# Cloud Operations

Techlogix Cloud Operation Team provides Infrastructure Support including Servers and Application Monitoring, Cost Management and Capacity Planning. Techlogix Cloud Operation offers an Industry Leading Cloud Service Level Agreement (SLA) with 100% uptime guarantee.

**Contents**

[***Cloud Operations:*** 1](#_Toc488862954)

**Revision Chart**: **…………………..………………………………………………………………………………………………………………3**

**Task: …………………………………………………………………………………………………………………………………………………..4**

**Solution: …………………………………………………………………………………………………………………………………………….4**

**Script: …………………………………………………………………………………………………………………………………………………5**

Revision Chart

This chart contains a history of this document’s revisions. The entries below are provided solely for purposes of illustration. Entries should be deleted until the revision they refer to has actually been created.

The document itself should be stored in revision control, and a brief description of each version should be entered in the revision control system. That brief description can be repeated in this section.

It can be removed from the client version of the document, however the local copy of the client version should contain the revision chart and version column should exhibit the “Client Version” in it.

Revision Chart

| Version | Author(s) | Description of Change | Effective Date |
| --- | --- | --- | --- |
| 1 | Saad Saood | Initial draft created for distribution and review comments | November 20, 2017 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

***Task:***

Import destination database from backup of source database in different/same subscription

1. Backups of two databases (source/destination).
2. Change location of source database .bacpac file from source container to destination container.
3. Delete destination database.
4. Import database from .bacpac file of source database.

***Solution:***

First of all you have to login to azure account and give parameters for source and destination databases and container you want to use.

1. ***Source Database parameters***
2. Select subscription
3. Select resource group
4. Select server
5. Select source database
6. Give credentials
7. Select storage account
8. Select storage container
9. ***Destination Database parameters***
10. Select subscription
11. Select resource group
12. Select server
13. Select source database
14. Give credentials
15. Select storage account
16. Select storage container

Now, script will run and take backups of both databases and change container. Script will then delete destination database and import it from backup of source database.

***Script:***

param (

[Parameter(Position=0)]

[ValidateNotNullOrEmpty()]

[string]$resourceGroupNameSource,

[Parameter(Position=1)]

[ValidateNotNullOrEmpty()]

[string]$sqlServerAdminSource,

[Parameter(Position=2)]

[ValidateNotNullOrEmpty()]

[string]$sqlServerPasswordSource,

[Parameter(Position=3)]

[switch]$statusBar,

[Parameter(Position=4)]

[ValidateNotNullOrEmpty()]

[string]$resourceGroupNameDest,

[Parameter(Position=5)]

[ValidateNotNullOrEmpty()]

[string]$sqlServerAdminDest,

[Parameter(Position=6)]

[ValidateNotNullOrEmpty()]

[string]$sqlServerPasswordDest

)

try

{

$AzureSubscriptionIdSource = (Get-AzureRmSubscription | Out-GridView -Title 'Select an Azure source Subscription' -OutputMode Single)

Set-AzureRmContext -SubscriptionID $AzureSubscriptionIdSource

}

catch

{

Login-AzureRmAccount

$AzureSubscriptionIdSource = (Get-AzureRmSubscription | Out-GridView -Title 'Select an Azure source Subscription' -OutputMode Single)

Set-AzureRmContext -SubscriptionID $AzureSubscriptionIdSource

}

if(!($PSBoundParameters.ContainsKey('resourceGroupNameSource')))

{

$resourceGroupNameSource = (Get-AzureRmResourceGroup | Out-GridView -Title 'Select a Source Resource Group' -OutputMode Single).ResourceGroupName

}

$sqlServerSource = Get-AzureRmSqlserver -ResourceGroupName $resourceGroupNameSource | Out-GridView -Title 'Select your source Azure SQL Server' -OutputMode Single

$sqlDBSource = (Get-AzureRmSqlDatabase -ResourceGroupName $resourceGroupNameSource -ServerName $sqlServerSource.ServerName | where {$\_.DatabaseName -ne 'master'} | Out-GridView -Title 'Select a source DB to export' -OutputMode Single).DatabaseName

if(!($PSBoundParameters.ContainsKey('sqlServerAdminSource')))

{

$sqlServerAdminSource = $sqlserverSource.SqlAdministratorLogin

}

if(!($PSBoundParameters.ContainsKey('sqlServerPasswordSource')))

{

$securePasswordSource = Read-Host "Enter in the password for $sqlServerAdminSource" -AsSecureString

}

else

{

$securePasswordSource = ConvertTo-SecureString -String $sqlServerPasswordSource -AsPlainText -Force

}

$credsSource = New-Object -TypeName System.Management.Automation.PSCredential -ArgumentList $sqlServerAdminSource, $securePasswordSource

# Generate a unique filename for the BACPAC of source

$bacpacFilenameSource = $sqlDBSource + (Get-Date).ToString("yyyyMMddHHmm") + ".bacpac"

# Storage account source info for the BACPAC

$storageAcctSource = Get-AzureRmStorageAccount | Out-GridView -Title 'Select a Source Storage Account' -OutputMode Single

$storageAcctKeySource = (Get-AzureRmStorageAccountKey -ResourceGroupName $storageAcctSource.ResourceGroupName -name $storageAcctSource.StorageAccountName)[0].value

$storageContextSource = New-AzureStorageContext -StorageAccountName $storageAcctSource.StorageAccountName -StorageAccountKey $storageAcctKeySource

$storageContainerSource = Get-AzureStorageContainer -Context $storageContextSource | select Name | Out-GridView -Title 'Select a source Container..' -OutputMode Single

$BaseStorageUriSource = "$($storageAcctSource.PrimaryEndpoints.Blob)$($storageContainerSource.Name)/"

$BacpacUriSource = $BaseStorageUriSource + $bacpacFilenameSource

# Select All Variables for destination subscription

try

{

$AzureSubscriptionIdDest = (Get-AzureRmSubscription | Out-GridView -Title 'Select a destination Azure Subscription' -OutputMode Single)

Set-AzureRmContext -SubscriptionID $AzureSubscriptionIdDest

}

catch

{

Login-AzureRmAccount

$AzureSubscriptionIdDest = (Get-AzureRmSubscription | Out-GridView -Title 'Select a destination Azure Subscription' -OutputMode Single)

Set-AzureRmContext -SubscriptionID $AzureSubscriptionIdDest

}

if(!($PSBoundParameters.ContainsKey('resourceGroupNameDest')))

{

$resourceGroupNameDest = (Get-AzureRmResourceGroup | Out-GridView -Title 'Select Destination Resource Group' -OutputMode Single).ResourceGroupName

}

$sqlServerDest = Get-AzureRmSqlserver -ResourceGroupName $resourceGroupNameDest | Out-GridView -Title 'Select your destination Azure SQL Server' -OutputMode Single

$sqlDBDest = (Get-AzureRmSqlDatabase -ResourceGroupName $resourceGroupNameDest -ServerName $sqlServerDest.ServerName | where {$\_.DatabaseName -ne 'master'} | Out-GridView -Title 'Select a destination DB to export' -OutputMode Single).DatabaseName

if(!($PSBoundParameters.ContainsKey('sqlServerAdminDest')))

{

$sqlServerAdminDest = $sqlserverDest.SqlAdministratorLogin

}

if(!($PSBoundParameters.ContainsKey('sqlServerPasswordDest')))

{

$securePasswordDest = Read-Host "Enter in the password for $sqlServerAdminDest" -AsSecureString

}

else

{

$securePasswordDest = ConvertTo-SecureString -String $sqlServerPasswordDest -AsPlainText -Force

}

$credsDest = New-Object -TypeName System.Management.Automation.PSCredential -ArgumentList $sqlServerAdminDest, $securePasswordDest

# Generate a unique filename for the destination BACPAC

$bacpacFilenameDest = $sqlDBDest + (Get-Date).ToString("yyyyMMddHHmm") + ".bacpac"

# Storage account Destination info for the BACPAC

$storageAcctDest = Get-AzureRmStorageAccount | Out-GridView -Title 'Select a destination Storage Account' -OutputMode Single

$storageAcctKeyDest = (Get-AzureRmStorageAccountKey -ResourceGroupName $storageAcctDest.ResourceGroupName -name $storageAcctDest.StorageAccountName)[0].value

$storageContextDest = New-AzureStorageContext -StorageAccountName $storageAcctDest.StorageAccountName -StorageAccountKey $storageAcctKeyDest

$storageContainerDest = Get-AzureStorageContainer -Context $storageContextDest | select Name | Out-GridView -Title 'Select a destination Container..' -OutputMode Single

$BaseStorageUriDest = "$($storageAcctDest.PrimaryEndpoints.Blob)$($storageContainerDest.Name)/"

$BacpacUriDest = $BaseStorageUriDest + $bacpacFilenameDest

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*command to backup source\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Set-AzureRmContext -SubscriptionID $AzureSubscriptionIdSource

$exportRequestSource = New-AzureRmSqlDatabaseExport -ResourceGroupName $resourceGroupNameSource -ServerName $sqlServerSource.ServerName `

-DatabaseName $sqlDBSource -StorageKeytype 'StorageAccessKey' -StorageKey $storageAcctKeySource -StorageUri $BacpacUriSource `

-AdministratorLogin $credsSource.UserName -AdministratorLoginPassword $credsSource.Password

[int]$counter = 0;

if ($PSBoundParameters.ContainsKey('statusBar'))

{

[int]$expStatusctr = 0

$expStatus = Get-AzureRmSqlDatabaseImportExportStatus -OperationStatusLink $exportRequestSource.OperationStatusLink

while ($expStatus.Status -ne 'Succeeded')

{

Write-Progress -Activity "Exporting Database $($sqlDBSource.DatabaseName)" -PercentComplete (($expStatusctr / 100) \* 100)

start-sleep -Milliseconds 200

$expStatus = Get-AzureRmSqlDatabaseImportExportStatus -OperationStatusLink $exportRequestSource.OperationStatusLink

if ($expStatus.StatusMessage)

{

$expStatus = $expStatus.StatusMessage.Split('=')

$expStatusctr=$expStatus[1].Trim('%')

}

}

Write-Host "Source Export complete! $BacpacUriSource" -ForegroundColor Green

}

else

{

$counter = $counter+1

Write-Host "Source Exporting to $BacpacUriSource. It should be done exporting soon..."

}

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*command to backup Destination\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Set-AzureRmContext -SubscriptionID $AzureSubscriptionIdDest

$exportRequestDest = New-AzureRmSqlDatabaseExport -ResourceGroupName $resourceGroupNameDest -ServerName $sqlServerDest.ServerName `

-DatabaseName $sqlDBDest -StorageKeytype 'StorageAccessKey' -StorageKey $storageAcctKeyDest -StorageUri $BacpacUriDest `

-AdministratorLogin $credsDest.UserName -AdministratorLoginPassword $credsDest.Password

[int]$temp = 0;

if ($PSBoundParameters.ContainsKey('statusBar'))

{

[int]$expStatusctr = 0

$expStatus = Get-AzureRmSqlDatabaseImportExportStatus -OperationStatusLink $exportRequestDest.OperationStatusLink

while ($expStatus.Status -ne 'Succeeded')

{

Write-Progress -Activity "Exporting Database $($sqlDBDest.DatabaseName)" -PercentComplete (($expStatusctr / 100) \* 100)

start-sleep -Milliseconds 200

$expStatus = Get-AzureRmSqlDatabaseImportExportStatus -OperationStatusLink $exportRequestDest.OperationStatusLink

if ($expStatus.StatusMessage)

{

$expStatus = $expStatus.StatusMessage.Split('=')

$expStatusctr=$expStatus[1].Trim('%')

}

}

Write-Host "Destination Export complete! $BacpacUriDest" -ForegroundColor Green

}

else

{

$temp = $temp+1

Write-Host "Destination Exporting to $BacpacUriDest. It should be done exporting soon..."

}

#moving containers from one subscription to another subcription

[int]$temp1 = 0

if($counter -eq 0 -and $temp -eq 0)

{

#Do the copy of only new created bcpacfile

$BlobCopy = Start-CopyAzureStorageBlob -Context $storageContextSource -SrcContainer $storageContainerSource.name -SrcBlob $bacpacFilenameSource -DestContext $storageContextDest -DestContainer $storageContainerDest.name -DestBlob $bacpacFilenameSource -Force

#Check Status of container changing

$CopyState = $BlobCopy | Get-AzureStorageBlobCopyState

$Message = $CopyState.Source.AbsolutePath + " " + $CopyState.Status + " {0:N2}%" -f (($CopyState.BytesCopied/$CopyState.TotalBytes)\*100)

Write-Host $Message -ForegroundColor Yellow

}

else

{

$temp1 = $temp1 + 1

Write-Host "Container NOT change" -ForegroundColor Yellow

}

#Delete database that has been exported from destination

[int]$temp2 = 0

if($counter -eq 0 -and $temp -eq 0 -and $temp1 -eq 0)

{

Remove-AzureRmSqlDatabase -ResourceGroupName $resourceGroupNameDest -ServerName $sqlServerDest.ServerName -DatabaseName $sqlDBDest -Force

Write-Host "Deleted database" $sqlDB -ForegroundColor Yellow

}

else

{

$temp2 = $temp2 + 1

Write-host "Deleting ERROR" -ForegroundColor Black

}

#import new database from blob storage

if($counter -eq 0 -and $temp -eq 0 -and $temp1 -eq 0 -and $temp2 -eq 0)

{

#get Uri of storage bacpac file

$bacpacurinew = $BaseStorageUriDest + $bacpacFilenameSource

$importRequest = New-AzureRmSqlDatabaseImport -ResourceGroupName $resourceGroupNameDest -ServerName $sqlServerDest.ServerName -DatabaseName $sqlDBDest -StorageKeyType "StorageAccessKey" -StorageKey $storageAcctKeyDest -AdministratorLogin $credsDest.UserName -AdministratorLoginPassword $credsDest.Password -StorageUri $bacpacurinew –Edition basic –ServiceObjectiveName basic -DatabaseMaxSizeBytes 500000

#import status

if ($PSBoundParameters.ContainsKey('statusBar'))

{

[int]$impStatusctr = 0

$impStatus = Get-AzureRmSqlDatabaseImportExportStatus -OperationStatusLink $importRequest.OperationStatusLink

while ($impStatus.Status -ne 'Succeeded')

{

Write-Progress -Activity "importing Database $($sqlDBDest.DatabaseName)" -PercentComplete (($impStatusctr / 100) \* 100)

start-sleep -Milliseconds 200

$impStatus = Get-AzureRmSqlDatabaseImportExportStatus -OperationStatusLink $importRequest.OperationStatusLink

if ($impStatus.StatusMessage)

{

$impStatus = $impStatus.StatusMessage.Split('=')

$impStatusctr=$impStatus[1].Trim('%')

}

}

}

}

else

{

Write-Host " NOT import" -ForegroundColor Yellow

}