

Tech Data

Archiving on Azure

Description

The Azure Storage: Archiving solution is particularly well-suited for the storage of infrequently accessed archival data. This solution is built on Azure managed service Blob Storage. This service runs in a high-availability environment, patched and supported, allowing you to focus on your solution instead of the environment they run in. Using Azure blob storage as an extension of the on-premises solution a cool tier on Azure Blob storage is used to back up data that's less frequently accessed, while a hot tier on Azure Blob storage is used to store data that's frequently accessed.

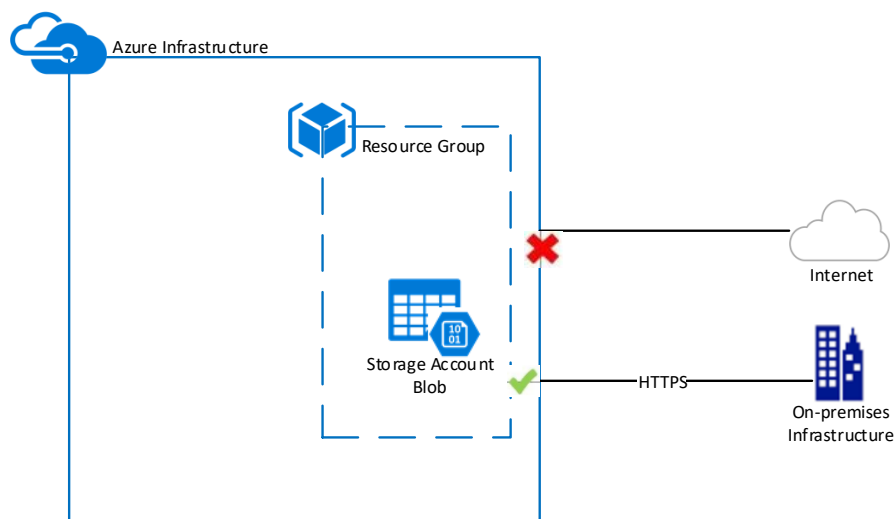
Use Cases:

- Serving images or documents directly to a browser.
- Storing files for distributed access.
- Streaming video and audio.
- Writing to log files.
- Storing data for backup and restore, disaster recovery, and archiving.
- Storing data for analysis by an on-premises or Azure-hosted service.

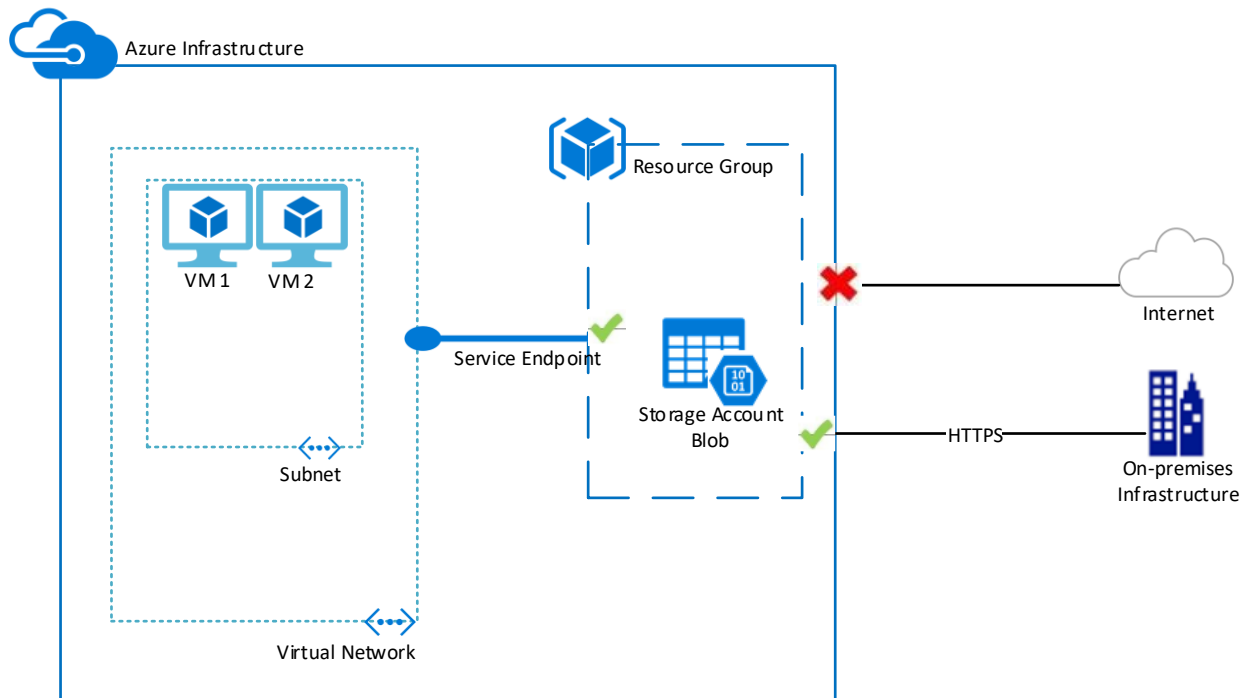
Features

- Optimized for storing massive amounts of unstructured data (text or binary data)
- Protects and secures your data
- Accessible using Azure Storage Explorer
- Achieve the unique scalability of the cloud paying for just what you need and grow into scale
- Choose from different SLA at a low-cost price
- Highly durable storage

Architecture



Example: Access the blob storage from on-prem or from an allowed VNET in your Azure subscription



Deploying the Solution

For this solution, we are deploying a **General-purpose v2 account** with some best practices, as well as additional configurations built into the deployment. This solution was designed to be used for data archiving, but if you are just wanting to use it for Hot or Cold Tier storage that is totally fine as well. The configuration is no different. Microsoft has a good article that explains the difference and depending on the tier the price per GB changes. You should review [this page](#) before building out this storage account for your customer.

“General-purpose v2 storage accounts support the latest Azure Storage features and incorporate all of the functionality of general-purpose v1 and Blob storage accounts. General-purpose v2 accounts deliver the lowest per-gigabyte capacity prices for Azure Storage, as well as industry-competitive transaction prices.”

Using a General-purpose v2 account gives us the option to expand the use case of the deployed storage account. At first you may just be using it to move archive data into the cloud, but as the customer moves more and more of their business into the cloud – you are ready to utilize all the additional Azure storage services on this Gen-purp v2 storage account, such as:

- Blobs (all types: Block, Append, Page)
- Files
- Disks
- Queues
- Tables

Parameters

Account Type: This sets the type of Storage account.

- **Standard_LRS** – Locally redundant storage with 99.999999999% (nine 9's) SLA. More info [here](#).
- **Standard_RAGRS** - Read-access Geo-redundant storage with 99.99999999999999% (16 9's) SLA. More info [here](#).

Account Tier: This sets the default tier of the Blob containers.

- **Hot** - Optimized for storing data that is accessed frequently.
- **Cold** - Optimized for storing data that is infrequently accessed and stored for at least 30 days

Enable Encryption: This sets the encryption on the storage account. The default is Enabled.

- **True** – Enable Encryption
- **False** – Disable Encryption (Not recommended)

Container One Name: This sets the name for the first preconfigured Blob container.

Container Two Name: This sets the name for the second preconfigured Blob container.

Container Access: By default, a container and any blobs within it may be accessed only by a user that has been given appropriate permissions. To grant anonymous users read access to a container and its blobs, you can set the container public access level. When you grant public access to a container, then anonymous users can read blobs within a publicly accessible container without authorizing the request.

- **Private** - The container and its blobs can be accessed only by the storage account owner. This is the default for all new containers.
- **Blob** - Blobs within the container can be read by anonymous request, but container data is not available. Anonymous clients cannot enumerate the blobs within the container.
- **Container** - All container and blob data can be read by anonymous request. Clients can enumerate blobs within the container by anonymous request but cannot enumerate containers within the storage account.

Default Access: If you select **Deny**, this will allow you to limit access to the Azure Storage Account to a public IP address/IP range that you specify. If you need to add additional IP addresses/ranges, you can do so in the storage account properties post-deployment.

- **Deny** – If Deny, specify an IP/IP range that you want to be able to access the storage account.
- **Allow** – If Allow, access to the storage account will not be limited to an IP address/IP range.

Limit IP Access: If setting Default Access to Deny, please specify your public IP address or IP range that you want to be able to access the blob's inside of the storage account. Examples: 97.96.157.22 or 97.96.157.0/24.

Post-Deployment

Azure Storage Explorer

One of the things that will make it easier for you to manage an Azure Storage account is their Storage Explorer application. To connect to your Storage Account using this tool you will need to obtain a connection string for the newly created storage account. We've wrote a PowerShell script to assist you in doing this. You'll need to update your \$rgName and \$saName variables. The values for these will be sent to you in the confirmation e-mail once the solution has successfully deployed.

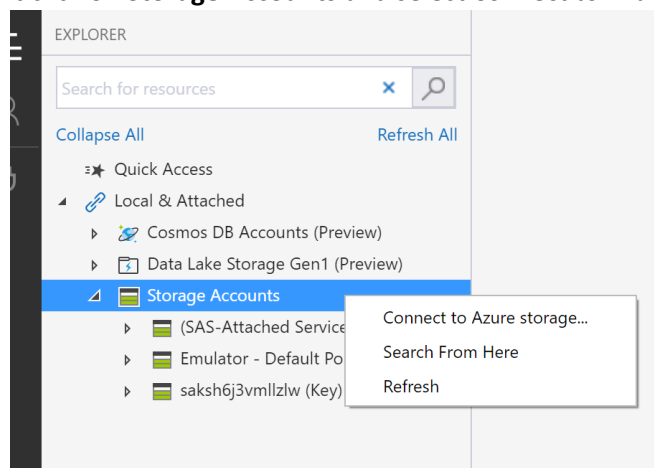
```
#Set Resource Group Name and Storage Account Name
$rgName = "RG-name "
$saName = "saName###"
#Login to AZ account
Login-AzAccount
#Build Storage Connection String - your string will be the last line of the script
output. Starting with 'Default' and ending with 'windows.net'.
$sa = Get-AzStorageAccount -StorageAccountName $saName -ResourceGroupName $rgName
$saKey = (Get-AzStorageAccountKey -ResourceGroupName $rgName -Name $saName)[0].Value
'DefaultEndpointsProtocol=https;AccountName=' + $saName + ';AccountKey=' + $saKey +
';EndpointSuffix=core.windows.net'
```

Once you run this script, the last line of the output will be the connection string that you will use to connect to your Azure Storage Account through Storage Explorer. It will look like this:

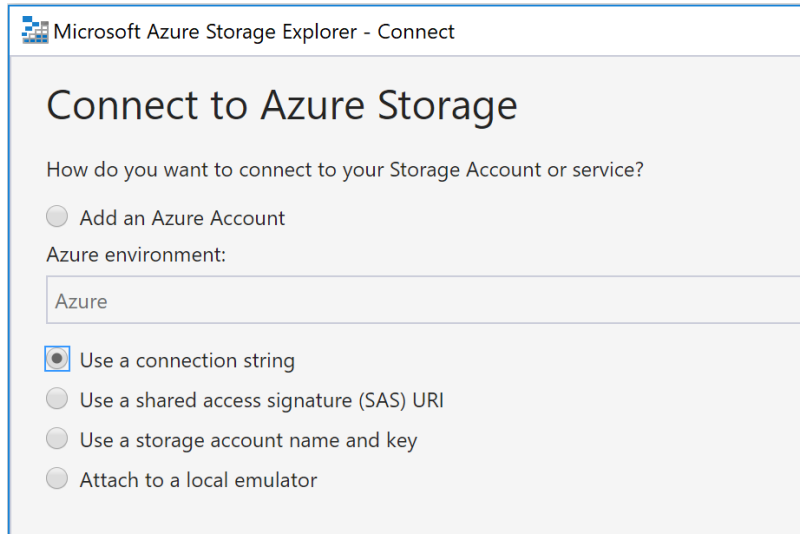
```
DefaultEndpointsProtocol=https;AccountName=saksh6j3vml1z1w;AccountKey=qR11Ywd/6w1h0dZ
4va00PgUm88F/X/NwTRnsS9ZR4a1MJG1mm5nqHY9ckXTmsCdbdFy0Q/sctwjR/VJoZ03yg==;EndpointSuffi
x=core.windows.net
```

You can download Azure Storage Explorer from here: <https://azure.microsoft.com/en-us/features/storage-explorer/>. Once installed, open the application.

1. Right click on **Storage Accounts** and select *Connect to Azure Storage*



2. Select *Use a connection string* and click *Next*.



Microsoft Azure Storage Explorer - Connect

Connect to Azure Storage

How do you want to connect to your Storage Account or service?

☐ Add an Azure Account

Azure environment:

Azure

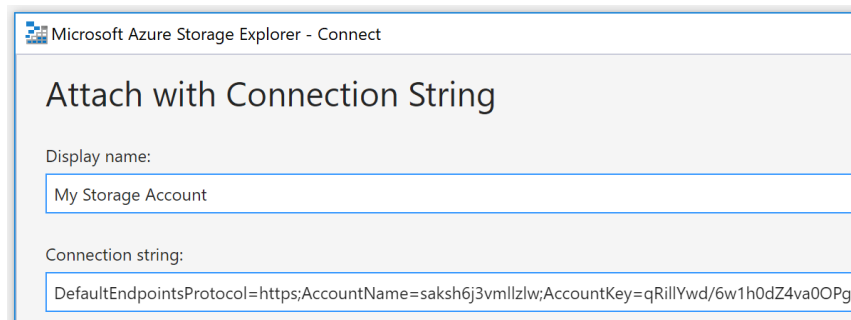
☒ Use a connection string

☐ Use a shared access signature (SAS) URI

☐ Use a storage account name and key

☐ Attach to a local emulator

3. Name your Storage Account connection, and then paste in the output of the PowerShell script you ran above, then click *Next*.
- 4.



Microsoft Azure Storage Explorer - Connect

Attach with Connection String

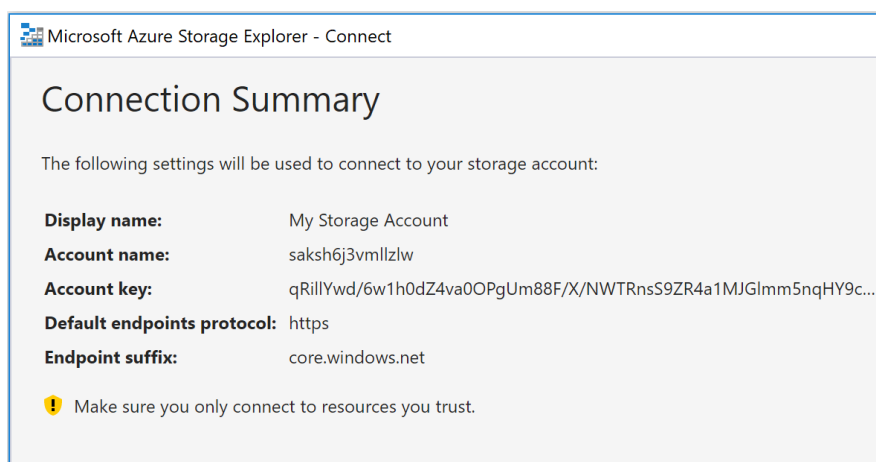
Display name:

My Storage Account

Connection string:

DefaultEndpointsProtocol=https;AccountName=saksh6j3vmlzlw;AccountKey=qRillYwd/6w1h0dZ4va0OPg

5. The Connection Summary page will display, showing the account you will be connecting to. You can verify this info, and then select *Connect*.



Microsoft Azure Storage Explorer - Connect

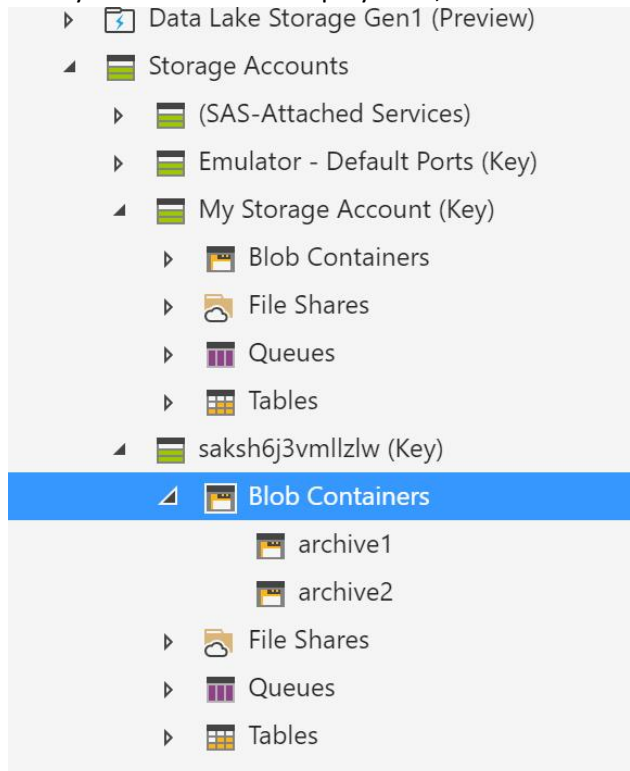
Connection Summary

The following settings will be used to connect to your storage account:

Display name:	My Storage Account
Account name:	saksh6j3vmlzlw
Account key:	qRillYwd/6w1h0dZ4va0OPgUm88F/X/NWTRnsS9ZR4a1MJGImm5nqHY9c...
Default endpoints protocol:	https
Endpoint suffix:	core.windows.net

⚠ Make sure you only connect to resources you trust.

6. Your newly created Storage Account will be added. You can see the two blob containers we've created by default with the deployment, and the files and folders underneath them.



Once you have your Storage Account added into Storage Explorer, you can more easily add/remove/modify the properties of the storage account. Doing such things as adding additional Blob containers, generating direct hyperlinks to them, modifying access, etc.

Setting containers to Archive Tier

Currently we cannot deploy containers as the Access Tier *Archive*. For this reason, we recommend that after you deploy the Azure Storage: Archiving click-to-run solution that you run the [PowerShell script below](#). Keep this script for your records as you will want to use it as a maintenance tool to ensure that the data in your blob containers gets set to the **Archiving** tier for cost savings.

Side Note: It's been an ongoing Azure feature request to have a blob container be able to be set as **Archiving** for good, but it's not in place yet. Marc Kean has done some great work with automating a script to do the same task that the one below does, but utilizing an Azure Automation Account to run it on a scheduled interval so it's more of an automated process, for more info on that visit [his blog](#).

```
### This code snippet is used to connect to an Azure Storage Account and tier all the
blobs to Azure Archive Storage Tier.
### This was converted to use Az cmdlets rather than AzureRm ones on 6/11/2019 by Ryan
Berger @ TechData
### You need PowerShell module Az to run the below commands
Install-Module Az -AllowClobber

### Import Azure PowerShell module
Import-Module Az

### Login to your Azure Account
Login-AzAccount
```

```

### Storage Account information

### In case you have several subscriptions select the proper one
Select-AzSubscription (Get-AzSubscription | Out-GridView -Title "Select your Azure
Subscription" -PassThru)

### Select the Azure Resource Group and Storage Account
$ResourceGroupName = (Get-AzResourceGroup | Out-GridView -Title "Select your
Resource Group" -PassThru).ResourceGroupName
$StorageAccount = Get-AzStorageAccount -ResourceGroupName $ResourceGroupName |
Out-GridView -Title "Select your Storage Account" -PassThru
$StorageAccountKey = (Get-AzStorageAccountKey -ResourceGroupName
$ResourceGroupName -Name $StorageAccount.StorageAccountName)[0].Value













### Create a storage context
$context = New-AzStorageContext -StorageAccountName
$StorageAccount.StorageAccountName -StorageAccountKey $StorageAccountKey
$containername = (Get-AzStorageContainer -Context $context | Out-GridView -Title
"Select your Storage Container" -PassThru).name
$blobs = Get-AzStorageBlob -Container $containerName -Context $context

### Use the .Net client library API to change the access tier
Foreach ($blob in $blobs) {
    $blob.ICloudBlob.SetStandardBlobTier("Archive")
}

```

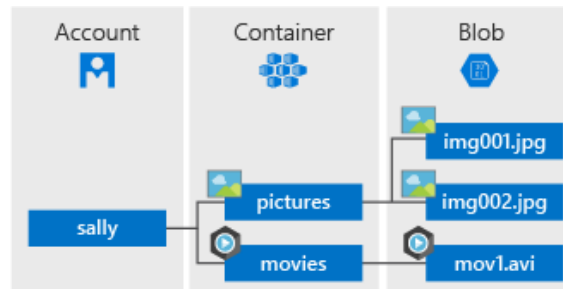
This will authenticate to your Azure tenant, allow you to select the subscription, resource group, and then storage account. It will then show you a list of the blob containers that exist, by default there will be two: archive1, and archive 2. Select one at a time and then the script will set all the existing files in that blob container to the **Archiving** tier.

As you can see below, I uploaded some files to my archive1 Blob container, and then ran the PowerShell script to set their tier to **Archive**. I then added some additional files, and they got put into the container as it's default Tier, in this case **Cool**. To also switch these files to **Archive**, I would need to run that PowerShell script again.

	_22_0850.bmp	Archive	6/12/2019, 9:57:44 AM
	_23_0849.bmp	Archive	6/12/2019, 9:57:45 AM
	_24_0848.bmp	Archive	6/12/2019, 9:57:45 AM
	_25_0847.bmp	Archive	6/12/2019, 9:57:45 AM
	_26_0846.bmp	Archive	6/12/2019, 9:57:46 AM
	_27_0845.bmp	Archive	6/12/2019, 9:57:46 AM
	_28_0844.bmp	Cool (inferred)	
	_29_0843.bmp	Cool (inferred)	
	_30_0842.bmp	Cool (inferred)	
	_31_0841.bmp	Cool (inferred)	
	_32_0840.bmp	Cool (inferred)	
	_34_0838.bmp	Cool (inferred)	

Components

The graphic below will help you understand the Storage Account vs. Container vs. Blob relationship. Think of the Storage Account as the file server, the Container as the share on the file server, and the blob's as the individual files in that share.



Storage Account

All access to data objects in Azure Storage happens through a storage account. It contains all your Azure Storage data objects (blobs in this case). Data is durable and highly available, secure, massively scalable, and accessible from anywhere in the world over HTTP or HTTPS. There are 3 types of storage accounts and the one used for this solution is a General-purpose v2.

Container

Organizes a set of blobs, similar to a folder in a file system. A storage account can contain an unlimited number of containers, and a container can store an unlimited number of blobs.

Blob

Blobs (Block) store text and binary data, up to about 4.7 TB. Block blobs are made up of blocks of data that can be managed individually.

Pre-Requisites

- Enrolled under CSP program with Tech Data
- Azure Subscription

Default Limits

Resource	Default Limit
Number of storage accounts per region per subscription, including both standard and premium accounts	200
Max storage account capacity	500 TiB (<i>5 PB hard limit</i>)
Max number of blob containers, blobs, file shares, tables, queues, entities, or messages per storage account	No limit
Maximum request rate per storage account (IOPS)	20,000 requests per second (<i>50,000 hard limit</i>)
Max ingress per storage account (US Regions)	10 Gbps if RA-GRS/GRS enabled, 20 Gbps for LRS/ZRS (<i>50 Gbps hard limit</i>)
Max egress per storage account (US Regions)	20 Gbps if RA-GRS/GRS enabled, 30 Gbps for LRS/ZRS (<i>50 Gbps hard limit</i>)
Max ingress per storage account (Non-US regions)	5 Gbps if RA-GRS/GRS enabled, 10 Gbps for LRS/ZRS (<i>50 Gbps hard limit</i>)
Max egress per storage account (Non-US regions)	10 Gbps if RA-GRS/GRS enabled, 15 Gbps for LRS/ZRS (<i>50 Gbps hard limit</i>)

NOTE:

- The *ingress* limit refers to all data (requests) being sent to a storage account. The *egress* limit refers to all data (responses) being received from a storage account.
- Azure storage accounts support higher limits for capacity, request rate, ingress, and egress by request. To request an increase in account limits, contact Azure Support.
- If RA-GRS is enabled, egress targets for the secondary location are identical to those for the primary location.

Resource	Target
Max size of single blob container	Same as max storage account capacity
Max number of blocks in a block blob or append blob	50,000 blocks
Max size of a block in a block blob	100 MiB
Max size of a block blob	50,000 X 100 MiB (approx. 4.75 TiB)
Max size of a block in an append blob	4 MiB
Max size of an append blob	50,000 x 4 MiB (approx. 195 GiB)
Max size of a page blob	8 TiB
Max number of stored access policies per blob container	5
Target throughput for single blob	Up to 60 MiB per second, or up to 500 requests per second