Table of Contents

[AZURE STORAGE SERVICE 3](#_Toc134470382)

[AZURE STORAGE ACCOUNT 3](#_Toc134470383)

[TYPES OF STORAGE ACCOUNT 3](#_Toc134470384)

[GPV2 CORE STORAGE SERVICES 4](#_Toc134470385)

[CREATING GENERAL PURPOSE V2 STORAGE ACCOUNT 4](#_Toc134470386)

[BENEFITS OF STORAGE SERVICES 4](#_Toc134470387)

[VM DISK AND STORAGE ACCOUNT 5](#_Toc134470388)

[STORAGE ACCOUNT END POINTS 5](#_Toc134470389)

[DATA REDUNDANCY IN AZURE STORAGE ACCOUNT 5](#_Toc134470390)

[BLOB STORAGE 13](#_Toc134470391)

[STEPS TO CREATE BLOB STORAGE 14](#_Toc134470392)

[AUTORIZATION TECHNIQUES IN AZURE STORAGE ACCOUNT 16](#_Toc134470393)

[ACCESS TIERS FOR BLOB SERVICE 24](#_Toc134470394)

[BLOB VERSIONING 31](#_Toc134470395)

[SOFT DELETE 32](#_Toc134470396)

[BLOB STORAGE – PROMGRAMATIC APPROACH 32](#_Toc134470397)

[AZ COPY TOOL (COMMAND LINE) 34](#_Toc134470398)

[AZURE BLOB CHANGE FEED 36](#_Toc134470399)

[TABLE SERVICE 37](#_Toc134470400)

[TABLE STORAGE COMPONENTS 37](#_Toc134470401)

[QUEUES SERVICE 39](#_Toc134470402)

[EXAMPLE 41](#_Toc134470403)

[FILE SHARE SERVICE 42](#_Toc134470404)

[NEED OF FILE STORAGE 42](#_Toc134470405)

[WHAT IS FILE STORAGE? 42](#_Toc134470406)

[USE CASE OF FILE STORAGE? 42](#_Toc134470407)

[BENEFITS OF FILE STORAGE? 42](#_Toc134470408)

[CREATING A AZURE FILE SHARE 43](#_Toc134470409)

[UPLOADING DATA IN FILE SHARE 43](#_Toc134470410)

[MOUNTING THE FILE SHARE TO A DRIVE 43](#_Toc134470411)

[AZURE FILE SYNC 44](#_Toc134470412)

[AZURE COSMOS DATABASE 46](#_Toc134470413)

[COSMOS DB API 46](#_Toc134470414)

[COSTING 46](#_Toc134470415)

[PROVISIONING MODE OF COSMOS DB 47](#_Toc134470416)

[PARTITIONS IN COSMOS DB 47](#_Toc134470417)

[CREATING A COSMOS DB ACCOUNT 48](#_Toc134470418)

[SELECTING COSMOS DB API 49](#_Toc134470419)

[COSMOS DB ACCOUNT CONFIGURATION 49](#_Toc134470420)

[CREATING A COSMOS DATABASE AND CONTAINER 50](#_Toc134470421)

[CREATING A COSMOS DATABASE AND CONTAINER 50](#_Toc134470422)

[QUERYING CONTAINER DATA 51](#_Toc134470423)

[AZURE SQL DATABASE 52](#_Toc134470424)

[CREATING A SQL DATABASE SERVER 52](#_Toc134470425)

[DATA MIGRATION 54](#_Toc134470426)

[OFFLINE DATA MIGRATION 55](#_Toc134470427)

[ONLINE DATA MIGRATION 55](#_Toc134470428)

# AZURE STORAGE SERVICE

* This suite of cloud-based Microsoft-managed storage services mainly comprises four types of storage services in Azure



# AZURE STORAGE ACCOUNT

* It’s a cloud based stoareg solution by Microsift . Hence to use the storage services – we need a storage account.
* Using this account, we can manage and access the storage resources
* All our storage data, including blobs, files, queues, and tables, resides in our storage account
* A storage account provides a unique namespace for our Azure Storage data, that's accessible from anywhere in the world over HTTP or HTTPS. Data in this account is secure, highly available, durable, and massively scalable.
* Azure Storage is also used by infrastructure as a service virtual machine, and platform as a service cloud services.
* 5 PB(1 Peta Byte = 1025 TB) is available for each storage account.

## TYPES OF STORAGE ACCOUNT

* **Microsoft offers multiple types of storage accounts, each capable of handling different types of storage data**

PERFORMANCE BASED CLASSIFICATION OF STORAGE ACCOUNT

|  |  |  |
| --- | --- | --- |
|  | | *Based on Performance – The storage accounts are classified in 2 types*   1. GENERAL PURPOSE STORAGE ACCOUNT 2. PREMIUM STORAGE ACCOUNT |
| **GENERAL PURPOSE ACCOUNT V2 (GPV2)** | The GPV2 Storage Account contains following service   * + BLOB SERVICE   + FILE SHARE SERVICE   + QUEUE SERVICE   + TABLE SERVICE | |
| ` | Types of Premium Storage Account   1. BLOCK BLOBS 2. PAGE BLOBS 3. FILE SHARES | |
| **PREMIUM BLOCK BLOBS ACCOUNT** | * This is specific storage account only meant for Blobs * These premium account gives a fast access to the blobs, high transaction rates | |
| **PREMIUM PAGE BLOBS ACCOUNT** | * Premium page blobs are used for storing the virtual hard disk of Azure VMs (VHD) | |
| **PREMIUM FILE SHARE ACCOUNT** | * This is used when we want fast access of files with high transaction rates | |

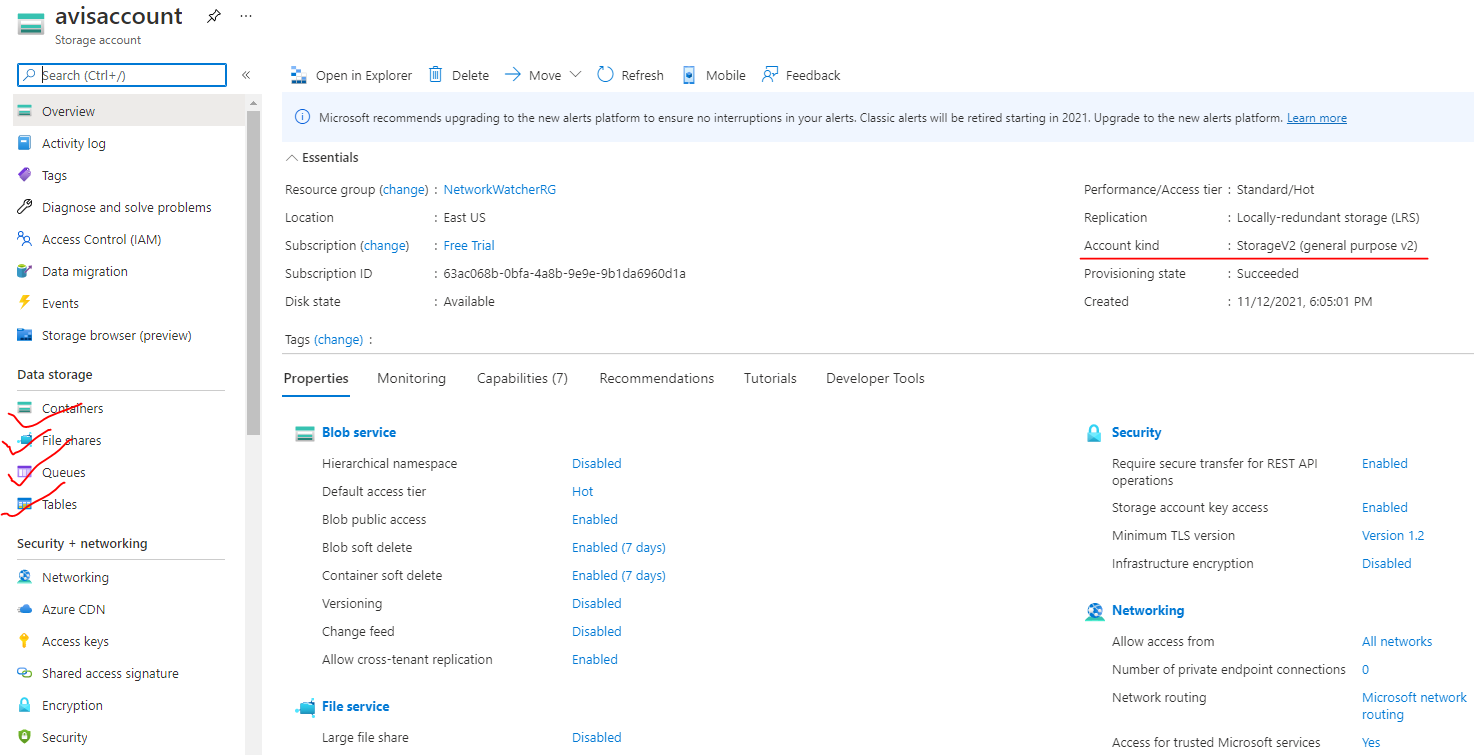
## GPV2 CORE STORAGE SERVICES

|  |  |
| --- | --- |
| BLOB STORAGE | * Azure Blob Storage is an object storage solution that we can use to store massive amounts of unstructured data, such as text or binary data. * Blob Storage is ideal for serving images or documents directly to a browser, storing data for archives or distributed access, streaming video and audio, and disaster recovery scenarios. |
| TABLE STORAGE | * Azure Table Storage offers a NoSQL data store for key value pairs using large scale datasets. * We can use Azure Table Storage to store petabytes of semi-structured data. |
| QUEUE STORAGE | * Azure Queue Storage provides asynchronous message queuing for communication between application components,whether they're running in the cloud, on the desktop,on premises, or on mobile devices |
| FILE STORAGE | * Azure File Storage offers fully managed file shares in the cloud, * They accessible using industry standard network protocols. * Mounting Azure file shares is just like connecting to shares on the local network. |
| DISK STORAGE | * Azure Disk Storage provides disks for virtual machines and applications to access and use as they need - similar to how they would access disks that were on premises. * Azure offers both solid state drives for higher performance workloads and conventional hard drives for your less critical business scenarios. |

### CREATING GENERAL PURPOSE V2 STORAGE ACCOUNT

* Create Resource 🡪 Storage Account 🡪Give the unique storage Account name
* The Storage account name must be unique because it – the storage account resources have unique URLs

|  |  |
| --- | --- |
| * *Standard storage accounts are backed by Standard Hard Disk Drive (HDD)* * *Premium Storage account are backed by Solid State Drive (SSD)* |  |



### BENEFITS OF STORAGE SERVICES

|  |  |
| --- | --- |
| SECURITY | Azure provides top-notch security as data stored or written in Azure Storage is encrypted. Azure Storage offers full control over who can and cannot access our data |
| ACCESSIBILITY | The data stored in Microsoft Azure Storage is made accessible over HTTP or HTTPS from anywhere in the world |
| SCALABILITY | Azure Storage is highly scalable to meet the on-demand requirements of modern applications |
| HIGH AVAILABILITY | Users are given the option of replicating their data across multiple data centers so that the data stays available even in the event of outage |

### VM DISK AND STORAGE ACCOUNT

|  |  |
| --- | --- |
|  | * Behind the scene – Azure stores the Managed Disks (OS and Data disks) in a storage account . * VM disks are part of page blob service of the storage account. * The managed disk of VM are managed by Azure itself * GPV2 account are backed by HDD on other hand – Premium stoarage account are backed by SSD(Better performance) |

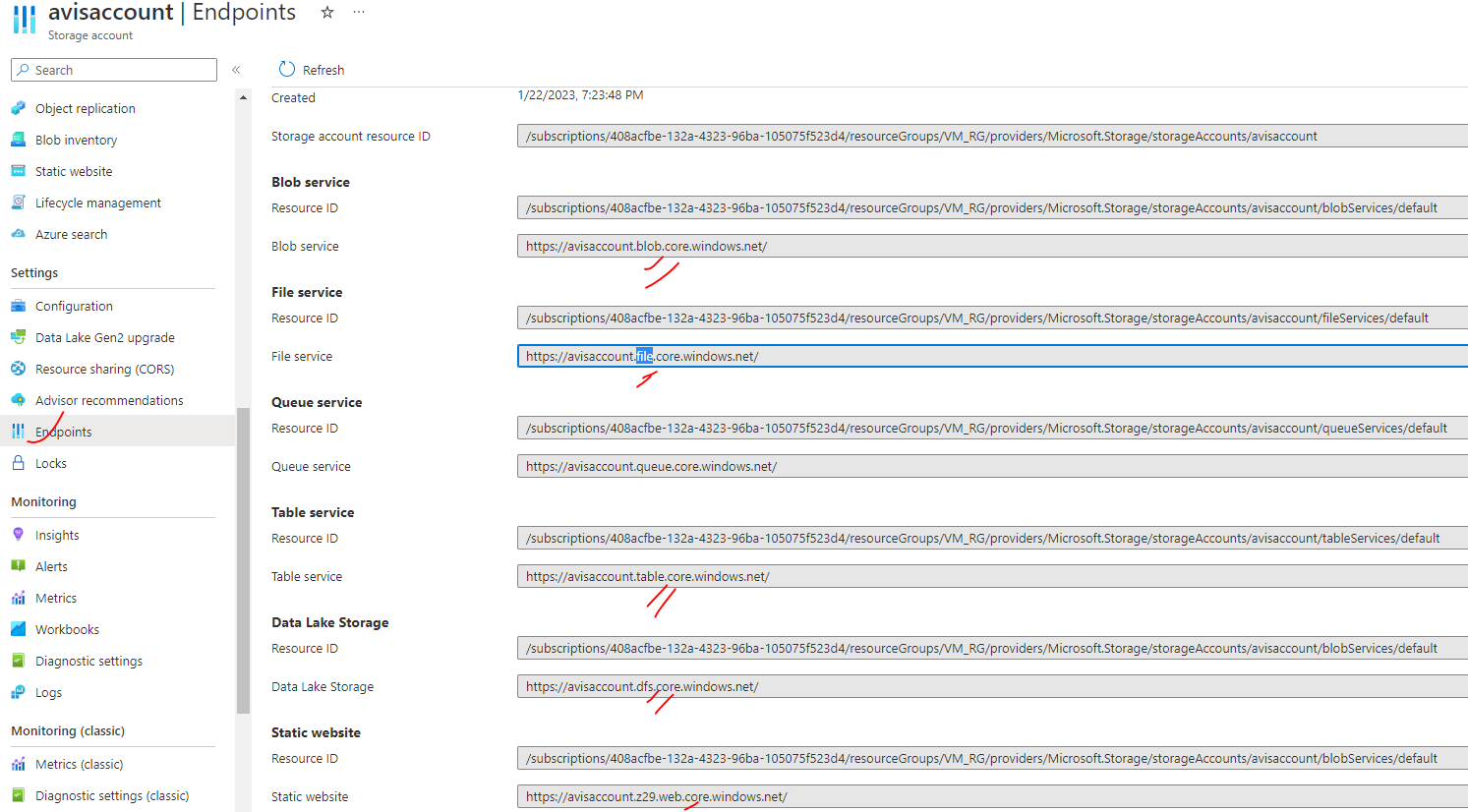
### STORAGE ACCOUNT END POINTS

* Each service in stoareg

**URL PATTERN**

* ***https:// <STORAGE\_ACCOUNT\_NAME>.<SERVICE\_NAME>/<CONTAINER\_NAME>/<BLOB\_OBJECT>***
* **EXAMPLE -** [**https://appstoreaccount.blob.core.windows.net/container/image.JPG**](https://appstoreaccount.blob.core.windows.net/container/image.JPG)

Since we are using blob service the service name is “**blob**.core.windows.net”. If we use file service – it will be “**file**.core.windows.net”



### DATA REDUNDANCY IN AZURE STORAGE ACCOUNT

#### WHAT IS REDUNDANCY?

|  |  |
| --- | --- |
|  | * When we store a data in a storage account – it is basically get stored in a physical storage device. * When azure store in the data in the storage devices – it stores multiple copies of the data * This helps in protecting the planned and unplanned events, transient hardware failures, network, and power outage. |
|  | The different replication options for Azure Storage account   1. LOCALLY REDUNDANT STORAGE 2. ZONE-REDUNDANT STORAGE 3. GEO-REDUNDANT STORAGE 4. READ ACCESS GEO REDUNDANT STORAGE 5. GEO-ZONE-REDUNDANT STORAGE 6. READ ACCESSS GEO-ZONE-REDUNDANT STORAGE |
| For read access type account we need to check the below checkbox- while creating the storage account (Advanced Tab) | |

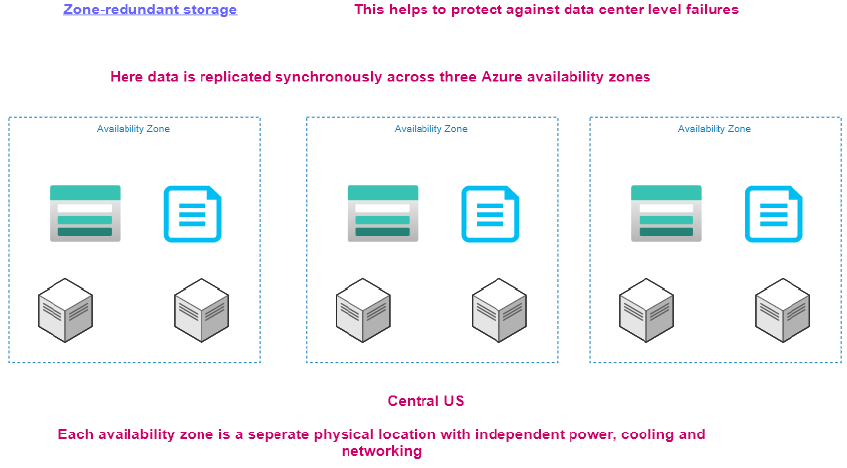
#### LOCALLY REDUNDANT STORAGE(LRS)

|  |  |
| --- | --- |
|  | * The data is replicated into 3 different storage devices/servers (on a different rack) within a single data center. Hence data will have 3 copies of data in data center * Hence event if there is failure (like rack or server failure) in one of the storage devices – the data will be still available. * Copying of data happens synchronously (Synchronous operations require that one operation must wait for another to complete before it can begin) * We get a success message in Azure Portal – only if all the 3 copies are replicated. |

#### ZONE-REDUNDANT STORAGE(ZRS)

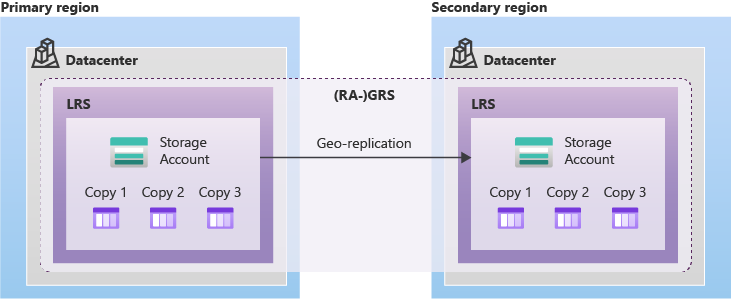
Zone always refers to an availability zone.

* Availability zones are physically separate datacenters within an Azure region.
* Each availability zone is made up of 3 or more datacenters. Each datacenter is equipped with independent power, cooling, and networking.
* An availability zone is set up to be an isolation boundary. If one zone goes down, the other continues working.
* Availability zones are connected through high-speed, private fiber-optic networks

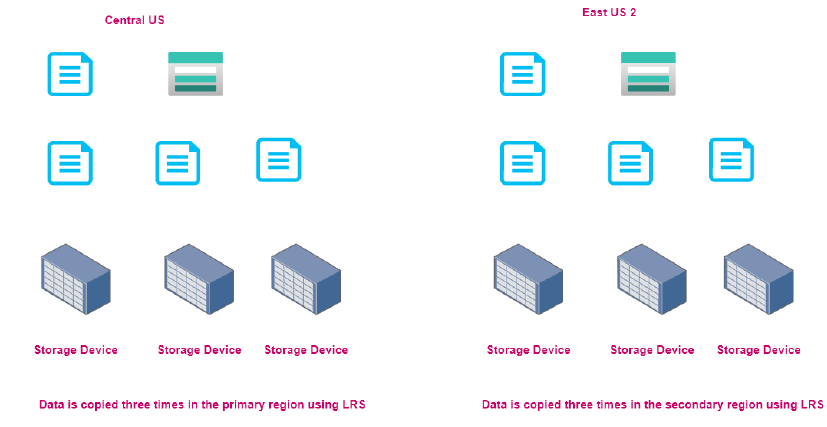


|  |  |
| --- | --- |
|  | * This help in protecting against failure in a data center. * Here also we have 3 copies of data i.e., one in each zone. Hence the data will be available even if a data center fails. * Since each availability zone resides in separate physical location hence even if the one of the zones not available, the data will be from other zone with a availability zone. * The replication in the zones happens synchronously |

#### GEO-REDUNDANT STORAGE(GRS)

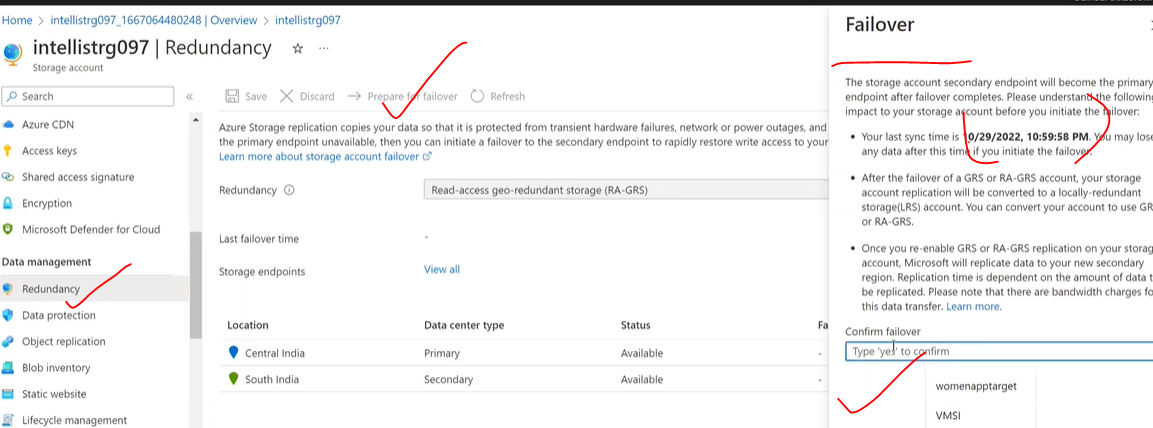


* This redundancy option protect against the region level failure.
* The data is replicated between paired zone (primary and secondary region) asynchrounously.
* Since the replication is asynchronous, the delay in replication from Primary to secondary maximum upto 16 mins.
* ***The date with a data center are replicated using LRS.i,e. 3 copies of the data will be present within a data center***
* Paired zone reference : <https://learn.microsoft.com/en-us/azure/availability-zones/cross-region-replication-azure#azure-cross-region-replication-pairings-for-all-geographies>



* If the primary data center is down – the data cannot be read/write to secondary location immediately.
* To read/write data from secondary location - failover must happen otherwise data cannot be read / write from secondary region.
* Failover can be “Automatic” or “Manual”
  + *AUTOMATIC FAILOVER* – Automatic failover happen – when Microsoft identify the issue in a storage account.
  + *MANUAL FAILOVER* – When a customer identifies an issue in the storage account – they too can trigger the failover from the Storage account.
* After failover happen – the storage account will be converted into LRS in the secondary region. But if we try to convert the storage account to GRS – then the previous primary region will be the new secondary region.

TRIGGERING A FAILOVER



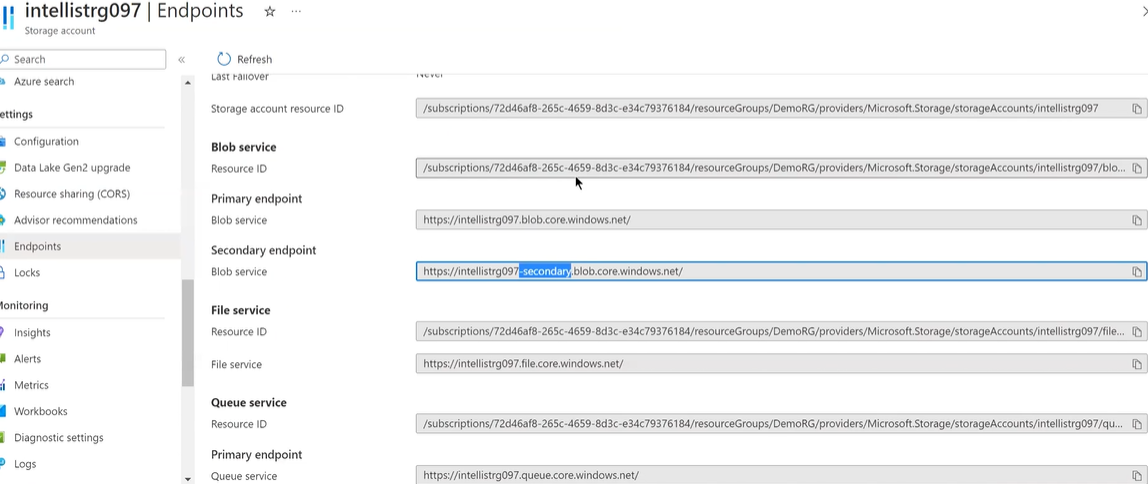
Note :

Before Failover

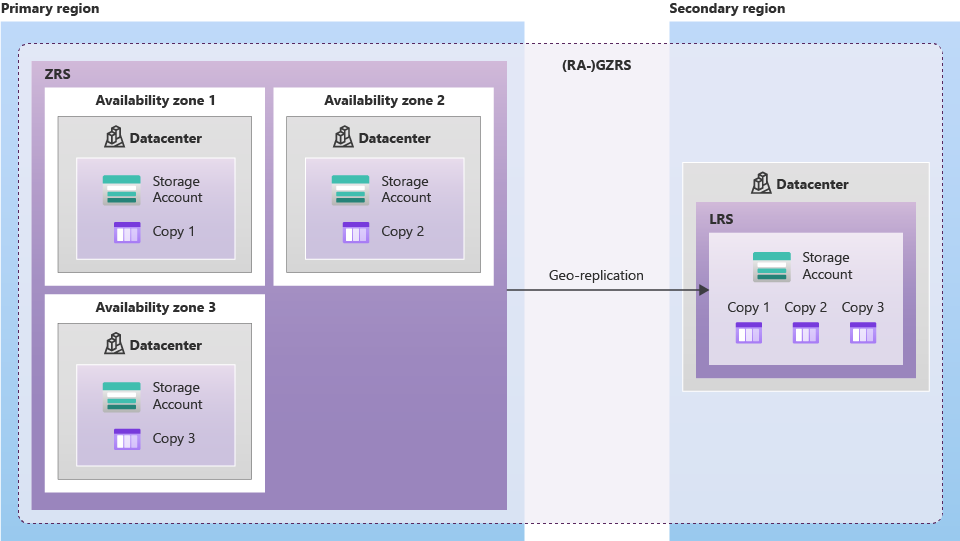
After Failover

#### READ ACCESS GEO REDUNDANT STORAGE (RAGRS)

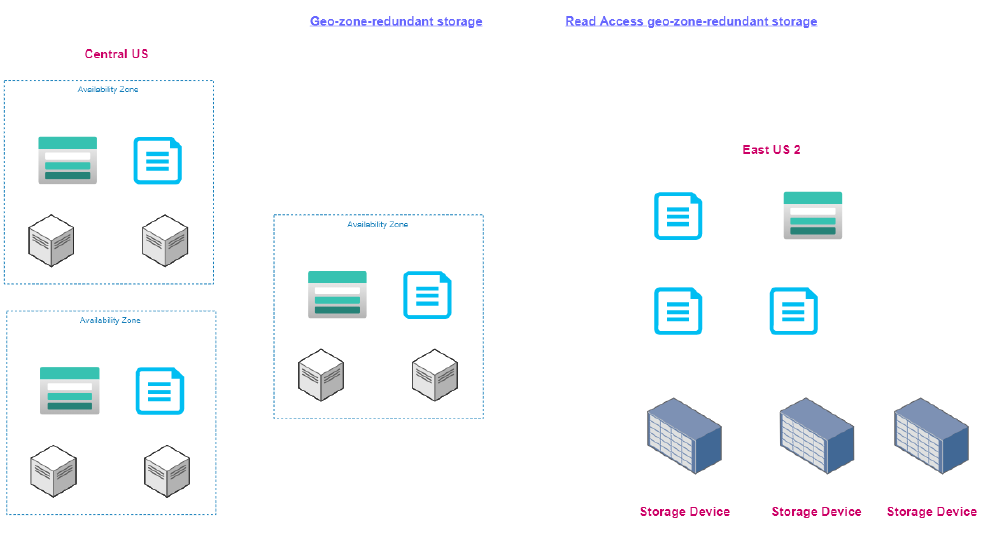
* It’s more like GRS – with a difference that with even initiating a failover- by default a read access has been given at the secondary region.
* In RAGRS – unlike GRS the data will be available for read operation the data is made available from the secondary region if the primary region goes down. Hence, we don’t have to wait for the failover process to finish, at least for read operation.
* Once the failover is triggered – we can be able to read + write in the secondary region. The storage account will be converted to LRS (like GRS)
* The RA- GRS expose 2 different endpoints one for primary and another for secondary. Hence the custom application must be designed in a way such that when primary goes down – we can be able to switch the endpoint to secondary (Just for read operations)



#### GEO-ZONE-REDUNDANT STORAGE



* The data is first replicated to availability zones in a primary region. Hence there will be 3 copies of in a primary region, i.e one copy of data in each zone.
* The data gets replicated to secondary zone in asychronously.
* In the secondary region 3 copies of data get created in the same data center i.e. LRS



BENEFITS OF GZRS OVER GRS

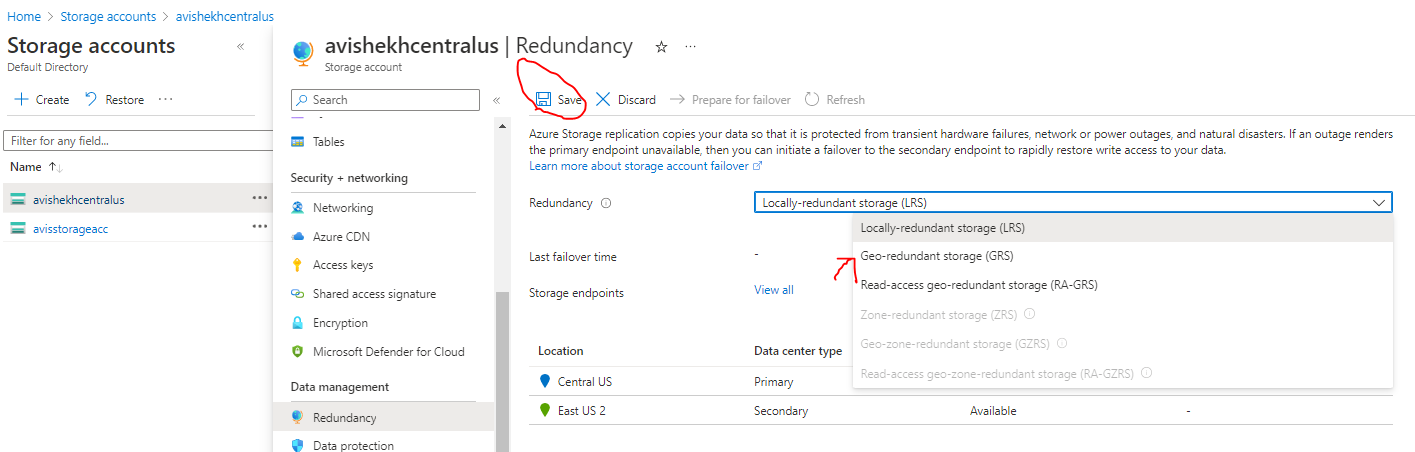
* *In GRS – if the primary data center goes down – we need to trigger a failover to start the read/ write operation in the secondary region.*
* *Unlike GRS, GZRS its very unlikely where we need to trigger the failover because we have extra redundancy in the primary region itself- because the primary will be having data redundancy at availability zone level (ZRS)*

#### READ ACCESS GEO-ZONE-REDUNDANT STORAGE

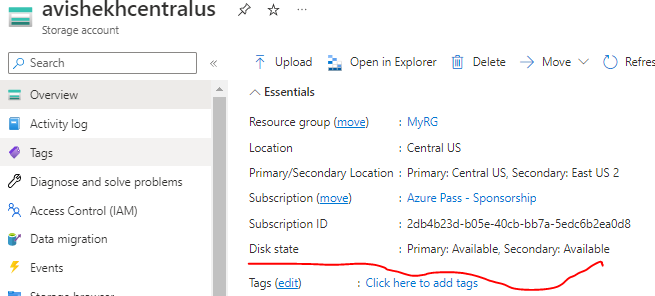
* In RAGZRS – secondary zone will have read access rest other behavior will remain same as of RAGRS

#### CONFIGURING REDUNDANT STORAGE

* To configure data redundancy for storage account, Go to Storage account 🡪 Select Redundancy
* Let’s select “Geo Redundant Storage”



* Since we selected Geo-redundant storage -Hence data will be replicated to paired zone(as shown below)



#### OBJECT REPLICATION

* **This feature can be used to copy blobs between a source and destination storage account.**
* You can create rules to specify which objects get replicated from the source to the destination.
* This feature is supported by –General Purpose V2 and Premium Blob storage accounts.

NOTE :

* *Blob versioning should be enabled on both the source and destination storage account.*
* *Change feed is enabled on the source storage account.*

|  |  |
| --- | --- |
|  | * Let’s say we have 2 storage account – both are in different locations  1. [***avisstorageacc***](https://portal.azure.com/#@amitsinhaazuregmail.onmicrosoft.com/resource/subscriptions/2db4b23d-b05e-40cb-bb7a-5edc6b2ea0d8/resourceGroups/MyRG/providers/Microsoft.Storage/storageAccounts/avisstorageacc) – Source Account from where we will copy the blobs(East US ) 2. [***avidestinationstorage***](https://portal.azure.com/#@amitsinhaazuregmail.onmicrosoft.com/resource/subscriptions/2db4b23d-b05e-40cb-bb7a-5edc6b2ea0d8/resourceGroups/MyRG/providers/Microsoft.Storage/storageAccounts/avidestinationstorage) – Destination storage account where the blobs will be copied (Central US)  * We have containers in both source and destination storage accounts |
| STEP1 : ENABLE BLOB VERSION IN BOTH SOURCE AND DESTINATION STORAGE ACCOUNT | |
| STEP 2:ENABLE CHANGE FEED IN SOURCE STORAGE ACCOUNT | |
| STEP 3: GO TO OBJECT REPLICATION OF SOURCE STORAGE ACCOUNT   * Create Replication Rule * Select Destination Storage Account * Select the container from source storage and container of destination storage account * The blobs will be copied to destination container. | |

## BLOB STORAGE

|  |  |
| --- | --- |
| * **Azure Blob Storage is an object storage solution for the cloud. It can store massive amounts of data, such as text or binary data**. * Azure Blob Storage is **unstructured**, meaning that there are no restrictions on the kinds of data it can hold. | * Blob Storage can manage thousands of simultaneous uploads, massive amounts of video data, constantly growing log files, and can be reached from anywhere with an internet connection.   **Blob Storage is ideal for:**   * Serving images or documents directly to a browser. * Storing files for distributed access. * Streaming video and audio. * Storing data for backup and restore, disaster recovery, and archiving. * Storing data for analysis by an on-premises or Azure-hosted service. * Storing up to 8 TB of data for virtual machines. |

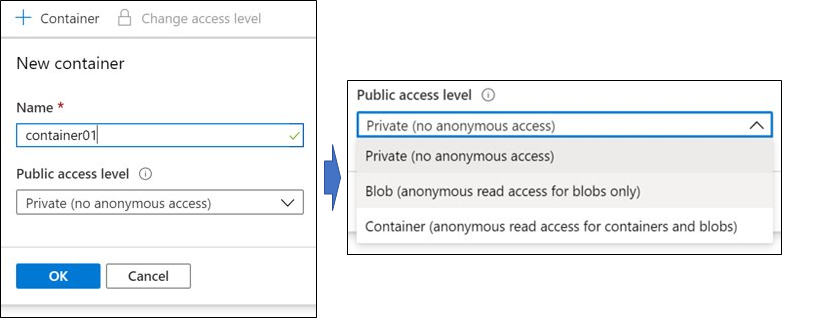
### STEPS TO CREATE BLOB STORAGE

* Containers act as a root folder for all the binary object we are going to store as blobs .
* The containers helps as to organize the blobs.

#### CREATING A CONTAINER AND UPLOADING BLOBS

|  |  |
| --- | --- |
|  |  |
|  | |
| * Enter the name of the container. * The container will hold all the blobs like image and videos * Using the upload option, we can be able to upload the images and videos * In the advanced options, if the folder name is given. Then a folder will be created with the same name and new file will be uploaded in the folder * Note: ***Anything uploaded in a container will have unique URL.***   **URL PATTERN**   * ***https:// <STORAGE\_ACCOUNT\_NAME>.<SERVICE\_NAME>/<CONTAINER\_NAME>/<BLOB\_OBJECT>*** * **EXAMPLE -** [**https://appstoreaccount.blob.core.windows.net/container/image.JPG**](https://appstoreaccount.blob.core.windows.net/container/image.JPG) * Since we are using blob service the service name is “**blob**.core.windows.net”. If we use file service – it will be “**file**.core.windows.net” | |

##### ACCESS LEVELS OF BLOBS



Public access level: Specifies whether data in the container may be accessed publicly. By default, container data is private to the account owner.

* Use **Private** to ensure there is no anonymous access to the container and blobs.
* Use **Blob** to allow anonymous public read access to individual blobs only.

|  |  |
| --- | --- |
|  | Use **Container** Access Level   * To allow anonymous public to read and list access to the entire container, including the blobs * URL to list the Blobs in the container – This will list only the blobs in XML format. Append and page blobs will not eb part of the list * <https://demostgacc.blob.core.windows.net/test?res=directory&comp=list> |

##### TYPES OF BLOBS

|  |  |
| --- | --- |
| Screenshot of the Upload Blob page. The Advanced section with Authentication type, blob types, and block size. | Azure Storage offers three types of blobs:   1. BLOCK BLOBS, 2. PAGE BLOBS, 3. APPEND BLOBS.   We specify the blob type when we are creating the blob object   * **BLOCK BLOBS (default):**  consist of blocks of data assembled to make a blob. Most scenarios using Blob storage employ block blobs. Block blobs are ideal for storing text and binary data in the cloud, like files, images, and videos. * **APPEND BLOBS:** are like block blobs in that they are made up of blocks, but they are optimized for append operations, so they are useful for logging scenarios. * **PAGE BLOBS**:   + Can be up to 8 TB in size   + Efficient for frequent read/write operations.   + The Page blob is blob which are divided into Pages each having fixed size of 512 bytes. Azure virtual machines use page blobs as OS and data disks.     The reason -why page blobs are designed so -because as this type of storage is used to store VM disk - this help in faster read / write operation of the VM disk from the random locations (pages) |

### AUTORIZATION TECHNIQUES IN AZURE STORAGE ACCOUNT

|  |  |
| --- | --- |
|  | **If a user or an application want to use a service within the Storage account. They need to authorize themselves. There are multiple techniques by which they can be able authorize themself**   * **ACCESS KEYS** * **SHARED ACCESS SIGNATURE** * **AZURE ACTIVE DIRECTORY** |

#### USING ACCESS KEYS

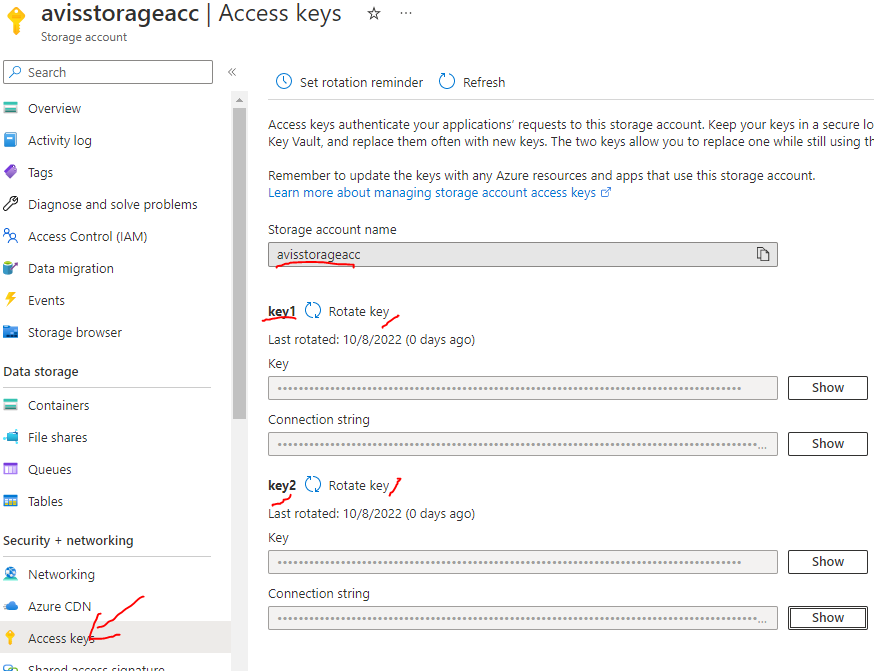
* The access to storage account can be given to another user via access key as well.

***Note: When we use access key -we are giving access to all the services within the storage account***

* To view the access key associated to a storage account 🡪 Go to Access Keys in the left navigation.

**WHY WE HAVE 2 KEYS ASSOCIATED TO A STORAGE ACCOUNT?**

* Multiple access keys are useful – if the 1st key is compromised (i.e., if it goes into wrong hand).
* In those scenarios – we switch to key-2 and “Rotate” the key1 (using rotate key option) to make the previous unusable.



##### ACCESS THE STORAGE ACCOUNT USING ACCESS KEY - STORAGE EXPLORER

|  |  |
| --- | --- |
|  | * Open Storage Explorer 🡪 View 🡪 Account Management * Select 🡪 “Use Storage Account name and key” * Provide the details * ***Display Name:*** The name that will appear to the user * ***Account name***: Name of the storage account to which we want user to connect with * ***Account key***: The account key (either key1 or key2)   *Note: When we use access key -we are giving access to all the services within the storage account* |
|  |

#### SHARED ACCESS SIGNATURE

|  |  |
| --- | --- |
|  | The shared access signature access authorization method can be used –   * When we want to give an access to Blob/ storage account for specific time. * Or Access to a specific IP (specific host machine)   Shared access signature can be used to give an access at   * **BLOB LEVEL** * **STORAGE ACCOUNT LEVEL** |

##### SHARED ACCESS SIGNATURE – BLOB LEVEL

* To make use of SAG for a Blob object 🡪 Select the blob object 🡪 Shared Access Tokens

|  |  |
| --- | --- |
| SIGNING KEY | This is the key(key1 or key 2) associated with the storage account |
| START AND EXPIRY | This is the timeline when we want to give the access on the BLOB |
| ALLOWED IPS | The list of IPs of the machine – which can be able to access the blob object |

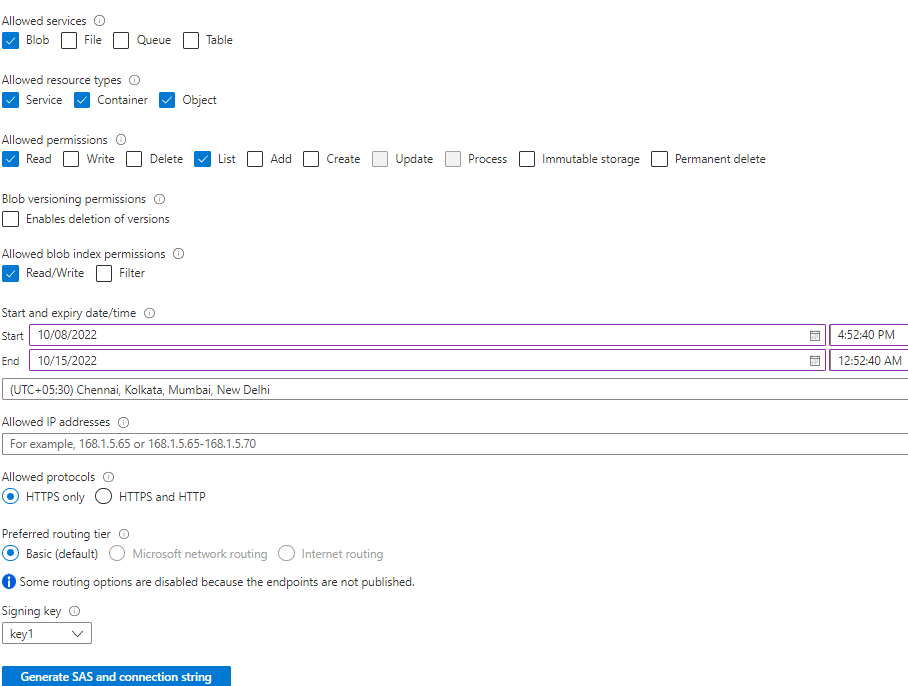
* Generate SAS token and URL based on above details.

**Copy the BLOB SAS URL to access the BLOB using browser. Note the validity of SAS till the expiry date (if it is set)**

|  |
| --- |
|  |

##### SHARED ACCESS SIGNATURE – STORAGE ACCOUNT LEVEL

* Navigate to storage account



|  |  |
| --- | --- |
| ALLOWED SERVICES | * Allowed service of the storage account * In above example – We are allowing only Blob service |
| RESOURCE TYPES | Allowed Resource types. In above example we allowed the service, container, and blob object |
| ALLOWED PERMISSION |  |
| START AND EXPIRY DATE | This is the timeline when we want to give the access on the BLOB |
| ALLOWED IP ADDRESS | The list of IPs of the machine – which can be able to access the blob object |
| SIGNING KEY | This is the key (key1 or key 2) associated with the storage account |

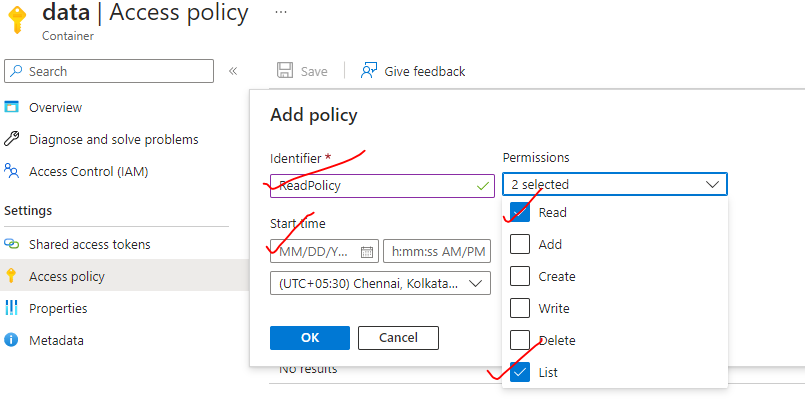
|  |  |
| --- | --- |
|  | * With the above access we can be able to connect with storage account with the STORAGE EXPLORER * Step 1: Capture the BLOB SAS URL to connect with storage account using Explorer |
|  | * Paste the SAS URL 🡪 Next * After we connect to Storage explore – we can observe that only blob services are available with only read permission (we cannot able to upload a blob using storage explorer) |

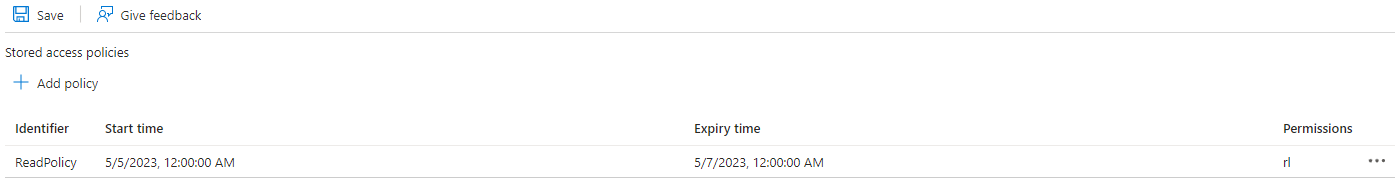
#### STORED ACCESS POLICY

* Shared access signature gives us a more finer control when it comes to authorization on Azure storage account. But if in case shared access is compromised.
* In such cases – **we can invalidate a SAS using stored access policy** (they are attached to that shared access signature)

STEP 1: CREATE THE ACCESS POLICY

* Go to storage Account 🡪 Container🡪 Access Policy 🡪 Give the required permission and Start





STEP 2: CREATE THE ACCESS POLICY

|  |  |  |
| --- | --- | --- |
|  |  | |
|  | |  |

We basically have two types of policies.

Now.

I'll go on to a small storage explorer.

And yeah, I'm logged in basically as my as the admin account.

I'll go on to my storage account.

Yeah.

Then I'll go onto my data container.

All right.

Click All Tools to get a shared access signature.

I now here I can actually assign or use that access policy so that access policy only has a set of permissions

in place and then hit on create.

So now we have this shared access signature in place.

I'm just going to copy the U.

Auto and then close this.

Yeah.

I'll quickly disconnect from the earlier shared access and ensure that I have.

Yeah.

I'll add an account this time.

I need to choose a block container.

I'll choose shared access signature.

I'll go on to next pleased his source you order here this is the display name I'll go onto next I'll

hit on connect open up the explorer We just closed all of this so I'll go on to detail on SAS and I

can see my objects.

So here we just closed this.

So the main thing is data on shared access signature.

Now what happens?

As I mentioned before, if these shared access center gets in the wrong hands, then what you can do

is you can then go on to be stored access policy.

You can edit the policy.

You're in the permissions.

You can now disable those permissions and then hit on a key and then click on Save.

So now we actually remove the permissions from the policy that was assigned onto the shared access signature.

Now, here, let me hit on refresh on I'll go on to local and attached onto my storage accounts.

So on to my attach containers.

BLOCK Container Data one.

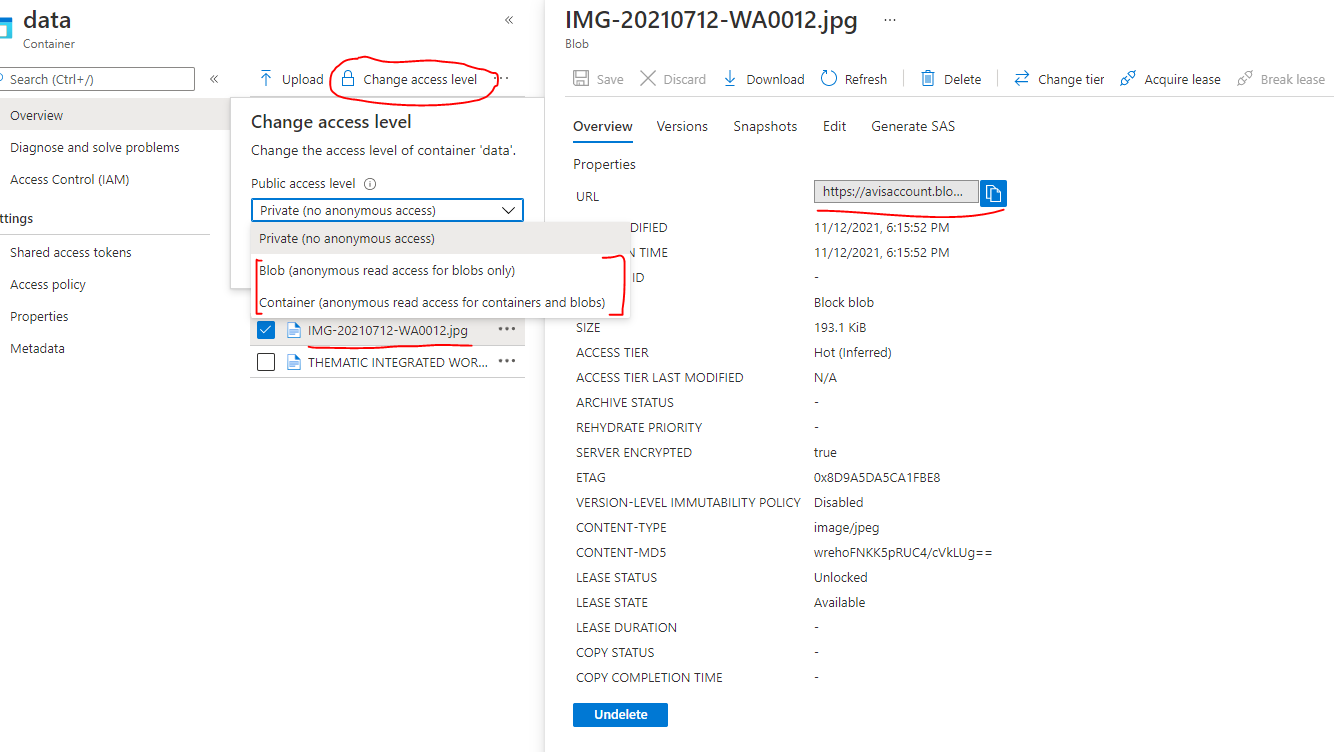
And I can see it as fail to authenticate the request because now we have gain the permissions of the

stored access policy.

It has gone ahead and invalidated these shared access signature.

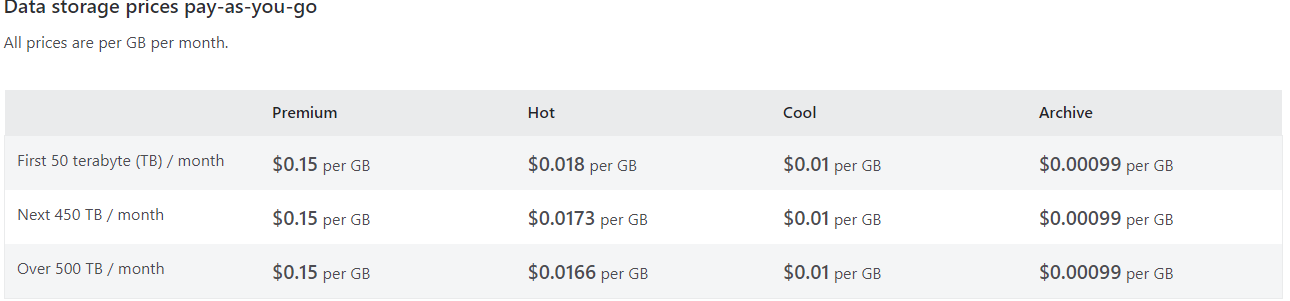
#### PUBLIC ACCESS

* All the BLOB items have a unique url. Click on the items to view its property
* The URl field will give the get the unique url of the BLOB Item(e.g. : <https://avisaccount.blob.core.windows.net/data/IMG-20210712-WA0012.jpg> )
* ***The items will be accessible when the Access level is not private.***



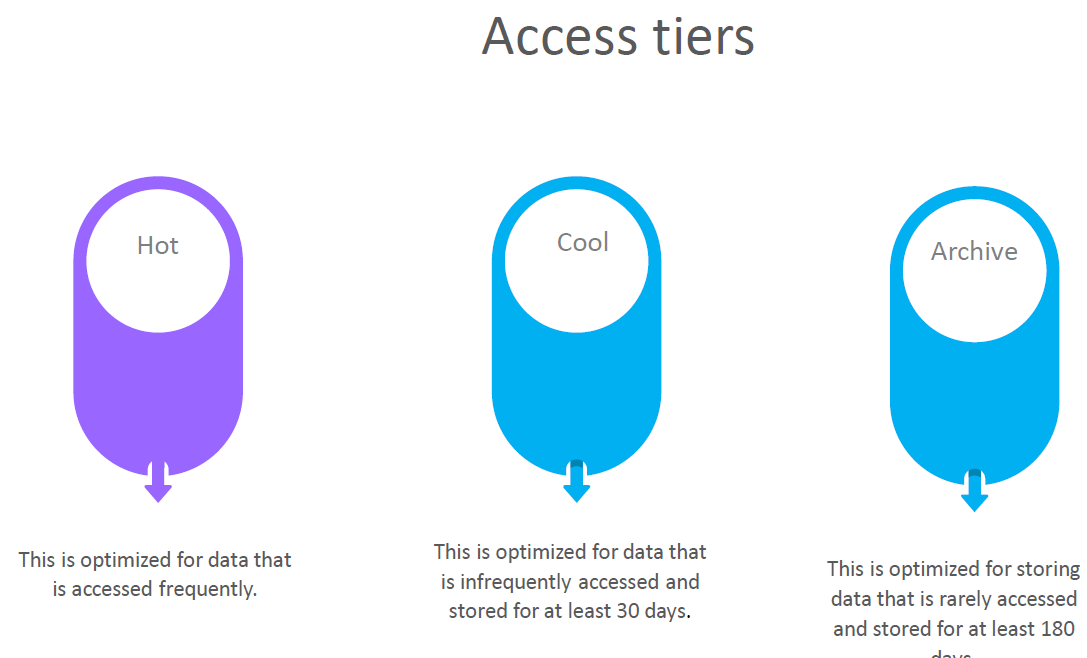
### ACCESS TIERS FOR BLOB SERVICE

* **Access tier is an option provided for the Blob service for the storage account that can be used to optimize the costs for using Azure storage based on how frequently the stored data is accessed.**
* **The price of the data storage varies with the access tier. By default, when a blob is uploaded in blob storage – it is uploaded as hot access tier.**



#### BLOB ACCESS TIERS

Azure blob service offers different access tiers for the blob storage, helping us store object data in the most cost-effective manner. The available access tiers include:



|  |  |
| --- | --- |
| **HOT ACCESS TIER** | * The hot tier is typically used for storing data that is accessed regularly. * This access tier provides low latency, and hence it's comparatively more expensive than the cool tier * Optimized for storing data that is accessed |
| **COOL ACCESS TIER** | * The cool tier is used to store less-accessed data or archived data. It provides higher latency than the hot tier. Hence, it's best suited for data that is not accessed frequently * Optimized for data that is infrequently accessed **and stored for at least 30 days** (for example, invoices for your customers). |
| **ARCHIVE ACCESS TIER** | * Appropriate for data that is rarely accessed and stored **for at least 180 days**, with flexible latency requirements (for example, long-term backups). * **Behind the scenes the data is moved to Tape Drive. Tape Drive are slow but can retain the data for longer duration.** * When we move the data from Archive Tier to Cool or Hot tier – it goes through a rehydration process. The amount to time it takes for rehydration depends on amount of data we are trying to rehydrate. * Till the time the Blob is in archive tier or getting rehydrated – it will not be accessible |

* ***If we try to access a resource in the cool tier before 30 days cut off time – an early deletion charge will be charged***

#### CONFIGURING ACCESS TIERS

##### HOT AND COOL ACCESS TIERS CONFIGURATION

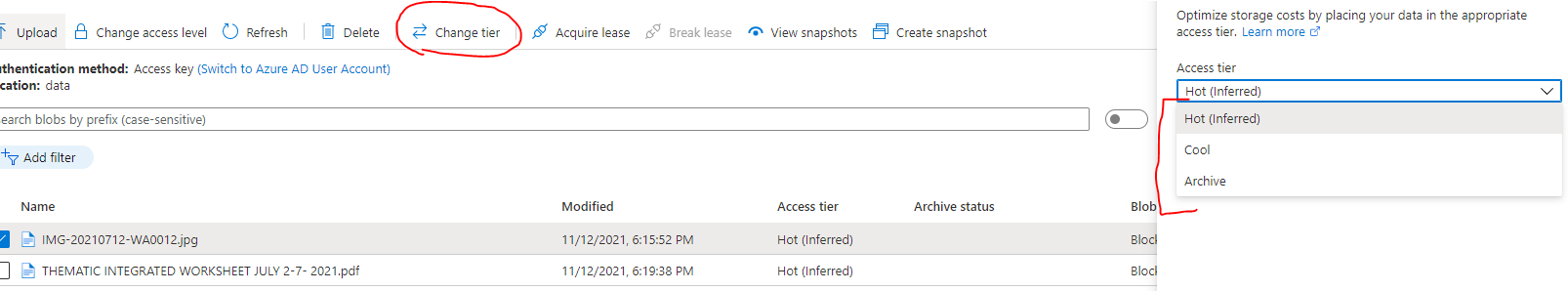
|  |  |
| --- | --- |
|  | * We can change the default access tier for the blobs from the storage account level (while creating or even after creating the storage account) |
|  | * The default access tier for the blobs are be can be set while creating the storage account itself. |
| * Hot, cool, and archive tiers can be set at the blob level, during upload or after upload. * The Access Tiers can be changed as shown below. Select the BLOB item 🡪 Change Tier 🡪 Select the Access Tier. | |

##### ACCESS TIER CONFIGURATION

|  |  |
| --- | --- |
|  | We can change the access tier of entire storage account  *Storage Account 🡪 Configuration 🡪 Blob Access Tier* |
|  | * We can do the configuration while creating the storage account or even after creating the Storage as well. |

##### ARCHIVE TIERS CONFIGURATION

* **The archive access can be only at the blob level.**



* Setting the access tier to "Archive" will make the blob inaccessible until it is rehydrated back to "Hot" or "Cool", which may take several hours. Moving the data from Archive tier to Hot or Cool tier is called Rehydration

|  |  |
| --- | --- |
|  | * When we want to change the access tier from Archive to cool/ Hot access tier – we need to rehydrate the blob * We can set the priority of the rehydration. It can be Standard or High priority. * Archive Tier Rehydration - <https://learn.microsoft.com/en-us/azure/storage/blobs/archive-rehydrate-overview> |

#### LIFE CYCLE MANAGEMENT FOR ACCESS TIER

* Let's say that we have an application that is uploading objects onto the storage account - initially all of them are basically part of the hot access tier.
* In the uploaded blobs - some of these objects mightnot accessed that frequently. Hence to save the cost – we can change the access tier for those Blob objects.
* **We can achieve using lifecycle management rules. Using lifecycle management rules based on a particular condition, we can go out and tell the Azure Blob service to change the access to or even delete an object as well.**

|  |  |
| --- | --- |
|  | * *To add the Rule 🡪 Go to Storage account 🡪 Life Cycle Management 🡪 Add Rule* |
| * Add a rule name, rule scope, blob type and Blog subtype. |  |
|  | * As per the given rule the blob will be moved to “cool” storage if it is not modified/ accessed for 7 days |

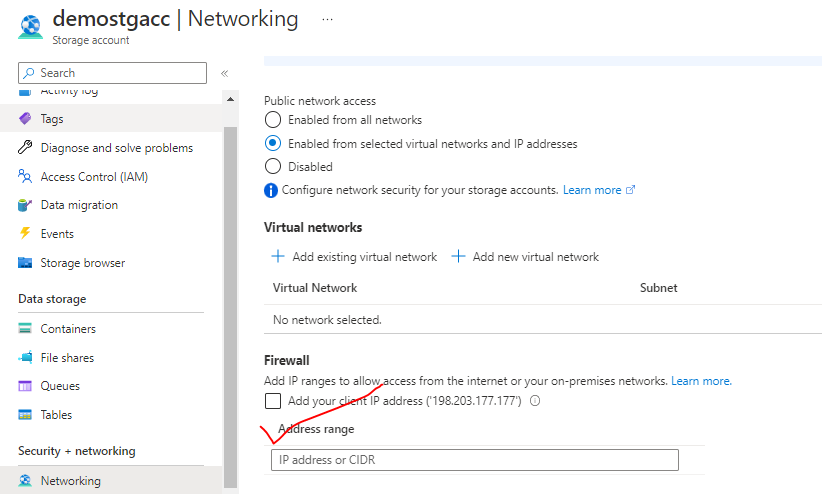
#### ACCESS TO STORAGE IN TERMS OF NETWORKING



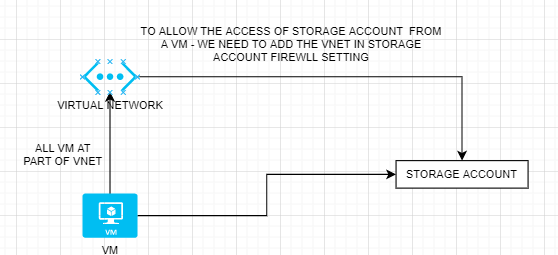
|  |  |
| --- | --- |
| ENABLED FOR ALL NETWORKS | Storage account will have anonymous access |
| ENABLED FOR SELECTED VNET AND IP ADDRESS | Can be accessed from specific IP and Virtual network |
| DISABLED | No Access (Not even from Azure Portals) |

#### ACCESSING THE STORAGE ACCOUNT FROM IPADDRESS

* Storage Account can be restricted to a public address only. Note so far IP4 address are only supported , not IP6



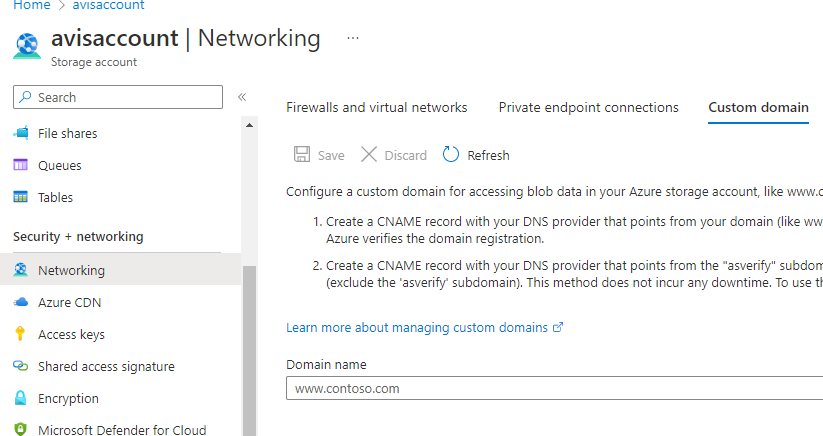
#### CONNECTING VM WITH THE STORAGE ACCOUNT



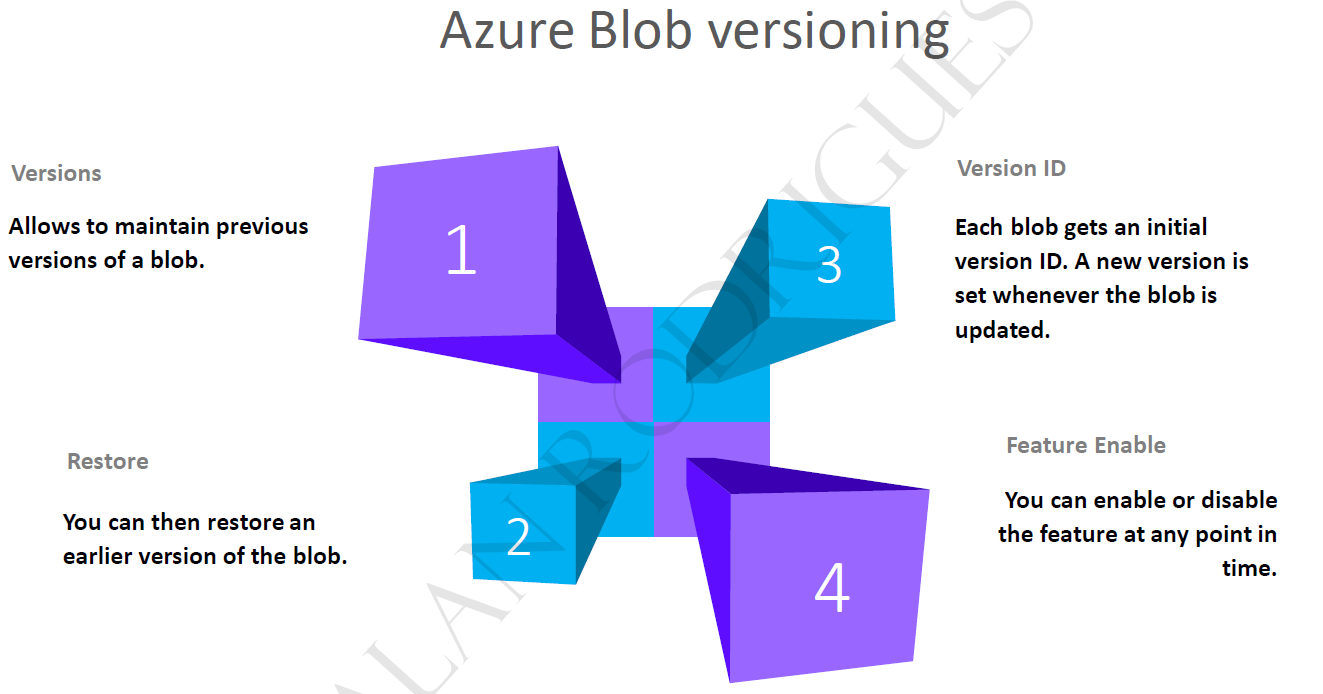
* All the VM are part of a Virtual network. To access the storage account from the VM – we need to add the VNET to the Storage Account Firewall setting.

#### CUSTOM DOMAIN

* To access the items in the Storage account make use of a URL (the url varies based in what service we are consuming)
* We can also add a custom domain while accessing the Stoarge account services.

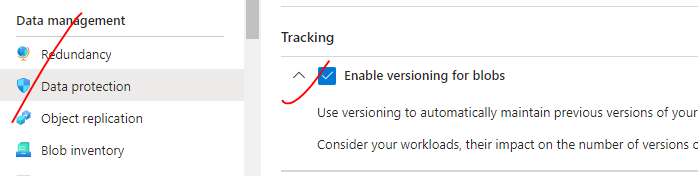


### BLOB VERSIONING



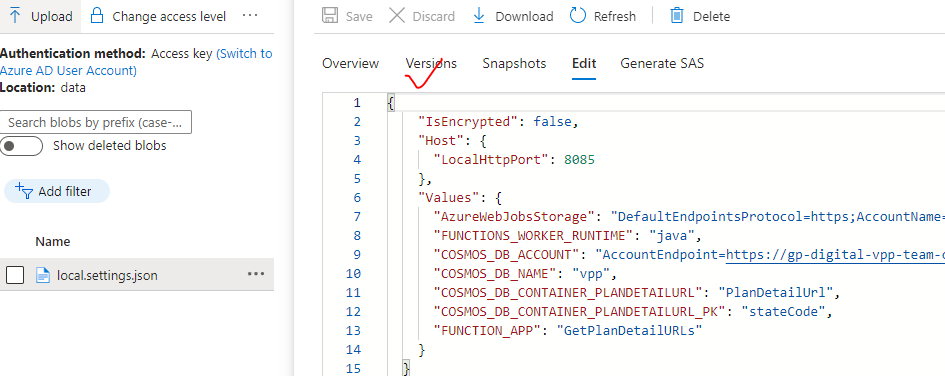
#### STEP 1: ENABLE THE BLOB VERSIONING

* Go to storage account 🡪 data protection 🡪 Enable versioning for blobs 🡪 Save

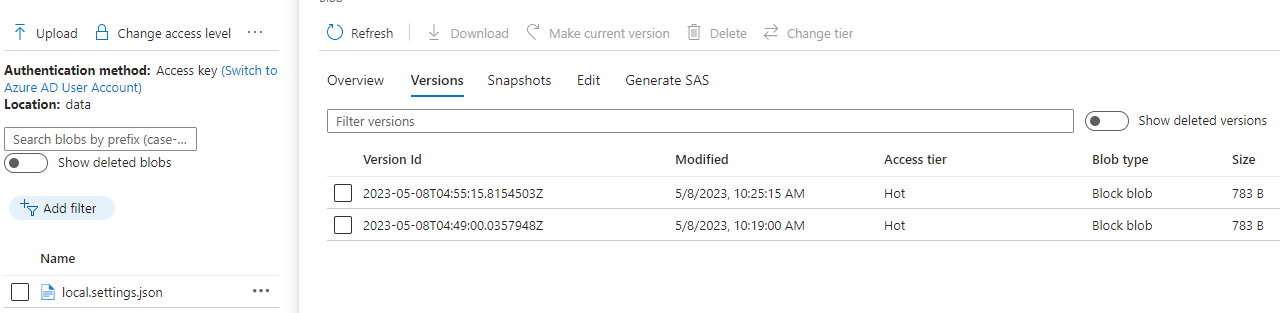


#### STEP 2: VALIDATE THE VERISONING

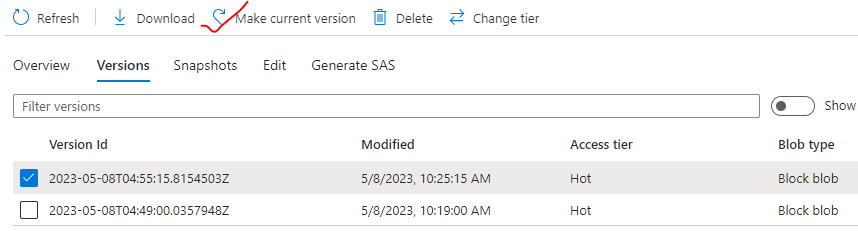
* To validate the versioning 🡪 Go to the blob and make a change to it. For example, the below JSON file
* For every update🡪 Save. Will create a new version of the blob



* T**he version can be viewed in the “Versions” tab of the blob(as below)**

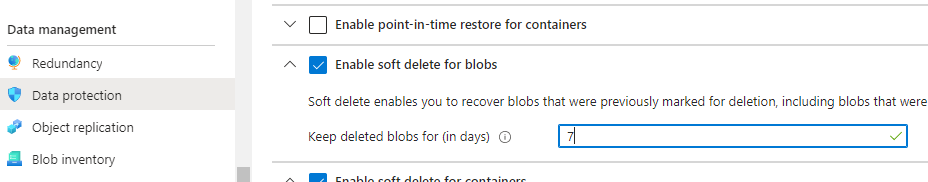


We can then select a desired version to make it a current version

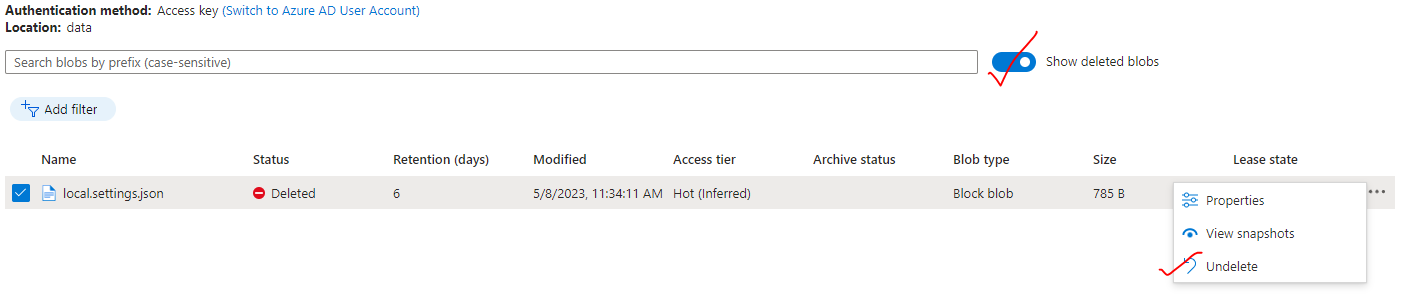


### SOFT DELETE

* In blob soft delete, we can retain the objects even after deletion which helps recovering the blobs in case helps against the accidental deletion.
* We need to specify the retention period of the deleted object from 1 to 365 days (This can be changed at any point of time). Depending upon the retention period, the data will be made available after it has been deleted or even overridden.
* During the retention period we can restore the blob along with its snapshot.
* By default - The soft delete feature with retention period of 7 days is already enabled for the blobs



* To view the deleted blobs 🡪Enable the show deleted blobs
* Undelete the blob to recover the blob



### BLOB STORAGE – PROMGRAMATIC APPROACH

#### BLOB API

|  |  |
| --- | --- |
|  | Azure Blob Storage is optimized for storing massive amounts of unstructured data. Unstructured data doesn't adhere to a particular data model or definition, such as text or binary data.  Blob storage offers three types of resources:   1. *The storage account* 2. *A container in the storage account* 3. *A blob in the container* |

Use the following Java classes to interact with these resources:

|  |  |
| --- | --- |
| [**BlobServiceClient**](https://learn.microsoft.com/en-us/java/api/com.azure.storage.blob.blobserviceclient) | The BlobServiceClient class allows us to manipulate Azure Storage resources and blob containers. The storage account provides the top-level namespace for the Blob service. |
| [**BlobServiceClientBuilder**](https://learn.microsoft.com/en-us/java/api/com.azure.storage.blob.blobserviceclientbuilder) | The BlobServiceClientBuilder class provides a fluent builder API to help aid the configuration and instantiation of BlobServiceClient objects |
| [**BlobContainerClient**](https://learn.microsoft.com/en-us/java/api/com.azure.storage.blob.blobcontainerclient) | The BlobContainerClient class allows you to manipulate Azure Storage containers and their blobs |
| [**BlobClient**](https://learn.microsoft.com/en-us/java/api/com.azure.storage.blob.blobclient) | The BlobClient class allows you to manipulate Azure Storage blobs |
| [**BlobItem**](https://learn.microsoft.com/en-us/java/api/com.azure.storage.blob.models.blobitem) | The BlobItem class represents individual blobs returned from a call to [listBlobs](https://learn.microsoft.com/en-us/java/api/com.azure.storage.blob.blobcontainerclient.listblobs) |

RESOURCES: <https://github.com/Azure-Samples/AzureStorageSnippets/tree/master/blobs/howto/Java>

<https://learn.microsoft.com/en-us/azure/storage/blobs/storage-quickstart-blobs-java?tabs=powershell%2Cmanaged-identity%2Croles-azure-portal%2Csign-in-azure-cli#create-the-project>

**CREATING A JAVA PROJECT**

mvn archetype:generate `

--define interactiveMode=n `

--define groupId=com.blobs.quickstart `

--define artifactId=blob-quickstart `

--define archetypeArtifactId=maven-archetype-quickstart `

--define archetypeVersion=1.4

To Build and run project

1. Set the Env variable in setX command:

**setx** AZURE\_\_TABLE\_STORAGE\_CONNECTION\_STRING "DefaultEndpointsProtocol=https;AccountName=avistgacc;AccountKey=fCkzORQpmYoMSrNgBCEnooNkJlY1bxq2c2hUGTcMpYWxKpK/Sh+4JeeXedk3Avy7t50gnykduKOS+AStthsBuQ==;EndpointSuffix=core.windows.net"

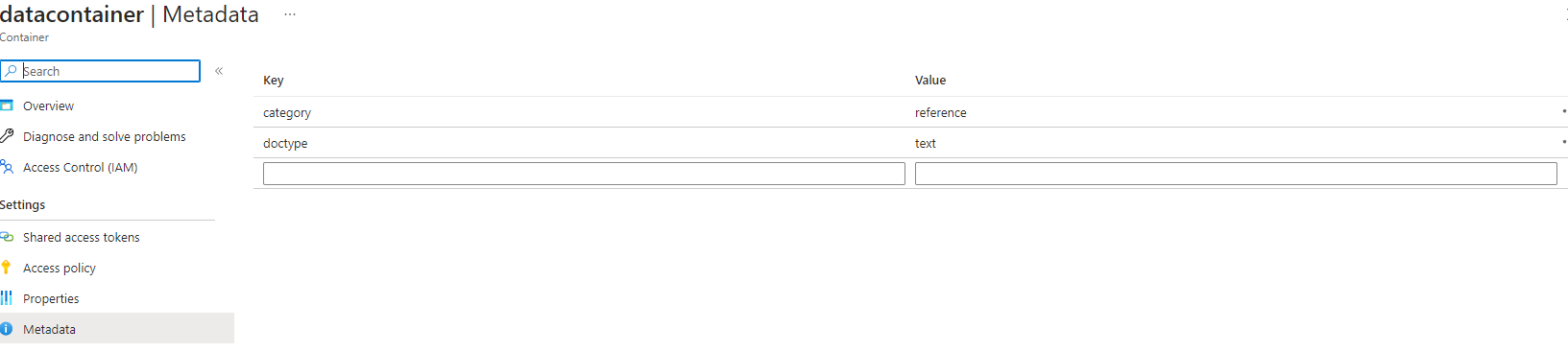
1. mvn compile
2. mvn package
3. Add the below dependency 🡪 mvn exec:java

|  |
| --- |
| <plugin>  <groupId>org.codehaus.mojo</groupId>  <artifactId>exec-maven-plugin</artifactId>  <version>1.4.0</version>  <configuration>  <mainClass>com.blobs.quickstart.App</mainClass> 🡨 Update this Main class path  <cleanupDaemonThreads>false</cleanupDaemonThreads>  </configuration>  </plugin> |

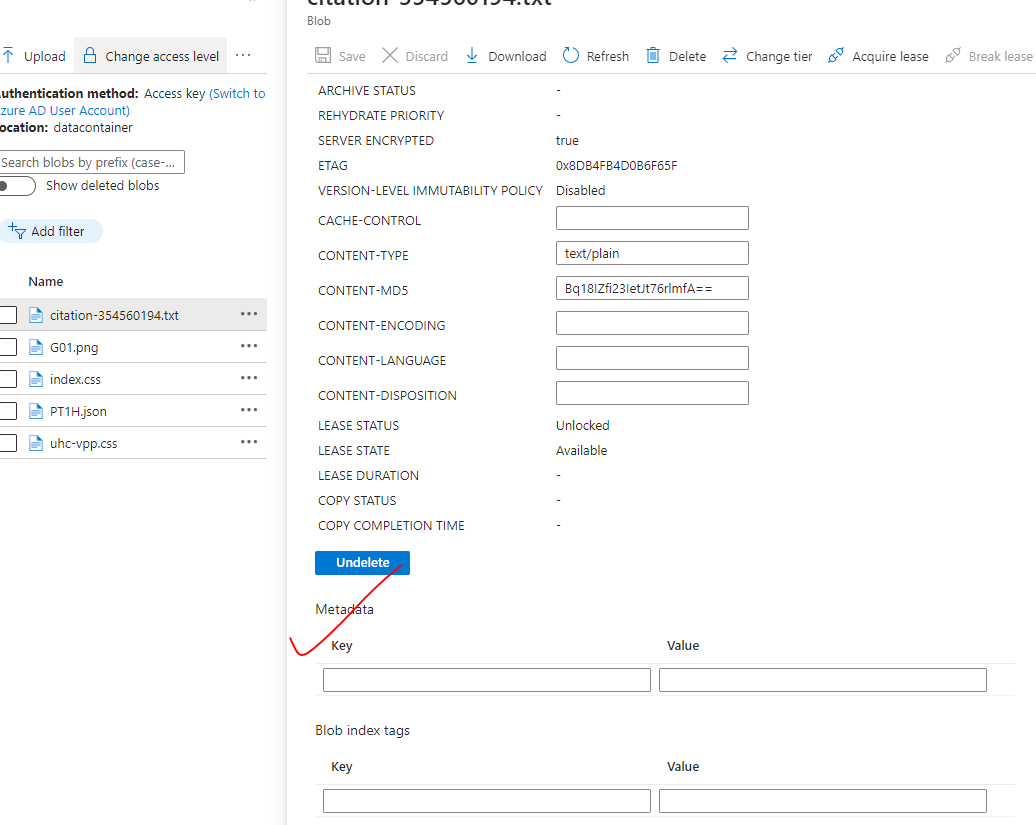
#### CONTAINER AND BLOB METADATA

Every Container is associated with Metadata information. These metadata can be set from Portal and programmatically as well

**CONTAINER METADATA**



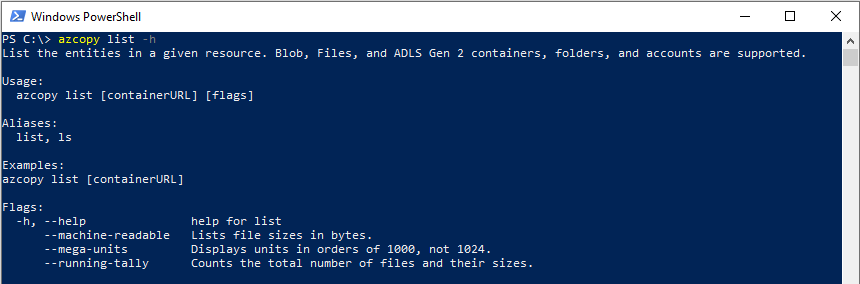
**BLOB METADATA**



### AZ COPY TOOL (COMMAND LINE)

* Reference : <https://learn.microsoft.com/en-us/azure/storage/common/storage-use-azcopy-v10>
* AzCopy is a command-line utility that you can use to copy blobs or files to or from a storage account

#### AZ HELP



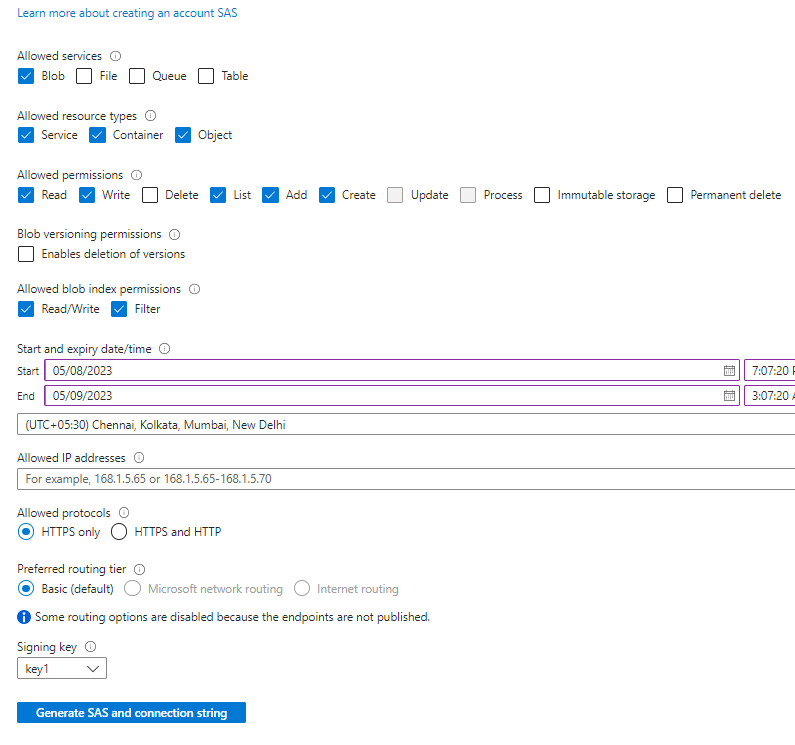
**To copy the blobs/ files we need to provide authorization credentials. These 2 different methods of authorization**

1. **AZURE AD**
2. **STORAGE ACCOUNT SAS TOKEN.**

#### AZ COPY USING SAS

##### GENERATING SAS

* To generate the SAS 🡪 Go to storage account🡪 Generate the SAS
* In the below example – We are allowing Read + Write permission on blobs

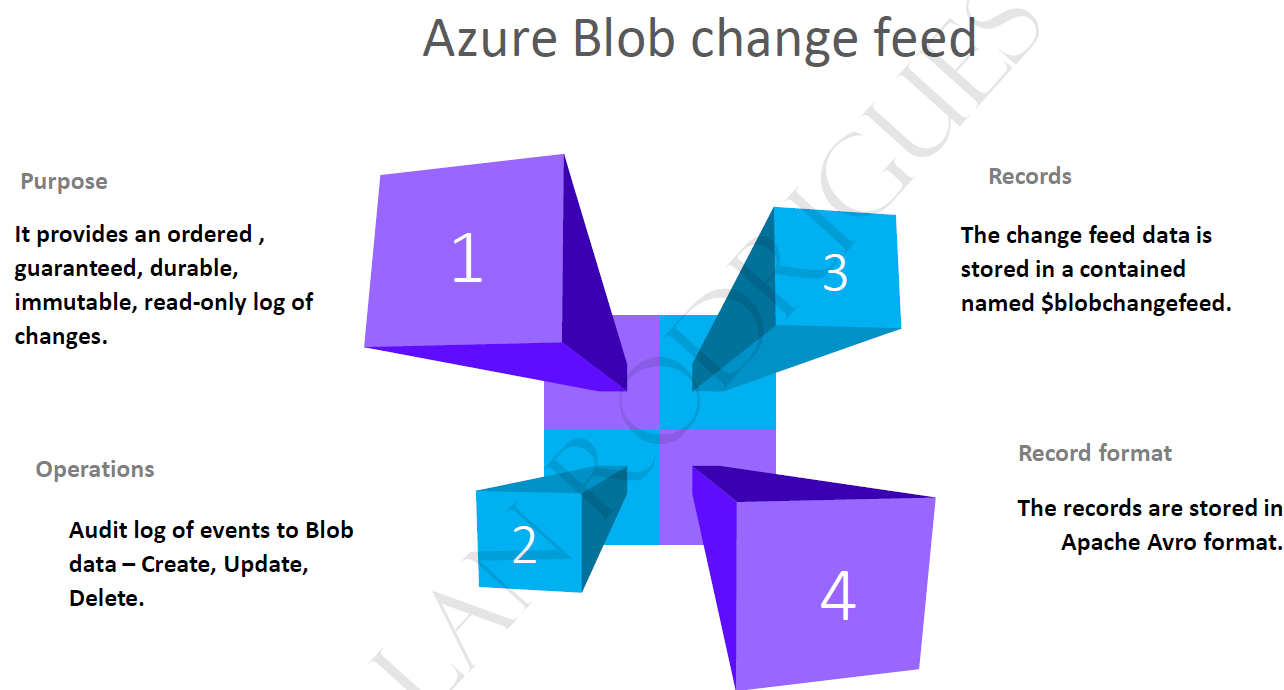


##### AZ COPY SYNTAX

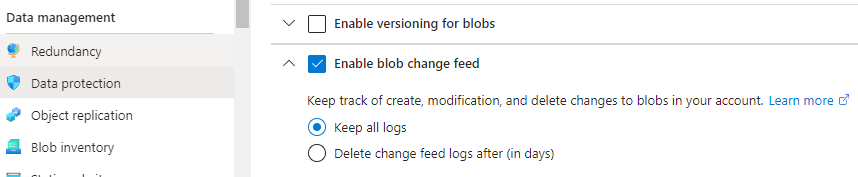
1. **SYNTAX - azcopy make** [**https://[account-name].[blob,file,dfs].core.windows.net/[top-level-resource-name]**](https://[account-name].[blob,file,dfs].core.windows.net/%5btop-level-resource-name%5d)
2. **Example: azcopy make** [**https://avistgacc.blob.core.windows.net/mycontainer?sv=2022-11-02&ss=b&srt=sco&sp=rwlactf&se=2023-05-08T21:37:20Z&st=2023-05-08T13:37:20Z&spr=https&sig=H6NZmBBpwDpNPMW4jlGnpPN6DNWa3rJRW%2FlDu%2Fi%2FVbc%3D**](https://avistgacc.blob.core.windows.net/mycontainer?sv=2022-11-02&ss=b&srt=sco&sp=rwlactf&se=2023-05-08T21:37:20Z&st=2023-05-08T13:37:20Z&spr=https&sig=H6NZmBBpwDpNPMW4jlGnpPN6DNWa3rJRW%2FlDu%2Fi%2FVbc%3D)**”**

|  |  |
| --- | --- |
| **List of commands** | |
| **Command** | **Description** |
| [azcopy bench](https://learn.microsoft.com/en-us/azure/storage/common/storage-ref-azcopy-bench?toc=/azure/storage/blobs/toc.json) | Runs a performance benchmark by uploading or downloading test data to or from a specified location. |
| [azcopy copy](https://learn.microsoft.com/en-us/azure/storage/common/storage-ref-azcopy-copy?toc=/azure/storage/blobs/toc.json) | Copies source data to a destination location |
| [azcopy doc](https://learn.microsoft.com/en-us/azure/storage/common/storage-ref-azcopy-doc?toc=/azure/storage/blobs/toc.json) | Generates documentation for the tool in Markdown format. |
| [azcopy env](https://learn.microsoft.com/en-us/azure/storage/common/storage-ref-azcopy-env?toc=/azure/storage/blobs/toc.json) | Shows the environment variables that can configure AzCopy's behavior. |
| [azcopy jobs](https://learn.microsoft.com/en-us/azure/storage/common/storage-ref-azcopy-jobs?toc=/azure/storage/blobs/toc.json) | Subcommands related to managing jobs. |
| [azcopy jobs clean](https://learn.microsoft.com/en-us/azure/storage/common/storage-ref-azcopy-jobs-clean?toc=/azure/storage/blobs/toc.json) | Remove all log and plan files for all jobs. |
| [azcopy jobs list](https://learn.microsoft.com/en-us/azure/storage/common/storage-ref-azcopy-jobs-list?toc=/azure/storage/blobs/toc.json) | Displays information on all jobs. |
| [azcopy jobs remove](https://learn.microsoft.com/en-us/azure/storage/common/storage-ref-azcopy-jobs-remove?toc=/azure/storage/blobs/toc.json) | Remove all files associated with the given job ID. |
| [azcopy jobs resume](https://learn.microsoft.com/en-us/azure/storage/common/storage-ref-azcopy-jobs-resume?toc=/azure/storage/blobs/toc.json) | Resumes the existing job with the given job ID. |
| [azcopy jobs show](https://learn.microsoft.com/en-us/azure/storage/common/storage-ref-azcopy-jobs-show?toc=/azure/storage/blobs/toc.json) | Shows detailed information for the given job ID. |
| [azcopy list](https://learn.microsoft.com/en-us/azure/storage/common/storage-ref-azcopy-list?toc=/azure/storage/blobs/toc.json) | Lists the entities in a given resource. |
| [azcopy login](https://learn.microsoft.com/en-us/azure/storage/common/storage-ref-azcopy-login?toc=/azure/storage/blobs/toc.json) | Logs in to Azure Active Directory to access Azure Storage resources. |
| [azcopy login status](https://learn.microsoft.com/en-us/azure/storage/common/storage-ref-azcopy-login-status) | Lists the entities in a given resource. |
| [azcopy logout](https://learn.microsoft.com/en-us/azure/storage/common/storage-ref-azcopy-logout?toc=/azure/storage/blobs/toc.json) | Logs the user out and terminates access to Azure Storage resources. |
| [azcopy make](https://learn.microsoft.com/en-us/azure/storage/common/storage-ref-azcopy-make?toc=/azure/storage/blobs/toc.json) | Creates a container or file share. |
| [azcopy remove](https://learn.microsoft.com/en-us/azure/storage/common/storage-ref-azcopy-remove?toc=/azure/storage/blobs/toc.json) | Delete blobs or files from an Azure storage account. |
| [azcopy sync](https://learn.microsoft.com/en-us/azure/storage/common/storage-ref-azcopy-sync?toc=/azure/storage/blobs/toc.json) | Replicates the source location to the destination location. |
| [azcopy set-properties](https://learn.microsoft.com/en-us/azure/storage/common/storage-ref-azcopy-set-properties?toc=/azure/storage/blobs/toc.json) | Change the access tier of one or more blobs and replace (overwrite) the metadata, and index tags of one or more blobs. |

### AZURE BLOB CHANGE FEED



* The purpose of this feature is that it provides an order guaranteed , durable, immutable read only log of changes. For example, we modify/create/delete a particular blob, that change can be recorded in a log in **AVRO format**. This is very useful if we want to have an audit trail for the changes being made on the Blob.
* To enable change feed for blobs 🡪 Go to Storage account 🡪 Data protection 🡪 Enable blob change feed



* **Once the blob change feed is enabled – it will create a dedicated container for store the feed logs “$blobchangefeed”**



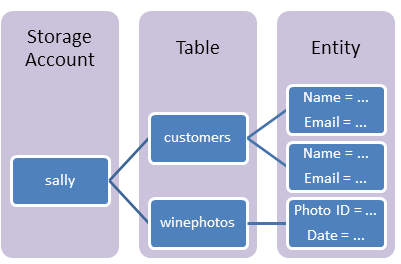
* Once any blob is updated/deleted /created will be logged in the $***blobchangefeed*** container. These avro files can be downloaded and covered in readable format like JSON

|  |  |
| --- | --- |
|  |  |

## TABLE SERVICE

* Azure tables are ideal for **storing structured, non-relational data** in the cloud.
* Tables are store as key /value pair with a schemeless design.
* Due to schemeless design - it's easy to adapt the data as the needs of the application evolve. Access to Table storage data is fast and cost-effective for many types of applications and is typically lower in cost than traditional SQL for similar volumes of data.
* We can use Table storage to store flexible datasets like user data for web applications, address books, device information, or other types of metadata your service requires.

### TABLE STORAGE COMPONENTS



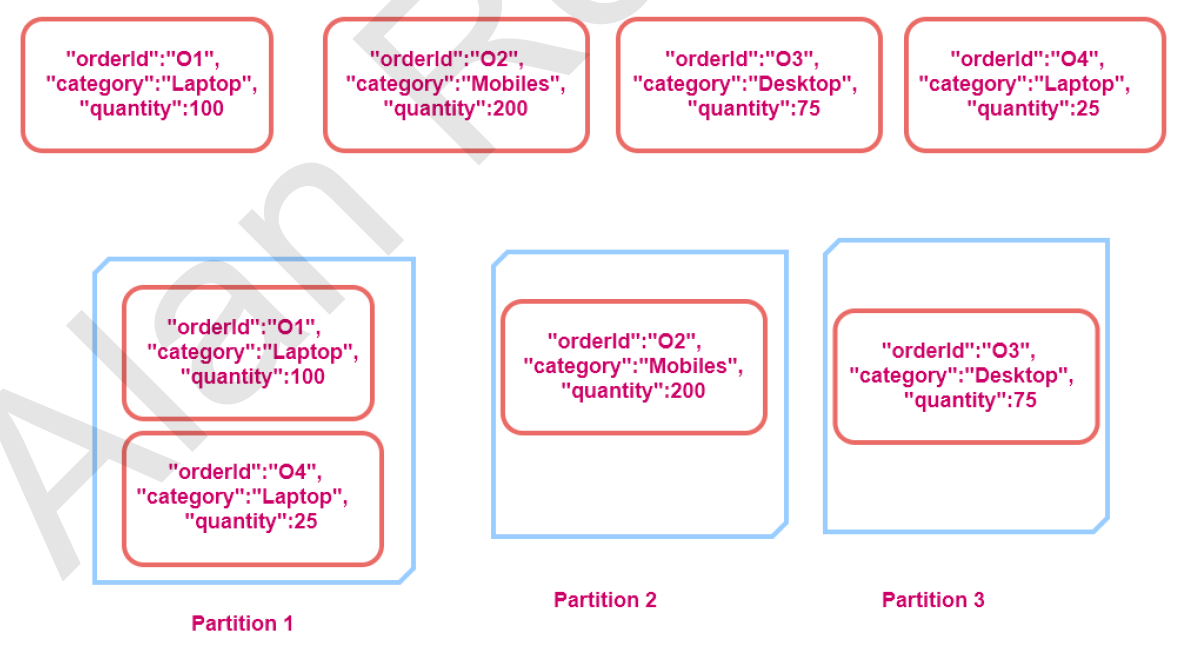
|  |  |
| --- | --- |
| **URL FORMAT** | * Azure Table Storage accounts format:   http://<storage account>.table.core.windows.net/<table>   * Azure Cosmos DB Table API accounts format:   http://<storage account>.table.cosmosdb.azure.com/<table>  **These URLs can be consumed by the application to perform operation on the table storage** |
| **TABLE** | A table is a collection of entities. Tables don't enforce a schema on entities, which means a single table can contain entities that have different sets of properties. |
| **ENTITY** | An entity is a set of properties, like a database row. **An entity in Azure Storage can be up to 1MB in size. An entity in Azure Cosmos DB can be up to 2MB in size** |
| **PROPERTIES** | * A property is a name-value pair. * Each entity can include up to *252 properties to store data*. * *Each entity also has three system properties that specify a partition key, a row key, and a timestamp*. **Entities with the same partition key can be queried more quickly and inserted/updated in atomic operations.** * An entity's row key is its unique identifier within a partition. |

#### ENTITY AND PARTITION

An Entity in a table has 2 parts

|  |  |
| --- | --- |
| **PARTITION KEY** | This is a string value . This identifies the partition that entity belongs to |
| **ROWKEY** | The is a string value. This uniquely identifies each entity within the partition |
| **Note – The partition key along with the Row key helps in uniquely identify the entity within the table** | |

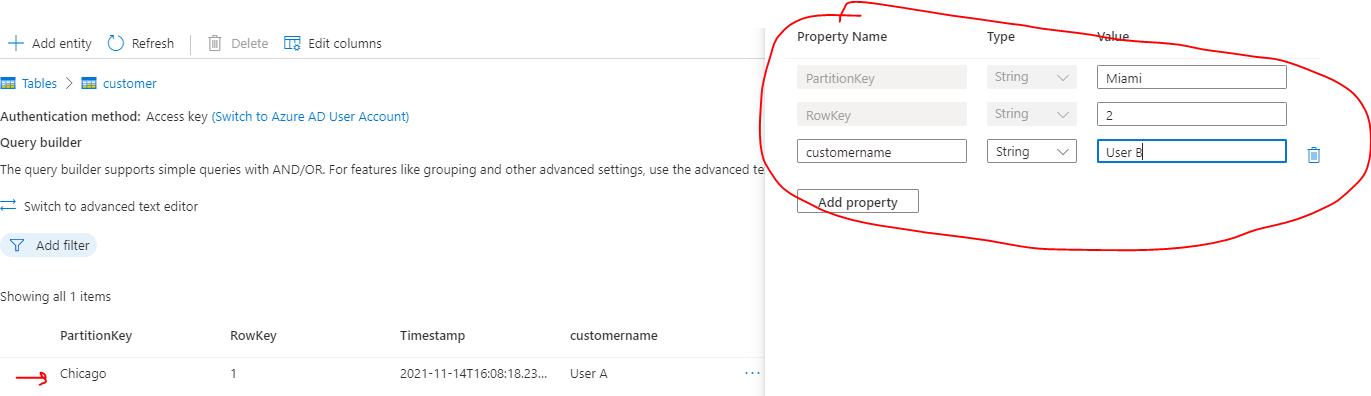
Example : - in the below example - We created ”category” as Partition key.



#### ADDING DATA TO TABLE

|  |  |
| --- | --- |
| * To work with a table – it is preferred to use storage explorer. * The data can be added to table storage using CSV File well. * The import/export of data in CSV can be done in from Storage Explorer. | ***SAMPLE CSV FOR for Userdata***  PartitionKey,RowKey,Name,SSN  UserData,006,Alex6,9892 |

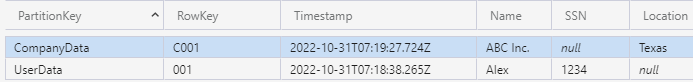
|  |  |
| --- | --- |
|  | * For table storage – we can create tables in the azure storage account. * To add data to the table 🡪 Navigate to Storage Browser 🡪 Navigate to the table 🡪 Add Entity |



##### PARTITION KEY AND ROWKEY

PARTITION KEY

* + If we have huge amount of data in a table, dividing the table into partition – it becomes easier to search for an entity in particular partition.

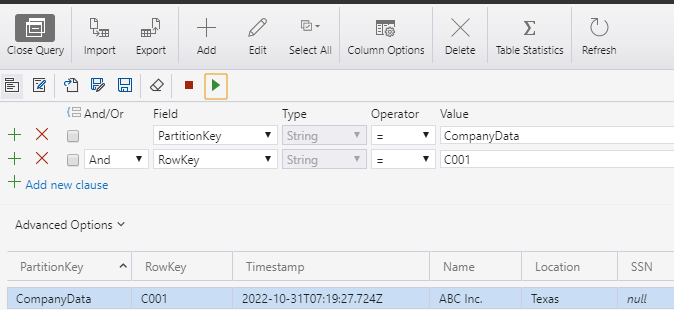


* ROWKEY: Rowkey helps in searching the data within the partition.

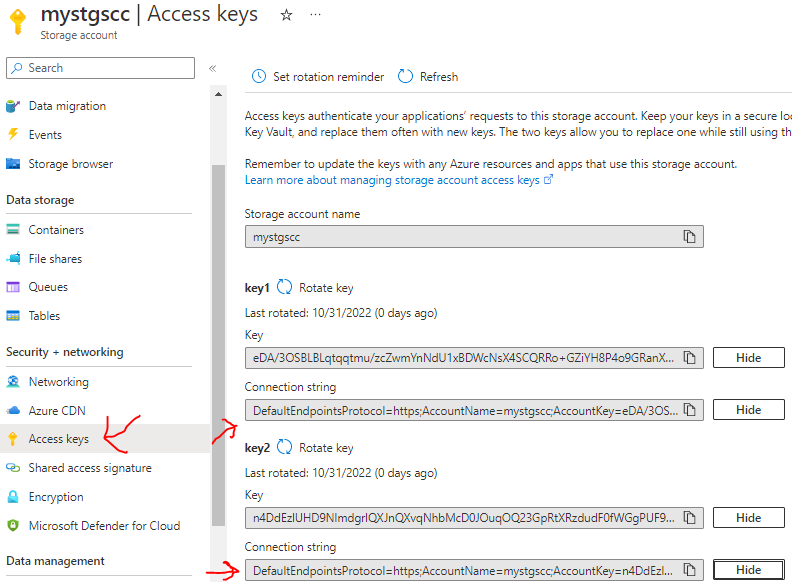
Example: Partition key can be considered as a type of data. For example, if the table has to store - “company” & “user” information. Then all the Company information will be stored as *CompanyData* Partition Key and User Information stored *UserData* partition key. This makes the searching fast – especially when the data is huge in a table. Row key help in searching the data withing a partition

##### QUERYING DATA

* We usually query the data in table storage using connection string.
* From Storage explorer we have an option to query data



##### CONNECTING TO TABLE STORAGE

**

* To programmatically – to access the table we can make use of access keys
* Note - The data in table storage cannot be accessed in browser (using URL) unlike BLOB storage.

### PROGRAMATICALLY -WORKING WITH TABLE STORAGE

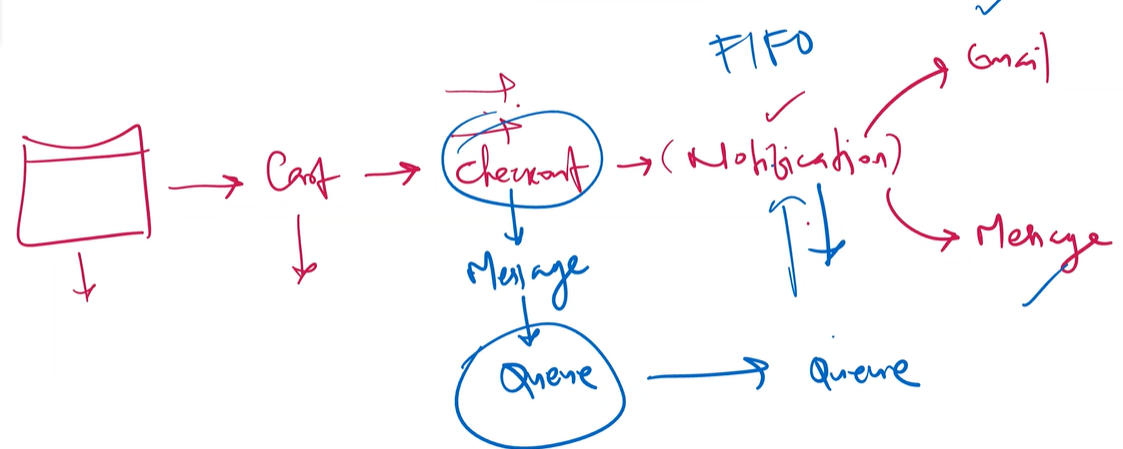
Reference : <https://learn.microsoft.com/en-us/azure/cosmos-db/table/how-to-use-java>

CREATING THE JAVA PROJECT:

mvn archetype:generate --define interactiveMode=n --define groupId=com.table.quickstart --define artifactId=table-quickstart --define archetypeArtifactId=maven-archetype-quickstart --define archetypeVersion=1.4

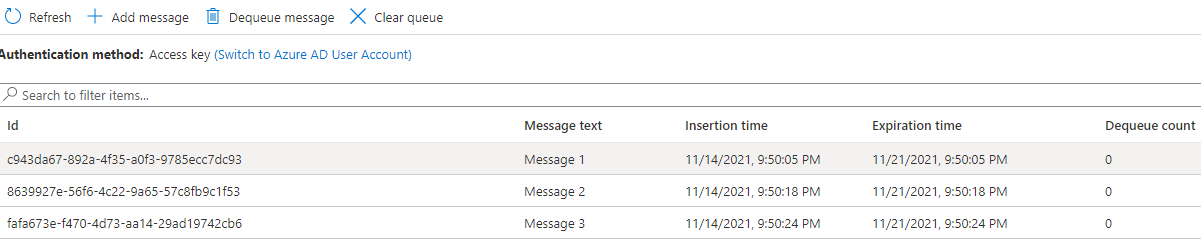
## QUEUES SERVICE

USE CASE: E COMMERCE CHECKOUT AND NOTIFICATION



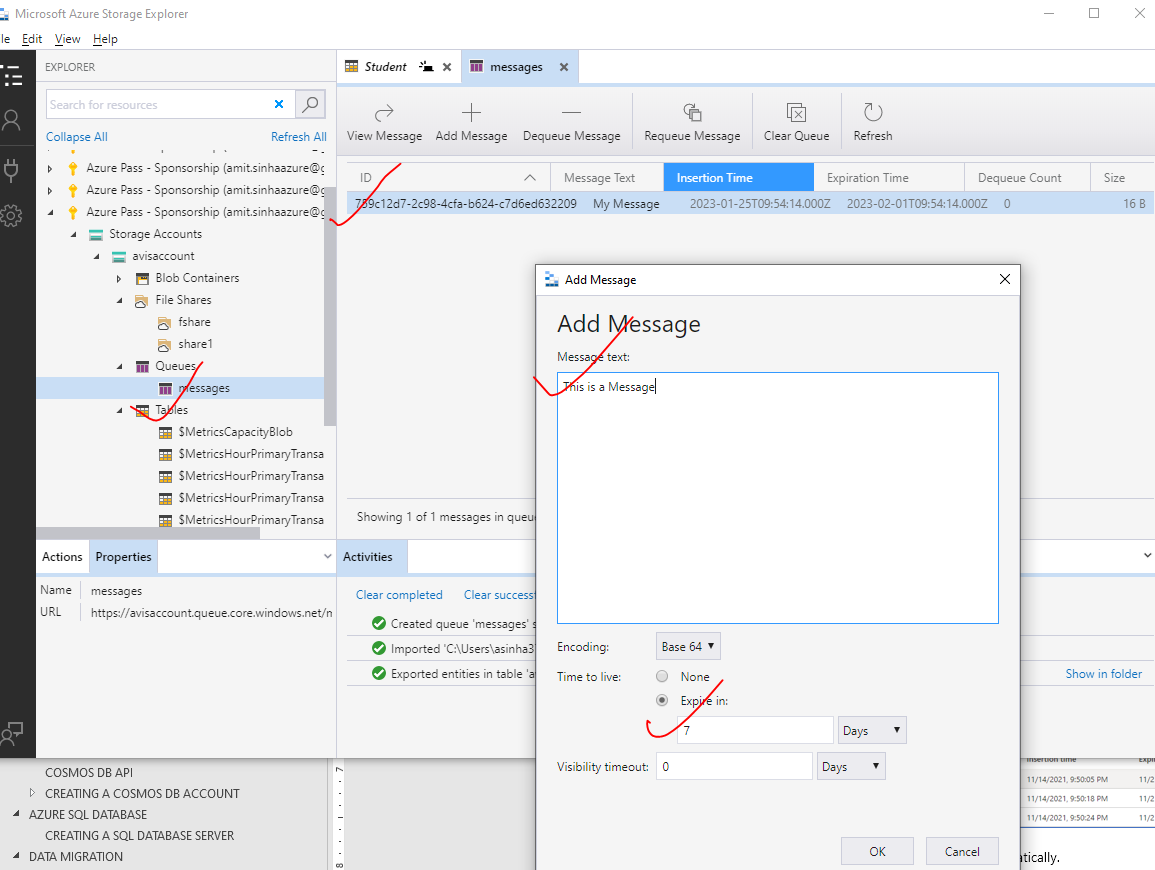
|  |  |
| --- | --- |
|  |  |

* Dequeue message will remove the message from top of the queue
* Clear queue will remove all the message of the queue.

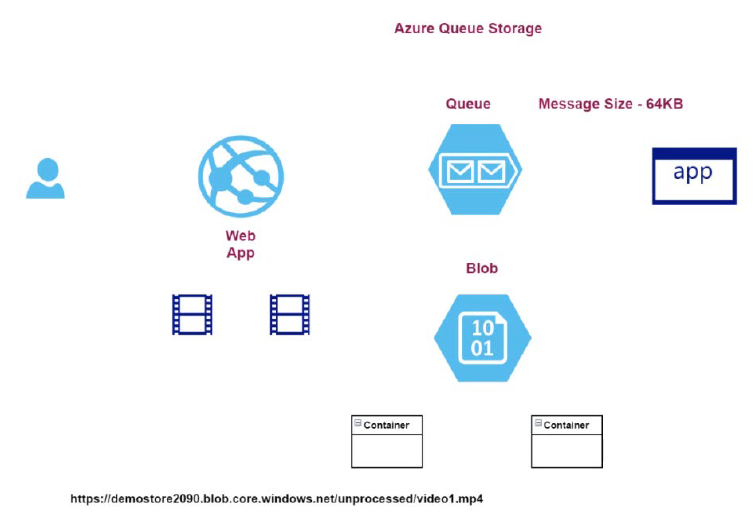


Note in the queue

* The messages are added / dequeued from the queue programmatically.
* By default – the validity period of messages is 7 days(after 7 days the message will be dequeued)
* The maximum size of each message can be up to 64KB



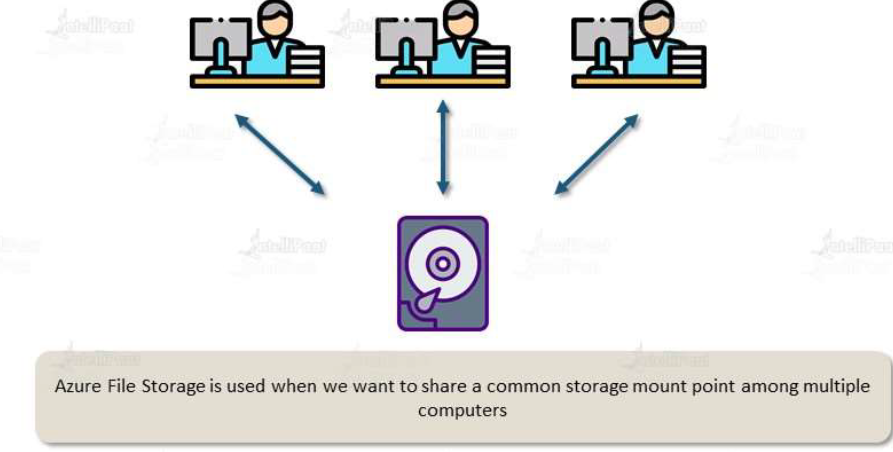
### EXAMPLE



## FILE SHARE SERVICE

* File Share is cloud based file share which is using SMB or NFS protocol

### NEED OF FILE STORAGE



* In an organization – if the employee needs a central storage space for storing documents so that they could access the files from anywhere.
* To meet this need - companies set up a dedicated a file server - a small chunk of the storage is allocated to each user. The user can be able to access the data via file server from anywhere in the organization.
* In this case - A file server needs to be maintained by company. Hence if the demand for storage starts to increase, then the company needs to purchase additional hardware, additional storage.
* Along with infrastructure – companies have place security control when the fileserver - if documents are getting accessed over internet.
* ***Unlike creating a dedicated file share service, Azure File share solves the above problems. Using file Share service, we don't need to think about the underlying file server and storage because everything is managed for us***.

### WHAT IS FILE STORAGE?

* The file storage can be used to store the data of any kind
* It can be mounted on any type of operating system like Windows / Mac and Linux.
* With Azure file storage we also get authentication protocol call SMB (Server Message Block) – which is used by the server whenever a transaction happens with the azure file storage.

|  |  |
| --- | --- |
| BLOB SERVICE | FILE SERVICE |
| Uses HTTP / HTTPS protocol  Port -80 | SMB protocol  Port – 445 |
| Cannot be mounted. Can be accessed via an URL | Can be mounted |

### USE CASE OF FILE STORAGE?

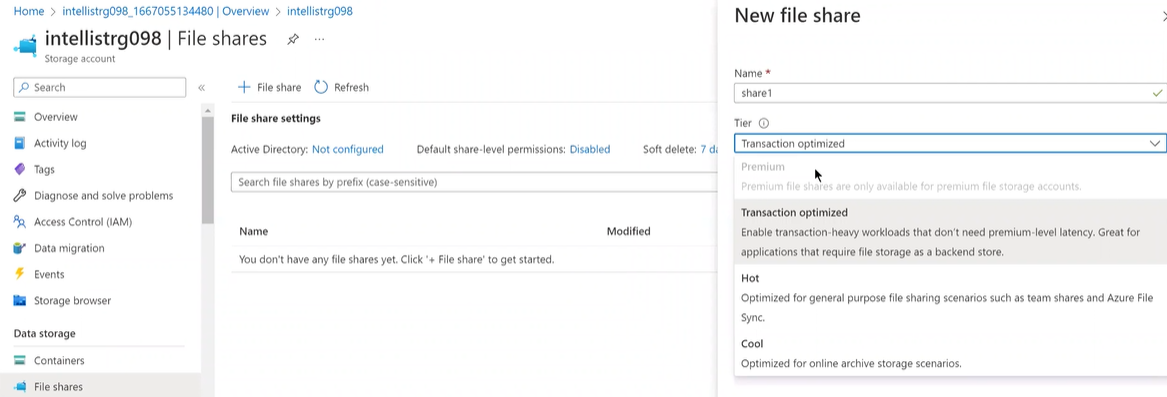
|  |  |
| --- | --- |
| SHARED APPLICATION SETTINGS | We can store configuration files in a centralized location where they can be accessed from many applications instance |
| DIAGNOSTICS SHARE | * Application can store their logs, metrics and crash dumbs in file share (and then that file share can be mounted to developer machine for debugging purpose) |
| DEV/ TEST / DEBUG | Azure File Storage can be used to commonly used tools and utilities -which can be accessed by developers and admin. |

### BENEFITS OF FILE STORAGE?

1. **SHARED ACCESS**: Since Azure file shared support the SMB protocol, we can easily replace our on-premises file share with azure file share
2. **FULLY MANAGED:** File shares can be created without a need to manage hardware or OS
3. **RESILIENCY:** Azure file share are extremely reliable and fault tolerant.

### CREATING A AZURE FILE SHARE

|  |  |
| --- | --- |
| TIERS OF FILE SHARE | |
| TRANSACTION OPTIMIZED |  |
| HOT TIER |  |
| COOL TIER |  |



* ***The premium file share is backed by SSD disk. Note – The we can create premium file share only if the storage account is premium type***

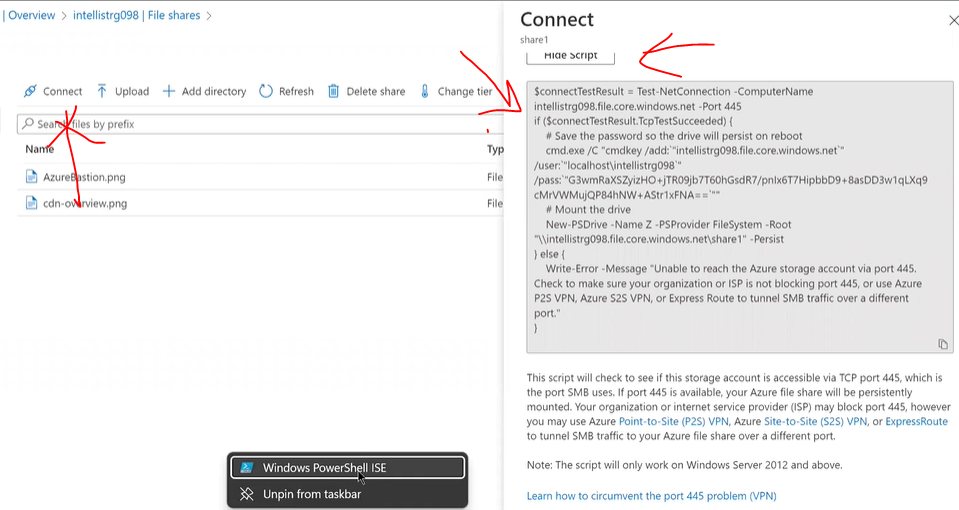
|  |  |
| --- | --- |
| By default, Azure file share are created with 5TB capacity, but “Enable large file share” will create a file share of 100TB capacity |  |
| * Quota of file share can be updated as well | |

### UPLOADING DATA IN FILE SHARE



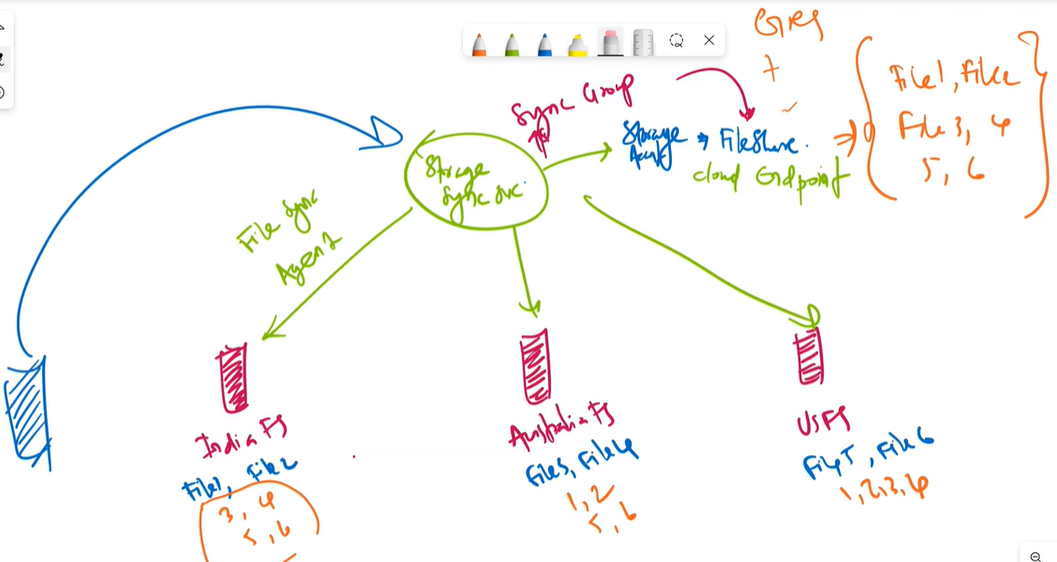
### MOUNTING THE FILE SHARE TO A DRIVE

* To connect to a file share, click on Connect
* Azure will give a Scripts for the connection for different Operating system
* Copy the Script in based on OS and Start Powershell in normal mode (not administrator)
* Paste the script in powershell console.
* This will mount the file share in the local machine.



|  |  |
| --- | --- |
| CONNECTING USING POWER SHELL SCRIPT   * This command will mount the file share in the local machine (as a drive) |  |
| MANUALY MAPPING THE FILE SHARE |  |

## AZURE FILE SYNC



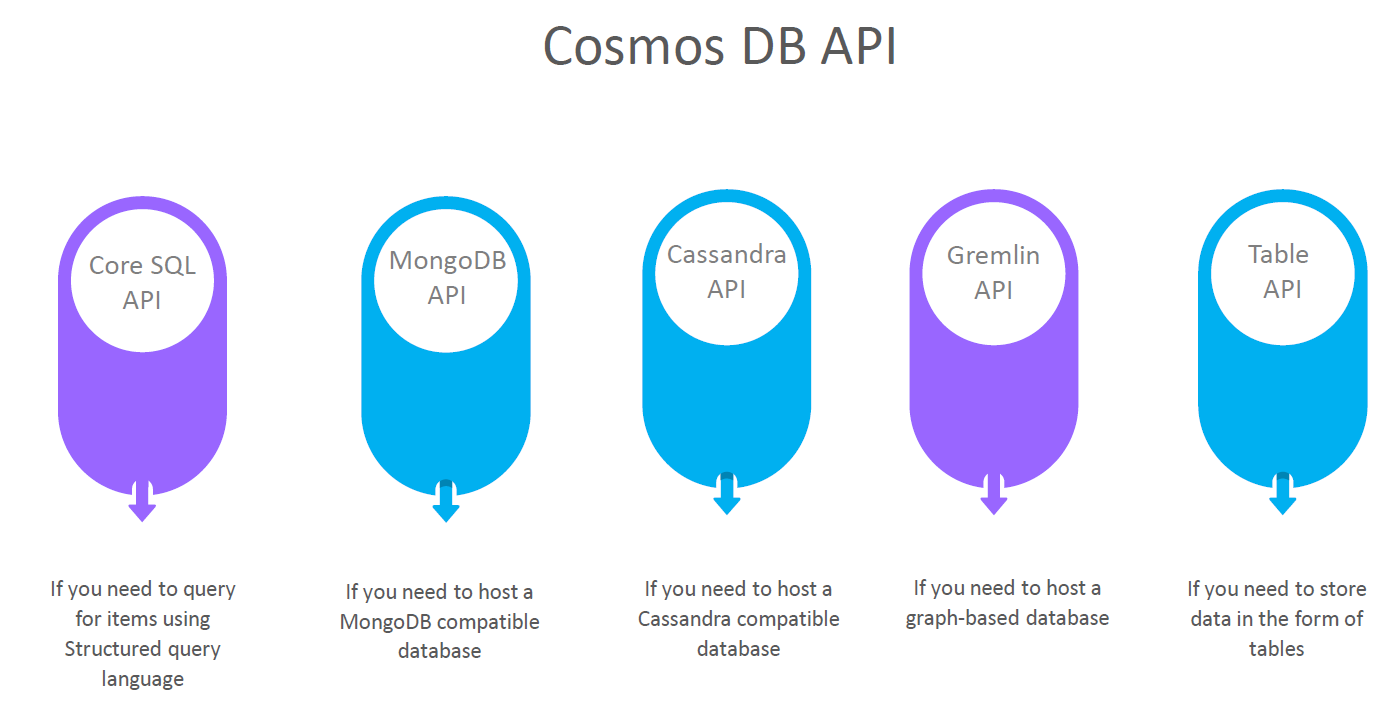
* So normally companies would allow users to have their own file shares.
* So if they want to store some data somewhere, they would have file shares that would be hosted on,
* let's see, indoor service.
* Now, with the help of the OCR Sync service, you can actually sync or have those files stored on file
* shares in a storage account.
* As we all know, the easier storage account is highly available and a durable service that you don't
* have to think about the storage.
* Then maybe as your file sync service, you can have the files in the Azoff File shares.
* The most commonly used one actually available on Windows service and use can access those files seamlessly.
* So this is actually a further integration of having file shares with the use up as your storage accounts
* by the File Share Service, you are using the power of giving the ability of users to access files very
* easily from Windows servers, but also ensuring that those files are stored in a highly available and
* durable service has file shares.

# AZURE COSMOS DATABASE

* This is a fully managed NoSQL database. where underlying infrastructure is completely managed by Azure.
* The database provides fast response time and is highly scalable.
* Commonly used for web, mobile, gaming and IoT applications that need to handle massive amounts of data.
* Azure Cosmos DB is a globally distributed, multi-model database service.
* You can elastically and independently scale throughput and storage across any number of Azure regions worldwide.
* Azure Cosmos DB supports schema-less data, which lets you build highly responsive and "Always On" applications to support constantly changing data.
* We can use this feature to store data that's updated and maintained by users around the world.

|  |  |
| --- | --- |
| **SQL DB** | **COSMOS DB** |
| We need to have relationship between tables | No SQL data store |
| When we you want to have constraints like foreign key constraint | Flexible Schema  No need of joins between data structures. |

## COSMOS DB API



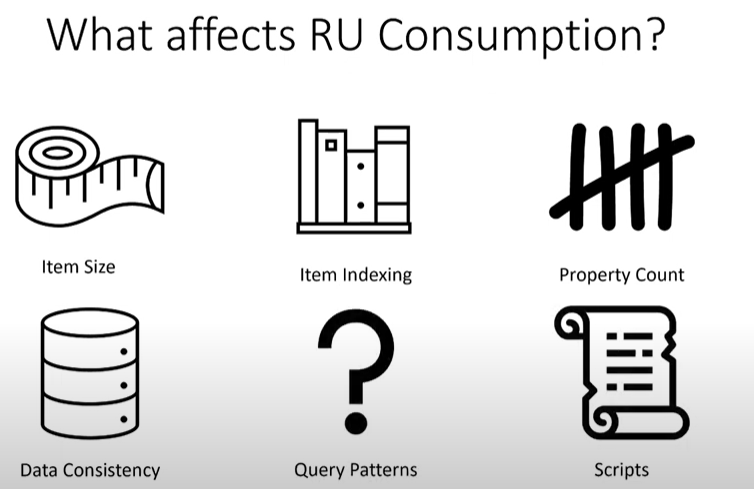
### COSTING

***What is Throughput? - Throughput is a measure of how many units of information a system can process in a given amount of time.***

#### REQUEST UNIT

|  |  |
| --- | --- |
|  | * The cost of database operations of cosmos DB is measured in terms of request units. * **Request units is a measure of the CPU, the IOPS and the memory. So instead of charging separately for CPU, the IOPS and the memory everything is taken as a single measure and that's called request units.** * RU is the currency of throughput |

##### FACTORS IMPACT RU



### PROVISIONING MODE OF COSMOS DB

The type of Azure Cosmos DB account you're using determines the way consumed RUs get charged. There are three modes in which you can create an account:

1. **PROVISIONED THROUGHPUT MODE**:
2. **SERVERLESS MODE**:
3. **AUTOSCALE MODE**

#### PROVISIONED THROUGHPUT MODE

* In provisioned throughput - we pre-assign the number of RUs for an application on a per-second basis. **Hence for this type of provisioning need Capacity Planning to make sure we satisfy the workload of the application**. This kind of provisioning is best suited for the application which are stable and predictable workload in
* The RU can be incremented in terms of 100 RUs per second. This can be used to scale the application

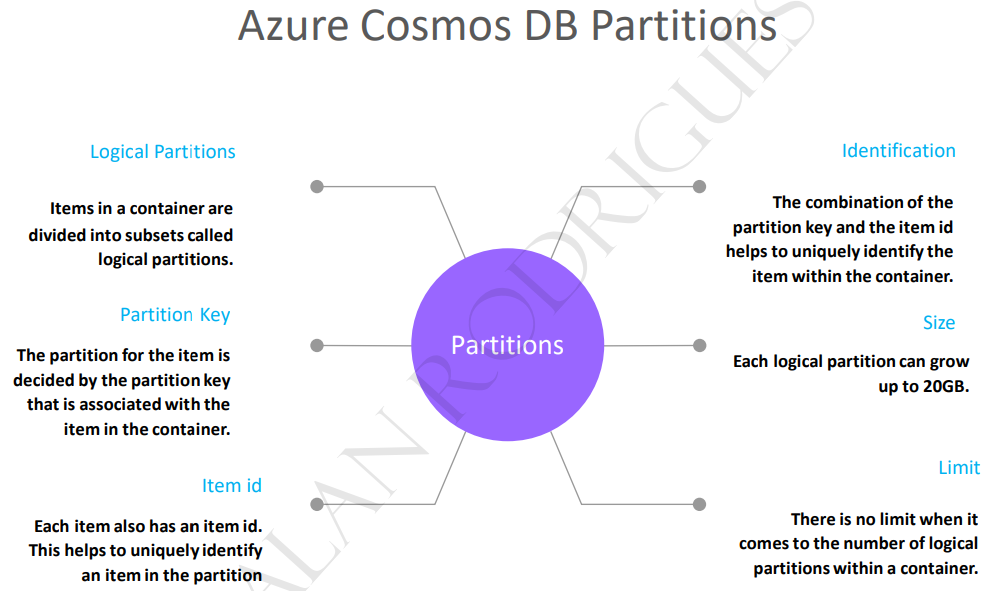
#### AUTOSCALE MODE

* In this mode, you can automatically and instantly scale the throughput (RU/s) of your database or container based on its usage.
* This scaling operation doesn't affect the availability, latency, throughput, or performance of the workload.
* This mode is well suited for mission-critical workloads that have variable or unpredictable traffic patterns and require SLAs on high performance and scale.

#### SERVERLESS MODE

* In this mode, you don't have to assign any throughput when creating resources in your Azure Cosmos DB account.
* At the end of your billing period, you get billed for the number of Request Units consumed by your database operations.

### PARTITIONS IN COSMOS DB



* The items within a container are divided into subsets called logical partitions. It helps for the faster querying of data if the data is spread across multiple partitions.
* The logical division get partitioned is via a partition key.
* One of the properties of the item behave as the partition key.
* **Each item within a container** also gets an item id which help in uniquely identify an item within the partition itself.
* The combination of the partition key and the item id helps to uniquely identify the item within the entire container.

#### SELECTION OF PARTITION KEY

While selecting the partition kry

1. **Choose a property to be partition key in which value does not change**
2. **The property should have wide range of values**
3. **Once we decide on partition key for a container – we can’t change it.**

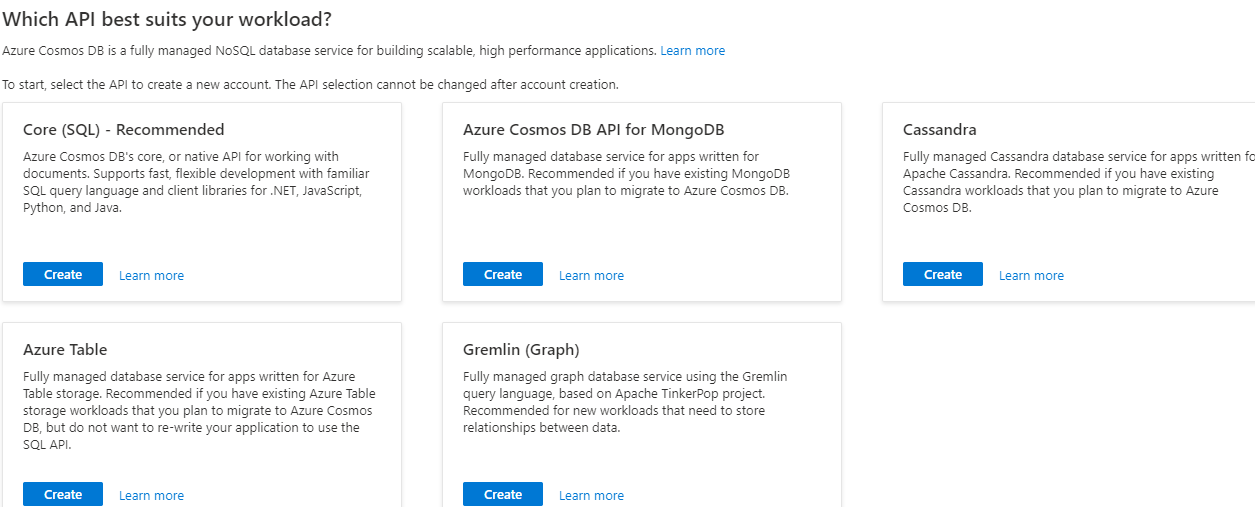
## CREATING A COSMOS DB ACCOUNT

|  |  |
| --- | --- |
|  | Step 1: Create Database account. During account creation we need to choose the type of API.  Step 2: Within the database account, we create a database.  Step 3: Create Container within database  Step 4: Within database – we create items  Note – The terminologies changes when with the type of API selected |

|  |  |  |  |
| --- | --- | --- | --- |
| API TYPE | DATABASE | CONTAINER | ITEM |
| SQL API | ***Database*** | ***Container*** | ***Item*** |
| MONGO DB API | ***Database*** | ***Collection*** | ***Document*** |
| GREMLIN API | ***Database*** | ***Graph*** | ***Node/ Edge*** |
| CASSANDRA API | ***Keyspace*** | ***Table*** | ***Row*** |
| TABLE API | ***NA*** | ***Table*** | ***Item*** |

### SELECTING COSMOS DB API

1. We want to host a Data where we can issue SQL based command 🡪 Select Core(SQL) API.
2. All the data is stored in form of JSON when it comes to underlying document.



### COSMOS DB ACCOUNT CONFIGURATION

|  |  |
| --- | --- |
|  | 1. When it comes to costing of the cosmos Db account it can be    1. Provisioned throughput    2. Serverless   PROVISION THROUGHOUT:  In this case, we are not charged separately for CPU , Memory etc. but we are charged based on RU(request unit) which is kind bundle of CPU , memory etc.. |

### CREATING A COSMOS DATABASE AND CONTAINER

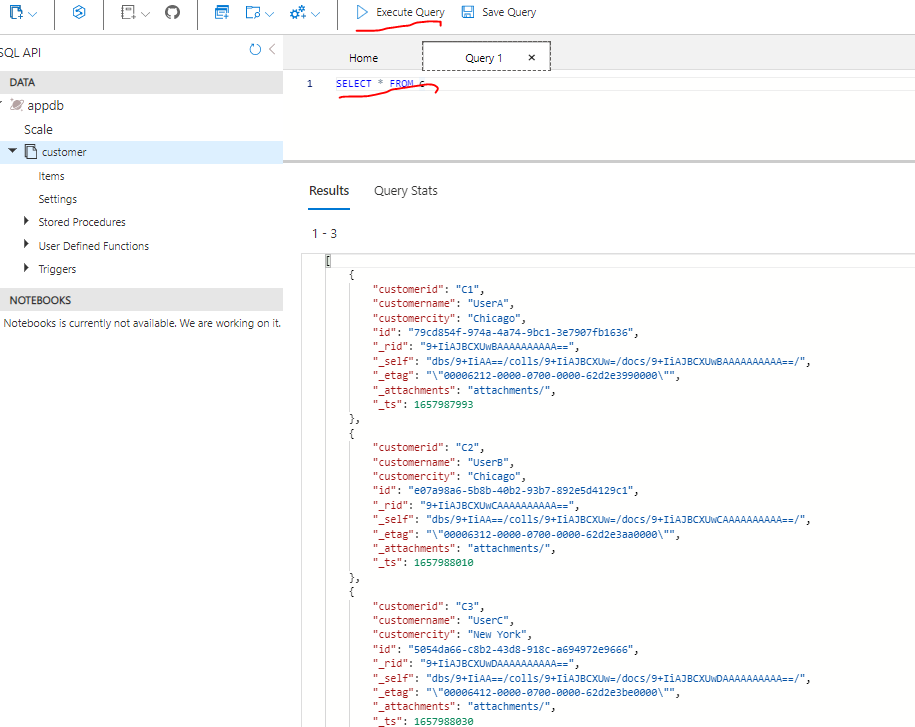
* Go to Data explorer 🡪 New Container

|  |  |
| --- | --- |
|  | * Step 1: We need to create a database * Step 2: The database contains container. Each database can have multiple containers. Container is equivalent to table in relational DB * Step 3: Container can have items. Item is equivalent to one record in relational DB   **CREATED DB AND CONTAINER** |

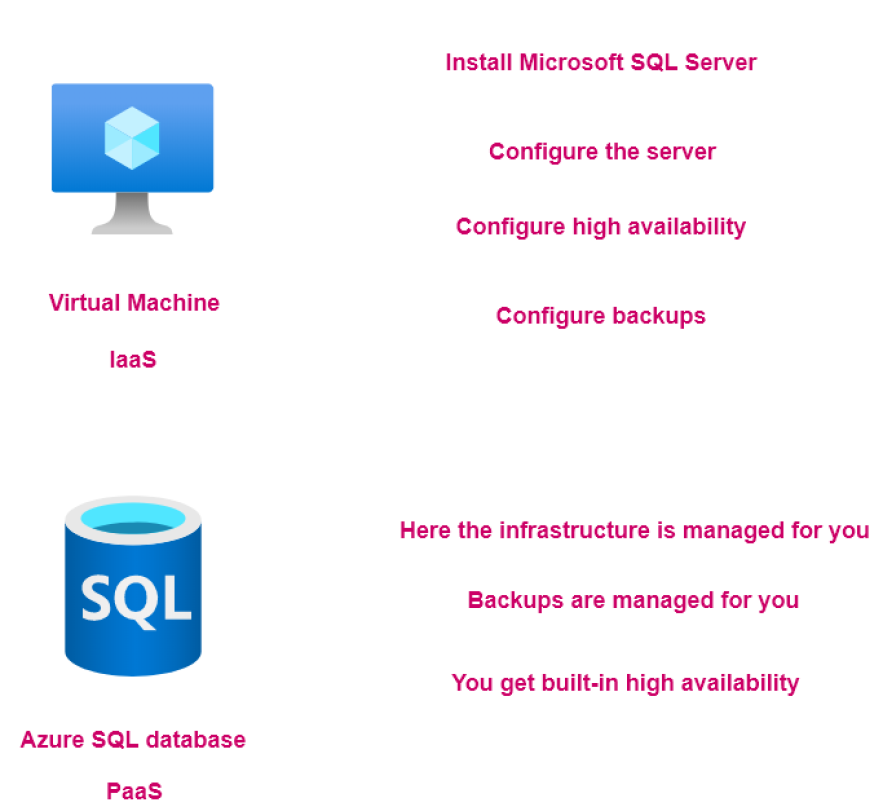
### CREATING A COSMOS DATABASE AND CONTAINER

|  |  |
| --- | --- |
|  | * To add new items in the container 🡪 New Items   DATA FORMAT  {  "customerid":"C1",  "customername":"UserA",  "customercity":"Chicago"  } |

### QUERYING CONTAINER DATA



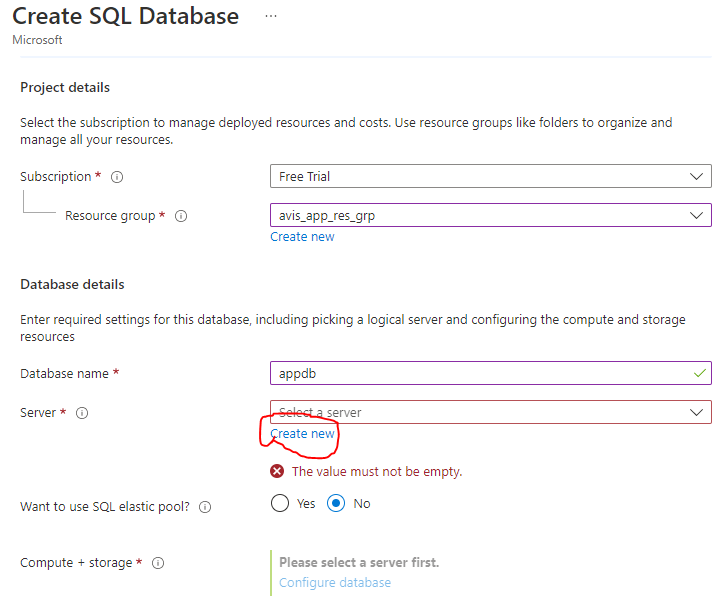
# AZURE SQL DATABASE



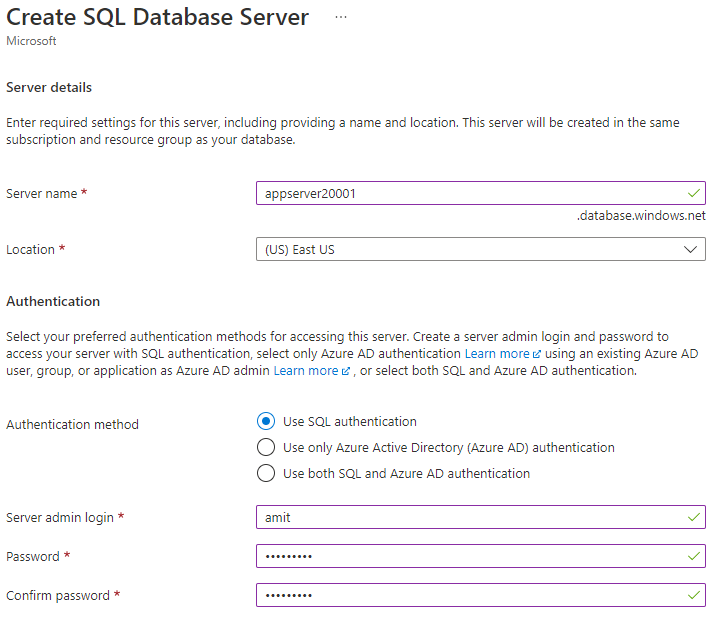
## CREATING A SQL DATABASE SERVER

* When we use SQL Database service – it creates two resources
  + Azure SQL Database Server: This is managed by Azure itself. That’s the Azure SQL database is known has PaaS (Platform as a Service)
  + Azure SQL Database: To host the tables of data

STEP 1: CREATE THE SQL DATABASE



STEP 2: CREATE THE SQL DATABASE SERVER



STEP 3: CONFIGURING THE SQL DATABASE (COMPUTE AND STOARAGE CONFIGURATION)

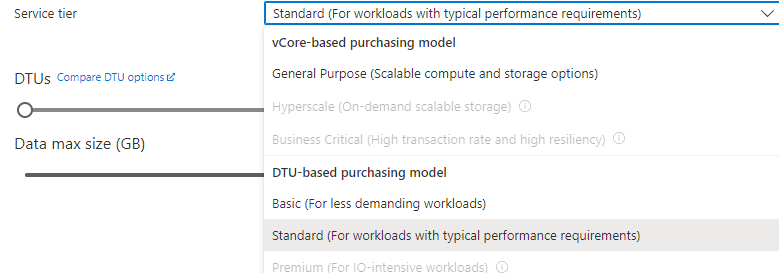
SQL database service pricing tier are classified in two broad categories:

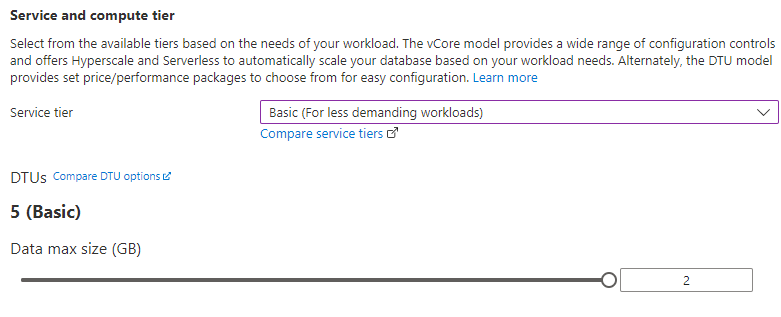
**DTU –DATABASE TRANSACTION UNITS.**

* This is a blended measure of CPU, Memory and Input/Output.
* There are different pricing tiers when it comes to the DTU model.

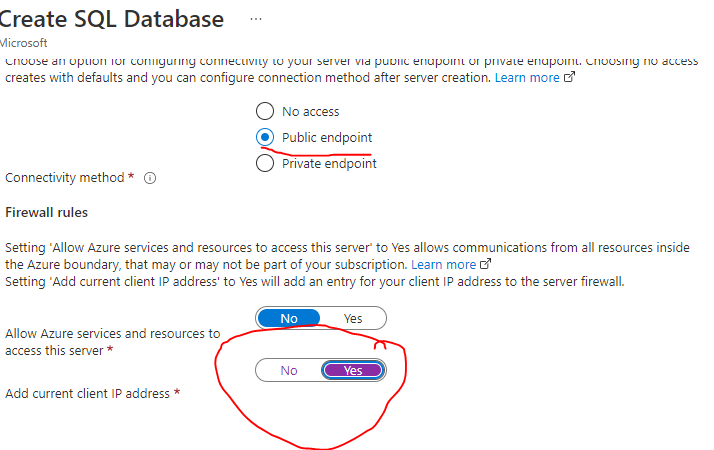
**VCORE-BASED PURCHASING MODEL.**

* Here you can independently scale compute and storage.
* You can make use of the hybrid benefit model. Here you can save on costs if you have existing SQL Server licenses.

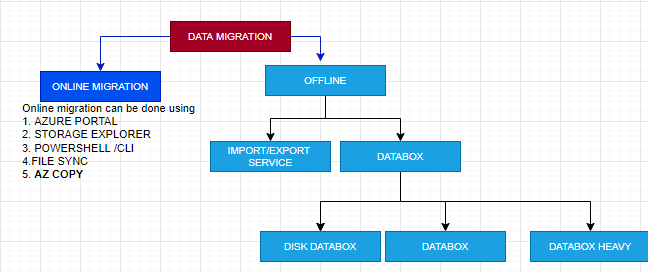




NETWORKING



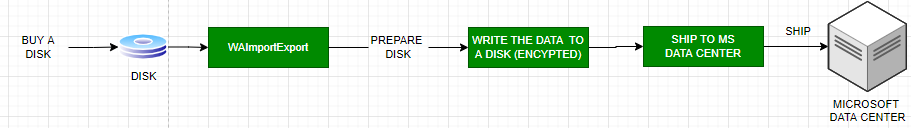
# DATA MIGRATION



Note - Offline methods of data migration are used when the data size is too huge to transfer over network.

## OFFLINE DATA MIGRATION

* Offline data migration – make use of physical storage device.
* Azure Import/Export service is used to securely import large amounts of data to Azure Blob storage and Azure Files by shipping disk drives to an Azure datacenter. This service can also be used to transfer data from ***Azure Blob storage*** to disk drives and ship to on-premises sites.
* Data from one or more disk drives can be imported either to Azure Blob storage or Azure Files.
* ***In Azure Import/Export service - we need to supply our own disk drives.***



* We can also use disk drives supplied by Microsoft. ***If we want to transfer data using disk drives supplied by Microsoft, we can use Azure Data Box Disk to import data into Azure.*** Microsoft ships up to 5 encrypted solid-state disk drives (SSDs) with a 40 TB total capacity per order, to your datacenter through a regional carrier. You can quickly configure disk drives, copy data to disk drives over a USB 3.0 connection, and ship the disk drives back to Azure.

## ONLINE DATA MIGRATION

#### AZCOPY

|  |  |
| --- | --- |
|  | * ***AzCopy is online mode of data migration/ copying to Storage account. Its a command line tool available as a zip file*** * Download Link (Zip File) : <https://learn.microsoft.com/en-us/azure/storage/common/storage-use-azcopy-v10> |

1. To copy data to storage account we need authorize