**1. Create an Azure Storage Account & Explore Options**

**Steps:**

1. **Login to Azure Portal**: <https://portal.azure.com>
2. Navigate to **Storage accounts** > Click **+ Create**
3. **Basics Tab:**
   * Subscription: Choose your subscription
   * Resource Group: Create or choose existing
   * Storage Account Name: Unique name, e.g., adityastorage123
   * Region: Choose nearest or required region
   * Performance: Standard or Premium (Premium uses SSD, faster)
   * Account kind: Select **StorageV2 (general purpose v2)** (most flexible)
   * Replication: Choose from
     + LRS (Locally Redundant Storage)
     + GRS (Geo-Redundant Storage)
     + RA-GRS (Read-Access Geo-Redundant Storage)
     + ZRS (Zone-Redundant Storage)
   * Access tier (default): Hot or Cool (affects cost & access speed)
4. **Advanced Tab:**
   * Security options: Enable secure transfer required (recommended)
   * Blob soft delete, versioning, large file shares, etc. can be enabled here
5. **Networking Tab:**
   * Public endpoint or private endpoint (VNet integration)
   * Configure firewall rules if needed
6. **Data Protection Tab:**
   * Enable soft delete for blobs and files (helps to recover deleted data)
7. **Tags Tab:**
   * Add any tags if needed for billing or management
8. **Review + Create** and then **Create**

**2. Upload and Access Blob**

1. Go to your storage account in portal > **Containers**
2. Create a new container (public/private access)
3. Open container > Upload > select files or create blobs
4. After upload, click blob to get URL, test access from browser if container is public
5. Try downloading and viewing the blob content

**3. Authentication Techniques & Test**

Azure Blob Storage supports:

* **Shared Key Authorization** (using Access keys)
* **Shared Access Signature (SAS) tokens**
* **Azure AD Authentication**

**Test these:**

* Use **Access keys** (from storage account > Access keys)
* Generate **SAS tokens** and access blobs with limited permissions & expiry
* Use **Azure AD OAuth2** authentication (requires role assignments, e.g., Storage Blob Data Reader/Contributor roles)

**4. Use Azure Storage Explorer**

* Download & install [Azure Storage Explorer](https://azure.microsoft.com/en-us/features/storage-explorer/)
* Connect to your storage account using:
  + Account name & key (access keys)
  + SAS token
  + Azure AD login
* Explore containers, blobs, file shares, upload/download blobs interactively

**5. Provision Access Keys & Use for Connection**

* Go to your Storage Account > **Access keys**
* Copy **key1** or **key2**
* Use these in your code or tools to connect:

**Example connection string:**

DefaultEndpointsProtocol=https;AccountName=adityastorage123;AccountKey=<your\_key\_here>;EndpointSuffix=core.windows.net

**6. Create Shared Access Signature (SAS) & Check Scope**

1. Go to Storage Account > **Shared access signature**
2. Set permissions (Read, Write, Delete, List, Add, Create)
3. Set start & expiry time
4. Restrict IPs or protocols (HTTP/HTTPS)
5. Generate SAS token & URL
6. Test access by using URL with SAS in browser or tools (e.g., Azure Storage Explorer with SAS)

**7. Create Stored Access Policy on Container & Use with SAS**

* Go to Container > **Access Policy** tab
* Create a policy with expiry, permissions
* Use this policy to generate SAS token with limited control over SAS lifetime
* Test SAS with policy to confirm correct permissions

**8. Access Tiers and Use Cases**

Azure Blob Storage access tiers:

* **Hot:** Frequently accessed data, higher storage cost, lower access cost
* **Cool:** Infrequently accessed data, lower storage cost, higher access cost
* **Archive:** Rarely accessed, long-term retention, very low storage cost, data needs to be rehydrated before use

**Use cases:**

* Hot for active app data
* Cool for backup and disaster recovery data
* Archive for compliance and long-term backups

**9. Apply Lifecycle Management Policy**

* Go to Storage Account > **Lifecycle management**
* Create rule:
  + Define filter for blobs (prefix, blob types)
  + Actions: Move blobs between tiers (Hot -> Cool -> Archive), or delete after certain days
* Save and test by uploading blobs and waiting for lifecycle action or force trigger via API

**10. Test Object Replication in Blob**

* Object replication is supported on RA-GRS and GZRS accounts
* Enable replication rules between source and destination storage accounts
* Upload blob in source, verify replicated blob in destination
* Test data consistency and failover

**11. Create File Share and Test Functionality**

* Go to Storage Account > **File shares**
* Create a new file share with quota
* Mount file share on local machine (Windows, Linux)
* Upload/download files
* Test SMB protocol access

**12. Setup Azure File Sync**

* Create **Azure File Sync** resource in Azure portal
* Register your Windows Server (on-prem or VM) with Storage Sync Service
* Create a sync group, add your Azure file share and server endpoint
* Sync files between on-prem and Azure cloud
* Test file sync, conflict resolution, and tiering

**1. Create Storage Account (Azure CLI)**

# Variables

RESOURCE\_GROUP="myResourceGroup"

LOCATION="eastus"

STORAGE\_ACCOUNT\_NAME="adityastorage$(date +%s)" # unique name

# Create resource group

az group create --name $RESOURCE\_GROUP --location $LOCATION

# Create storage account with standard performance, general-purpose v2

az storage account create \

--name $STORAGE\_ACCOUNT\_NAME \

--resource-group $RESOURCE\_GROUP \

--location $LOCATION \

--sku Standard\_LRS \

--kind StorageV2 \

--access-tier Hot

**2. Create Blob Container and Upload Blob (Azure CLI)**

# Create container

CONTAINER\_NAME="mycontainer"

az storage container create --name $CONTAINER\_NAME --account-name $STORAGE\_ACCOUNT\_NAME

# Upload file (assume local file 'sample.txt')

az storage blob upload \

--container-name $CONTAINER\_NAME \

--name sample.txt \

--file ./sample.txt \

--account-name $STORAGE\_ACCOUNT\_NAME

**3. Get Storage Account Keys (Azure CLI)**

# List storage account keys

az storage account keys list --resource-group $RESOURCE\_GROUP --account-name $STORAGE\_ACCOUNT\_NAME

# Save key to variable

STORAGE\_KEY=$(az storage account keys list --resource-group $RESOURCE\_GROUP --account-name $STORAGE\_ACCOUNT\_NAME --query '[0].value' -o tsv)

**4. Access Blob Using Shared Key (Azure CLI Example)**

# Download blob using storage key

az storage blob download \

--container-name $CONTAINER\_NAME \

--name sample.txt \

--file downloaded\_sample.txt \

--account-name $STORAGE\_ACCOUNT\_NAME \

--account-key $STORAGE\_KEY

**5. Create SAS Token (Azure CLI)**

# Create SAS token with read, write, list permissions for 1 day

EXPIRY=$(date -u -d "1 day" '+%Y-%m-%dT%H:%MZ')

sas\_token=$(az storage account generate-sas \

--account-name $STORAGE\_ACCOUNT\_NAME \

--permissions rwl \

--resource-types sco \

--services b \

--expiry $EXPIRY \

--https-only \

-o tsv)

echo "SAS Token: $sas\_token"

# Full URL example

echo "Blob SAS URL: https://${STORAGE\_ACCOUNT\_NAME}.blob.core.windows.net/${CONTAINER\_NAME}/sample.txt?$sas\_token"

**6. Using Stored Access Policy (Azure CLI)**

# Create stored access policy on container with read/write permissions expiring in 7 days

EXPIRY\_POLICY=$(date -u -d "7 days" '+%Y-%m-%dT%H:%MZ')

az storage container policy create \

--account-name $STORAGE\_ACCOUNT\_NAME \

--container-name $CONTAINER\_NAME \

--name "mypolicy" \

--permissions "rw" \

--expiry $EXPIRY\_POLICY

Now generate SAS using this policy:

sas\_token\_policy=$(az storage container generate-sas \

--account-name $STORAGE\_ACCOUNT\_NAME \

--name $CONTAINER\_NAME \

--policy-name "mypolicy" \

--https-only \

-o tsv)

echo "SAS with policy: $sas\_token\_policy"

**7. Change Access Tier (Azure CLI)**

# Change access tier of blob to Cool

az storage blob set-tier \

--account-name $STORAGE\_ACCOUNT\_NAME \

--container-name $CONTAINER\_NAME \

--name sample.txt \

--tier Cool

**8. Apply Lifecycle Management Policy (JSON + CLI)**

Create a JSON file lifecycle-policy.json:

{

"rules": [

{

"enabled": true,

"name": "moveToCoolAfter30Days",

"type": "Lifecycle",

"definition": {

"filters": {

"blobTypes": ["blockBlob"],

"prefixMatch": ["sample"]

},

"actions": {

"baseBlob": {

"tierToCool": {

"daysAfterModificationGreaterThan": 30

}

}

}

}

}

]

}

Apply with CLI:

az storage account management-policy create \

--account-name $STORAGE\_ACCOUNT\_NAME \

--resource-group $RESOURCE\_GROUP \

--policy @lifecycle-policy.json

**9. Create File Share and Upload (Azure CLI)**

# Create file share with 100GB quota

az storage share create --name myfileshare --quota 100 --account-name $STORAGE\_ACCOUNT\_NAME

# Upload file to file share

az storage file upload \

--share-name myfileshare \

--source ./sample.txt \

--path sample.txt \

--account-name $STORAGE\_ACCOUNT\_NAME

**10. Set up Azure File Sync (High-Level)**

Azure File Sync setup is more involved; here are the steps:

* Create **Storage Sync Service** in Azure portal.
* Register your Windows Server with Storage Sync Service (using the Azure File Sync agent).
* Create a Sync Group and add your Azure file share and Server endpoint.
* Sync files automatically.

No direct CLI commands but you can automate some steps with PowerShell and Azure CLI for registration:

# Install Azure File Sync agent on your Windows Server (manual download/install)

# Register the server with Storage Sync Service (PowerShell)

$StorageSyncServiceName = "<your-storage-sync-service-name>"

$ResourceGroup = "<your-resource-group>"

Register-AzStorageSyncServer -ResourceGroupName $ResourceGroup -StorageSyncServiceName $StorageSync