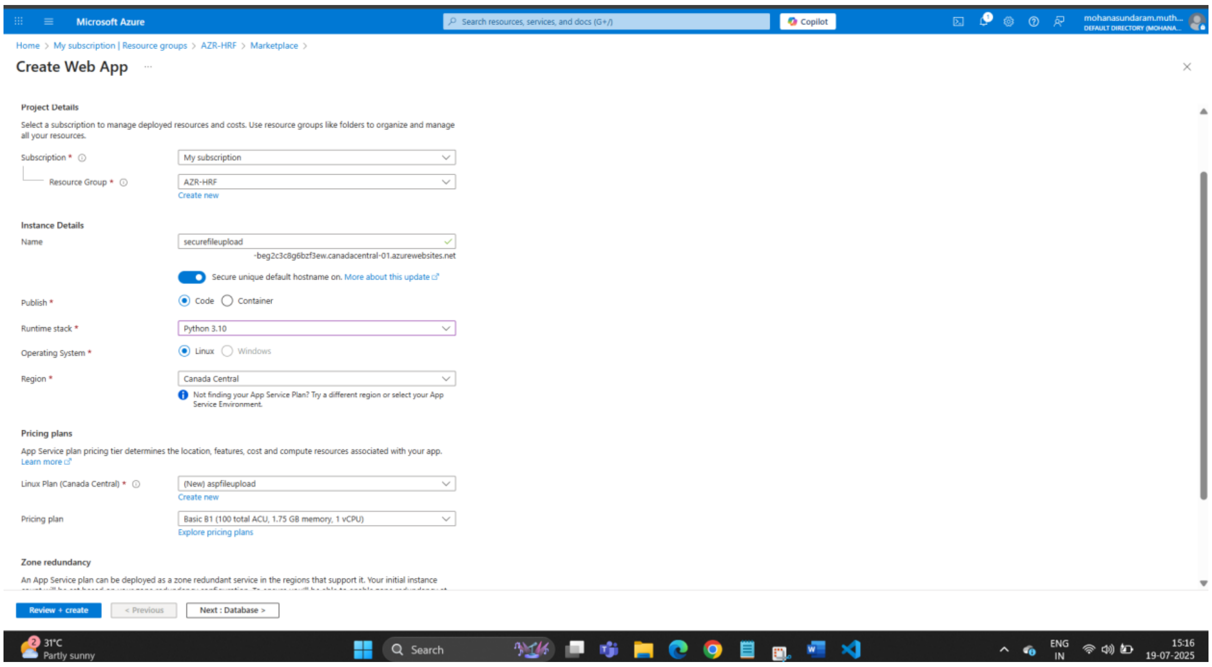
Phase 1: web app and storage

1.Created web app by below configurations:


Microsoft Azure
Search resources, services, and docs (G+/)
Copilot
mohanasundaram.muth...
DEFAULT DIRECTORY (MOHANA...

Home > My subscription | Resource groups > AZR-HRF > Marketplace >

Create Web App

Project Details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *	My subscription
Resource Group *	AZR-HRF (Create new)

Instance Details

Name	securefileupload -beg2c3c8g6bzf3ew.canadacentral-01.azurewebsites.net
	Secure unique default hostname on. More about this update
Publish *	[✓ Code] Container
Runtime stack *	Python 3.10
Operating System *	[✓ Linux] Windows
Region *	Canada Central
	Not finding your App Service Plan? Try a different region or select your App Service Environment.

Pricing plans

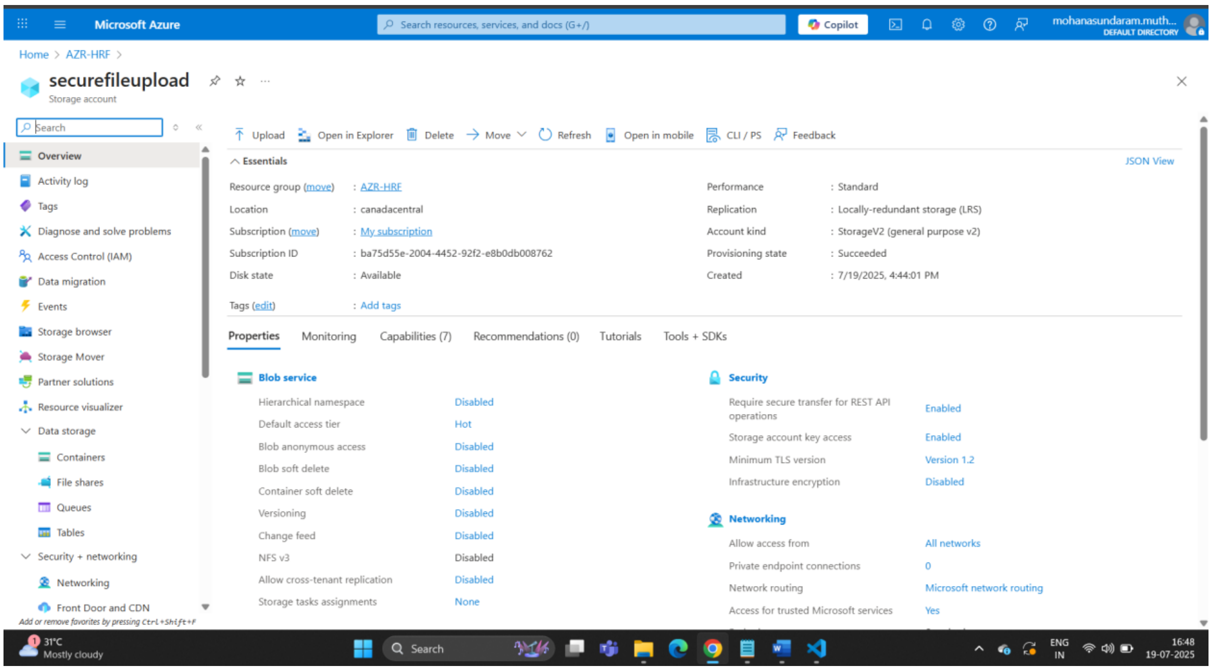
App Service plan pricing tier determines the location, features, cost and compute resources associated with your app. Learn more

Linux Plan (Canada Central) *	(New) aspfileupload (Create new)
Pricing plan	Basic B1 (100 total ACU, 1.75 GB memory, 1 vCPU) (Explore pricing plans)

Zone redundancy

An App Service plan can be deployed as a zone redundant service in the regions that support it. Your initial instance

2.Created LRS storage with public:



securefileupload
Storage account

Essentials

Resource group (move)	: AZR-HRF	Performance	: Standard
Location	: canadacentral	Replication	: Locally-redundant storage (LRS)
Subscription (move)	: My subscription	Account kind	: StorageV2 (general purpose v2)
Subscription ID	: ba75d55e-2004-4452-92f2-e8b0db008762	Provisioning state	: Succeeded
Disk state	: Available	Created	: 7/19/2025, 4:44:01 PM
Tags (edit)	: Add tags		

Blob service

Hierarchical namespace	Disabled
Default access tier	Hot
Blob anonymous access	Disabled
Blob soft delete	Disabled
Container soft delete	Disabled
Versioning	Disabled
Change feed	Disabled
NFS v3	Disabled
Allow cross-tenant replication	Disabled
Storage tasks assignments	None

Security

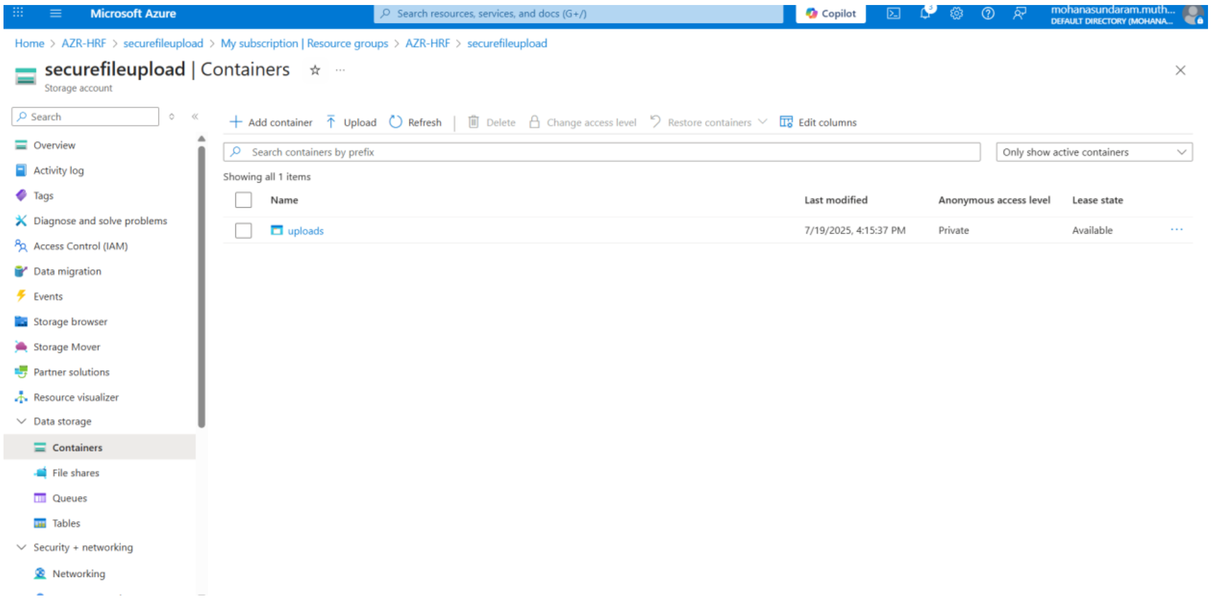
Require secure transfer for REST API operations	Enabled
Storage account key access	Enabled
Minimum TLS version	Version 1.2
Infrastructure encryption	Disabled

Networking

Allow access from	All networks
Private endpoint connections	0
Network routing	Microsoft network routing
Access for trusted Microsoft services	Yes

3.

Create a blob container name “uploads”

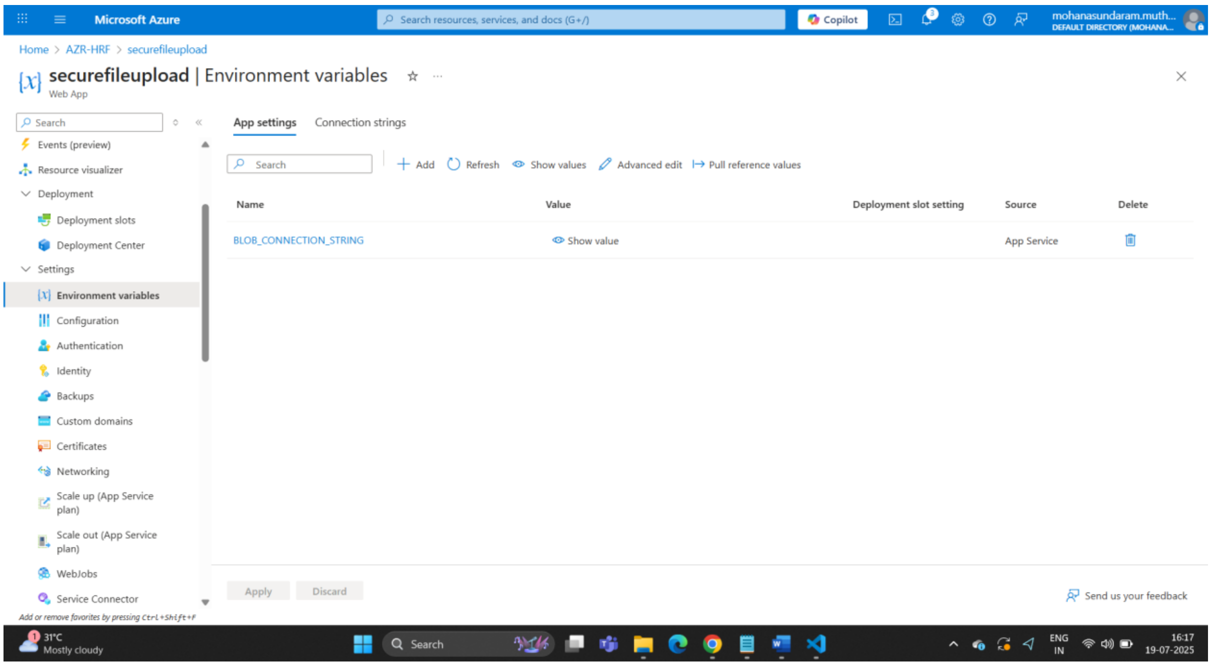


Showing all 1 items

Name	Last modified	Anonymous access level	Lease state
uploads	7/19/2025, 4:15:37 PM	Private	Available

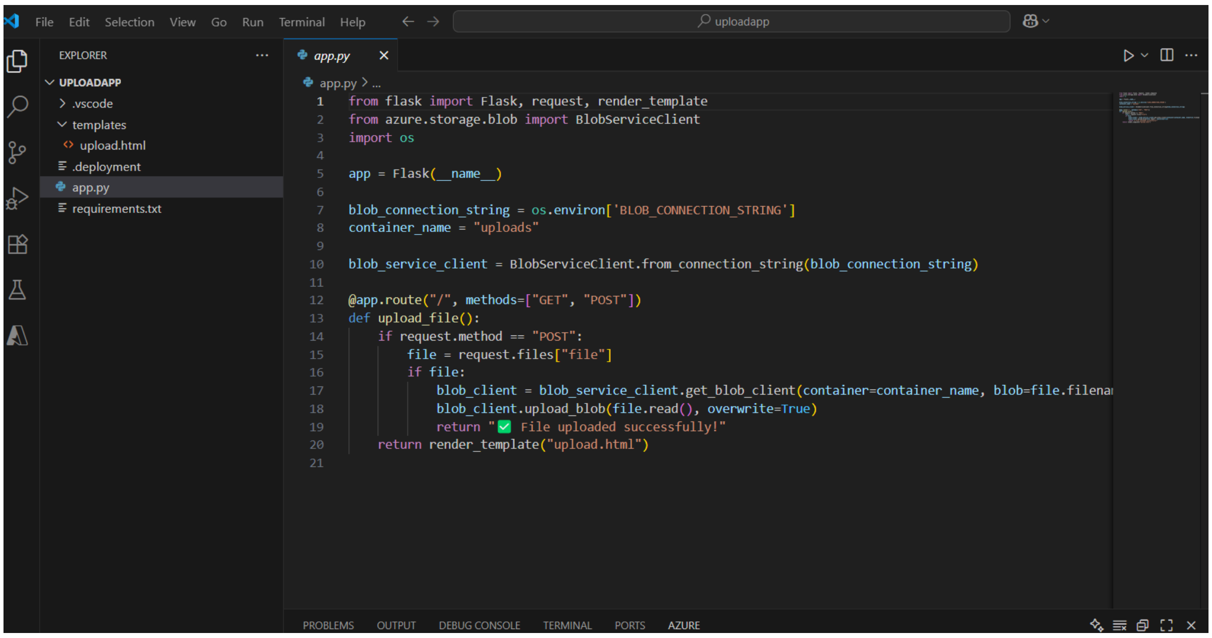
3. Noted down the connection string for storage "securefileupload"

4. Created the environment variables connection string name as "BLOB_CONNECTION_STRING" and Value as connection string



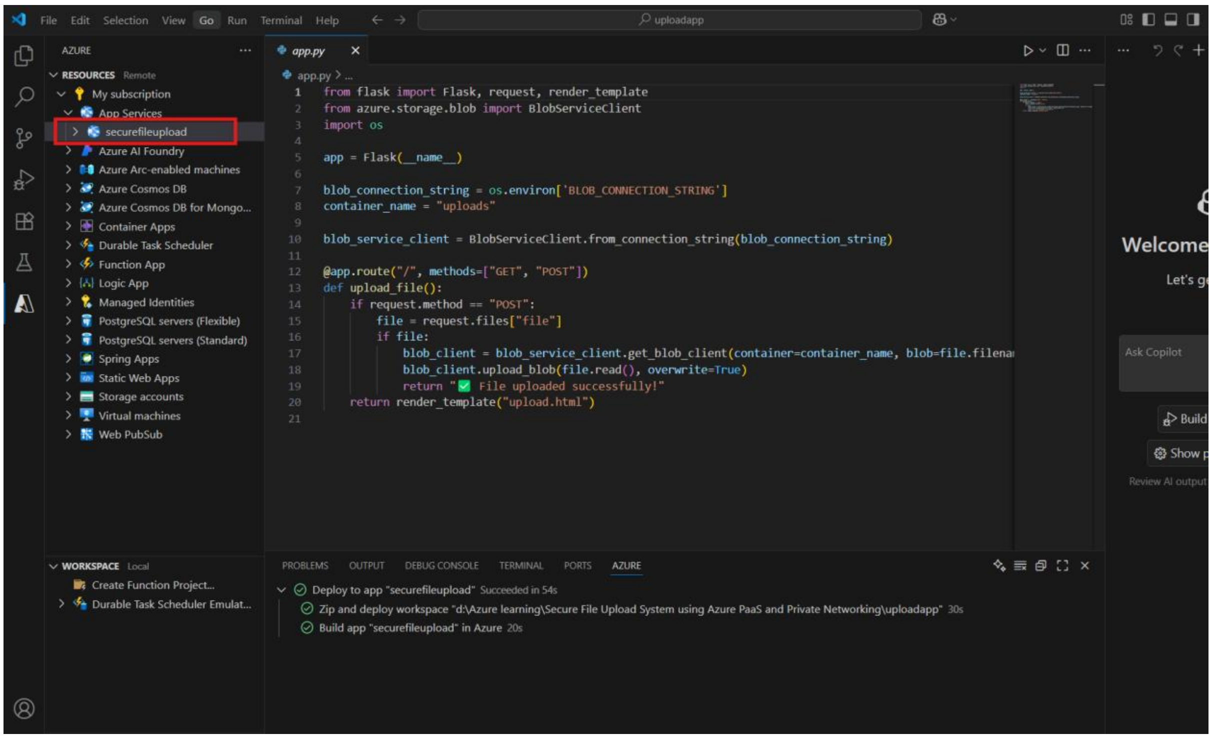
Name	Value	Deployment slot setting	Source	Delete
BLOB_CONNECTION_STRING	Show value		App Service	

5. Open the Zip folder of web app project



```python
from flask import Flask, request, render_template
from azure.storage.blob import BlobServiceClient
import os

app = Flask(__name__)

blob_connection_string = os.environ['BLOB_CONNECTION_STRING']
container_name = "uploads"

blob_service_client = BlobServiceClient.from_connection_string(blob_connection_string)

@app.route("/", methods=["GET", "POST"])
def upload_file():
    if request.method == "POST":
        file = request.files["file"]
        if file:
            blob_client = blob_service_client.get_blob_client(container=container_name, blob=file.filena
            blob_client.upload_blob(file.read(), overwrite=True)
            return "✅ File uploaded successfully!"
    return render_template("upload.html")
```

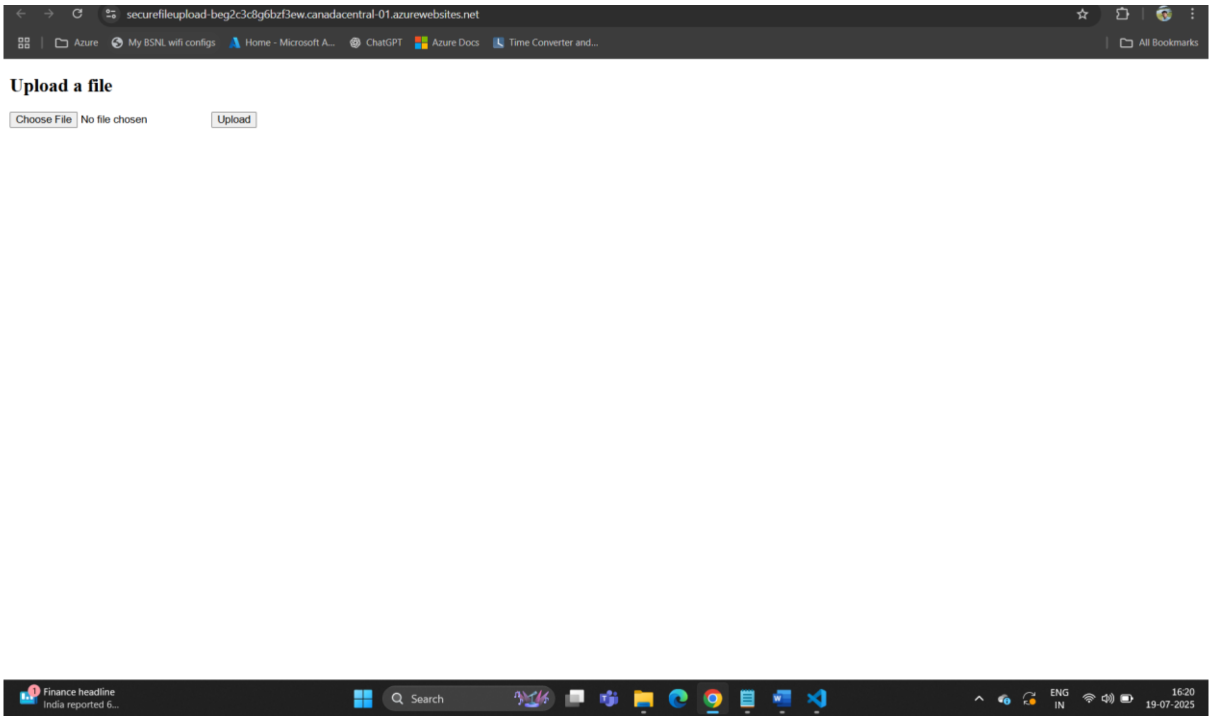
6.Next go to bottom tools and open Azure tools and expand our Azure subscription and deploy to web app.



```python
from flask import Flask, request, render_template
from azure.storage.blob import BlobServiceClient
import os

app = Flask(__name__)

blob_connection_string = os.environ['BLOB_CONNECTION_STRING']
container_name = "uploads"

blob_service_client = BlobServiceClient.from_connection_string(blob_connection_string)

@app.route("/", methods=["GET", "POST"])
def upload_file():
    if request.method == "POST":
        file = request.files["file"]
        if file:
            blob_client = blob_service_client.get_blob_client(container=container_name, blob=file.filena
            blob_client.upload_blob(file.read(), overwrite=True)
            return "✅ File uploaded successfully!"
    return render_template("upload.html")
```

```
Deploy to app "securefileupload" Succeeded in 54s
  Zip and deploy workspace "d:\Azure learning\Secure File Upload System using Azure PaaS and Private Networking\uploadapp" 30s
  Build app "securefileupload" in Azure 20s
```

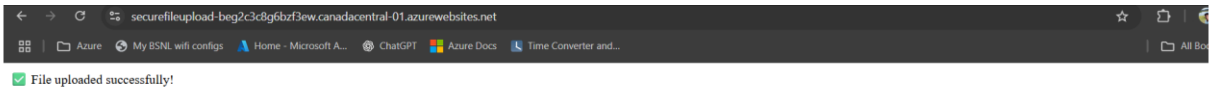
7. Next upload any pdf file from local machine



securefileupload-beg2c3c8g6bzf3ew.canadacentral-01.azurewebsites.net

Upload a file

8.After successfully uploaded it will receive in our blob storage under “uploads” container:



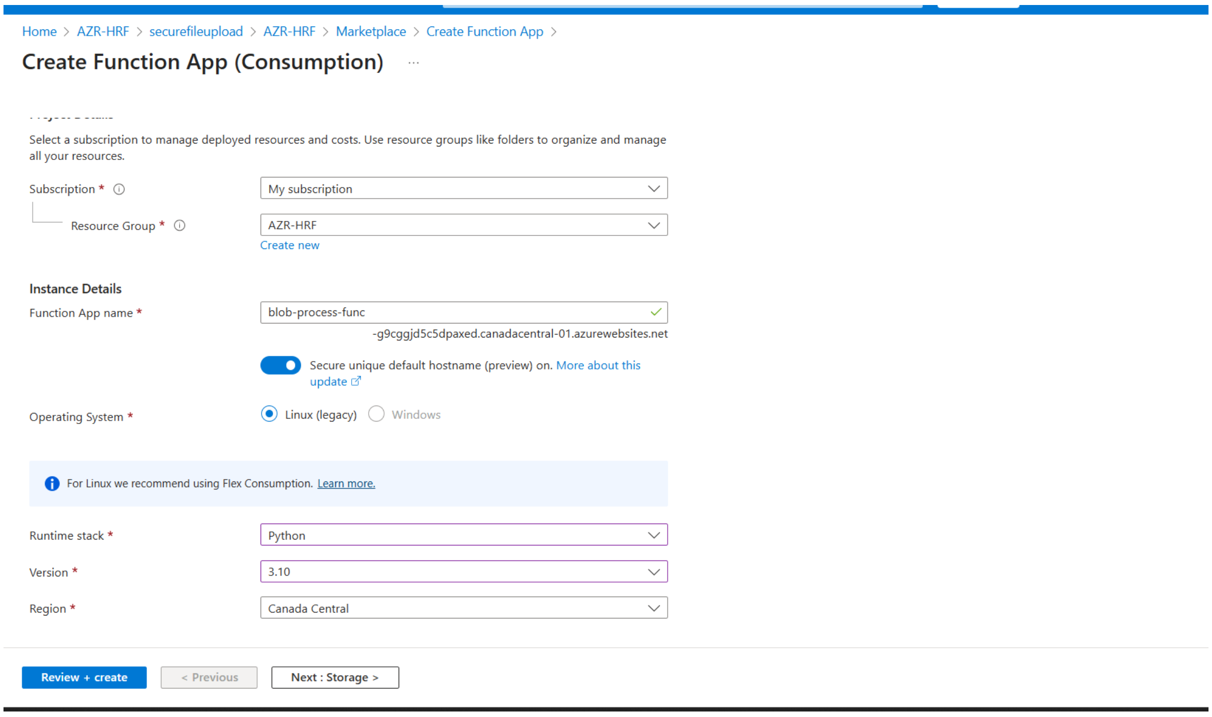
securefileupload-beg2c3c8g6bzf3ew.canadacentral-01.azurewebsites.net

✅ File uploaded successfully!

Phase 2:

Function app+Azure Table storage+computer vision:

1. Create func app with below config:


Home > AZR-HRF > securefileupload > AZR-HRF > Marketplace > Create Function App >

Create Function App (Consumption) ...

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * : My subscription

Resource Group * : AZR-HRF

Create new

Instance Details

Function App name * : blob-process-func

-g9cggjd5c5dpaxed.canadacentral-01.azurewebsites.net

Secure unique default hostname (preview) on. More about this update

Operating System * : [✓ Linux (legacy)] Windows

For Linux we recommend using Flex Consumption. Learn more.

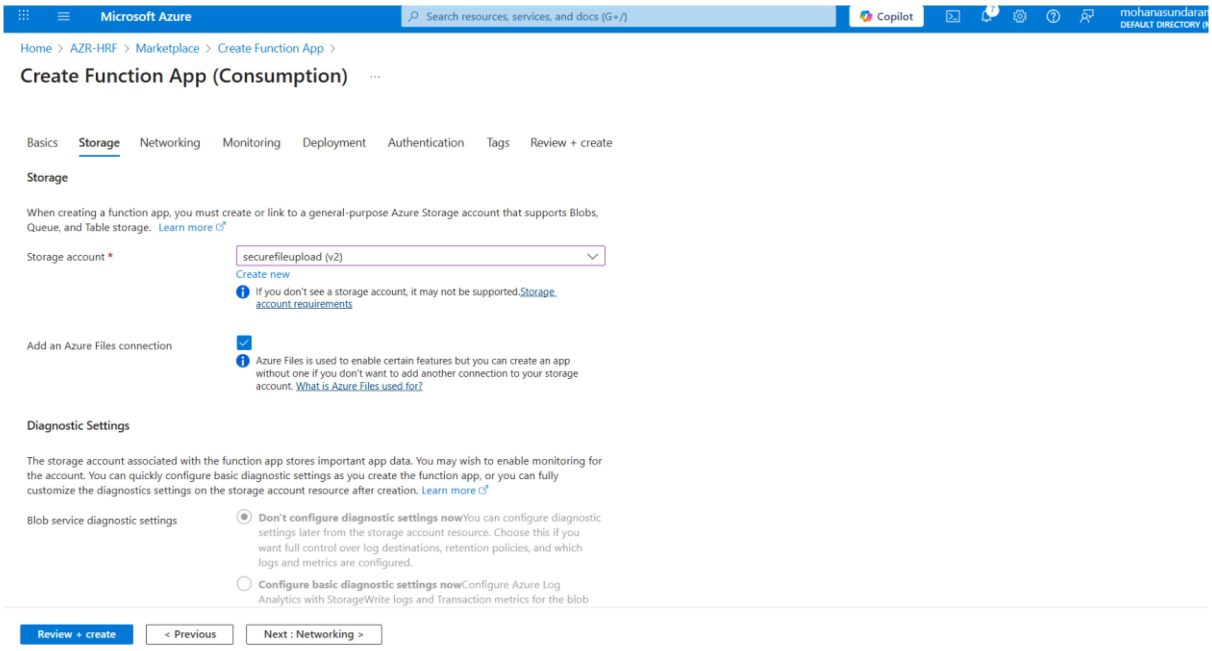
Runtime stack * : Python

Version * : 3.10

Region * : Canada Central

Review + create | < Previous | Next : Storage >

For storage give your existing storage:


Home > AZR-HRF > Marketplace > Create Function App >

Create Function App (Consumption) ...

Basics | Storage | Networking | Monitoring | Deployment | Authentication | Tags | Review + create

Storage

When creating a function app, you must create or link to a general-purpose Azure Storage account that supports Blobs, Queue, and Table storage. Learn more

Storage account * : securefileupload (v2)

Create new

If you don't see a storage account, it may not be supported. Storage account requirements

Add an Azure Files connection : ✓

Azure Files is used to enable certain features but you can create an app without one if you don't want to add another connection to your storage account. What is Azure Files used for?

Diagnostic Settings

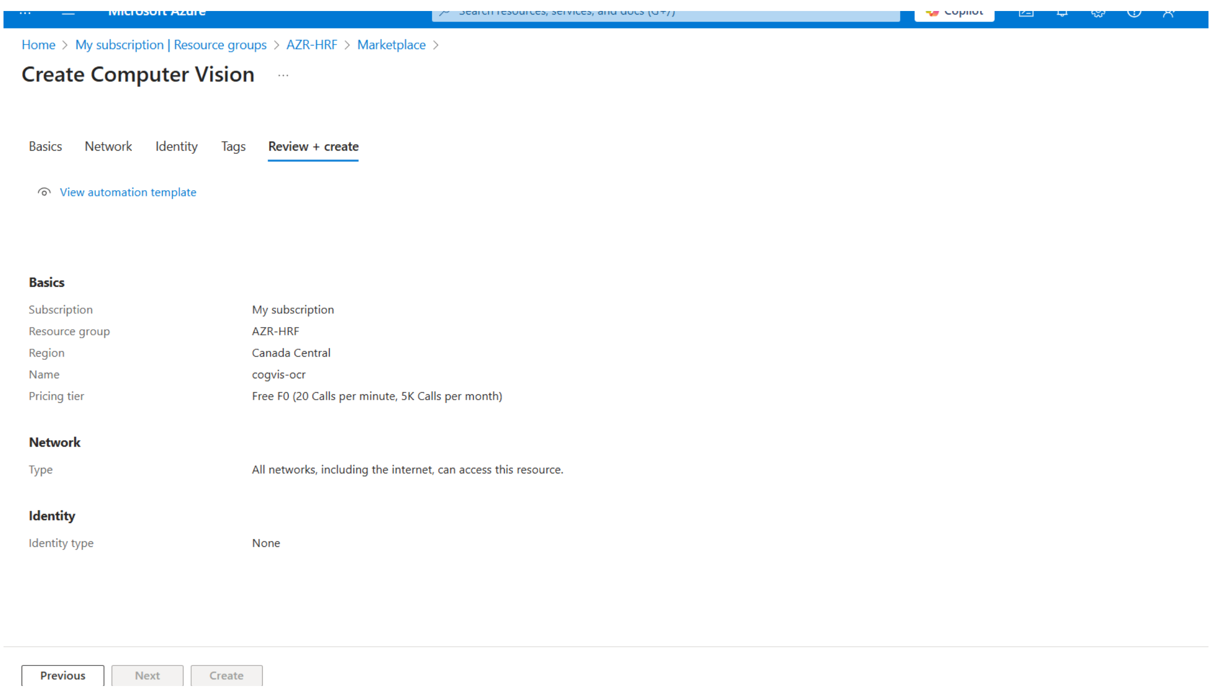
The storage account associated with the function app stores important app data. You may wish to enable monitoring for the account. You can quickly configure basic diagnostic settings as you create the function app, or you can fully customize the diagnostics settings on the storage account resource after creation. Learn more

Blob service diagnostic settings : [✓ Don't configure diagnostic settings now] You can configure diagnostic settings later from the storage account resource. Choose this if you want full control over log destinations, retention policies, and which logs and metrics are configured.

Configure basic diagnostic settings now Configure Azure Log Analytics with StorageWrite logs and Transaction metrics for the blob

Review + create | < Previous | Next : Networking >

2. Create computer vision service with Below configuration:



Home > My subscription | Resource groups > AZR-HRF > Marketplace >

Create Computer Vision

Basics

Subscription	My subscription
Resource group	AZR-HRF
Region	Canada Central
Name	cogvis-ocr
Pricing tier	Free F0 (20 Calls per minute, 5K Calls per month)

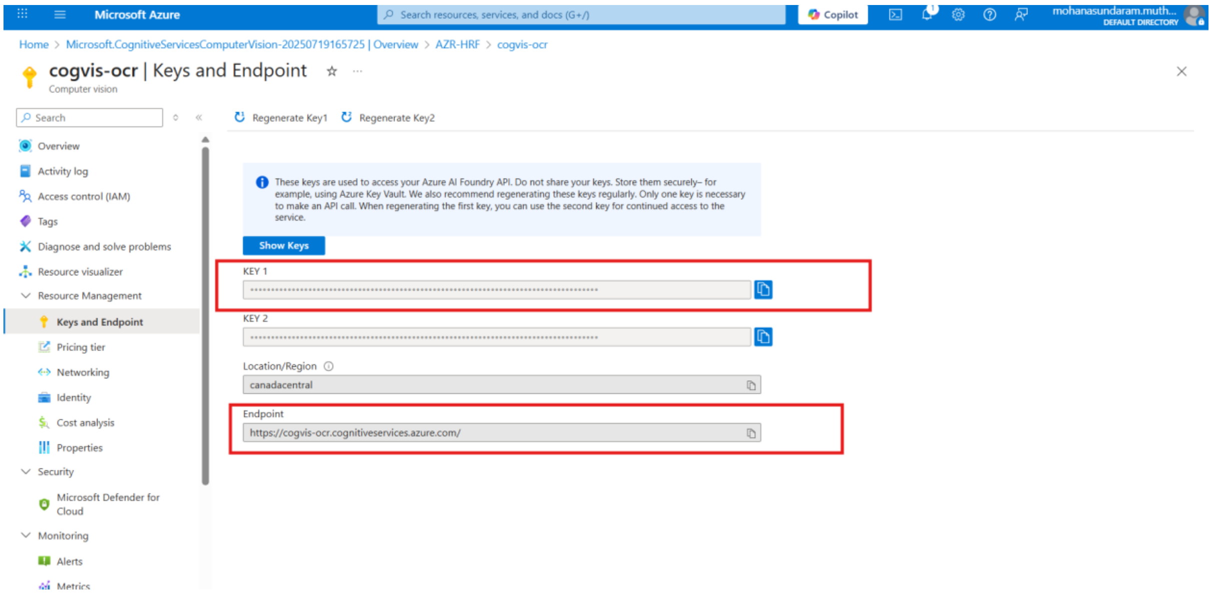
Network

Type	All networks, including the internet, can access this resource.

Identity

Identity type	None

3. After computer vision service provisioned note down the Key and endpoint:



Home > Microsoft.CognitiveServicesComputerVision-20250719165725 | Overview > AZR-HRF > cogvis-ocr

cogvis-ocr | Keys and Endpoint

Computer vision

These keys are used to access your Azure AI Foundry API. Do not share your keys. Store them securely– for example, using Azure Key Vault. We also recommend regenerating these keys regularly. Only one key is necessary to make an API call. When regenerating the first key, you can use the second key for continued access to the service.

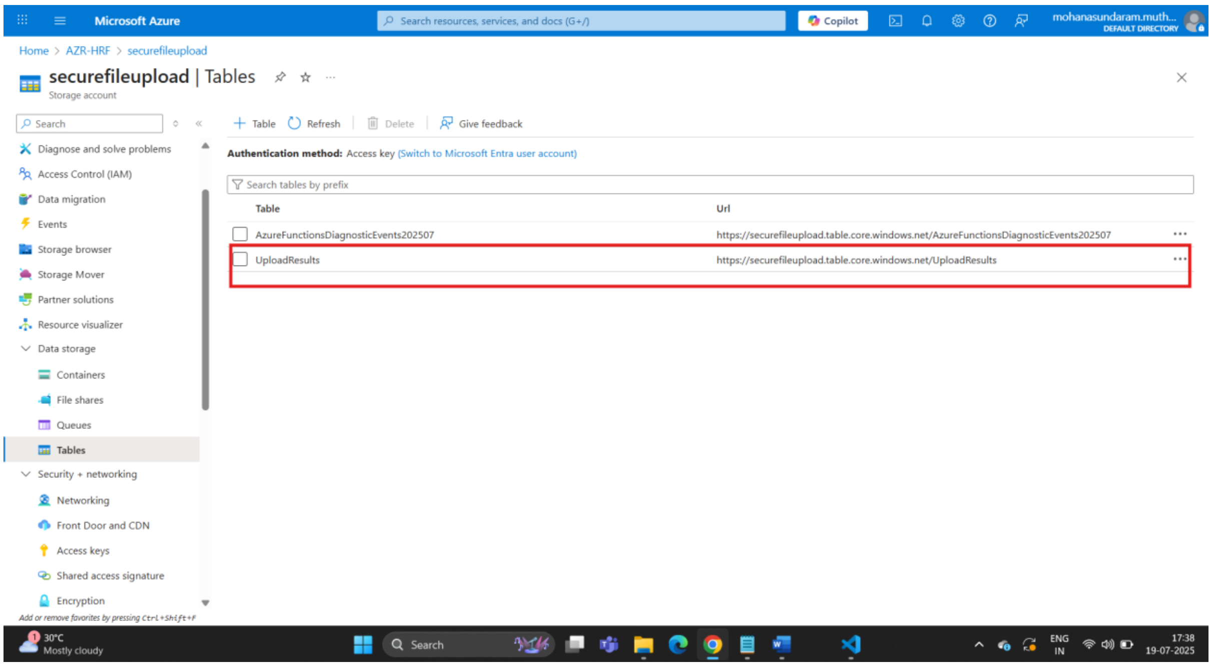
KEY 1: ••

KEY 2: ••

Location/Region: canadacentral

Endpoint: https://cogvis-ocr.cognitiveservices.azure.com/

4. Create the Azure Table storage in our existing storage acc "**securefileupload** "

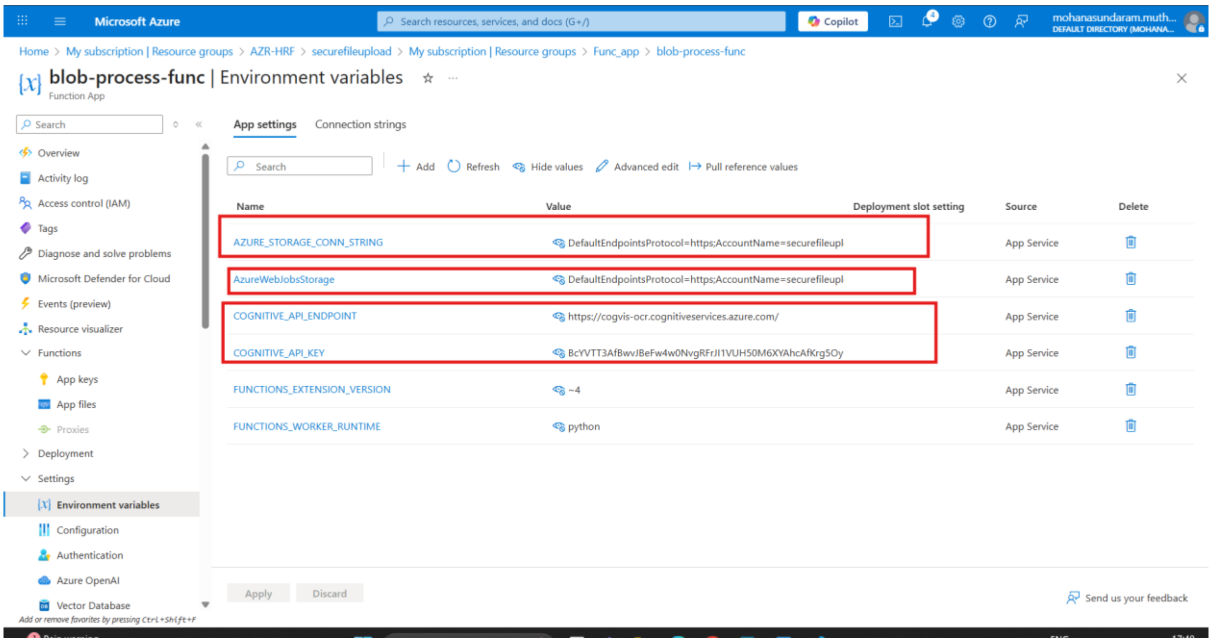

securefileupload | Tables

Storage account

Authentication method: Access key (Switch to Microsoft Entra user account)

Table	Url
AzureFunctionsDiagnosticEvents202507	https://securefileupload.table.core.windows.net/AzureFunctionsDiagnosticEvents202507
UploadResults	https://securefileupload.table.core.windows.net/UploadResults

5. `COGNITIVE_API_KEY` - key of cognitive
   `COGNITIVE_API_ENDPOINT`-endpoint of cognitive
   `AZURE_STORAGE_CONN_STRING` - connection string of storage where table storage output need to be place
   AzureWebJobsStorage- where blob trigger can be known by this storage connection string
   Every function app will created by storage as default we are using same storage for all input and output

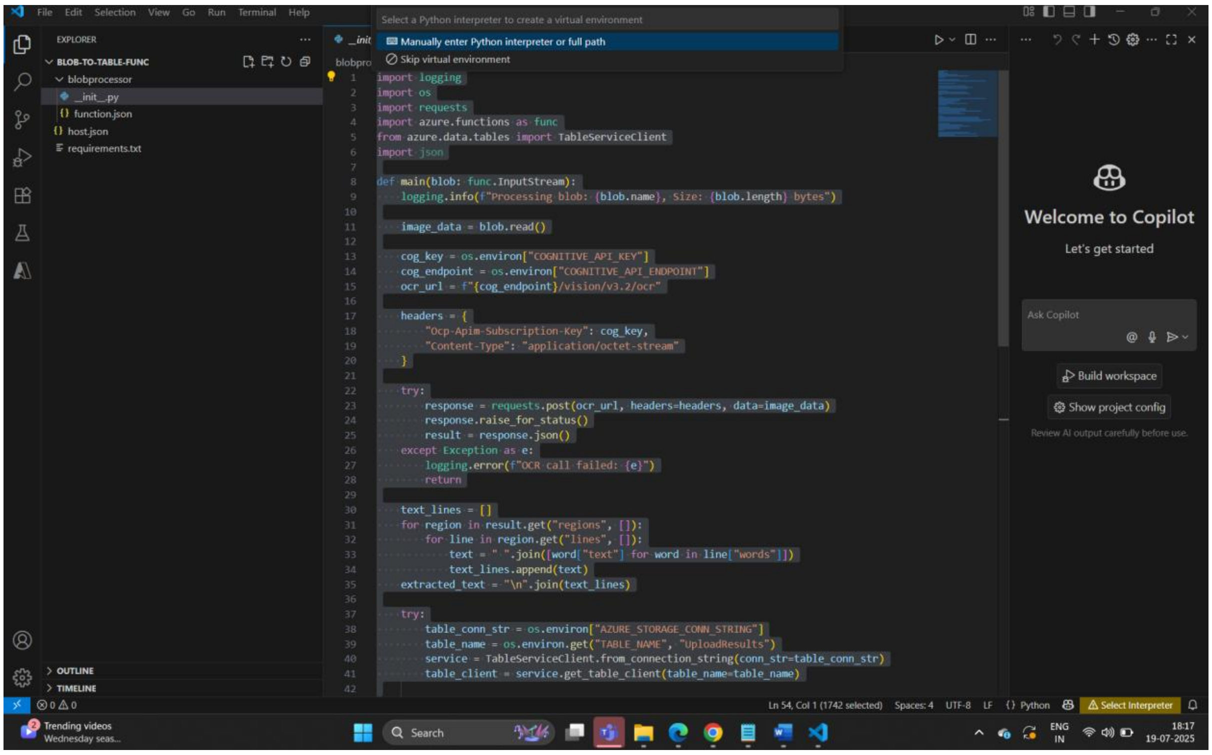

blob-process-func | Environment variables

Function App

App settings | Connection strings

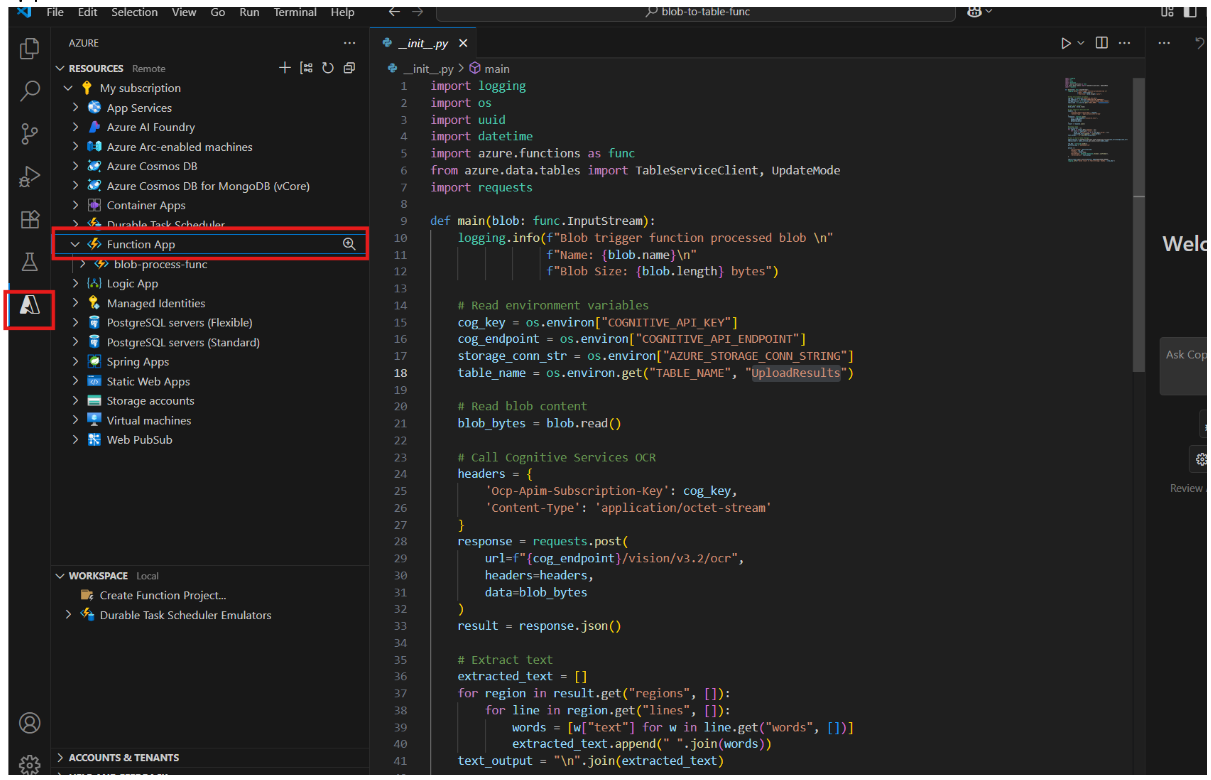
Name	Value	Deployment slot setting	Source	Delete
AZURE_STORAGE_CONN_STRING	DefaultEndpointsProtocol=https;AccountName=securefileupl		App Service	
AzureWebJobsStorage	DefaultEndpointsProtocol=https;AccountName=securefileupl		App Service	
COGNITIVE_API_ENDPOINT	https://cogvis-ocr.cognitiveservices.azure.com/		App Service	
COGNITIVE_API_KEY	BcYVTT3AfBwvJBeFw4w0NvgRFrJI1VUH50M6XYAhcAfKrg5Oy		App Service	
FUNCTIONS_EXTENSION_VERSION	~4		App Service	
FUNCTIONS_WORKER_RUNTIME	python		App Service	

6. Next open folder of our function app code in VS code:



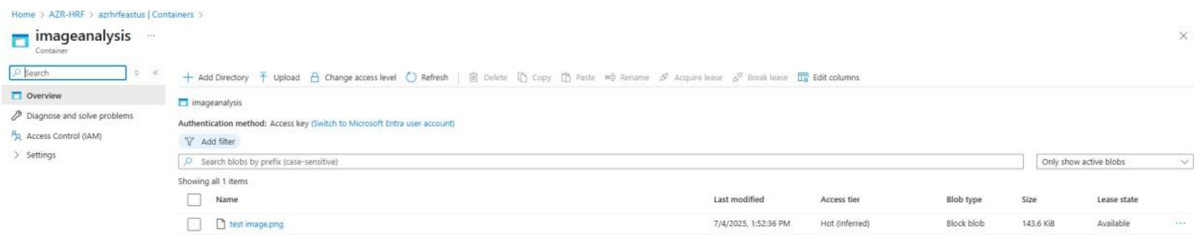
```python
import logging
import os
import requests
import azure.functions as func
from azure.data.tables import TableServiceClient
import json

def main(blob: func.InputStream):
    logging.info(f"Processing blob: {blob.name}, Size: {blob.length} bytes")

    image_data = blob.read()

    cog_key = os.environ["COGNITIVE_API_KEY"]
    cog_endpoint = os.environ["COGNITIVE_API_ENDPOINT"]
    ocr_url = f"{cog_endpoint}/vision/v3.2/ocr"

    headers = {
        "Ocp-Apim-Subscription-Key": cog_key,
        "Content-Type": "application/octet-stream"
    }

    try:
        response = requests.post(ocr_url, headers=headers, data=image_data)
        response.raise_for_status()
        result = response.json()
    except Exception as e:
        logging.error(f"OCR call failed: {e}")
        return

    text_lines = []
    for region in result.get("regions", []):
        for line in region.get("lines", []):
            text = " ".join([word["text"] for word in line["words"]])
            text_lines.append(text)
    extracted_text = "\n".join(text_lines)

    try:
        table_conn_str = os.environ["AZURE_STORAGE_CONN_STRING"]
        table_name = os.environ.get("TABLE_NAME", "UploadResults")
        service = TableServiceClient.from_connection_string(conn_str=table_conn_str)
        table_client = service.get_table_client(table_name=table_name)
```

7. Go to Azure tools and right click on Azure function and deploy the code into Azure function app:



```python
import logging
import os
import uuid
import datetime
import azure.functions as func
from azure.data.tables import TableServiceClient, UpdateMode
import requests

def main(blob: func.InputStream):
    logging.info(f"Blob trigger function processed blob \n"
                 f"Name: {blob.name}\n"
                 f"Blob Size: {blob.length} bytes")

    # Read environment variables
    cog_key = os.environ["COGNITIVE_API_KEY"]
    cog_endpoint = os.environ["COGNITIVE_API_ENDPOINT"]
    storage_conn_str = os.environ["AZURE_STORAGE_CONN_STRING"]
    table_name = os.environ.get("TABLE_NAME", "UploadResults")

    # Read blob content
    blob_bytes = blob.read()

    # Call Cognitive Services OCR
    headers = {
        'Ocp-Apim-Subscription-Key': cog_key,
        'Content-Type': 'application/octet-stream'
    }
    response = requests.post(
        url=f"{cog_endpoint}/vision/v3.2/ocr",
        headers=headers,
        data=blob_bytes
    )
    result = response.json()

    # Extract text
    extracted_text = []
    for region in result.get("regions", []):
        for line in region.get("lines", []):
            words = [w["text"] for w in line.get("words", [])]
            extracted_text.append(" ".join(words))
    text_output = "\n".join(extracted_text)
```

Once I have upload the file in web app ui, then the image is stored in Blob container


Home > AZR-HRF > azrhrfeastus | Containers >

imageanalysis
Container

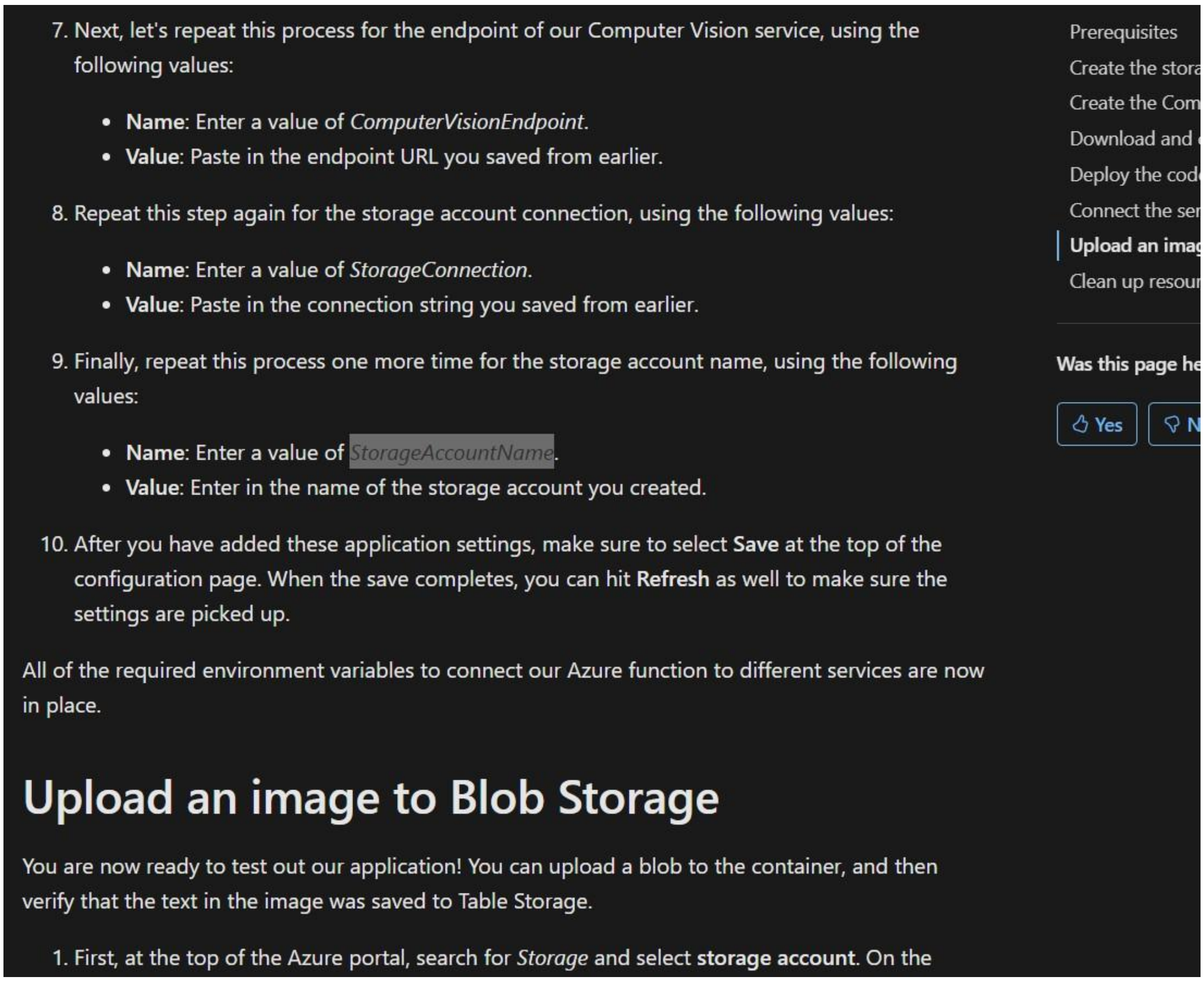
Authentication method: Access key (Switch to Microsoft Entra user account)

Only show active blobs

Showing all 1 items

Name	Last modified	Access tier	Blob type	Size	Lease state
test image.png	7/4/2025, 1:52:36 PM	Hot (Inferred)	Block blob	143.6 KiB	Available

I have uploaded the below image as input



7. Next, let's repeat this process for the endpoint of our Computer Vision service, using the following values:
   - **Name**: Enter a value of *ComputerVisionEndpoint*.
   - **Value**: Paste in the endpoint URL you saved from earlier.
8. Repeat this step again for the storage account connection, using the following values:
   - **Name**: Enter a value of *StorageConnection*.
   - **Value**: Paste in the connection string you saved from earlier.
9. Finally, repeat this process one more time for the storage account name, using the following values:
   - **Name**: Enter a value of *StorageAccountName*.
   - **Value**: Enter in the name of the storage account you created.
10. After you have added these application settings, make sure to select **Save** at the top of the configuration page. When the save completes, you can hit **Refresh** as well to make sure the settings are picked up.

All of the required environment variables to connect our Azure function to different services are now in place.

Upload an image to Blob Storage

You are now ready to test out our application! You can upload a blob to the container, and then verify that the text in the image was saved to Table Storage.

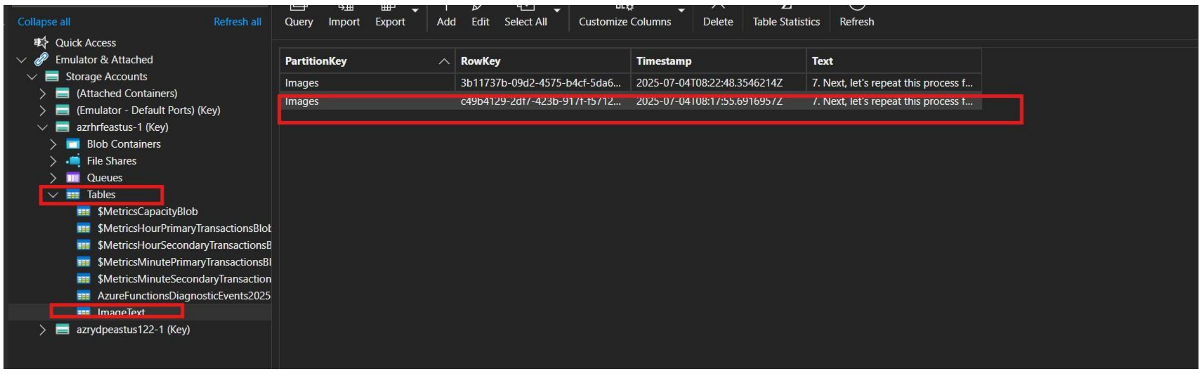
1. First, at the top of the Azure portal, search for *Storage* and select **storage account**. On the

 The Function is triggered automatically via a **Blob Trigger** when an image is uploaded to the container imageanalysis.

 Inside the Function (ProcessImageUpload.cs):

- Reads environment variables:
  - StorageConnection
  - StorageAccountName
  - ComputerVisionKey
  - ComputerVisionEndpoint
- Constructs the full image URL
- Sends it to the Computer Vision API
- Receives extracted text response
- Saves it to Azure Table Storage (ImageText)

Get the SAS Token of storage and open MS Storage explorer and table storage you can able to see image is converted to text and stored.



PartitionKey	RowKey	Timestamp	Text
Images	3b11737b-09d2-4575-b4cf-5da6...	2025-07-04T08:22:48.3546214Z	7. Next, let's repeat this process f...
Images	c49b4129-2df7-423b-917f-f5712...	2025-07-04T08:17:55.6916957Z	7. Next, let's repeat this process f...

Output text stored in table:
7. Next, let's repeat this process for the endpoint of our Computer Vision service, using thePrerequisitesfollowing values:Create the storName: Enter a value of ComputerVisionEndpoint.Create the ComValue: Paste in the endpoint URL you saved from earlier.Download andDeploy the cod8. Repeat this step again for the storage account connection, using the following values:Connect the se| Upload an ima· Name: Enter a value of
StorageConnection.. Value: Paste in the connection string you saved from earlier.Clean up resou9.
Finally, repeat this process one more time for the storage account name, using the followingWas this page hevalues:Yes· Name: Enter a value of StorageAccountName.Value: Enter in the name of the storage account you created.10. After you have added these application settings, make sure to select Save at the top of theconfiguration page. When the save completes, you can hit Refresh as well to make sure thesettings are picked up.All of the required environment variables to connect our Azure function to different services are nowin place.Upload an image to Blob StorageYou are now ready to test out our application! You can upload a blob to the container, and thenverify that the text in the
image was saved to Table Storage.1. First, at the top of the Azure portal, search for Storage and select storage account. On the