

Database migrations to Azure SQL DB Managed Instance - Restore with Full and Differential backups

Prepared by

Data SQL Ninja Team (datasqlninja@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.

© 2020 Microsoft. All rights reserved.

Note: The detail provided in this document has been harvested as part of a customer engagement sponsored through the [Data SQL Ninja Engineering](#).

Table of Contents

1	Introduction	4
2	SQL Server Migration Scenarios and Requirements	5
2.1	Possible scenarios	5
2.2	Applications and tools required	5
3	Planning the migration	6
3.1	High Level Process	6
4	Initiate the Migration.....	7
4.1	Run the full backup	7
4.2	Copy the backup files to Azure Blob	7
4.3	Initiate the restore	7
5	Initiate the Cutover	11
5.1	Stop writing processes into the source database	11
5.2	Run the Differential backup	11
5.3	Copy the differential backup file to Azure Blob.....	11
5.4	Cutover with Advanced REST Client	11
6	PowerShell Automation Option.....	14
6.1	Introduction.....	14
6.2	Prerequisites.....	14
6.3	Initiate the migration from full backup	14
6.4	Initiate the cutover from differential backup	16
7	Appendix.....	19
7.1	Backup Tuning Considerations.....	19

1 Introduction

This Whitepaper has been developed to help accelerating migrations from on-premises, IaaS and/or relevant SQL Server implementation that need to be migrated to an Azure SQL DB Managed Instance, with little downtime, and having full and differential backups as the only options from source (e.g. no log backup capabilities).

If log backups are available, it is strongly recommended to use [Data Migration Service](#) instead.

2 SQL Server Migration Scenarios and Requirements

2.1 Possible scenarios

This whitepaper covers these possible scenarios:

- ✓ There is little tolerance for downtime for the migration cutover (e.g. less than 1 hour)
- ✓ There is no log backup option available from source
- ✓ There is no possibility to use [Azure Data Migration Service](#)
- ✓ Source can backup with these options:
 - Full Backup with 1 or to multiple files (recommended)
 - Differential backup to 1 file (mandatory)
 - Compression can be enabled (recommended)
 - Checksum enabled (mandatory)
- ✓ Source is SQL Server 2005 or later

2.2 Applications and tools required

- ✓ Access to source database as backup operator or administrator, to execute full backup and differential backup commands
- ✓ Administrative access to the target Azure SQL DB Managed Instance from a SQL interface tool like SSMS (SQL Server Management Studio)
- ✓ Access to the Azure resource portal that allows users to access the target Azure SQL DB Managed Instance
- ✓ Access to the Azure Blob Storage container to create SAS keys to:
 - Write the backup files
 - Read and List the backup files (these 2 options are the only ones that must be enabled for the web portal restore to work properly)
- ✓ Access to download and Install [Advanced REST client](#)
- ✓ Access to download and Install azcopy:
 - [Windows 64-bit](#) (zip)
 - [Windows 32-bit](#) (zip)
 - [Linux](#) (tar)
 - [MacOS](#) (zip)

3 Planning the migration

Recommendation is to plan the migration, to ensure that there is as little data as tolerable acceptable for the differential backup file creation.

It is also recommended to proceed when the database is not been intensively used for writes.

3.1 High Level Process

1. Run a full backup on source into multiple files (e.g. 10), with compression (recommended) and checksum (mandatory)
2. Copy the full backup files from source to an Azure Blob Storage Container with azcopy.
3. Initiate the Restore via web portal with access to your subscription
4. Obtain the Authorization from the Request Preview after running the restore from portal
5. Monitor the restore process on the Azure SQL DB Managed instance target with SSMS (or tool of preference)
6. Obtain the *session_activity_id* from the restore process on the Azure SQL DB Managed Instance with SSMS (or tool of preference)
7. Once determined that the full backup restore is about to finalize, stop write connectivity to the source database
8. Run a differential backup on source into 1 file with compression (recommended) and checksum (mandatory).
9. Transfer the differential backup files from source to the same Azure Blob Storage container used to store the full backup
10. Execute the cutover with the Advanced REST client

4 Initiate the Migration

4.1 Run the full backup

Sample command:

```
BACKUP DATABASE [SAMPLE]
```

```
TO
```

```
DISK='C:\BACKUP\SAMPLEBackup01.bak',
```

```
DISK='C:\BACKUP\SAMPLEBackup02.bak',
```

```
DISK='C:\BACKUP\SAMPLEBackup03.bak',
```

```
DISK='C:\BACKUP\SAMPLEBackup04.bak',
```

```
DISK='C:\BACKUP\SAMPLEBackup05.bak',
```

```
DISK='C:\BACKUP\SAMPLEBackup06.bak',
```

```
DISK='C:\BACKUP\SAMPLEBackup07.bak',
```

```
DISK='C:\BACKUP\SAMPLEBackup08.bak',
```

```
DISK='C:\BACKUP\SAMPLEBackup09.bak',
```

```
DISK='C:\BACKUP\SAMPLEBackup10.bak'
```

```
WITH COMPRESSION, CHECKSUM;
```

You can review optional [backup tuning considerations section](#) for options to further tune backup speed.


4.2 Copy the backup files to Azure Blob

Assumption is that the SAS key for writing files is already available

Sample command:

```
azcopy cp "C:\BACKUP\*.*)" "https://account.blob.core.windows.net/container?SASKey"
```

4.3 Initiate the restore

1. Access <https://docs.microsoft.com/en-us/rest/api/sql/manageddatabases/createorupdate>
2. Click on 
3. Sign in and choose the appropriate account

4. Fill in (ensure to keep the api-version as below):

Parameters		
databaseName*	<input type="text" value="TheTargetDatabase"/>	
managedInstanceName*	<input type="text" value="TheMangedInstanceName"/>	
resourceGroupName*	<input type="text" value="ResourceGroupNameOfTheManagedInstance"/>	
subscriptionId*	<input type="text" value="Choose the Subscription Name"/>	▼
api-version*	<input type="text" value="2017-10-01-preview"/>	
<input type="text" value="name"/>	<input type="text" value="value"/>	+

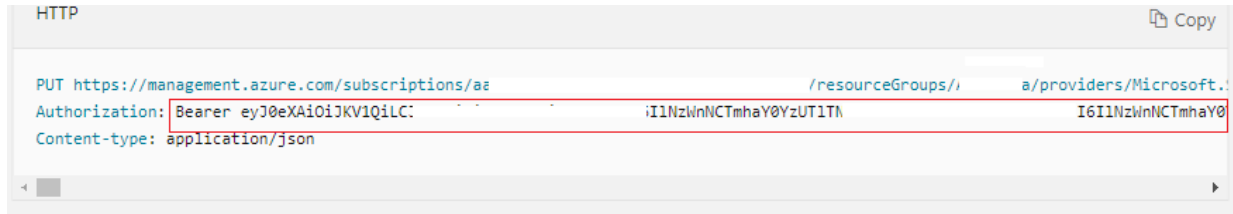
Headers		
Content-Type*	<input type="text" value="application/json"/>	
<input type="text" value="name"/>	<input type="text" value="value"/>	+

5. Paste into the Body (SAS Key below should just have **only** list and read capabilities from container):
- Location: e.g. *West US 2*

```
{
  "properties": {
    "createMode": "RestoreExternalBackup",
    "storageContainerUri": "https://account.blob.core.windows.net/container",
    "storageContainerSasToken": "SASKey",
    "collation": "SQL_Latin1_General_CP1_CI_AS"
  },
  "location": "RegionOfTheManagedInstance"
}
```

Run ▶

- E.g.: *Bearer verylonstringverylonstringverylonstringverylonstring*



- ## 10. Monitor the status on the Managed Instance with a tool like SSMS

```
SELECT session_id as SPID, command, a.text AS Query, start_time, percent_complete
      , dateadd(second,estimated_completion_time/1000, getdate()) as
estimated_completion_time
FROM sys.dm_exec_requests r
CROSS APPLY sys.dm_exec_sql_text(r.sql_handle) a
WHERE r.command in ('BACKUP DATABASE','RESTORE DATABASE');
```

11. Once completed (e.g. no output from above query), obtain the `session_activity_id` of the pending restore (e.g. EAB63459-9AF0-4433-1BFF-CCE A70B67BC1)

```
SELECT [session_activity_id] as operationId
      ,[resource_type]
      ,[resource_type_desc]
      ,[major_resource_id]
      ,[minor_resource_id]
      ,[operation]
```

```
, [state]
, [state_desc]
, [percent_complete]
, [error_code]
, [error_desc]
, [error_severity]
, [error_state]
, [start_time]
, [last_modify_time]

FROM [master].[sys].[dm_operation_status] where state_desc = 'IN_PROGRESS';
```

5 Initiate the Cutover

5.1 Stop writing processes into the source database

You can opt to stop the applications connecting to the source database, to ensure that no new transactions are applied after the differential backup has been executed.

5.2 Run the Differential backup

Sample command:

```
BACKUP DATABASE [SAMPLE]
TO
    DISK='C:\BACKUP\SAMPLEBackupDIF.bak'
WITH DIFFERENTIAL, COMPRESSION, CHECKSUM;
```

5.3 Copy the differential backup file to Azure Blob

Assumption is that the SAS key for writing files is already available

Sample command:

```
azcopy cp "C:\BACKUP\ SAMPLEBackupDIF.bak"
"https://account.blob.core.windows.net/container?SASKey"
```

5.4 Cutover with Advanced REST Client

1. Install [Advanced REST Client](#) and run it
2. Build out the POST Request URL. Have these handy
 - o Azure Subscription ID
 - o The Location (region) of the Managed Instance
 - o The session_activity_id obtained of the pending restore (e.g. EAB63459-9AF0-4433-1BFF-CCE A70B67BC1)
 - o Sample below:

```
https://management.azure.com/subscriptions/YourSubscriptionID/providers/Microsoft.Sql/locations/MILocation (e.g. westus2)/managedDatabaseRestoreAzureAsyncOperation/session_activity_id/completeRestore?api-version=2017-03-01-preview
```

3. Choose Method: *POST*
4. Request URL: Paste the above created
5. Add in HEADERS Section (use value from Initiate the Restore section)
 - Header: *Authorization*
 - Parameter Value: *Bearer verylonstringverylonstringverylonstringverylonstring*
6. Add in Headers Section
 - Header: *Content-Type*
 - Parameter Value: *application/json*
7. Add in the BODY section

```
{
  lastBackupName: "SAMPLEBackupDIF.bak"
}
```

8. Click SEND and expect a 202 Accepted

Method
POST ▼

Request URL
https://management.azure.com/subscriptions/aa841...

SEND

⋮

Request parameters ^

HEADERS BODY AUTHORIZATION ACTIONS CONFIG CODE

COPY SOURCE VIEW

<input checked="" type="checkbox"/>	Header name Authorization	Parameter value Bearer verylonstringverylonstringverylonstringver	⊖
<input checked="" type="checkbox"/>	Header name Content-Type	Parameter value application/json	⊖

⊕ ADD HEADER

Browser tabs: < HTTPS://MANAGEMENT... x + >

Method: POST ▾ Request URL: https://management.azure.com/subscriptions/aa841 ▾ **SEND** ⋮

Request parameters ▴

HEADERS BODY AUTHORIZATION ACTIONS CONFIG CODE

Body content type: application/json ▾

FORMAT JSON MINIFY JSON COPY

```
1 {  
2   lastBackupName: "SAMPLEBackupDIF.bak"  
3 }
```

202 Accepted 745.06 ms DETAILS ▾

9. Database should finalize restore and open.

6 PowerShell Automation Option

6.1 Introduction

The PowerShell option below can help you automate steps depicted prior in the document, to go forward with a full backup restore plus differential cutover.

6.2 Prerequisites

To successfully execute the PowerShell commands, validate the state of the restore operation one should have:

- ✓ The PowerShell or PowerShell Core (7.0 or newer) installed to initiate the restore operation via REST API. We recommend the latest versions. The code may work with older versions too, but it hasn't been tested.
- ✓ Az, Az.Sql, Az.Accounts modules installed.
- ✓ Access to the Managed Instance via TDS endpoint to verify the restore operation progress and states.

6.3 Initiate the migration from full backup

The steps below describe how to initiate the restore operation using PowerShell and assume the full backup files are already in the specified storage account container. **We recommend having a dedicated container per database or restore operation.** Mixing multiple backup files for different databases in a single container may cause the operation to fail.

The overall process completes the following steps:

- ✓ Retrieve Managed Instance properties such as:
 - Id
 - Collation
 - LocationThey will be needed to invoke the request.
- ✓ Retrieve the container SAS token with Read and List permissions only
- ✓ Obtain the authentication token
- ✓ Invoke restore operation via REST API
- ✓ Monitor the status and progress of the restore operation

1. Login to Azure and select appropriate subscription

```
Login-AzAccount  
Select-AzSubscription -SubscriptionName "XXXXXXX"
```

2. Import Modules

```
Import-Module Az.Sql  
Import-Module Az.Accounts
```

3. Get Managed Instance properties

```
$ManagedInstanceName = "managedinstancename"  
$miId = (Get-AzSqlInstance -InstanceName $ManagedInstanceName).Id  
$collation = (Get-AzSqlInstance -InstanceName $ManagedInstanceName).Collation  
$location = (Get-AzSqlInstance -InstanceName $ManagedInstanceName).Location
```

4. Build the REST URI (as an example AdventureWorksDW2017 database will be used)

```
$DatabaseName = "AdventureWorksDW2017"  
$uriFull = "https://management.azure.com{0}/databases/{1}?api-version=2017-10-01-  
preview" -f $miId, $DatabaseName
```

5. Obtain container SAS token. It will be valid for 24 hours, but it may changes as needed. The container URI and SAS token will be used in the REST call body

```
$StorageAccountName = "storageaccountname"  
$Container = "containername"  
$containerUri = "https://{0}.blob.core.windows.net/{1}" -  
f $StorageAccountName, $Container  
$context = (Get-AzResource -Name $StorageAccountName | Get-  
AzStorageAccount).Context  
$now = Get-Date  
$sas = New-AzStorageContainerSASToken -Name $Container -Permission "r1" -  
StartTime $now.AddHours(-1) -ExpiryTime $now.AddHours(24) -Context $context
```

6. Create REST call body. The variables \$containerUri, \$sas, \$collation and \$location have been assigned in the previous steps

```
$bodyFull = @"  
    {
```

```

        "properties": {
            "createMode": "RestoreExternalBackup",
            "storageContainerUri": "$containerUri",
            "storageContainerSasToken": "$sas",
            "collation": "$collation"
        },
        "location": "$location"
    }
"@

```

7. Obtain authentication token and create REST header

```

$azProfile = [Microsoft.Azure.Commands.Common.Authentication.Abstractions.AzureRmProfileProvider]::Instance.Profile
$currentAzureContext = Get-AzContext
$profileClient = New-Object Microsoft.Azure.Commands.ResourceManager.Common.RMProfileClient($azProfile)
$token = $profileClient.AcquireAccessToken($currentAzureContext.Tenant.TenantId)
$authToken = $token.AccessToken
$headers = @{}
$headers.Add("Authorization", "Bearer "+"$authToken")

```

8. Invoke FULL restore request

```

Invoke-RestMethod -Method Put -Headers $headers -Uri $uriFull -
ContentType "application/json" -Body $bodyFull

```

9. Monitor the status of the restore process as described in the 4.3 subchapter of this document
10. Obtain the ***session_activity_id*** associated with the restore operation as described in the 4.3 subchapter of this document. It will be needed to invoke differential backup restore.

6.4 Initiate the cutover from differential backup

Once the full backup restore is completed the last differential backup can be restored. The steps below describe how to complete the restore operation using PowerShell and assume the differentials file (there must be a single diff backup file) is already in the specified storage account container. **We recommend having a dedicated container per database or restore operation.** Mixing multiple backup files for different databases in a single container may cause the operation to fail.

The overall process completes the following steps:

- ✓ Retrieve Managed Instance Id and Subscription Id
- ✓ Obtain the authentication token
- ✓ Invoke restore operation via REST API
- ✓ Monitor the status and progress of the restore operation

1. Login to Azure and select appropriate subscription

```
Login-AzAccount  
Select-AzSubscription -SubscriptionName "XXXXXXX"
```

2. Import Modules

```
Import-Module Az.Sql  
Import-Module Az.Accounts
```

3. Get Managed Instance Id and Subscription Id

```
$ManagedInstanceName = "XXXXXXX"  
$location = (Get-AzSqlInstance -InstanceName $ManagedInstanceName).Location  
$subscriptionId = (Get-AzContext).Subscription.Id
```

4. Build the REST URI. \$RestoreSessionActivityId is the session_activity_id associated with the previously initiated restore operation (e.g. 5FFDEDFE-FEE5-481D-9138-F22E33A1FB7D). It can be found in the sys.dm_operation_status as described in subchapter 4.3 of this document

```
$RestoreSessionActivityId = "5FFDEDFE-FEE5-481D-9138-F22E33A1FB7D"  
$uriDiff = "https://management.azure.com/subscriptions/$subscriptionId/providers/  
Microsoft.Sql/locations/$location/managedDatabaseRestoreAzureAsyncOperation/$RestoreSessionActivityId/completeRestore?api-version=2017-03-01-preview"
```

5. Create REST call body. As an example AdventureWorksDW2017_DIFF.bak file is used. The file must be already copied to the storage container.

```
$LastDifferentialBackupFileName = "AdventureWorksDW2017_DIFF.bak"  
$bodyDiff = @"  
{  
  "lastBackupName": "$LastDifferentialBackupFileName"  
}  
"@
```

6. Obtain authentication token and create REST header

```
$azProfile = [Microsoft.Azure.Commands.Common.Authentication.Abstractions.AzureRmProfileProvider]::Instance.Profile
$currentAzureContext = Get-AzContext
$profileClient = New-Object Microsoft.Azure.Commands.ResourceManager.Common.RMProfileClient($azProfile)
$token = $profileClient.AcquireAccessToken($currentAzureContext.Tenant.TenantId)
$authToken = $token.AccessToken
$headers = @{}
$headers.Add("Authorization", "Bearer "+"$authToken")
```

7. Invoke DIFF restore request

```
Invoke-RestMethod -Method Post -Headers $headers -Uri $uriDiff -
ContentType "application/json" -Body $bodyDiff
```

8. Monitor the status of the restore process as described in a [prior subchapter](#) of this document

7 Appendix

7.1 Backup Tuning Considerations

- Using MAXTRANSFERSIZE is needed to compress TDE enabled databases (SQL 2016 CU7 / SP1 CU4 / SP2 +).
- We have seen a 4MB (4194304) MAXTRANSFERSIZE to give best performance in various engagements.
- BUFFERCOUNT requires more precise tuning. A quick section on TF3213 (for Backup/Restore info) and TF3605 to output this information to the error log would be a good idea to optimize backup.
- Watch out for 512byte aligned disk I/O on source and the impact this has on the restore speed. It may be needed in advance to set TF1800 on the source on-prem SQL Server in order to get around potential issues.