Restoring Database from Microsoft Azure Blob Block Storage

**Overview**  
The purpose of this document is to provide step by step instructions on how to restore database from Microsoft Azure Blob Storage Service. There are two types of blobs that can be stored in Microsoft Azure Blob storage service: ***block and page*** blobs. This document is limited to discuss on restoring database from block blob storage service. This is due to the fact that we are migrating from V1 which was configured with page blobs to V2 where most of the recently provisioned database servers QC's send files to block blob (Default as of 11/20/2017).

If you are interested to restore database from block blob, you should have Shared Access Signature (SAS) because BACKUP to a block blob requires SAS that saved in a SQL Server credential. This document will outline the necessary steps to restore database from Block Blob storage using SAS which includes how to login to Azure portal from local working station, retrieve SAS token, create credential using SAS and finally restoring database using GUI or t-SQL . Restoring database from Microsoft azure blob block storage requires SQLDBA team to follow the following three basic steps:

*Step 1: Retrieve SAS token of full-bkup-block container/tlog-bkup-block*

SQL Server backup and restore processes use credential to authenticate to the Microsoft Azure Blob storage service and its container and blob objects. The block blob credential stores container URL and its SAS token therefore it is necessary to retrieve SAS token of full-bkup-block container/tlog-bkup-block before executing the syntax of RESTORE statements. If you are interested to restore only full back up, you only need to retrieve SAS token from full-bkup-block container. However if you are interested to restore both full and tlog back up, you also need to retrieve SAS token from tlog-bkup-block container. Use the below PowerShell script to retrieve the existing SAS.  
  
### login to the portal

Login-AzureRmAccount

### Common code to retrieve sAS token

$subscriptionName='your-subscription-here'  
$resourceGroupName='your-resource-group-here' # Set a variable for the name of the resource group you will create or use  
$storageAccountName= 'backup-storage-here' # the storage account name you will create or use  
$policyName = $storageAccountName # the name of storage account name (foreasier usage)

$fullBkupContainerName = 'full-bkup-block' # the storage container name to which you will attach the SAS policy with its SAS token  
$logBkupContainerName = 'tlog-bkup-block'  
  
# set the tenant, subscription and environment for use in the rest of  
Set-AzureRmContext -SubscriptionName $subscriptionName  
# Get the access keys for the ARM storage account  
$accountKeys = Get-AzureRmStorageAccountKey -ResourceGroupName $resourceGroupName -Name $storageAccountName  
# Create a new storage account context using an ARM storage account  
$storageContext = New-AzureStorageContext -StorageAccountName $storageAccountName -StorageAccountKey $accountKeys[0].Value

######## Storage Account key  
Write-Host 'Storage Account Name:'$storageAccountName''  
Write-Host 'Storage Account Key 1:'$accountKeys[0].Value''

######## full-bkup-block Container SAS Token for Full Backup  
# Get Full backup container in blob storage  
$fullBkupContainer = Get-AzureStorageContainer -Context $storageContext -Name $fullBkupContainerName  
$fullBkupcbc = $fullBkupContainer.CloudBlobContainer  
# Gets the Shared Access Signature for the policy  
$policy = new-object 'Microsoft.WindowsAzure.Storage.Blob.SharedAccessBlobPolicy'  
$fullBkupSASToken = $fullBkupcbc.GetSharedAccessSignature($policy, $policyName)  
$FullBkupBlockTokenPrimary = $fullBkupSASToken.Substring(1)  
Write-Host 'full-bkup-block SAS Token:'$($fullBkupSASToken.Substring(1))''

######## log-bkup-block Container SAS Token for log Backup  
# Get Log backup container in blob storage  
$logBkupContainer = Get-AzureStorageContainer -Context $storageContext -Name $logBkupContainerName  
$logBkupcbc = $logBkupContainer.CloudBlobContainer  
# Gets the Shared Access Signature for the policy  
$policy = new-object 'Microsoft.WindowsAzure.Storage.Blob.SharedAccessBlobPolicy'  
$logBkupSASToken = $logBkupcbc.GetSharedAccessSignature($policy, $policyName)  
$logBkupBlockTokenPrimary = $logBkupSASToken.Substring(1)  
Write-Host 'log-bkup-block SAS Token:'$($logBkupSASToken.Substring(1))''  
If you want to implement Azure related tasks from local station, you need to install Azure PowerShell on your machine. Follow the steps to install Azure PowerShell on your machine <https://docs.microsoft.com/en-us/powershell/azure/install-azurerm-ps?view=azurermps-6.5.0>

*Step 2: Create Credential on the target SQL instance where you are going to restore the database*

* Credential Name: **https:// < storage account name here>** **.**[**blob.core.windows.net/full-bkup-block**](http://blob.core.windows.net/full-bkup-block)
* Identity : **SHARED ACCESS SIGNATURE**
* Password : **SAS token**

**Remember**: a SQL Server credential is an object that is used to store authentication information required to connect to a resource outside of SQL Server. So if you are interested to restore database on the same server from where the backup is configured to send to block blob, there is no need to create credential because the backup and restore processes will use same credential to authenticate to the Microsoft Azure block blob. Just go to step 3 for such case. Creating credential is mandatory if you are interested to restore database from one environment to another environment where they do not have same credential. Backup and restore TDE certification for database encrypted may also be required if necessary in addition to credentials

**How to Create Credential:** It is possible to create credential either using GUI or t-sql

**Using GUI**

Go to security > Credential > Right-click **Credential** and select **New** **Credential**

A screenshot of a computer

Description automatically generated  
  
Pass the following value

* **Credential Name**: https:// < storage account name here> .[blob.core.windows.net/full-bkup-block](https://blob.core.windows.net/full-bkup-block)
* **Identity** : SHARED ACCESS SIGNATURE
* **Password** : SAS token
* **Confirm Password** : SAS token

See example below. This credential is created on different server (let us say ABCDSQ1) using storage account and SAS token of XYZSQ1\SQL1 server. Here the target is we are interested to restore Database backed up from server XYZSQ1\SQL1 to ABCDSQ1\SQL1  
  
A screenshot of a computer screen

Description automatically generated

**t-SQL**

Use the following script to create a SAS credential - providing the URL for the storage account container and the SAS key as indicated

IF NOT EXISTS   
(SELECT \* FROM sys.credentials   
WHERE name = 'https://<storageaccountname>.blob.core.windows.net/<containername>')   
CREATE CREDENTIAL [https://<storageaccountname>.blob.core.windows.net/<containername>]  
WITH IDENTITY = 'SHARED ACCESS SIGNATURE',   
SECRET = '<SAS\_TOKEN>';

***Example***  
  
USE [master]  
GO  
IF NOT EXISTS (SELECT \* FROM sys.credentials WHERE name = '<https://xyzsq1storage.blob.core.windows.net/full-bkup-block> ')  
CREATE CREDENTIAL [<https://xyzsq1storage.blob.core.windows.net/full-bkup-block>]  
WITH IDENTITY = N'SHARED ACCESS SIGNATURE',  
SECRET = N'henoktsv=2017-07-29&sr=c&si=xyzsq1storage=ORDRMi0Lh%%3D'  
Go

*Step 3 :Restoring from Microsoft Azure storage Using SQL Server Management Studio*

The Restore Database task includes **URL** as a device to restore from. The following steps describe using the Restore task to restore from the Microsoft Azure Blob storage service using GUI (2016 and above) or using t-sql  
  
Right-click **Databases** and select **Restore Database...**.  
  
  
On the **General** page, select **Device** under the **Source** section. Click the browse (...) button to open the **Select backup devices** dialog box  
  
  
  
Select **URL** from the **Backup media type:** drop-down list. Click **Add** to open the **Select a Backup File Location** dialog box  
  
  
Select backup File Location  
  
  
Click on **Azure storage container** drop down and select the pre-registered container from the drop down or manually enter the fully qualified container name and pass SAS token on **Shared Access Signature Box** and then click Ok.   
  
A screenshot of a computer

Description automatically generated

SQL Server connects to Microsoft Azure storage using the SQL Credential information you provided and opens the **Locate Backup File in Microsoft Azure** dialog**.**

**Remember** if you create credential correctly, this is the step where you will see the Azure Storage Container of the other server.

The backup files residing in the storage container are displayed on this page. Select the file you want to use to restore and click **OK.** Example: DBAdmin\_20230807\_043000.BAK   
  
This takes you back to the **Select Backup Devices** dialog where URL, Backup file location and bak file being selected. Click Ok  
  
  
Clicking **OK** takes you back to the main **Restore** dialog where you will be able **modify on files and options and** then complete the restore  
  
  
  
You can also script out and may schedule if the DB is Big. Example If I want to change the DB name and move the file to other location with t-sql  
USE [master]  
RESTORE DATABASE [DBAdmin]  
FROM URL = N''<https://xyzsq1storage.blob.core.windows.net/full-bkup-block>/ DBAdmin\_20230807\_043000.BAK' WITH FILE = 1,  
MOVE N'DBAdmin' TO N'G:\DATA\DBAdmin.mdf',  
MOVE N'DBAdmin\_log' TO N'G:\DATA\DBAdmin\_log.ldf'  
GO  
If everything is okay - DB will be restored successfully

Additional reference: [Perform on-premises SQL Server database backups using maintenance plans to Azure Blob Storage (mssqltips.com)](https://www.mssqltips.com/sqlservertip/4900/perform-onpremises-sql-server-database-backups-using-maintenance-plans-to-azure-blob-storage/)