

Task-3

(Data Backup and Restore with Azure Blob Storage)

Aim: Implement a basic data backup and restore system using Azure Blob Storage. Write a Python or Node.js script to upload files from your local machine to Azure Blob Storage. Then, create a mechanism to download and restore these files back to your local machine. This project will give you hands-on experience with cloud-based storage services and data management.

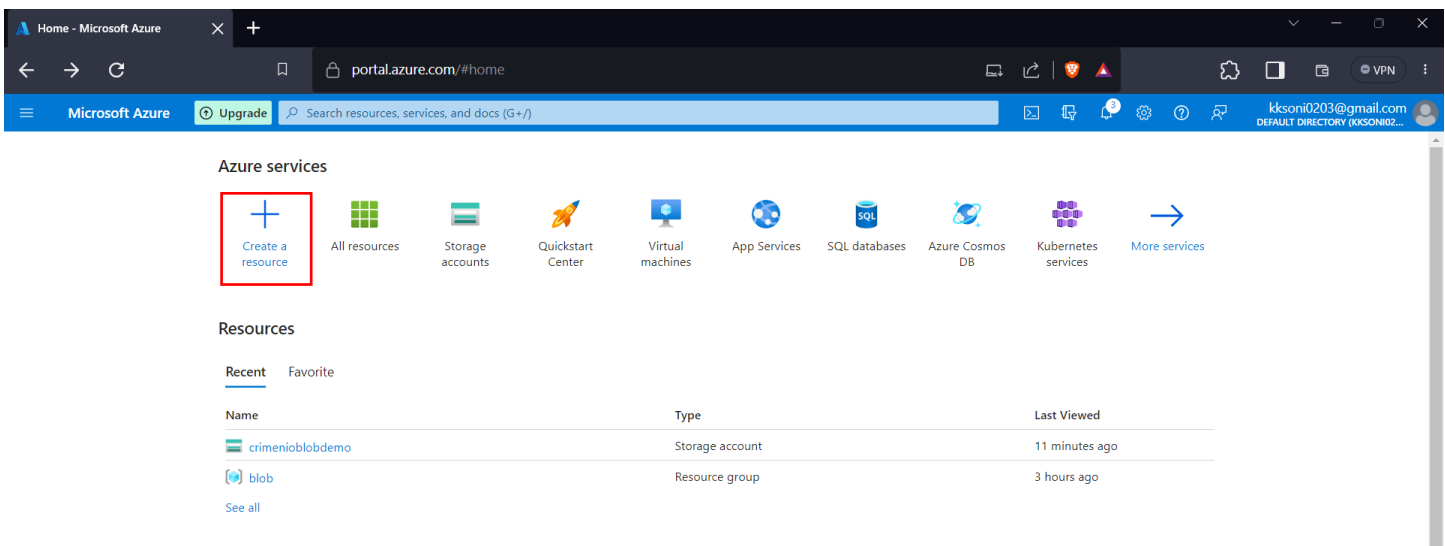
Step 1: Sign in to Azure Portal

- 1) Open a web browser and go to the Azure Portal: <https://portal.azure.com/>
- 2) Sign in with your Azure account.

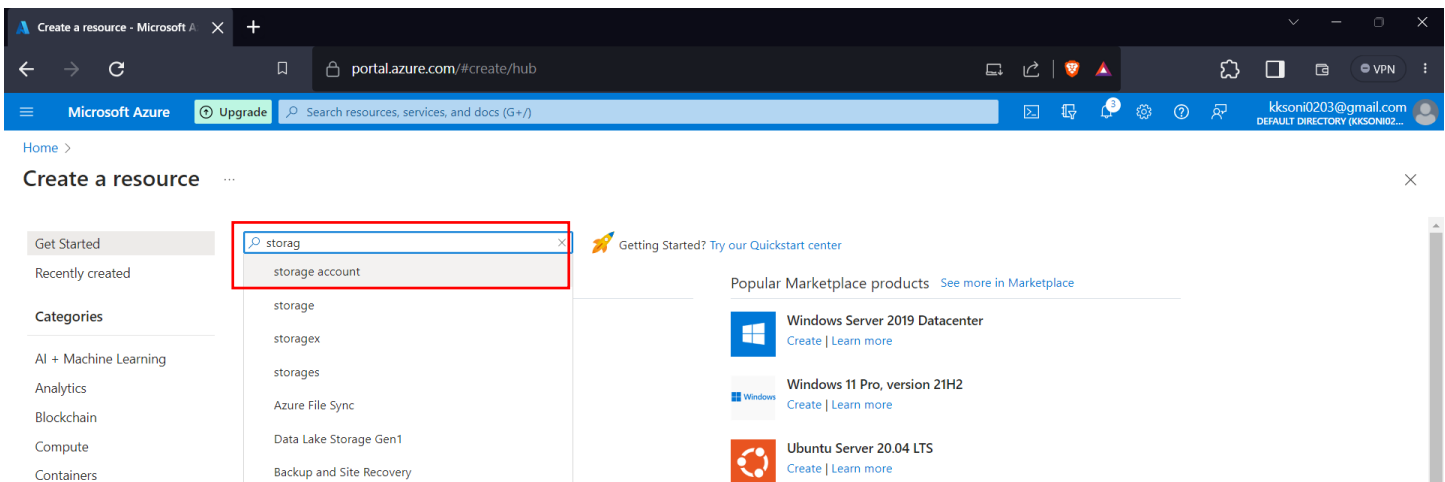
Step 2: Create a Resource Group

A resource group is a logical container for resources deployed in Azure.

- 1) In the Azure Portal, click on the "Create a resource" button (+) on the left side.



- 2) Search for "Storage account" and select "Storage account - blob, file, table, queue."



3) Select "Storage account" and click the "Create" button to create a new resource group.

The screenshot shows the Azure Marketplace search results for 'storage account'. The search bar at the top contains 'storage account'. Below the search bar, there are filters for Pricing, Operating System, Publisher Type, Product Type, and Publisher name, all set to 'All'. A checkbox for 'Azure services only' is present. The results show 1 to 20 of 172 results. The first result, 'Storage account' by Microsoft, is highlighted with a red box. It is an Azure Service that uses Blobs, Tables, Queues, Files, and Data Lake Gen 2 for reliable, economical cloud storage. The 'Create' button is visible at the bottom of the card.

The screenshot shows the details page for the 'Storage account' service in the Azure Marketplace. The page title is 'Storage account' and it is by Microsoft. It has a 4.2 rating from 1827 ratings. The plan is 'Storage account' and the 'Create' button is highlighted with a red box.

4) Provide a unique name for the resource group and choose a subscription.

The screenshot shows the 'Create a storage account' wizard in the Azure portal. The 'Basics' tab is selected. The 'Project details' section is visible, showing the 'Subscription' dropdown set to 'Free Trial' and the 'Resource group' dropdown set to '(New) blob-storage'. The 'Create new' link is visible below the resource group dropdown.

5) Provide storage account name, select a region, select standard as performance and select LRS.

Instance details

Storage account name ⓘ *

Region ⓘ *

Performance ⓘ * ☒ Standard: Recommended for most scenarios (general-purpose v2 account)

Redundancy ⓘ *

[Deploy to an edge zone](#)

[Review](#) [< Previous](#) [Next : Advanced >](#) [Give feedback](#)

6) Review the settings and then click “Create”.

Create a storage account - Mic...

portal.azure.com/#create/Microsoft.StorageAccount-ARM

Microsoft Azure Upgrade Search resources, services, and docs (G+)

Home > Create a resource > Marketplace > Storage account >

Create a storage account

Running final validation...

Basics Advanced Networking Data protection Encryption Tags Review

Blob retainment period in days	7
Container soft delete	Enabled
Container retainment period in days	7
File share soft delete	Enabled
File share retainment period in days	7
Versioning	Disabled
Blob change feed	Disabled
Version-level immutability support	Disabled

Encryption

Encryption type	Microsoft-managed keys (MMK)
Enable support for customer-managed keys	All service types (blobs, files, tables, and queues)
Enable infrastructure encryption	Disabled

[Create](#) [< Previous](#) [Next >](#) [Download a template for automation](#) [Give feedback](#)

Step 3: Create a Container within the Storage Account

- 1) Once your storage account is created, go to the resource group containing your storage account.
- 2) In the storage account overview page, navigate to the "Containers" section in the left menu.

The screenshot shows the Microsoft Azure portal interface for a storage account named 'crimenio'. The left-hand navigation pane is visible, with 'Containers' highlighted under the 'Data storage' section. The main content area displays the 'Essentials' tab, showing account details such as Resource group (blob-storage), Location (East US), Subscription (Free Trial), and Subscription ID (f6973940-bfe6-4afb-803d-ad10fb39c04b). The 'Properties' tab is active, showing 'Blob service' settings: Hierarchical namespace (Disabled), Default access tier (Hot), and Blob public access (Enabled). The 'Security' section shows 'Require secure transfer for REST API operations' (Enabled) and 'Storage account key access' (Enabled).

- 3) Click the "+ Container" button to create a new container and choose a public access level for the container. For private access, select "Private" (recommended). Click the "Create" button.

The screenshot shows the 'New container' dialog box in the Microsoft Azure portal. The dialog box is open, and the 'Name' field is highlighted with a red rectangle. The 'Public access level' dropdown menu is also highlighted with a red rectangle, showing 'Private (no anonymous access)' selected. The 'Create' button is visible at the bottom of the dialog box. The background shows the 'Containers' section of the storage account, with a table listing existing containers. The '+ Container' button in the left navigation pane is also highlighted with a red rectangle.

Name	Last modified	Public access level
<input type="checkbox"/> \$logs	8/21/2023, 1:04:00 PM	Private

Step 2: Install Required Libraries

Ensure you have the azure-storage-blob package installed. You can install it using the following command:

```
# pip install azure-storage-blob
```

Step 3: Write the Upload Script

Create a Python script, for example, upload.py, and write the following code:

```
from azure.storage.blob import BlobServiceClient
import os
# import data

storage_account_key = "t0cwQcQoth/eLzOnTNuxRdXzMRLSR+wVv2oDk3SXn6wMl6L/hRJ/3ooq6o33EOHwrH4sdPqtaZjt+ASt4tUwlg=="
storage_account_name = "crimenioblobdemo"
connection_string = "DefaultEndpointsProtocol=https;AccountName=crimenioblobdemo;AccountKey=t0cwQcQoth/eLzOnTNuxRdXzMRLSR+wVv2oDk3SXn6wMl6L/hRJ/3ooq6o33EOHwrH4sdPqtaZjt+ASt4tUwlg==;EndpointSuffix=core.windows.net"
container_name = "backup"

def uploadToBlobStorage(file_path, file_name):
    blob_service_client = BlobServiceClient.from_connection_string(connection_string)
    blob_client = blob_service_client.get_blob_client(container=container_name, blob=file_name)

    with open(file_path, "rb") as data:
        blob_client.upload_blob(data)
        print("Upload "+file_name+" file")

uploadToBlobStorage('C:\\Users\\Krishnakant\\Desktop\\Task\\Task.jpg', 'Task')
```

Note: Replace "your_connection_string" with the actual connection string you obtained in Step 1. Also, set the local_directory to the path of the directory containing files you want to upload.

Step 4: Write the Download Script

Create another Python script, for example, download.py, and write the following code:

```
from azure.storage.blob import BlobServiceClient
import os

storage_account_key = "t0cwQcQoth/eLzOnTNuxRdXzMRLSR+wVv2oDk3SXn6wMl6L/hRJ/3ooq6o33EOHwrH4sdPqtaZjt+AS4tUwlg=="
storage_account_name = "crimenioblobdemo"
connection_string = "DefaultEndpointsProtocol=https;AccountName=crimenioblobdemo;AccountKey=t0cwQcQoth/eLzOnTNuxRdXzMRLSR+wVv2oDk3SXn6wMl6L/hRJ/3ooq6o33EOHwrH4sdPqtaZjt+AS4tUwlg==;EndpointSuffix=core.windows.net"
container_name = "backup"

download_directory = "C:\\Users\\Krishnakant\\Desktop\\Blob"

def download_files():
    blob_service_client = BlobServiceClient.from_connection_string(connection_string)
    container_client = blob_service_client.get_container_client(container_name)

    # Ensure the download directory exists
    os.makedirs(download_directory, exist_ok=True)

    # Download and save each blob to the local directory
    for blob in container_client.list_blobs():
        blob_client = container_client.get_blob_client(blob)
        blob_data = blob_client.download_blob()
        blob_file_path = os.path.join(download_directory, blob.name)

        with open(blob_file_path, "wb") as blob_file:
            blob_file.write(blob_data.readall())

        print(f"Downloaded: {blob.name}")

if __name__ == "__main__":
    download_files()
    print("Download process completed.")
```

Note: Replace "your_connection_string" with the actual connection string you obtained in Step 1. Also, set the download_directory to the path where you want to save the downloaded files.

Step 5: Run the Scripts

Open a terminal or command prompt and navigate to the directory where your scripts are located. Run the following commands:

For uploading:

```
# python upload.py
```

For downloading:

```
# python download.py
```

Note: These scripts will upload the files from your local directory to Azure Blob Storage and then download and restore them back to the specified local directory.

Remember to adapt the paths and filenames according to your preferences