

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 .

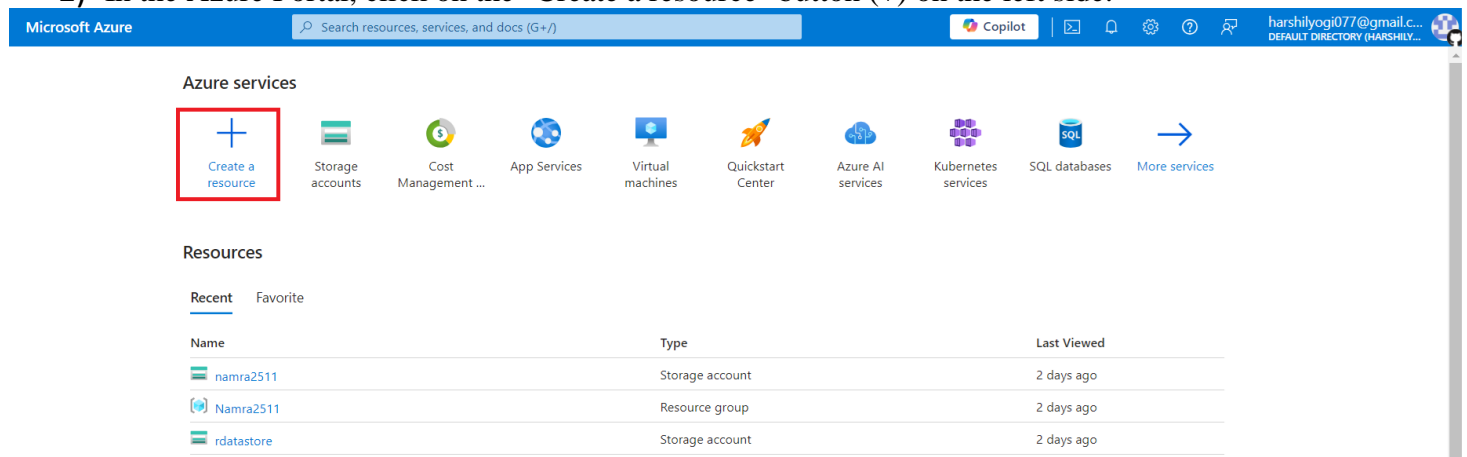
Step 1: Sign in to Azure Portal

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

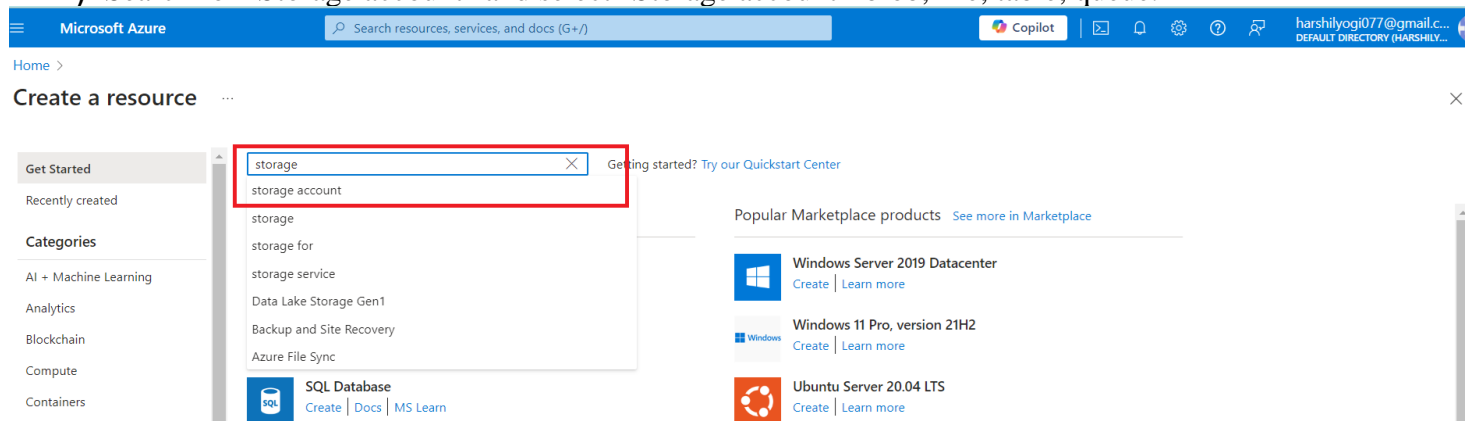
Step 2: Create a Resource Group

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.

Microsoft Azure

Search resources, services, and docs (G+)

Copilot

harshilyogi077@gmail.c...
DEFAULT DIRECTORY (HARSHILY...

Home > Create a resource > Marketplace

Get Started

Service Providers

Management

Private Marketplace

Private Offer Management

My Marketplace

Favorites

My solutions

Recently created

Private plans

Categories

storage account

Pricing: All Operating System: All Publisher Type: All Product Type: All Publisher name: All

☐ Azure services only

Showing 1 to 20 of 196 results for 'storage account'. [Clear search](#)

Tile view

Storage account

Microsoft

Azure Service

Use Blobs, Tables, Queues, Files, and Data Lake Gen 2 for reliable, economical cloud storage.

Create

Storage Account Using ARM Template

FortuneCloud LLC

Azure Application

storage account arm template

Price varies

Create

Storage task - Azure Storage Actions

Microsoft

Azure Service

Perform common operations on millions of objects based on logical conditions using object properties for Blobs and Data Lake Storage Gen2.

Create

Azure Storage Mover

Microsoft

Azure Service

Azure Storage Mover is a migration service that migrates your on-premises file shares to Azure Storage.

Create

Storage Account Using ARM

DIGISTORM LTD.

Azure Application

storage account arm

Price varies

Create

Microsoft Azure

Search resources, services, and docs (G+)

Copilot

harshilyogi077@gmail.c...
DEFAULT DIRECTORY (HARSHILY...

Home > Create a resource > Marketplace > Storage account

Storage account

Microsoft

Storage account [Add to Favorites](#)

Microsoft | Azure Service

★ 4.2 (1880 ratings)

Plan

Storage account

Create

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

Microsoft Azure

Search resources, services, and docs (G+)

Copilot

harshilyogi077@gmail.c...
DEFAULT DIRECTORY (HARSHILY...

Home > Create a resource > Marketplace > Storage account > Create a storage account

Basics Advanced Networking Data protection Encryption Tags Review + create

Azure Storage is a Microsoft-managed service providing cloud storage that is highly available, secure, durable, scalable, and redundant. Azure Storage includes Azure Blobs (objects), Azure Data Lake Storage Gen2, Azure Files, Azure Queues, and Azure Tables. The cost of your storage account depends on the usage and the options you choose below. [Learn more about Azure storage accounts](#)

Project details

Select the subscription in which to create the new storage account. Choose a new or existing resource group to organize and manage your storage account together with other resources.

Subscription * Azure for Students

Resource group * myblob

[Create new](#)

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

Instance details

Storage account name * ⓘ

storedatar

Region * ⓘ

(US) East US

Deploy to an Azure Extended Zone

Performance * ⓘ

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

☐ Premium: Recommended for scenarios that require low latency.

Redundancy * ⓘ

Locally-redundant storage (LRS)

6) Review the setting and then click “Create”

Create a storage account ...

- Basics
- Advanced
- Networking
- Data protection
- Encryption
- Tags
- Review + create

View automation template

Basics

Subscription	Azure for Students
Resource group	myblob
Location	East US
Storage account name	storedatar
Performance	Standard
Replication	Locally-redundant storage (LRS)

Advanced

Enable hierarchical namespace	Disabled
Enable SFTP	Disabled
Enable network file system v3	Disabled
Allow cross-tenant replication	Disabled
Access tier	Hot
Enable large file shares	Enabled

Security

Previous

Next

Create

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 Azure portal interface for a storage account named 'storedatar'. The left-hand navigation pane is expanded to 'Data storage', where the 'Containers' option is highlighted with a red rectangle. The main content area displays the 'Essentials' section with various account details and a 'Properties' tab. The 'Properties' tab shows settings for 'Blob service' and 'Security'.

Property	Value
Resource group (move)	myblob
Location	eastus
Subscription (move)	Azure for Students
Subscription ID	9d01d180-dd22-465e-9e40-3dd2e4de8c0a
Disk state	Available
Tags (edit)	Add tags
Performance	Standard
Replication	Locally-redundant storage (LRS)
Account kind	StorageV2 (general purpose v2)
Provisioning state	Succeeded
Created	6/21/2024, 9:55:46 PM

Property	Value
Hierarchical namespace	Disabled
Default access tier	Hot
Blob anonymous access	Disabled
Blob soft delete	Enabled (7 days)
Container soft delete	Enabled (7 days)

Property	Value
Require secure transfer for REST API operations	Enabled
Storage account key access	Enabled
Minimum TLS version	Version 1.2
Infrastructure encryption	Disabled

- 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 Azure portal interface for a storage account named 'storedatar'. The left-hand navigation pane is expanded to 'Data storage', where the 'Containers' option is highlighted. The main content area displays the 'Containers' section with a table of existing containers. A modal dialog is open on the right, showing the 'Name' field with the value 'uploadfile' and the 'Anonymous access level' dropdown set to 'Private (no anonymous access)'. A blue informational message states: 'The access level is set to private because anonymous access is disabled on this storage account.' The 'Create' button is visible at the bottom of the modal.

Name	Last modified	Anonymous
\$logs	6/21/2024, 9:56:10 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

#change credentials details with your's details

storage_account_key = "GrL5NVUtlIwRntY+NhGnwldXatdGn1xqKGvwhhF0B3Wxf8LSK3Z+gGcvKrmEZf3cHud7qLWKf/p+AstcCaNMA=="
storage_account_name = "storedatar"
connection_string = "DefaultEndpointsProtocol=https;AccountName=storedatar;AccountKey=GrL5NVUtlIwRntY+NhGnwldXatdGn1xqKGvwhhF0B3Wxf8LSK3Z+gGcvKrmEZf3cHud7qLWKf/p+AstcCaNMA=="
container_name = "uploadfile"

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\\Hrashil Yogi\\OneDrive\\Documents\\cloud\\task.png', 'cloud')

```

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 = "GrL5NVUtlIwRntY+NhGnwldXatdGn1xqKGvwhhF0B3Wxf8LSK3Z+gGcvKrmEZf3cHud7qLWKf/p+AstcCaNMA=="
storage_account_name = "storedatar"
connection_string = "DefaultEndpointsProtocol=https;AccountName=storedatar;AccountKey=GrL5NVUtlIwRntY+NhGnwldXatdGn1xqKGvwhhF0B3Wxf8LSK3Z+gGcvKrmEZf3cHud7qLWKf/p+AstcCaNMA=="
container_name = "uploadfile"
download_directory = "C:\\OneDrive\\Documents\\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.")

```

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`