



SQL Server database files and backups on Azure Storage for SAP workloads

By Ravi Alwani
AzureCAT

June 2018

Contents

1	Introduction	1
2	SQL Server data files on blob storage	2
2.1	Use PowerShell	2
2.1.1	Create credentials	4
2.1.2	Create a test database	4
2.2	Use Azure portal.....	5
3	File-snapshot backups for database files on Azure.....	10
3.1	Back up database using file-snapshot backup	10
3.2	Generate activity and backup logs using file-snapshot backup	12
3.3	Restore a database to a point in time	14
	Learn more.....	15

Authored by Ravi Alwani. Edited by Nanette Ray. Reviewed by Cameron Gardiner.

© 2018 Microsoft Corporation. This document is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY. The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

1 Introduction

Many SAP users are turning to Microsoft Azure to run their SAP workloads. When migrating to Azure, it's recommended that you store Microsoft SQL Server data files (.mdf, .ldf, and .bak files) on Azure. Doing so offers several benefits, including:

- Easy, fast migration.
- High availability and disaster recovery.
- Added security.
- File-snapshot backups.

This article addresses questions and requests from SAP users migrating to Azure. The goal is to provide concise, step-by-step guidance for storing SQL Server data files using Azure Blob Storage (in [Section 2](#)) and for setting up file-snapshot backups (in [Section 3](#)).

NOTE: In the latest version of SAP Software Provisioning Manager (SWPM), SAPinst can't create data files directly on Azure Blobs. However, if the target SAP database is created before running SAPinst, the installer detects the database and installs it normally. It's therefore recommended to create the target SAP database with data files on blobs before running SAPinst.

System size (number of cores)	Number of data files	Notes
Small	4 data files	Systems usually run on dedicated database servers that have 4 to 8 CPU cores.
Medium	8 or 16 data files	Systems usually run on dedicated database servers that have 8 to 16 CPU cores.
Large	16 (minimum) and 32 (maximum) data files	Systems today run on hardware with 16 to 32 CPU cores or up to 64 threads.
Extra large	32 data files for systems with 64 to 256 or more cores	Contact your Microsoft representative for very large databases in excess of 20 TB.

IMPORTANT: For SAP applications using SQL Server as the back-end database, the data files must be the same size. SQL Server allocates space for new data proportional to the free space in each of the files.

2 SQL Server data files on blob storage

You can use Azure PowerShell scripts and SQL queries to store SQL Server data files as Azure Blobs. Using scripts, you can easily create a database in SQL Server running on premises or in a virtual machine on Azure and set up a dedicated storage location for your data in Azure Blob Storage.

You can do the same thing using the Azure portal instead of PowerShell scripts. Both approaches are outlined in this section.

2.1 Use PowerShell

The PowerShell script that follows sets up a storage account, creates SQL Server credentials, and creates a test database with files stored in blobs. The script uses the Azure Resource Manager deployment model to create these new resources on Azure:

- [Resource group](#)
- [Storage account](#)
- [Container](#)
- [Shared access signature](#) (SAS)

Resource Manager is recommended, but if you need to work with a classic Azure Standard Manager storage account, you can. Modify the script as the comments indicate.

When you run the following script, note the output—a SQL statement to create credentials on SQL Server.

```
<#
This script uses the Resource Manager model and creates a new Resource Manager storage
account. Modify this script to use an existing Resource Manager.
#>
# Define global variables for the script
$prefixName = '<<add Prefix>>' # used as the prefix for the name for various objects
$subscriptionName="<<your Subscription>>" # the name of the subscription you will use
$locationName = '<<specify datacenter region>>' # the data center region you will use
$storageAccountName= $prefixName + 'storage' # the storage account name you will
create or use
$containerName= $prefixName + 'container' # the storage container name to which you
will attach the SAS policy with its SAS token
$policyName = $prefixName + 'policy' # the name of the SAS policy

# Set a variable for the name of the resource group you will create or use
$resourceGroupName=$prefixName + 'rg'

# adds an authenticated Azure account for use in the session
Login-AzureRmAccount

# set the tenant, subscription and environment for use in the rest of
Set-AzureRmContext -SubscriptionName $subscriptionName
```

```

# create a new resource group - comment out this line to use an existing resource
group
New-AzureRmResourceGroup -Name $resourceGroupName -Location $locationName

# Create a new Resource Manager storage account - comment out this line to use an
existing Resource Manager storage account.

New-AzureRmStorageAccount -Name $storageAccountName -ResourceGroupName
$resourceGroupName -Type Premium_LRS -Location $locationName

# Get the access keys for the Resource Manager storage account
$accountKeys = Get-AzureRmStorageAccountKey -ResourceGroupName $resourceGroupName -
Name $storageAccountName

# Create a new storage account context using a Resource Manager storage account
$storageContext = New-AzureStorageContext -StorageAccountName $storageAccountName -
StorageAccountKey $accountKeys[0].Value

<#
Using the classic deployment model
Use the following four lines to use an existing classic storage account
#>
#Classic storage account name
#Add-AzureAccount
#Select-AzureSubscription -SubscriptionName $subscriptionName #provide an existing
classic storage account
#$accountKeys = Get-AzureStorageKey -StorageAccountName $storageAccountName
#$storageContext = New-AzureStorageContext -StorageAccountName $storageAccountName -
StorageAccountKey $accountKeys.Primary

# The remainder of this script works with either the Resource Manager or classic
sections of code above

# Creates a new container in blob storage
$container = New-AzureStorageContainer -Context $storageContext -Name $containerName
$cbc = $container.CloudBlobContainer

# Sets up a Stored Access Policy and a Shared Access Signature for the new container
$permissions = $cbc.GetPermissions();
$policyName = $policyName
$policy = new-object 'Microsoft.WindowsAzure.Storage.Blob.SharedAccessBlobPolicy'
$policy.SharedAccessStartTime = $(Get-Date).ToUniversalTime().AddMinutes(-5)
$policy.SharedAccessExpiryTime = $(Get-Date).ToUniversalTime().AddYears(10)
$policy.Permissions = "Read,Write,List,Delete"
$permissions.SharedAccessPolicies.Add($policyName, $policy)
$cbc.SetPermissions($permissions);

# Gets the Shared Access Signature for the policy
$policy = new-object 'Microsoft.WindowsAzure.Storage.Blob.SharedAccessBlobPolicy'
$sas = $cbc.GetSharedAccessSignature($policy, $policyName)
Write-Host 'Shared Access Signature= '$($sas.Substring(1))'

```

```
# Outputs the Transact SQL to the clipboard and to the screen to create the credential
using the Shared Access Signature
Write-Host 'Credential T-SQL'
$tSql = "CREATE CREDENTIAL [{0}] WITH IDENTITY='Shared Access Signature',
SECRET='{1}'" -f $cbc.Uri,$sas.Substring(1)
$tSql | clip
Write-Host $tSql
```

2.1.1 Create credentials

Next, create credentials based on the output of the script. SQL Server needs these credentials to store the security information it uses to write to and read from the Azure container that is created. The credential name and secret are embedded in the signature returned by the script. For example:

<https://sqladbstorage1.blob.core.windows.net/sqladbcontainer?sr=c&si=sapsqlpolicy&sig=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1234567890k>

- For CREDENTIAL, use the text before the question mark (?).
- For SECRET, use the text (highlighted in yellow) after the question mark (?).

To create the credential, run the following script:

```
CREATE CREDENTIAL [myid_credential] WITH IDENTITY='Shared Access Signature',  
SECRET='mysecret'
```

For example:

[illegible]

To see all available credentials, run:

```
SELECT * from sys.credentials
```

2.1.2 Create a test database

To create a test database (**testdb**) with files stored in a blob, run the following script:

```
CREATE DATABASE <<SID>>
ON
( NAME = <<SID>>_data1,
  FILENAME =
'https://sqldbstorage1.blob.core.windows.net/<SID>container/<SID>Data1.mdf' )
,
( NAME = <<SID>>_data2,
  FILENAME =
'https://sqldbstorage1.blob.core.windows.net/<SID>container/<SID>Data2.ndf' )
LOG ON
( NAME = testdb_log,
  FILENAME =
'https://sqldbstorage1.blob.core.windows.net/<SID>dbcontainer/TestLog1.ldf' )
```

For example:

```
CREATE DATABASE testdb
ON
( NAME = testdb_dat,
  FILENAME =
'https://mabsapsqlstorage1.blob.core.windows.net/mabsapsqlcontainer/TestData1.mdf' )
LOG ON
( NAME = testdb_log,
  FILENAME =
'https://mabsapsqlstorage1.blob.core.windows.net/mabsapsqlcontainer/TestLog1.ldf' )
```

To verify the data file path, use [SQL Server Management Studio](#) as shown in Figure 1.

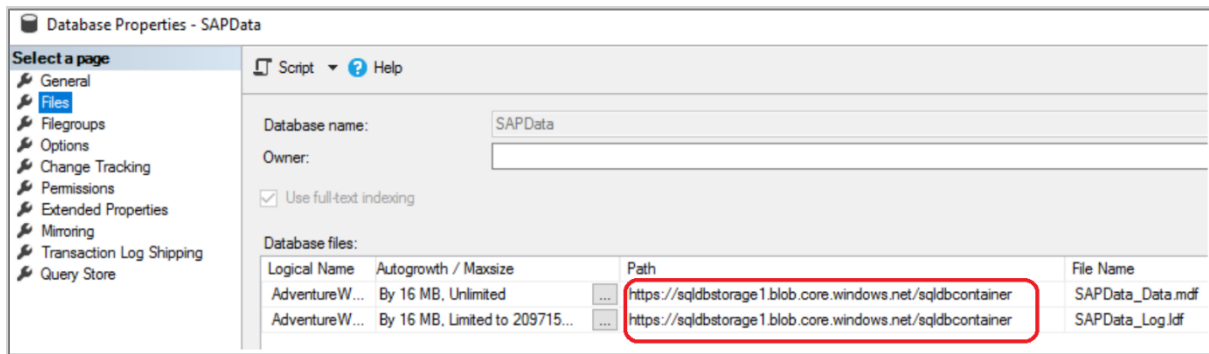


Figure 1. SAPData database properties include the data file path (circled in red) in SQL Server Management Studio

2.2 Use Azure portal

You can use the portal to set up a storage account, create SQL Server credentials, and create a test database with files stored in blobs. Some steps use [Azure Storage Explorer](#).

1. Log on to [Azure portal](#).
2. Select **Storage Account**, then **Add**.
3. Specify the settings as follows:
 - a. For **Name**, type the name of the storage file.
 - b. For **Deployment model**, choose **Resource manager**.
 - c. In the **Account kind** box, choose **Storage (general purpose)**.
 - d. For **Location**, choose the region you want.
 - e. In the **Replication** box, choose **Locally redundant storage (LRS)**.
 - f. For **Performance**, select **Premium**.
 - g. In the **Subscription** box, select the subscription you want to use.
 - h. For **Resource group**, click **Create new**, and then type a name in the box.
 - i. For **Secure transfer required** and **Virtual networks**, the default (**Disabled**) is OK.

The cost of your storage account depends on the usage and the options you choose below. [Learn more](#)

* Name ⓘ
 mabsapsqlstorage11 × ✓
 .core.windows.net

Deployment model ⓘ
 Resource manager Classic

Account kind ⓘ
 Storage (general purpose v1) ▾

* Location
 Central US ▾

Replication ⓘ
 Locally-redundant storage (LRS) ▾

Performance ⓘ
 Standard Premium

* Secure transfer required ⓘ
 Disabled Enabled

* Subscription
 AzureCAT ▾

* Resource group
☒ Create new ☐ Use existing
 sapdata_rg ✓

Virtual networks
 Configure virtual networks ⓘ
 Disabled Enabled

☐ Pin to dashboard

Create Automation options

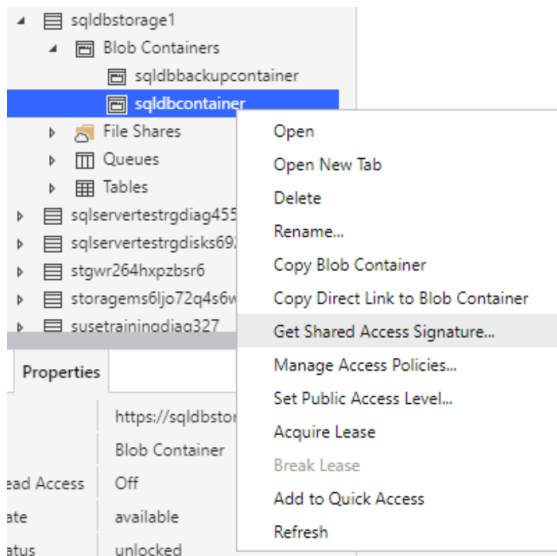
4. Click **Create**.
5. Select the storage account, click the **Containers** tab, and then click **Add Container** at the bottom of the screen.
6. In the **New container** box, enter a name for the container, and select **Private** for **Access type**. When you set the access to private, the container and blob data can only be read by the Azure account owner.

New container
Blob service (mabsapsqlstorage1)

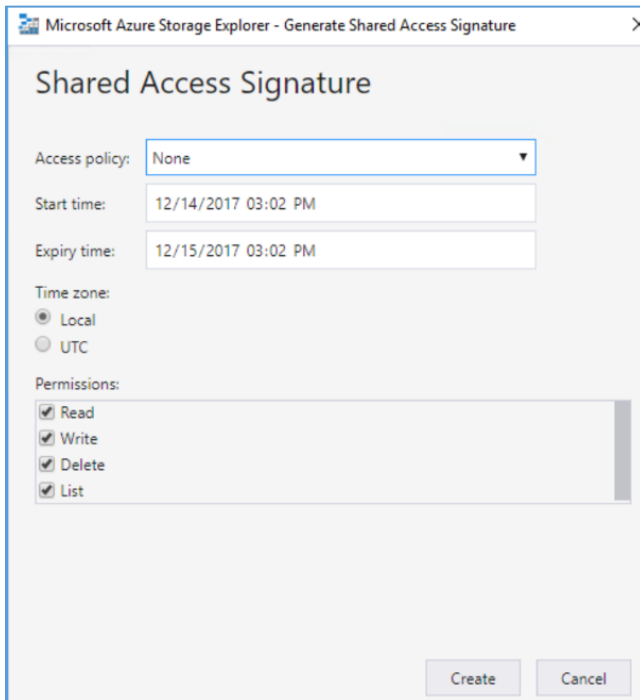
* Name
mabsapsqlcontainer1

Access type ⓘ
Private

7. If needed, install Azure Storage Explorer from <https://azure.microsoft.com/features/storage-explorer/>.
8. Add the newly created storage account in Azure Storage Explorer by providing the storage account name and storage account key under **How do you want to connect to your Storage Account or Service?**
9. Right-click the container and select **Get Shared Access Signature**.



10. Create an access policy, then click **Create**:



Microsoft Azure Storage Explorer - Generate Shared Access Signature

Shared Access Signature

Access policy: None

Start time: 12/14/2017 03:02 PM

Expiry time: 12/15/2017 03:02 PM

Time zone:

☒ Local

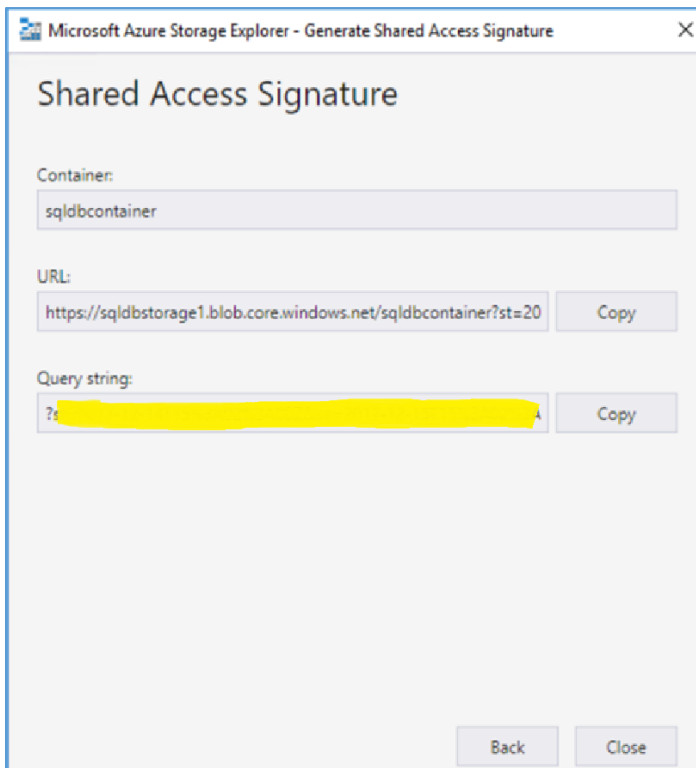
☐ UTC

Permissions:

- ☒ Read
- ☒ Write
- ☒ Delete
- ☒ List

Create Cancel

11. In the **Shared Access Signature** box, use the **Copy** button to copy the URL to the clipboard.



Microsoft Azure Storage Explorer - Generate Shared Access Signature

Shared Access Signature

Container: sqlldbcontainer

URL: https://sqlldbstorage1.blob.core.windows.net/sqlldbcontainer?st=20 Copy

Query string: ?<redacted> Copy

Back Close

12. Open SQL Server Management Studio and create a new credential from the copied signature as follows:

New Credential

Select a page
General

Script ? Help

Credential name: https://sqlbstorage1.blob.core.windows.net/sqlbcontainer

Identity: SHARED ACCESS SIGNATURE

Password:

Confirm password:

☐ Use Encryption Provider

Provider:

Connection

Server:
SQLTest

Connection:
SQLTest\sqladmin

[View connection properties](#)

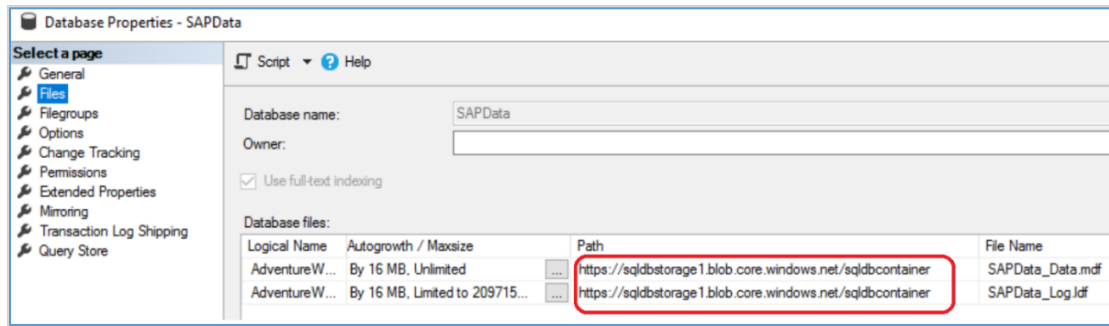
- For **Credential name**, paste the signature, keeping only the text before the question mark (?).
- For **Identity**, type *SHARED ACCESS SIGNATURE*.
- For **Password**, paste the signature, keeping only the text after the question mark (?). This text is the secret required for outgoing authentication. For example, in the following sample signature, the highlighted portion is used for Password:

<https://sqladbstorage1.blob.core.windows.net/sqladbcontainer?sr=c&si=sapsqlpolicy&sig=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1234567890k>

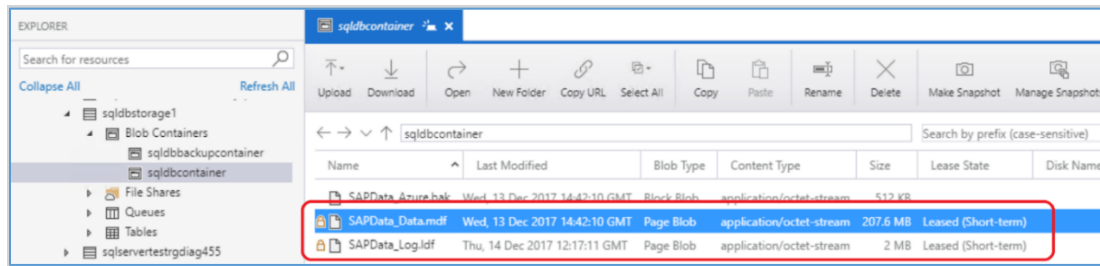
13. In the SQL Server Management Studio query window, create a test database (**testdb**) by executing the following statement:

```
CREATE DATABASE <SID>
ON
( NAME = <SID>_data1,
  FILENAME =
'https://sqldbstorage1.blob.core.windows.net/<SID>container/<SID>Data1.mdf' )
,
( NAME = <SID>_data2,
  FILENAME =
'https://sqldbstorage1.blob.core.windows.net/<SID>container/<SID>Data2.ndf' )
LOG ON
( NAME = testdb_log,
  FILENAME =
'https://sqldbstorage1.blob.core.windows.net/<SID>container/TestLog1.ldf' )
```

14. To confirm that the database files appear, use either SQL Server Management Studio:



Or use Azure Storage Explorer:



3 File-snapshot backups for database files on Azure

To simplify your backup and restore policies, you can specify SQL Server file-snapshot backup. This option uses Azure snapshots to provide nearly instantaneous backups and quicker restores for database files stored using the Azure Blob Storage service.

File-snapshot backups are extremely useful for SAP administrative tasks, such as applying SAP support packs. The backup and restore time is typically measured in seconds—a big advantage. As a best practice, back up your SAP applications before applying support packs and upgrades or performing other major data operations, such as archiving.

In addition, although you can store the SQL data files using either Standard or Premium storage, the backup file itself cannot be stored using Premium storage. For more information, see [File-Snapshot Backups for Database Files in Azure](#).

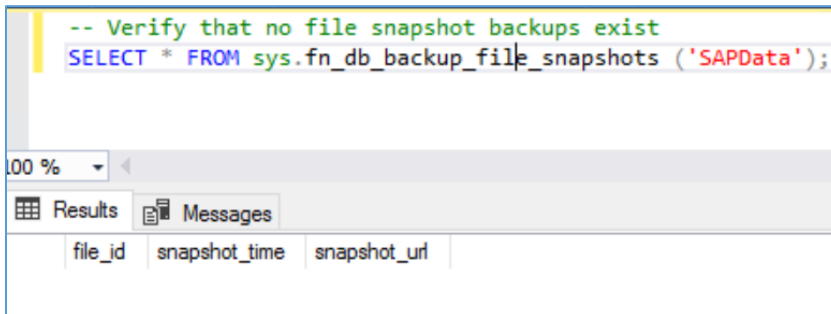
3.1 Back up database using file-snapshot backup

1. Connect to SQL Server Management Studio.
2. Open a new query window and connect to the SQL Server 2016 instance of the database engine in your Azure virtual machine.

- To view the existing file-snapshot backups for each file that comprises a specified database, execute the **sys.fn_db_backup_file_snapshots** system function in a stored procedure as follows:

```
SELECT * FROM sys.fn_db_backup_file_snapshots ('mydatabase')
```

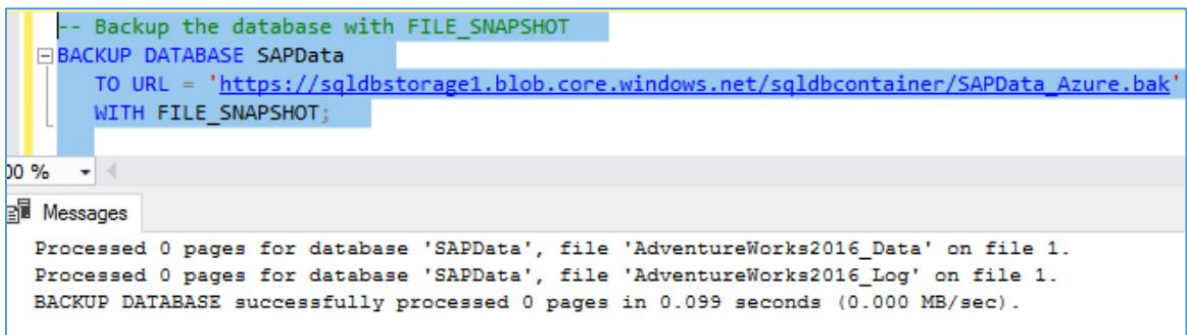
For example, there are no file-snapshot backups for this database:



- To back up your database with file snapshot, execute the following statement, modifying the URL to specify your storage account name and container:

```
BACKUP DATABASE <mydatabase>  
TO URL =  
'http[s]://<myaccountname>.blob.core.windows.net/<mycontainername>/<myfilename.bak>'  
WITH FILE_SNAPSHOT
```

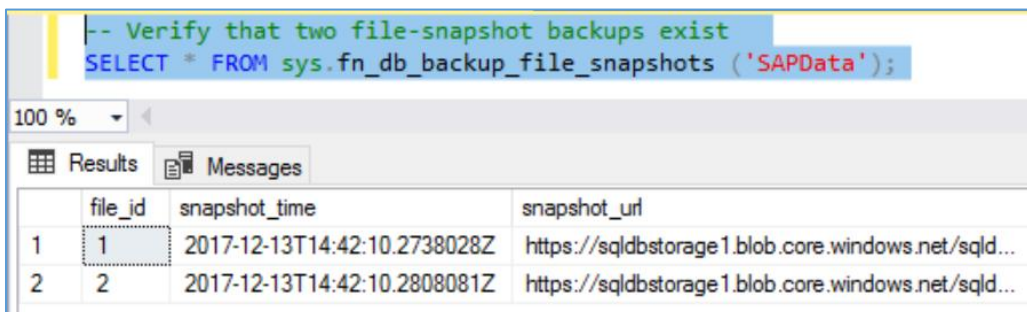
For example:



- To see the results of the file-snapshot backup operation, execute the following again:

```
SELECT * FROM sys.fn_db_backup_file_snapshots ('mydatabase')
```

File-snapshots of both the data and log file are generated. For example:



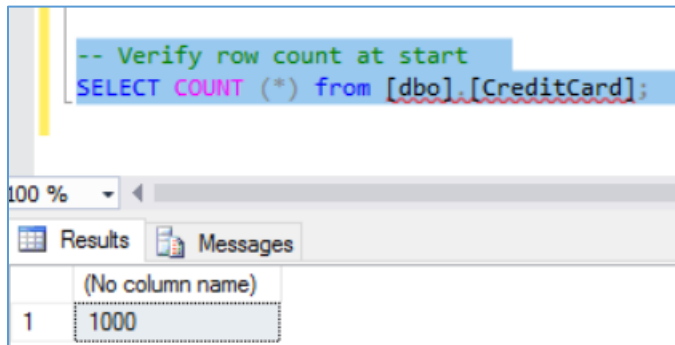
6. In SSMS Object Explorer, connect to your Azure Storage account. Expand **Containers**, then expand the container you specified earlier. The backup created in step 4 appears. For example:



3.2 Generate activity and backup logs using file-snapshot backup

To see how the backup works, you can generate activity in the database and periodically create transaction log backups using file-snapshot backups.

1. Before adding rows, check the count of rows present in the table. For example, the CreditCard table in the MABSAPDB database has 1,000 rows:



2. Open a query window and copy and paste the following script, which adds rows into the CreditCard table. Change the names to match your database and tables.

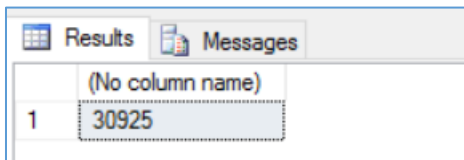
```
USE [SAPData]
GO
-- Insert 30,000 new rows into the CreditCard table in the MABSAPDB database
-- in batches of 75
DECLARE @count INT=1, @inner INT;
WHILE @count < 400
BEGIN
    BEGIN TRAN;
        SET @inner =1;
        WHILE @inner <= 75
        BEGIN;
            INSERT INTO [dbo].[CreditCard]([CardType],[CardNumber]
,[ExpMonth] ,[ExpYear] ,[ModifiedDate])
                VALUES( LEFT( NEWID(), 5 ), LEFT( NEWID(), 16 ) ,5 ,2020
,GETDATE())
            SET @inner = @inner + 1;
        END;
    COMMIT;
    WAITFOR DELAY '00:00:01';
    SET @count = @count + 1;
END;
```

```
SET @count = @count + 1;
END;
SELECT COUNT (*) from [dbo].[CreditCard];
```

- Open a second query window and copy and paste the following script, which creates multiple transaction log backups. Use the URL for your storage account name and the container that you specified earlier.

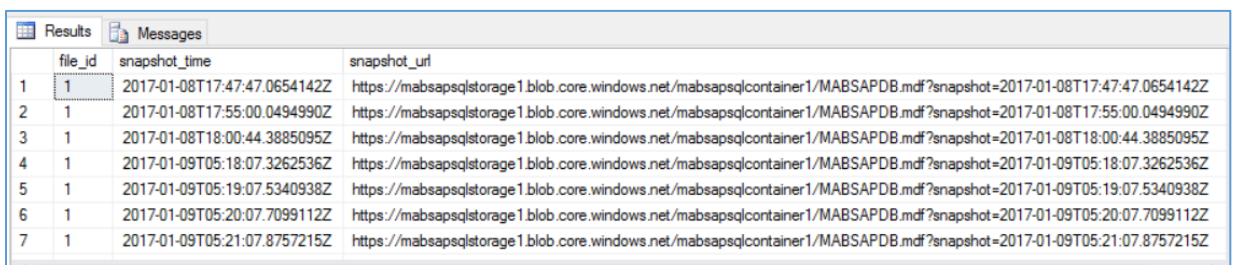
```
--take 7 transaction log backups with FILE_SNAPSHOT, one per minute,
--and include the row count and the execution time in the backup file name
DECLARE @count INT=1, @device NVARCHAR(120), @numrows INT;
WHILE @count <= 7
BEGIN
    SET @numrows = (SELECT COUNT (*) FROM [dbo].[CreditCard]);
    SET @device =
'https://mabsapsqlstorage1.blob.core.windows.net/mabsapsqlcontainer1/MABSAP-' +
CONVERT (varchar(10),@numrows) + '-' + FORMAT(GETDATE(), 'yyyyMMddHHmmss') +
'.bak';
    BACKUP LOG MABSAPDB TO URL = @device WITH FILE_SNAPSHOT;
    SELECT * from sys.fn_db_backup_file_snapshots ('MABSAPDB');
    WAITFOR DELAY '00:1:00';
    SET @count = @count + 1;
END;
```

- Execute the two scripts simultaneously in the separate query windows, then wait five to seven minutes for the scripts to complete.
- Review the output of the first script and notice that the final row count is updated.



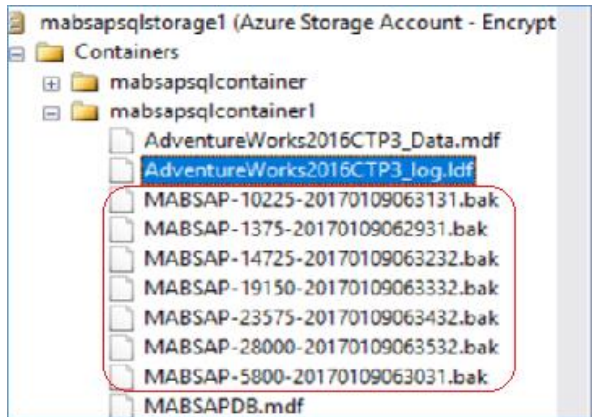
	(No column name)
1	30925

- Review the output of the second script. Notice that each time the BACKUP LOG statement is executed, two new file snapshots are created—one file snapshot of the log file and one file snapshot of the data file—for a total of two file snapshots per database file.



	file_id	snapshot_time	snapshot_url
1	1	2017-01-08T17:47:47.0654142Z	https://mabsapsqlstorage1.blob.core.windows.net/mabsapsqlcontainer1/MABSAPDB.mdf?snapshot=2017-01-08T17:47:47.0654142Z
2	1	2017-01-08T17:55:00.0494990Z	https://mabsapsqlstorage1.blob.core.windows.net/mabsapsqlcontainer1/MABSAPDB.mdf?snapshot=2017-01-08T17:55:00.0494990Z
3	1	2017-01-08T18:00:44.3885095Z	https://mabsapsqlstorage1.blob.core.windows.net/mabsapsqlcontainer1/MABSAPDB.mdf?snapshot=2017-01-08T18:00:44.3885095Z
4	1	2017-01-09T05:18:07.3262536Z	https://mabsapsqlstorage1.blob.core.windows.net/mabsapsqlcontainer1/MABSAPDB.mdf?snapshot=2017-01-09T05:18:07.3262536Z
5	1	2017-01-09T05:19:07.5340938Z	https://mabsapsqlstorage1.blob.core.windows.net/mabsapsqlcontainer1/MABSAPDB.mdf?snapshot=2017-01-09T05:19:07.5340938Z
6	1	2017-01-09T05:20:07.7099112Z	https://mabsapsqlstorage1.blob.core.windows.net/mabsapsqlcontainer1/MABSAPDB.mdf?snapshot=2017-01-09T05:20:07.7099112Z
7	1	2017-01-09T05:21:07.8757215Z	https://mabsapsqlstorage1.blob.core.windows.net/mabsapsqlcontainer1/MABSAPDB.mdf?snapshot=2017-01-09T05:21:07.8757215Z

- Open Object Explorer and connect to your Azure Storage account. Notice that seven new backup files appear (circled):

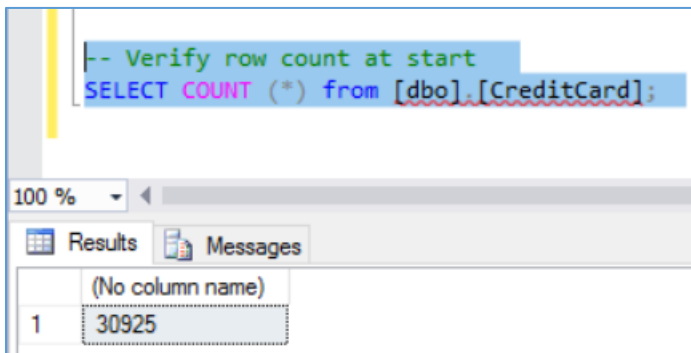


3.3 Restore a database to a point in time

To see how to restore snapshot backups, you can restore a database to its state at a specified point in time using two transaction log file-snapshot backup sets.

For example, to restore the sample database using the transaction logs for the CreditCard table, the process goes like this:

- Check the row count. For example, the CreditCard table has 30,925 rows as shown:



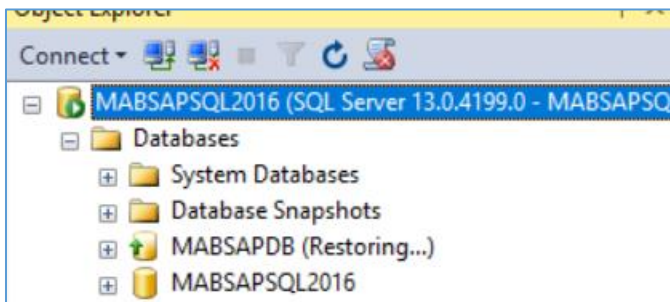
- Select two adjacent log backup files and convert the file name into the date and time needed for the recovery script.
- Edit the script (provided in step c) as follows:
 - Add the first and second backup file names.
 - Specify the STOPAT time in the 'Jan 09, 2017 06:33 AM' format.
 - Substitute your storage account name and the container that you specified.

```
-- restore and recover to a point in time between the times of two transaction log
backups, and then verify the row count
ALTER DATABASE SAPData SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
```



```
RESTORE DATABASE SAPData
FROM URL =
'https://mabsapsqlstorage1.blob.core.windows.net/mabsapsqlcontainer1/MABSAP-1375-20170109062931.bak'
WITH NORECOVERY, REPLACE;
RESTORE LOG MABSAPDB
FROM URL =
'https://mabsapsqlstorage1.blob.core.windows.net/mabsapsqlcontainer1/MABSAP-5800-20170109063031.bak'
WITH RECOVERY, STOPAT = 'Jan 09, 2017 06:33 AM';
ALTER DATABASE MABSAPDB set multi_user;
-- get new count
SELECT COUNT (*) FROM [dbo].[CreditCard];;
```

4. Run the script.
5. Review the output. Object Explorer indicates that the database is being restored as shown:



After the restore, the row count should be changed or reduced, representing the count from the backup taken at the specified time.

Learn more

- [SQL Server data files in Microsoft Azure](#)
- [Tutorial: Use Azure Blob storage service with SQL Server 2016](#)
- [Lesson 5: Backup database using file-snapshot backup](#)
- [Using Azure for hosting and running SAP workload scenarios](#)
- [SAP on Azure reference architectures](#)