCH_STATUS =0
                        BEGIN
                        IF @dbfilecount = 1
                        BEGIN
                                SET @restoreheadermsg = '-- Restore Full database scripts for
Database ' + @databasename + ' of size ' + CAST(@backupsize as VARCHAR) + ' on Server  ' +
@server + ' of type ' + @backuptype
                                SET @cmd1 = 'echo ' + @restoreheadermsg + ' >> ' +
@restorefullfile
                                EXEC master..xp_cmdshell @cmd1;

                                SET @restoremsg = 'Restore Database [' + @databasename + '] from
URL =' + '''' + @azureblobpath + @backupfile + '''' + ' WITH RECOVERY,'
                                SET @cmd2 = 'echo ' + @restoremsg + ' >> ' + @restorefullfile
                                EXEC master..xp_cmdshell @cmd2;

                                SET @maxfilecount = @dbfilerevcount;
                        END
```

```
                            -- Move data files to the specific location on target VM
                            SET @restoremsg = 'MOVE ' + '''' + @logicalname + '''' + ' TO ' +
'''' + @vmdatafilepath + @dbfilename + '''' + CASE WHEN @dbfilecount = @maxfilecount THEN ',
REPLACE, STATS = 10;' ELSE ',' END
                            SET @cmd2 = 'echo ' + @restoremsg + ' >> ' + @restorefullfile
                            EXEC master..xp_cmdshell @cmd2;
                            FETCH NEXT FROM RestoreFull_CURSOR INTO
@server,@databasename,@backuptype,@backupsize, @backupdevice, @backupfile;
                        END;
                        CLOSE RestoreFull_CURSOR;
                        DEALLOCATE RestoreFull_CURSOR;
        FETCH NEXT FROM AllDBs_CURSOR INTO @userdb, @logicalname, @physicalname, @dbfilename,
@dbfilecount, @dbfilerevcount;
END;
        CLOSE AllDBs_CURSOR
        DEALLOCATE AllDBs_CURSOR
GO
```

spBuildRestoreTL.sql is designed to build a restore command to apply the transaction logs which were generated after the Full backup. This script was tested in the demo environment but not in the POC where customer environment was quite different.

```
-----------------------------------------------------------------------------------------
-- Disclaimer
-- Any Sample Code, High Level Architecture, Migration Dispositions and guidelines in
this script is developed in consultation and collaboration with Microsoft Corporation
technical architects.
-- This script was developed based on the information provided to Microsoft by customer
and represents current Microsoft high level architectural guidelines and practices.
-- Because Microsoft must respond to changing market conditions, this script 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 artifact with the understanding
that customer would undertake detailed design and comprehensive reviews of the overall
solution before the solution
-- would be implemented/delivered.  Microsoft is not responsible for the final
implementation undertaken by customer.
-- MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, TO <<Customer>> OR <<Partner>> WITH
RESPECT TO THE INFORMATION CONTAINED HEREIN.
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-- Enable advanced options and is needed to configure xp_cmdshell

exec master..sp_configure 'xp_cmdshell', '1'
RECONFIGURE WITH OVERRIDE
-----------------------------------------------------------------------------------------
---------------------------

Drop Procedure IF EXISTS spBuildRestoreTL
Go
Create Procedure spBuildRestoreTL
As

Set Quoted_Identifier off
Set NoCount On
```

```
-----------------------------------------------------------------------------------------------------
----------------------
DECLARE @cmd1 varchar(2048)
Declare @cmd2 varchar(2048)
DECLARE @restoreheadermsg VARCHAR(2048)
DECLARE @restoremsg VARCHAR(2048)
DECLARE @restorelogfile      VARCHAR(128) = 'C:\Test\vmlogsrestore.SQL'
-- exec master..xp_cmdshell 'del c:\Test\vmlogsrestore.sql'
-- exec master..xp_cmdshell 'type c:\Test\vmlogsrestore.sql'
-----------------------------------------------------------------------------------------------------
----------------------
DECLARE @userdb                              VARCHAR(50) -- database name
DECLARE @databasename        VARCHAR(50) -- database name
DECLARE @server                              VARCHAR(50) -- server name
DECLARE @backuptype                  VARCHAR(15) -- backup type
DECLARE @backupsize                  INT                     -- backup size
DECLARE @backupdevice        VARCHAR(250)-- backup device URL
DECLARE @backupset                   VARCHAR(120)-- backup set

DECLARE AllDBs_CURSOR CURSOR FOR
   SELECT name
   FROM master..sysdatabases
   WHERE name NOT IN ( 'model','tempdb', 'pubs','northwind')
   AND (status & 32) =0         -- Do not include loading
   AND (status & 64) =0         -- Do not include loading
   AND (status & 128) =0        -- Do not include recovering
   AND (status & 256) =0        -- Do not include not recovered
   AND (status & 512) =0        -- Do not include offline
   AND (status & 32768) =0      -- Do not include emergency
   AND (status & 1073741824) =0 -- Do not include cleanly shutdown
OPEN AllDBs_CURSOR
FETCH NEXT FROM AllDBs_CURSOR INTO @userdb
WHILE (@@FETCH_STATUS = 0) -- Outer Cursor
BEGIN
                     DECLARE RestoreLogs_CURSOR CURSOR FOR
                            SELECT
                                         RTRIM(CONVERT(CHAR(100),
SERVERPROPERTY('Servername'))) AS Server,
                                         msdb.dbo.backupset.database_name,
                                         CASE msdb..backupset.type
                                                WHEN 'D' THEN 'Database'
                                                WHEN 'L' THEN 'Log'
                                         END AS backup_type,
                                         msdb.dbo.backupset.backup_size,
                                         msdb.dbo.backupmediafamily.physical_device_name
                                  FROM msdb.dbo.backupmediafamily
                                  INNER JOIN msdb.dbo.backupset ON
msdb.dbo.backupmediafamily.media_set_id = msdb.dbo.backupset.media_set_id
                                  WHERE msdb.dbo.backupset.database_name = @userdb
                                  AND msdb.dbo.backupset.type = 'L'
                                  AND (CONVERT(datetime,
msdb.dbo.backupset.backup_start_date, 102) >= GETDATE() - 7)
                                  AND msdb.dbo.backupmediafamily.physical_device_name like
'https://%blob.core.windows.net%'
                                  ORDER BY
                                  msdb.dbo.backupset.database_name,
                                  msdb.dbo.backupset.backup_finish_date
```

```
                        OPEN RestoreLogs_CURSOR;
                        FETCH NEXT FROM RestoreLogs_CURSOR INTO
@server,@databasename,@backuptype,@backupsize, @backupdevice;

                        IF @@CURSOR_ROWS != 0 and @@FETCH_STATUS = 0
                        BEGIN
                                SET @restoreheadermsg = '** Restore Transaction Logs scripts for
Database ' + @databasename + ' on Server  ' + @server + ' of type ' + @backuptype
                                SET @cmd1 = 'echo ' + @restoreheadermsg + ' >> ' +
@restorelogfile
                                EXEC master..xp_cmdshell @cmd1;
                        END;

                        WHILE @@FETCH_STATUS =0
                        BEGIN

                                SET @restoremsg = 'Restore Database ' + @databasename + ' from
URL =' + '''' + @backupdevice + '''' + ' WITH FILE = 1, NORECOVERY,  NOUNLOAD,  STATS = 5'
                                SET @cmd2 = 'echo ' + @restoremsg + ' >> ' + @restorelogfile
                                EXEC master..xp_cmdshell @cmd2;
                                FETCH NEXT FROM RestoreLogs_CURSOR INTO
@server,@databasename,@backuptype,@backupsize, @backupdevice;
                        END;
                        CLOSE RestoreLogs_CURSOR;
                        DEALLOCATE RestoreLogs_CURSOR;
        FETCH NEXT FROM AllDBs_CURSOR INTO @userdb
END;
        CLOSE AllDBs_CURSOR
        DEALLOCATE AllDBs_CURSOR
GO
```

3 References

- What is SQL Server on Azure Virtual Machines? (Windows)
- Performance guidelines for SQL Server in Azure VM
- Storage configuration for SQL Server VMs
- Get started with AzCopy

4 Feedback and Suggestions

If you have feedback or suggestions for improving this data migration asset, please contact the Data Migration Jumpstart Team (askdmjfordmtools@microsoft.com). Thanks for your support! Note: For additional information about migrating various source databases to Azure, see the Azure Database Migration Guide.