Agenda: Azure Storage Service

- Introduction to Storage Service
- Creating a Storage Account
- Working with Blob Storage
    - Types of Blobs (Block, Append, Page)
    - Container and Metadata
    - Soft delete, Snapshot
- Azure Storage Explorer
- Transfer Data using AzCopy
- Azure blob storage Lifecycle Management
- Storage account Security
    - Encryption at rest and transit
    - Role -based access control
    - Create and Manage Shared Access Signature
    - Account SAS vs Service SAS
    - Using Stored Access Policies
    - Regenerating Keys
    - **Encrypt Keys using Azure Key Vault integration**
    - Firewall and Virtual Network Integration
- Service EndPoint and Private EndPoint
    - Service EndPoint
    - Configure Networking with Service End Point
    - Azure Private Link
    - Private End Point
    - Configure Networking with Private End Point
- Working with Table Storage
    - Understanding NoSQL Database
    - Creating Table and Entities using Storage Explorer
    - Entities and Properties
- Azure SMB File Storage
    - Common usage of File Storage
    - Shares, Directory and File
    - Managing Using Azure Portal

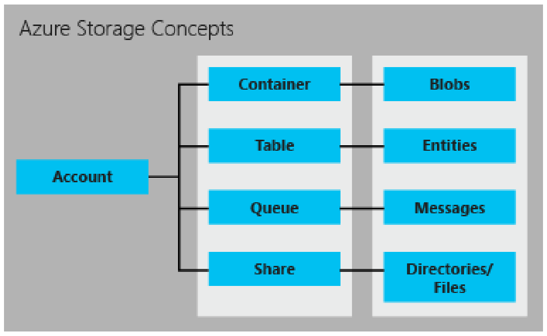
Introduction to Azure Storage Service

Cloud computing enables new scenarios for applications requiring **scalable, durable and highly available** storage for their data – which is exactly why Microsoft developed **Azure Storage Service.**

- Azure Storage is a **PaaS service** that you can use to store both **unstructured** and **partially structured** data.
- **Azure Storage is massively scalable and elastic:** It can store and process **hundreds of terabytes of data** to support the big data scenarios required by scientific, financial analysis, and media applications. Or you can store the **small amounts of data** required for a small business website.
- By default, you can create up to **100 storage accounts** in a single Azure subscription. Each standard storage account can contain up to **500 TB** of combined blob, queue, table and file data.
- As the demands on your storage application grow, Azure Storage **automatically allocates** the appropriate resources to meet them. **We are charged only for what we use**.
- SDKs for Azure Storage in a variety of languages – .NET, Java, Node.js, Python, PHP, Ruby, Go, and others – as well as REST APIs.It also supports scripting in Azure PowerShell or Azure CLI.

It offers **four types of storage services**, depending on the type of data that they are designed to store:

1. **Blob Storage** stores file data. A blob can be any type of **text or binary data**, such as a document, media file,datbase backup,VHD files or application installer. Blob Storage is sometimes referred to as **Object storage**.
2. **Table Storage** stores partially structured datasets. Table storage is a **NoSQL** key-attribute data store, which allows for rapid development and fast access to large quantities of data.
3. **Queue Storage** provides **reliable messaging** for workflow processing and for communication between components of cloud services.
4. **File Storage** Similar to blobs, these provide storage for unstructured files, but they offer support for file sharing in the same manner as traditional on-premises Windows file shares.



Azure Storage Account

An Azure storage account is a **secure account** that gives you access to services in Azure Storage..Your storage account provides the unique namespace for your storage resources. There are two types of storage accounts:

1. A **standard storage** account includes Blob, Table, Queue, and File storage. Standard use HDD Drives
2. A **premium storage** account is ideally supposed to be used for Azure Virtual Machine disks. Premiun use SSD Drives

*Note:*Combining data services(Blob, Table, Queue, and File storage) into a storage account lets you manage them as a group. The settings you specify when for the account, are applied to everything in the account. Deleting the storage account deletes all of the data stored inside it.

Creating Storage Account

The tools available for creating storage account are Portal,CLI,Powershell,Management client libraries.
The Azure CLI and Azure PowerShell let you write scripts, while the management libraries allow you to incorporate the creation into a client app.

Lab1: Create Storage Account Using Portal

1. Azure Portal → Browse Storage Accounts → **New** → **Data + Storage** → **Storage account**
2. Enter Name (must be all lowercase)
3. Deployment Model = Resource Manager
4. and select Subscription, Resource Group, Location
5. Performance: **Standard / Premium**

   Standard use HDD Drives and Premiun use SSD Drives

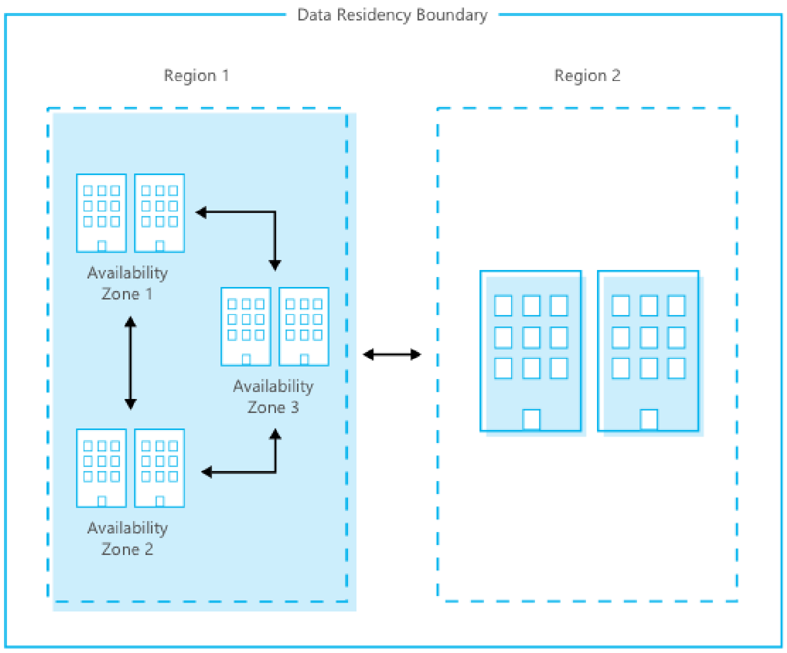
   Premium is used for disks of VMs (Page Blobs)

   Note that it is not possible to convert a Standard storage account to Premium storage account or vice versa.
6. Access tier: Cool / **Hot**

   Account kind: **Blob storage,** Performance**: Standard**

   - Access tier: **Hot**, if objects will be **more** frequently accessed. This allows you to store data at a **lower access cost.**
   - Access tier: **Cool**, if objects will be **less** frequently accessed. This allows you to store data at **a lower data storage cost.**
   - Acces tier: **Archive.** The archive tier is optimized for data that can tolerate several hours of retrieval latency and will remain in the Archive tier for at least 180 days. The archive tier is the most cost-effective option for storing data, but accessing that data is more expensive than accessing data in the hot or cool tiers. It is available at level of an individual blob only, not at the storage account level. Only block blobs and append blobs can be archived.
7. Replication: LRS

Azure Regions and availability Zones :



Regions

- A region is a set of datacenters deployed within a latency-defined perimeter and connected through a dedicated regional low-latency network.

Availability Zones

- Availability Zones are unique physical locations within an Azure region. Each zone is made up of one or more datacenters equipped with independent power, cooling, and networking.

Locally redundant storage (LRS):

- **Replicates 3 times within a single data center in a single region where Storage Account is created.**
- Locally redundant storage (LRS) provides at least 99.999999999% (11 nines) durability of objects over a given year.
- The replicas are spread across UDs and FDs within one storage scale unit (A storage scale unit is a collection of racks of storage nodes.)
- A request returns successfully only once it has been written to all three replicas.
- This architecture ensures your data is available if a hardware failure affects a single rack or when nodes are upgraded during a service upgrade.
- LRS is less expensive than GRS and also offers higher throughput.
- For Premium Storage accounts - This is the only option available.

Zone-redundant storage (ZRS)

- Replicates your data across three (3) storage clusters in a single region. Each storage cluster is physically separated from the others and resides in its own availability zone. Each availability zone, and the ZRS cluster within it, is autonomous, with separate utilities and networking capabilities.
- ZRS is not yet available in all regions.
- Once you have created your storage account and selected ZRS, you **cannot convert** it to use to any other type of replication, or vice versa.
- Consider ZRS for scenarios that require strong consistency, strong durability, and high availability even if an outage or natural disaster renders a zonal data center unavailable.
- **What happens when a zone becomes unavailable?** Your data is still accessible for both read and write operations even if a zone becomes unavailable. Microsoft recommends that you continue to follow practices for **transient fault handling**. These practices include implementing retry policies with exponential back-off.

Geo-redundant storage (GRS)

- GRS maintains **6 copies** of your data. 3 replicas in **primary region** and also replicates your data 3 additional times to a **secondary region** that is hundreds of miles away from the primary region.
- 99.99999999999999 (16 9s) Availablility is supported when calculated around a year.
- Data is durable even in the case of a complete regional outage or a disaster in which the primary region is not recoverable.
- If failure occurs in the primary region, Azure Storage **automatically failover** to the secondary region.
- An update is first committed to the primary region, where it is replicated three times. Then the update is replicated to the secondary region, where it is also replicated three times.
- Requests to write data are replicated **asynchronously** to the secondary region. It is important to note that opting for GRS does not impact latency of requests made against the primary region.
- The secondary region is **automatically determined** based on the primary region, and cannot be changed.

Read-access geo-redundant storage (RA-GRS)

- As with GRS, your data replicates asynchronously across two regions and synchronously within each region, yielding six copies of a storage account.
- This is default option when we create a storage account.
- In the event that data becomes unavailable in the primary region, your application can **read data** from the secondary region.

- If your primary endpoint for the Blob service is **myaccount.blob.core.windows.net**, then your secondary endpoint is **myaccount-secondary.blob.core.windows.net**. The **access keys** for your storage account are the **same** for both the primary and secondary endpoints.

More Details: https://azure.microsoft.com/en-in/documentation/articles/storage-redundancy/

8. Secure transfer required = **Disabled** / Enabled

   If Enabled only HTTPS requests will be accepted.

   This option doesn't work with Custom Domain Names for Storage account.
9. Select Subscription, Resource Group, Location
10. Virtual network: Disabled
11. Click on "Create".

Note: The number of storage accounts you need is typically determined by your data diversity, cost sensitivity, and tolerance for management overhead.

Data Diversity:Based on Loaction or based on Public/Private data

Cost sensitivity:Based on redundancy requirement or performance.

Pricing and Billing

All storage accounts use a pricing model for blob storage based on the tier of each blob.

When using a storage account, the following billing considerations apply:

- **Storage costs**: In addition to, the amount of data stored, the cost of storing data varies depending on the storage tier. The per-gigabyte cost decreases as the tier gets cooler.
- **Data access costs**: Data access charges increase as the tier gets cooler. For data in the cool and archive storage tier, you are charged a per-gigabyte data access charge for reads.
- **Transaction costs**: There is a per-transaction charge for all tiers that increases as the tier gets cooler.
- **Geo-Replication data transfer costs**: This charge only applies to accounts with geo-replication configured, including GRS and RA-GRS. Geo-replication data transfer incurs a per-gigabyte charge.
- **Outbound data transfer costs**: Outbound data transfers (data that is transferred out of an Azure region) incur billing for bandwidth usage on a per-gigabyte basis, consistent with general-purpose storage accounts.
- **Changing the storage tier**: Changing the account storage tier from cool to hot incurs a charge equal to reading all the data existing in the storage account. However, changing the account storage tier from hot to cool incurs a charge equal to writing all the data into the cool tier (GPv2 accounts only).

Working with Blob Storage

- **Blobs** are binary large objects. The Blob service stores text and binary data.
- Blob storage is also referred to as **object storage**.
- You can use Blob storage to store content such as:
    - Documents
    - Social data such as photos, videos, music, and blogs
    - Backups of files, computers, databases, and devices
    - Images and text for web applications
    - Configuration data for cloud applications
    - Big data, such as logs and other large datasets
- Every blob is organized into a **container**. Containers also provide a useful way to assign security policies to groups of objects. A storage account can contain any number of containers, and a container can contain any number of blobs, up to the **500 TB capacity** limit of the storage account.
- **Creating BLOB Hierarchies:** The blob service in Azure Storage is based on a **flat storage scheme**. This means that creating a container one level below the **root is the only true level of container**. However, you can specify a delimiter as part of the blob name to create your own **virtual hierarchy**. For example, you could create a blob named **/January/Reports.txt** and **/February/Reports.txt**, and filter based on /January or /February in most tools that support Azure Storage. Most third-party storage tools allow you to create folders within a container, but they are actually being clever with the name of the blob itself.

Blobs are addressable using the following URL format:

http(s)://<storage account name>.**blob**.core.windows.net/<container>/<blob name>

Lab2:Create container,Upload/download blobs,work with soft delete,snapshot,public and private blobs.

Types of blobs:

1. **Block blobs** are optimized for streaming (**sequential access**) and for uploads and downloads, and are a good choice for storing documents, media files, backups etc. Azure divides data into smaller blocks of up to 100 megabytes (MB) in size, which subsequently upload or download in parallel. Individual block blobs (file) can be up to 100 GB in size. One blob can have max of 50,000 blocks.
2. **Append blobs:** Append blobs are similar to block blobs, but are optimized for append operations. This works best with **logging and auditing** activities. Updating or deleting of existing blocks is not supported.
3. **Page blobs** are optimized for **random read/write** operations and provide the ability to write to a range of bytes in a blob. Blobs are accessed as pages, each of which is up to **512 bytes** in size. Each Page blob can be up to **8TB** each. Is best suited for **virtual machine disks**.

Soft Delete (Blob Service→Data Protection→Blob soft delete

- Enables blobs and checkpoints to be recovered if deleted.
- It utilizes special soft delete snapshot.
- Configured at storage account level for specific retention duration.

Immutable blob:

- It enables blob to be configured as read only so that it can not be altered or deleted.
- It is configured at container level.

Blob snapshot:

- Enables point in time copy of blob content to be taken.
- They are associated with parent blob.

Access Key:

- To acces the blob via application (java,C# etc) this key is required.
- Two keys are useful in scenario where you need to regenarate the key for security purpose and you do not want any downtime for your application.

Azure Storage Explorer

Microsoft Azure Storage Explorer is a standalone app from Microsoft that allows you to easily work with Azure Storage data.

Some of the benefits of Azure Storage Explorer are:

a) Access multiple accounts and subscriptions across Azure

b) Create, delete, view, and edit storage resources

c) View and edit Blob, Queue, Table, File, Cosmos DB storage and Data Lake Storage.

d) Obtain shared access signature (SAS) keys

e) Available for Windows, Mac, and Linux

Azure Storage Explorer has many uses when it comes to managing your storage. See the following articles to learn more. Also, check out the videos that follow this topic.

- **Connect to an Azure subscription:** Manage storage resources that belong to your Azure subscription.
- **Work with local development storage:** Manage local storage by using the Azure Storage Emulator.
- **Attach to external storage:** Manage storage resources that belong to another Azure subscription or that are under national Azure clouds by using the storage account's name, key, and endpoints.
- **Attach a storage account by using an SAS:** Manage storage resources that belong to another Azure subscription by using a shared access signature (SAS).

- **Attach a service by using an SAS:** Manage a specific storage service (blob container, queue, or table) that belongs to another Azure subscription by using an SAS.
- **Connect to an Azure Cosmos DB account by using a connection string:** Manage Cosmos DB account by using a connection string.

Lab3:Connect to storage Explorer using different option and work with it

Transfer data with the AzCopy

AzCopy is a command-line utility designed for copying data to/from Microsoft Azure Blob, File, and Table storage, using simple commands designed for optimal performance. You can copy data between a file system and a storage account, or between storage accounts.

There are two versions of AzCopy that you can download.

1. **AzCopy on Windows** is built with .NET Framework, and offers Windows style command-line options.
2. **AzCopy on Linux** is built with .NET Core Framework which targets Linux platforms offering POSIX style command-line options.

Path on Windows: C:\Program Files (x86)\Microsoft SDKs\Azure\AzCopy\AzCopy.exe

The basic syntax for AzCopy commands is:

```
AzCopy /Source:<source> /Dest:<destination> [Options]
```

Lab4:Use Azcopy to copy data from to and from local machine to storage

Copy single file from Local to blob storage:

Azcopy copy "D:\DP200\Tutorial\Data Storage Tutorial\orders\orders-2013.txt" “https://dssdemosa.blob.core.windows.net/mycontainer1/?sv=2020-02-10&ss=bfqt&srt=sco&sp=rwdlacupx&se=2021-05-10T23:41:01Z&st=2021-05-10T15:41:01Z&spr=https&sig=N6IvekStcKcowZYAmv96JYbb4sk5JjZqZRrYyc%2FV2%2B8%3D”

Copy complete folder from Local to blob storage:

Azcopy copy "D:\DP200\Tutorial\Data Storage Tutorial" “https://dssdemosa.blob.core.windows.net/mycontainer1/?sv=2020-02-10&ss=bfqt&srt=sco&sp=rwdlacupx&se=2021-05-10T23:41:01Z&st=2021-05-10T15:41:01Z&spr=https&sig=N6IvekStcKcowZYAmv96JYbb4sk5JjZqZRrYyc%2FV2%2B8%3D” –recursive

Refer for more information:

https://docs.microsoft.com/en-us/azure/storage/common/storage-use-azcopy-v10

What's new

- Synchronize a file system to Azure Blob or vice versa. Ideal for incremental copy scenarios.
- Supports Azure Data Lake Storage Gen2 APIs.
- Supports copying an entire account (Blob service only) to another account.
- Account to account copy is now using the new Put from URL APIs. No data transfer to the client is needed which makes the transfer faster.
- List/Remove files and blobs in a given path.
- Supports wildcard patterns in a path as well as –include and –exclude flags.
- Improved resiliency: every AzCopy instance will create a job order and a related log file. You can view and restart previous jobs and resume failed jobs. AzCopy will also automatically retry a transfer after a failure.
- General performance improvements.

Note: *To import file sizes above, Data Engineers must use PowerShell or Visual Studio. The AzCopy Tool supports a maximum file size of 1Tb and will automatically split into multiple files if the data file exceeds 200Gb*.

Azure blob storage Lifecycle Management

Azure Blob storage lifecycle management offers policy to transition your data to the appropriate access tiers (Hot,Cool,Archive)or expire at the end of the data's lifecycle.
By adjusting storage tiers in respect to the age of data, you can design the least expensive storage options for your needs. You can create a policy that deletes old snapshots based on snapshot age.

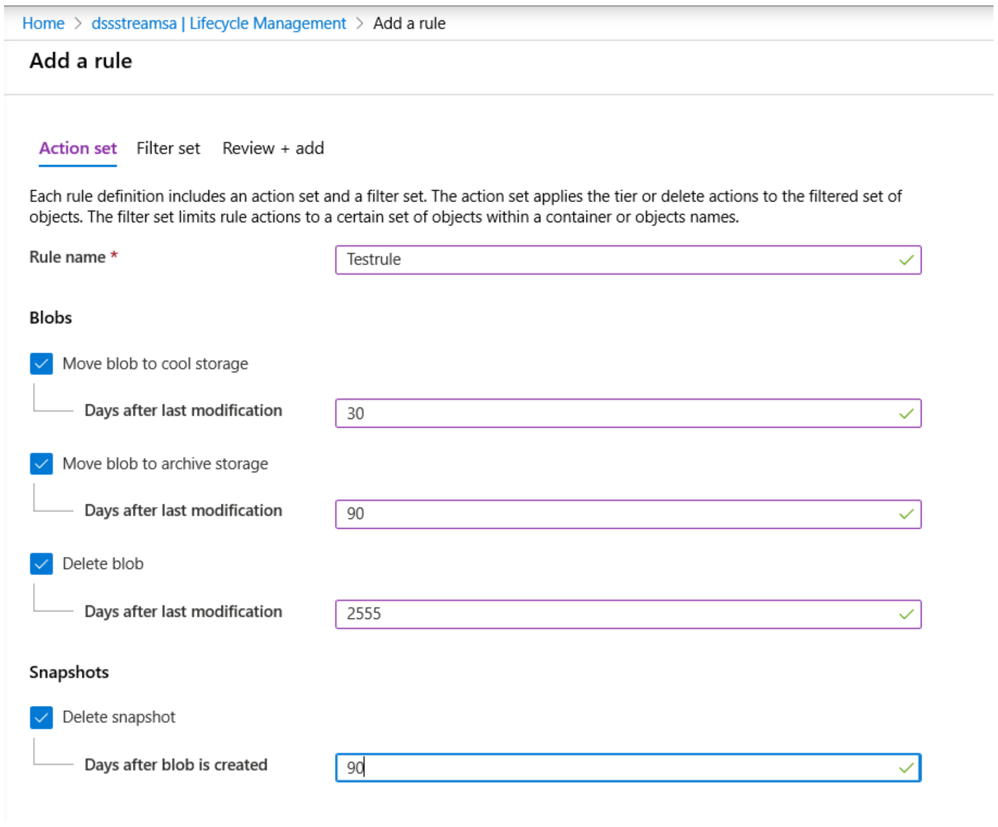
The lifecycle management policy lets you:

- Transition blobs to a cooler storage tier (hot to cool, hot to archive, or cool to archive) to optimize for performance and cost
- Delete blobs at the end of their lifecycles
- Define rules to be run once per day at the storage account level
- Apply rules to containers or a subset of blobs (using prefixes as filters)

Example:
Tier blob to cool tier 30 days after last modification
Tier blob to archive tier 90 days after last modification
Delete blob 2,555 days (seven years) after last modification

Delete blob snapshots 90 days after snapshot creation

Lab5 :Create Lifecycle management policy for blobs



Home > dssstreamsa | Lifecycle Management > Add a rule

Add a rule

Action set | Filter set | Review + add

Each rule definition includes an action set and a filter set. The action set applies the tier or delete actions to the filtered set of objects. The filter set limits rule actions to a certain set of objects within a container or objects names.

Rule name *: Testrule

Blobs

[✓] Move blob to cool storage
- Days after last modification: 30

[✓] Move blob to archive storage
- Days after last modification: 90

[✓] Delete blob
- Days after last modification: 2555

Snapshots

[✓] Delete snapshot
- Days after blob is created: 90

Azure Portal → Select Storage Account → Blob Service → Lifecycle Management

Code View:

```
{
 "rules": [
  {
   "name": "ruleFoo",
   "enabled": true,
   "type": "Lifecycle",
   "definition": {
    "filters": {
     "blobTypes": [ "blockBlob" ],
     "prefixMatch": [ "container1/foo" ]
    },
    "actions": {
     "baseBlob": {
      "tierToCool": { "daysAfterModificationGreaterThan": 30 },
      "tierToArchive": { "daysAfterModificationGreaterThan": 90 },
      "delete": { "daysAfterModificationGreaterThan": 2555 }
     },
     "snapshot": {
      "delete": { "daysAfterCreationGreaterThan": 90 }
     }
    }
   }
  }
 ]
}
```

Storage Account Security

- Azure storage is exposed via internet facing REST API endpoint via HTTP/HTTPS.
- This endpoint can be utilized using different operating systems and different programming languages via REST API or client libraries.
- Access is authenticated via **Shared Key** or **Shared access signature**
- Anonymous read access is also possible for some services.

Security features:

Azure Storage encryption for data at rest

- Azure Storage automatically encrypts your data when persisting it to the cloud by Storage Service Encryption (SSE)
- SSE automatically encrypts data when writing it to Azure Storage. When you read data from Azure Storage, Azure Storage decrypts the data before returning it.
- Storage accounts are encrypted regardless of their performance tier (standard or premium)
- All Azure Storage resources are encrypted, including blobs, disks, files, queues, and tables. All object metadata is also encrypted.
- Encryption does not affect Azure Storage performance.
- You can rely on **Microsoft-managed keys** for the encryption of your storage account, or you can manage encryption with your own keys.

Encryption in transit

- You can enable Encryption by setting **Secure transfer required**(Settings→Configuration)

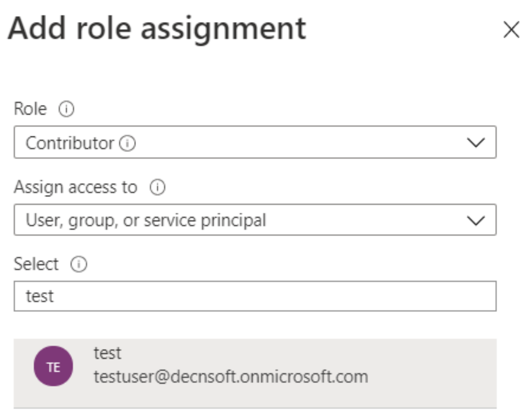
- When enabled you need to use https for Rest API call and SMB 3.0 with encryption for Azure Files

Role-based access control

- Azure Storage supports Azure Active Directory and role-based access control (RBAC) for both resource management and data operations.
- To security principals, you can assign RBAC roles that are scoped to the storage account.

Lab6: Give RBAC to user for storage account

1. Storage Account→Access Control(IAM)→Add role assignment→


Add role assignment

Role: Contributor

Assign access to: User, group, or service principal

Select: test

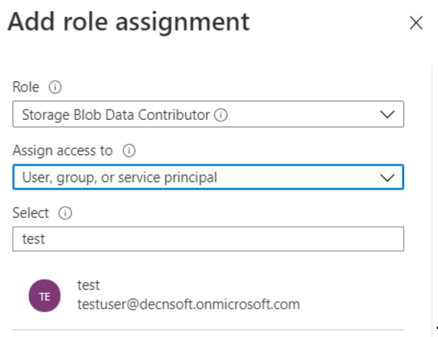
test
testuser@decnsoft.onmicrosoft.com

→Select the required user→save.

Login with that user and observe that user will be able to see the storage account.

2. Give permissions for container

   Container→demo→Access Control(IAM)→Add role assignment→


Add role assignment

Role: Storage Blob Data Contributor

Assign access to: User, group, or service principal

Select: test

test
testuser@decnsoft.onmicrosoft.com

→ Select the required user→save.

Shared Access Policy and Shared Access Signature (SAS) Token

There are two techniques for controlling access to objects within an Azure Storage account.

1. Using the **authentication key and storage account name** is one technique.
2. Granting access using a **shared access signature** or **via a policy** to allow granular access with expiration is another technique.

Authentication key /Shared keys are called storage account keys**. (Settings→Access Keys)**

- Azure creates two of these keys (primary and secondary) for each storage account you create. The keys give access to everything in the account.
- The client can embed the shared key in the HTTP Authorization header of every request, and the Storage account validates the key.
- It is recommended to use these keys only with trusted in-house applications that you control completely.

A shared access signature (SAS) is a URI that grants **restricted access rights** to Azure Storage resources. You can provide a shared access signature to clients who should not be trusted with your storage account key but whom you wish to delegate access to certain storage account resources. By distributing a shared access signature URI to these clients, you grant them access to a resource for a specified **period of time**.

- A shared access signature (SAS) is a **token** that can be appended to a URL that enables delegated access to a storage resource.
- Anyone who possesses the token can access the resource it points to with the permissions it specifies, for the period of time that it is valid.

Azure Storage supports two kinds of shared access signatures:

1. An **Account SAS** delegates access to resources in **one or more** of the storage services. You can also delegate access to read, write, and delete operations on blob containers, tables, queues, and file shares that are not permitted with a service SAS.
2. The **Service SAS** delegates access to a resource in **just one** of the storage services: Blob, Queue, Table, or File service.

   Note that stored access policies are currently not supported for an account-level SAS.

Lab7 :Creating an Account SAS (for many operations)

1. Azure Portal → Storage Accounts → Select Account
2. Settings → Shared access signature
3. Provide the options as required → Generate SAS
4. Copy the SAS token and share it with the client.

https://dssdemostorage.blob.core.windows.net/container1/Azure%20Storage%20Service.pdf?sv=2017-11-09&ss=b&srt=co&sp=r&se=2018-08-13T10:05:57Z&st=2018-08-11T02:05:57Z&spr=https&sig=hFYQeZ2fvj52%2BQI0kg%2BbPmErr8J%2F5hKrGkAnF7Q7u%2F4%3D

Stored Access Policies

A Shared Access Signature can take one of two forms:

- **An ad hoc SAS**. When you create an ad hoc SAS, the start time, expiration time, and permissions for the SAS are all specified on the SAS URI (or implied in the case where the start time is omitted). This type of SAS can be created on a container, blob, table, or queue.
- **An SAS with a Stored Access Policy**. A stored access policy is defined on a resource container—a blob container, table, or queue—and can be used to manage constraints for one or more Shared Access Signatures. When you associate an SAS with a stored access policy, the SAS inherits the constraints—the start time, expiration time, and permissions—defined for the stored access policy.

Note: Stored access policies give you the option to revoke permissions without having to regenerate the storage account keys. Set the expiration on these to be a very long time (or infinite), and make sure that it is regularly updated to move it further into the future.

Lab8 :Create Ad-hoc Service SAS using Portal

1. Azure Portal → Storage Accounts → Select Account
2. BLOB → Container → <Select Blob Item> → **Click on Generate SAS Tab**
3. Provide the details →
4. Copy the Token and use with Blob URL.

Lab 9: Create Stored Access Policy using Portal

1. Azure Portal → Storage Accounts → Select Account
2. BLOB → Container → Select the container → Access Policy → + Add Policy

Lab 10:Create SAS using Stored Access Policy using Portal

1. Azure Portal → Storage Accounts → Storage Explorer
2. BLOB → Container → Select the container / Blob → Right Click → Get Shared Access Signature

It's in query string format. This is a query string that can be appended to the full URI of the blob or container the SAS URI was created with, and passed to a client.

Example: http://dssdemostorage.blob.core.windows.net/secure/reports1.xlsx***?sv=2014-02-14&sr=b&sig=8qLv51D3ahgw9Zoyf9vhVfSvOuVdak%2Fdh1gIHmb6plI%3D&st=2014-11-29T12%3A17%3A50Z&se=2014-11-29T16%3A17%3A50Z&sp=rwd***

Note: SAS Token without policy cannot be revoked.

Summary

SAS Token

Account SAS

Always Ad-hoc

Same SAS token can be used for all services.

Service SAS

Ad-hoc / Policy based

Specific to only one service at a time.

Container SAS token works for all blobs in that container and can’t be generated at portal

Blob SAS token is only for the blob for which it is generated

Policy based SAS token can’t be generated at poral

Firewalls and Virtual Network Integration(Setttings→ Firewalls and Virtual Network)

- By default, the storage account endpoints are accessible to any client having required keys.
- You can configure network rules and, only applications requesting data over the specified set of networks can access a storage account.
- You can limit access to limited range of IP addresses
- You can limit the access from specific subnets in virtual networks within same and paired region in azure storage.
- You can use private endpoints for your Azure Storage accounts to allow clients on a virtual network (VNet) to securely access data over a Private Link. The private endpoint uses an IP address from the VNet address space for your storage account service. The connection between the private endpoint and the storage service uses a secure private link.

Service Endpoint and Private EndPoint

Azure offers two similar but distinct services to allow virtual network (VNet) resources to privately connect to other Azure services. Azure VNet Service Endpoints and Azure Private Endpoints .
Each network security by allowing VNet traffic to communicate with service resources without going over the internet

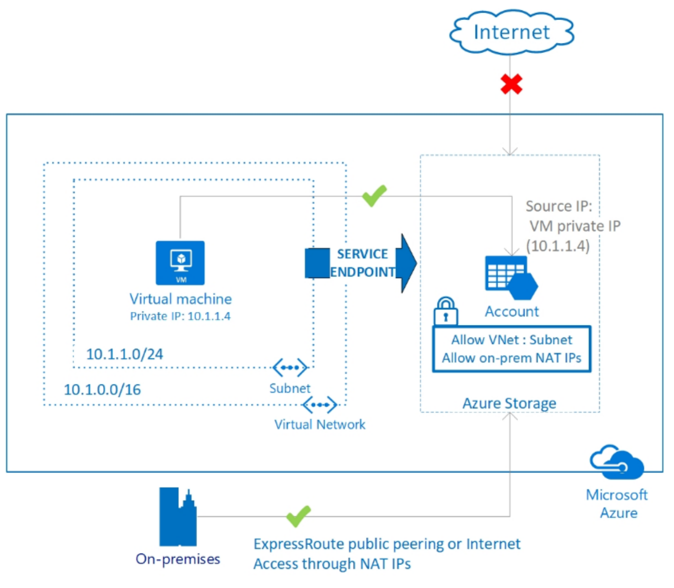
Service Endpoint:

- A service endpoint allows VNet resources to use private IP addresses to connect to an Azure service's public endpoint, meaning traffic flows to the service resource over the Azure backbone network -- instead of over the internet.
- Service endpoints enable enable you to limit network access to Azure service resources via virtual network subnets and IP addresses you specify. Thus providing an extra layer of security allowing you to secure your critical Azure service resources to your virtual networks.

- You need to enable this on your subnet. It can be enabled for one or multiple services (Storage, Azure SQL etc).
- When it is enabled, Identity of virtual network will be available to services (Storage, Azure SQL etc).
- Service Endpoint provides security by specifying firewall at service level so that it only accepts traffic from the subnet associated with the service endpoint.

Set up service End Point

1. Enable service endpoint in subnet
2. Configure firewall rule at service level for that subnet in given virtual machine

 Internet Source IP: VM private IP (10.1.1.4) SERVICE ENDPOINT Virtual machine Private IP: 10.1.1.4 Account Allow VNet : Subnet Allow on-prem NAT IPs 10.1.1.0/24 10.1.0.0/16 Subnet Virtual Network Azure Storage Microsoft Azure On-premises ExpressRoute public peering or Internet Access through NAT IPs	The virtual machine has the private IP address 10.1.1.4 but can access the storage account over a service endpoint. On-premises traffic cannot use service endpoints, and must go over the internet to access the storage account

Lab 11: Restrict access to Storage Account BLOB only from a VM in a given Subnet.

1. Create a virtual network
   a. One subnet with **disabled Service EndPoint**
   b. Another subnet with **enabled service endpoint**. Eg: Storage Service
2. Create a Storage Account
   a. Create a Container (Name = demo, Access Level = Blob)
   b. Upload a file to the container.
   c. Copy the URL of the uploaded file (eg: https://dsdemostorage.blob.core.windows.net/demo/Demo.txt)
3. Note that the URL is currently accessible from everywhere (from VM inside vNET and from my local machine outside vNET)
4. Restrict access to Azure PaaS service (Storage Account) only from **second** subnet.

   a. **Storage Account → Networking → Firewalls and virtual networks Tab**
   b. Select Selected networks
   c. Select VNet and Second Subnet
   d. Save.

5. Confirm access is denied to a resource from another subnet of vNET and also from the internet (my machine)

   Your Local Machine → Browser → URL of BLOB uploaded to Storage Account

   a. Note that you **are not able** to view the blob/file content

6. Confirm access to a resource from an allowed subnet.

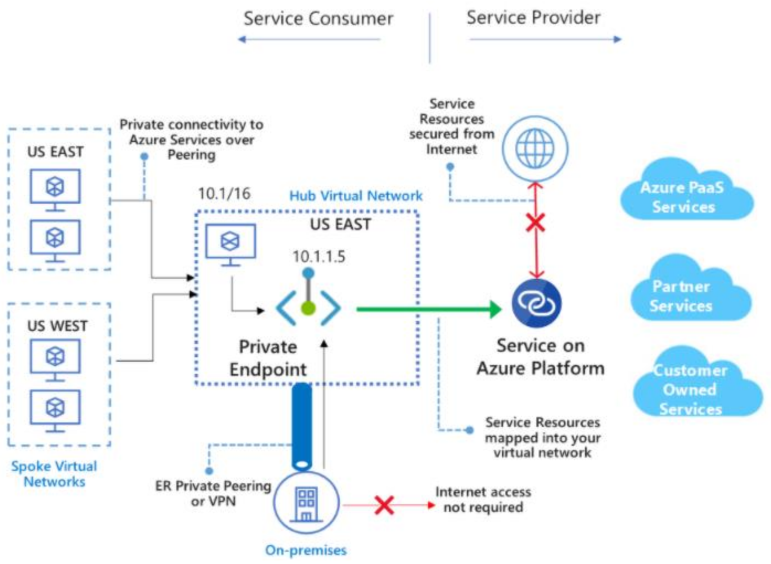
   VM created in second subnet → Browser → URL of BLOB uploaded to Storage Account

   a. Note that you are **able** to view the blob/file content.

Note: You can add your machine IP Address under filewall and access the Storage Account from your local machine.

Azure Private Link:

- Azure Private Link enables you to access Azure PaaS Services (for example, Azure Storage and SQL Database) and Azure hosted customer-owned/partner services over a private endpoint in your virtual network.
- This service that allows virtual network resources to privately connect to other resources as if they were part of the same network, carrying traffic across the Microsoft Azure backbone instead of the internet.
- Traffic between your virtual network and the service travels the Microsoft backbone network. Exposing your service to the public internet is no longer necessary. You can create your own private link service in your virtual network and deliver it to your customers.


Service Consumer
Service Provider
US EAST
Private connectivity to Azure Services over Peering
Service Resources secured from Internet
10.1/16
Hub Virtual Network
US EAST
10.1.1.5
Azure PaaS Services
Partner Services
Customer Owned Services
US WEST
Private Endpoint
Service on Azure Platform
Service Resources mapped into your virtual network
Spoke Virtual Networks
ER Private Peering or VPN
Internet access not required
On-premises

Private End Point:

- Private End Point is a network interface that provides a private IP address to a service that would normally only be accessible to a VNet via public IP address.
- Service resource (Storage etc) is extended in subnet in virtual network. The Resource is called Private Link Resource.
- NIC is created for that service and linked to service.
- Pass Service(Storage Account) also get private IP Address through NIC.
- Private endpoints enable traffic from on-premises to access private link resources without public peering or traversing the internet.
- So from On-Prem you can connect to Private link resource (Ex: Storage account) privately through Express Route OR site to site VPN

Lab12: Ensure that Azure PaaS service uses Private IP, if requested from vNET and public IP, if requested from outside the vNET.

1. Create a Virtual Machine to test the private endpoint is working from it.
2. Create a Storage Account, Add Public Container and upoad a file.
3. Create a **Private Endpoint**: **Search Private Link → Create private endpoint → Enter Basic details**
    1. **Resource type = Microsoft.Storage/storageAccounts**
    2. **Resource = <Storage Account>**
    3. **Target sub-resource = blob**
    4. Select Virtual Network and Subnet in which it Private IP will be created.
    5. Integrate with Private DNS zone = Yes and Either create or select DNS zone
    6. Review + Create
    7. This creates a Private DNS (<storageaccount>.blob.core.windows.net) name and NIC with Private IP addres (10.0.1.4).
4. Test connectivity to private endpoint from **outside** the VNet (your local machine)
    1. nslookup <storageaccount>.blob.core.windows.net
    2. Note that the IP address listed is **public IP**
    3. Try Access to Blob storage file from local machine and note that is accessible*.
5. *To restrict public access to storage account
    1. Go to Storage Account → Firewalls and virtual networks → Select **Selected Networks** and don't select any network.
    2. Try Access to Blob storage file from local machine and note that it is not accessible*.
6. Test connectivity to private endpoint from **inside** the VNet
    1. RDP to VM

2. **nslookup** <storageaccount>.blob.core.windows.net
3. Note that the IP address listed is **private IP (It's the IP Address of the NIC created for Private endpoint)**
4. Try Acceess to Blob storage file from VM and **it succeeds to connect**.

Azure Table Storage

Category(PKCategoryId, CategoryName, ...) - SQL Table

1, Furniture

2, Pets

3, Electronics

4, Plants

Product(PKProduct, FKCategoryId, ProductName, Price, Quantity) - SQL Table

1, 2, Dog01, 100, 3

2, 2, Cat01, 150, 2

3, 1, Chair01, 100, 100

4, 1, Table01, 50, 20

ProductAttributes (NoSQL Table) – NO FIXED SCHEMA

PartitionKey: 2 (Pets), RowKey:1 (Dog01), Age:3, Breed:Pomerian, Color:White

PartitionKey: 2 (Pets), RowKey:2 (Cat01), Age:1, Breed:Indian, Color:Black

PartitionKey: 1 (Furniture), RowKey:3 (Chair01), Weight: 10KG, Color: Brown, Type: Chair

PartitionKey: 1 (Furniture), RowKey:4 (Table01), Weight: 30KG, Color: Black, Type: Table

- The Azure Table storage service is not traditional table but Key/attribute(value) store.
- The Azure Table storage service stores large amounts of **partially structured** data offering high availability and massively scalable storage.
- The service is a **NoSQL datastore** which accepts **authenticated calls** from inside and outside the Azure cloud.

Common uses of the Table service include:

- You can use Table storage to store flexible datasets, such as ,Product catalogues for eCommerce applications ,user data for web applications, address books, device information, and any other type of metadata that your service requires.

- Storing datasets that **don't** require complex joins, foreign keys, or stored procedures and can be de-normalized for fast access.
- Quickly querying data using a clustered index (Combination of PartitionKey and RowKey).

You can use the Table service to store and query huge sets of structured, non-relational data, and your tables will scale as demand increases.

- **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. The number of tables that a storage account can contain is limited only by the storage account capacity limit.
- **Entity**: An entity is a set of properties, similar to a database row. An entity can be up to **1MB in size**. In an Azure Table Storage table, items are referred to as *rows*, and fields are known as *columns.*
- **Properties**: A property is a name-value pair. Each entity can include up to **252 custom properties** to store up to **1 MB** of data. Each entity also has **3 system** properties that specify a **partition key** (string upto 1KB in size)**, a row key** (string upto 1KB in size)**, 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.
- The partition key and row key effectively define a clustered index over the data.Items in the same partition are stored in row key order.
- Azure Table Storage tables have no concept of relationships, stored procedures, secondary indexes, or foreign keys. Data will usually be denormalized, with each row holding the entire data for a logical entity.

The URI for a specific table access is structured as follows:

http://<account>.**table**.core.windows.net/<TableName>

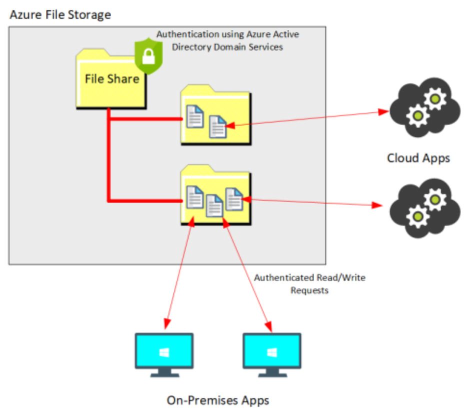
Lab13:Create table and Insert data.

Using Visual Studio Server Explorer

1. Server Explorer → … →Select Windows Azure Storage → Select and Expand Storage Account.
2. Expand to Tables → Create Table…, Enter Name of Table
3. Select Table → Right click → View Table
4. Use the Editor to manage Table.

Azure File Storage

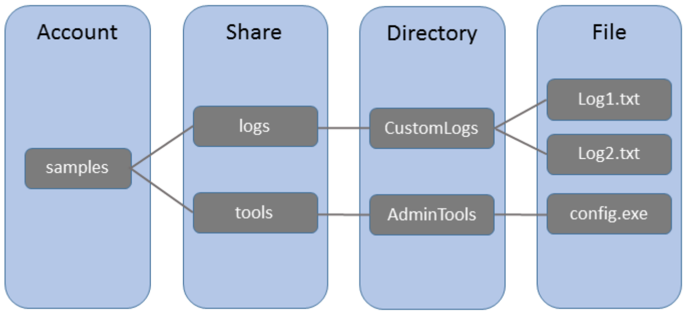
- Azure File storage is a service that offers file shares in the cloud using the standard Server Message Block (SMB) Protocol.
- With Azure File storage, you can migrate **legacy applications** that rely on file shares to Azure quickly and without costly rewrites.
- Microsoft Azure virtual machines can share file data across application components via mounted shares, and on-premises applications can access file data in a share via the File storage API.
- You can control access to shares in Azure File Storage using authentication and authorization services available through Azure Active Directory Domain Services.
- It enables you to share up to 100 TB of data in a single storage account Which can be distributed across any number of file shares in the account. Currently, Azure File Storage supports up to 2000 concurrent connections per shared file.


Azure File Storage
Authentication using Azure Active Directory Domain Services
File Share
Cloud Apps
Authenticated Read/Write Requests
On-Premises Apps

Common uses of File storage include:

- Replace or supplement on-premises file servers.
- "Lift and shift" of Legacy applications, that rely on file shares to run on Azure virtual machines or cloud services, without expensive rewrites.
- Simplify cloud development.
  - Shared application settings,for example in configuration files.
  - Diagnostic share such as logs, metrics, and crash dumps in a shared location
  - Dev/Test/Debug
- Storing tools and utilities needed for developing or administering Azure virtual machines or cloud services

File storage concepts:



- **Storage Account:** All access to Azure Storage is done through a storage account.
- **Share:** A File storage share is an SMB file share in Azure. All directories and files must be created in a parent share. An account can contain an unlimited number of shares, and a share can store an unlimited number of files, up to the 5 TB total capacity of the file share.
- **Directory:** An optional hierarchy of directories.
- **File:** A file in the share. A file may be up to 1 TB in size but you can set quotas to limit the size of each share below this figure.
- **Max size of a File Share = 5TB**

URL format: https**://<storage account>**.file.core.windows.net/**<share>**/**<directory/directory>**/**<file>**

The following example URL could be used to address one of the files in the diagram above:

http://samples.file.core.windows.net/logs/CustomLogs/Log1.txt

Lab14: Create File Share and Map file share in windows

Managing Using Azure Portal:

1. Azure Portal → **Storage accounts** → Create and Choose the Storage Account

   Note: File storage is replicated only via LRS or GRS right now…
2. Choose "Files" service → Click "+ File share" → New file share = Images, Quota=100GB
3. Optionally add directory and in the directory and upload the file[s].

Mapping the file share in Windows

4. Go to File Share Blade and click on **Connect** → Copy the net command edit the values
5. Open Command Prompt in Administrator Mode → Execute the net command
6. You can manage the File Share using the local drive.

Note: Ensure port 445 is open: Azure Files uses SMB protocol. SMB communicates over TCP port 445

Azcopy:

You can also use the *AzCopy copy* command to transfer files and folders to and from Azure File Storage to your local computer and between storage accounts.

Lab15: Download the files from a folder named *myfolder* in a file share named *myshare* to a local folder called *localfolder*,

```
azcopy copy "https://<storage-account-name>.file.core.windows.net/myshare/ myfolder<SAS-token>"
"localfolder" --recursive
```