Project Name:

SimpleTimeService-Docker-K8s-Terraform Using Azure Cloud

Table of Contents

1.Project Overview:

SimpleTimeService is a microservice that provides the current timestamp and the IP address. It is designed for minimalist deployment using Docker and Kubernetes.

This project also includes Terraform scripts to automate the deployment of the service on AZURE using AKS.

2. Pre-requisites:

- Docker
- Python
- Azure Cloud (AKS, VNET, RESOURCE GROUP)
- Terraform
- Azure CLI

3. Create a Simple Application:

After Completion of Python setup. Run the **pip install flask** because flask is a web framework to creates a Webservice.

SimpleTimeService.py

C: > Users > ktssk > Downloads > SimpleTimeService > SimpleTimeService.py

```python
from flask import Flask, jsonify, request
from datetime import datetime

app = Flask(__name__)

@app.route("/")
def home():
    return jsonify({
        "timestamp": datetime.utcnow().isoformat(),
        "ip": request.remote_addr
    })

if __name__ == "__main__":
    app.run()
```

Description of the Code:

from flask import Flask, jsonify, request

from datetime import datetime

- **Flask**: A Python web framework used to create this web service.
- **jsonify**: Converts Python dictionaries into JSON responses.
- **request**: Extract details from incoming HTTP requests
- **datetime**: Used to fetch the current date and time.

app = Flask(__name__)

- This creates an instance of the **Flask** web application.

@app.route("/")

def home():

return jsonify({

"timestamp": datetime.utcnow().isoformat(),

"ip": request.remote_addr

})

- @app.route("/"): Defines a **route** for the root URL (/), meaning when users visit the server's page.

- datetime.utcnow().isoformat(): Gets the current **UTC time** in **ISO 8601 format** .
- request.remote_addr: Extracts the **IP address** of the making the request.
- jsonify(...): Converts the response to **JSON format**.

```
if __name__ == "__main__":
    app.run()
```

- **if __name__ == "__main__":** ensures that the script runs only if executed directly.
- **app.run()** starts the Flask web server.

→ Run the Python code in the terminal.

```
C:\Windows\System32\cmd.e

C:\Users\ktssk\Downloads\SimpleTimeService>python ./SimpleTimeService.py
```

→Output is the response includes the client's IP address.

```
C:\Windows\System32\cmd.e

C:\Users\ktssk\Downloads\SimpleTimeService>python ./SimpleTimeService.py
 * Serving Flask app 'SimpleTimeService' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

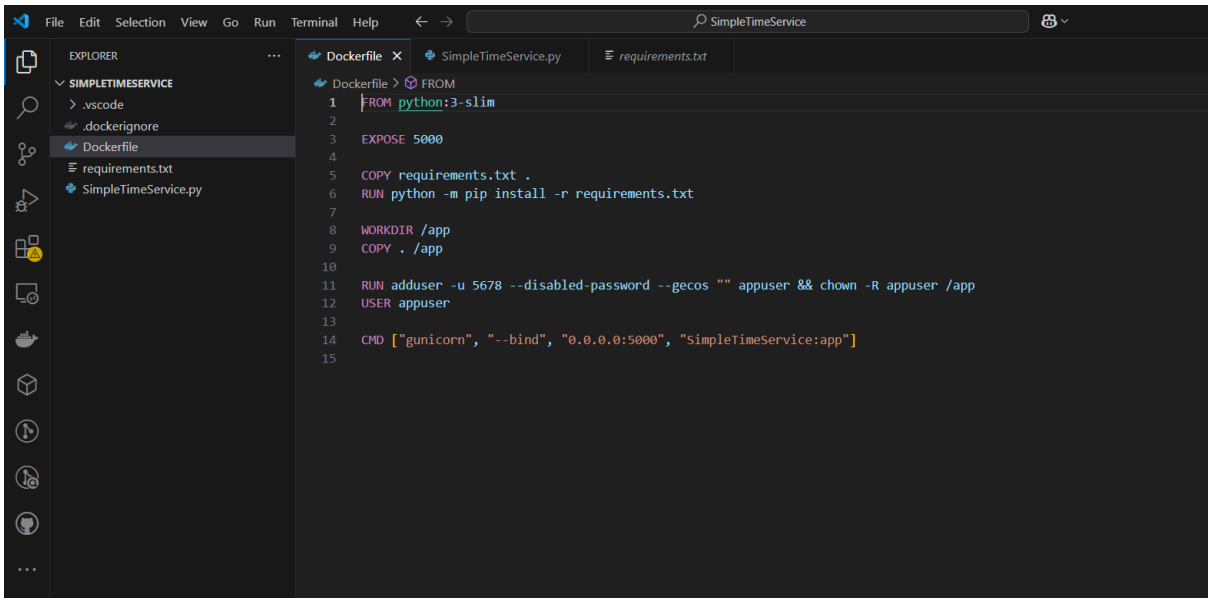
→Browse the Localhost URL to view the Application Webpage.

```
127.0.0.1:5000
127.0.0.1:5000
Pretty-print
{"ip":"127.0.0.1","timestamp":"2025-02-07T17:17:02.148979"}
```

4. Dockerizing the Application:

→Create a Docker file in the same Code Application Folder to load the code dependencies to the docker file.

→Write a Docker file based on the code.



→Build the Docker Image with Docker build command.

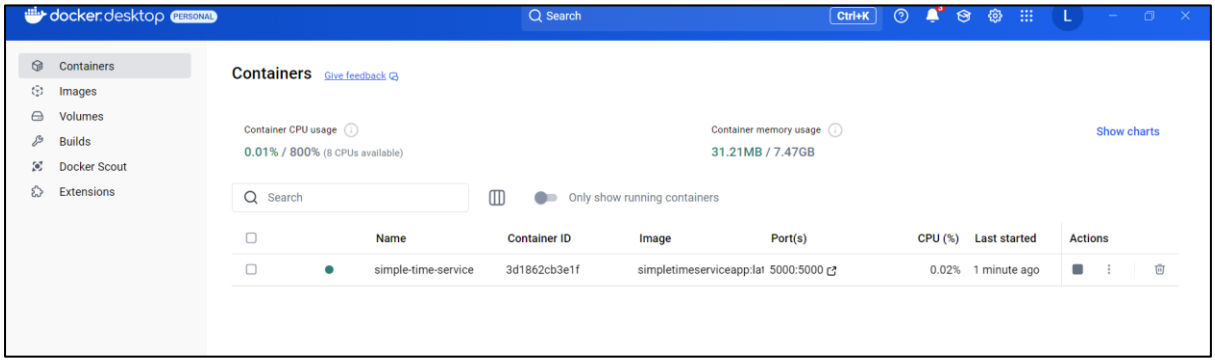
docker build -t simpletimeserviceapp .

```
PS C:\Users\ktssk\Downloads\SimpleTimeService> docker build -t simpletimeserviceapp .
[+] Building 20.2s (12/12) FINISHED                                                                    docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                   0.0s
 => => transferring dockerfile: 349B                                                                                   0.0s
 => [internal] load metadata for docker.io/library/python:3-slim                                                       3.8s
 => [auth] library/python:pull token for registry-1.docker.io                                                          0.0s
 => [internal] load .dockerignore                                                                                      0.0s
 => => transferring context: 409B                                                                                      0.0s
 => [1/6] FROM docker.io/library/python:3-slim@sha256:ae9f9ac89467077ed1efefb6d9042132d28134ba201b2820227d46c9effd3174  8.4s
 => => resolve docker.io/library/python:3-slim@sha256:ae9f9ac89467077ed1efefb6d9042132d28134ba201b2820227d46c9effd3174  0.0s
 => => sha256:34abb9d38fa3fcf8526765f370a3f2c6341456eca879f3af027c42062521e5bc 250B / 250B                            0.7s
 => => sha256:0d263a67e0dbb841a730efc09f84672f73b02427a19a57d9ef380b66eaaf0aec 12.58MB / 12.58MB                       6.7s
 => => sha256:91af2433aaf067a250607e067549a710adc587e7f5cf57c4ee096224cc5834dc 3.51MB / 3.51MB                         1.6s
 => => sha256:c29f5b76f736a8b555fd191c48d6581bb918bcd605a7cbcc76205dd6acff3260 28.21MB / 28.21MB                       6.8s
 => => extracting sha256:c29f5b76f736a8b555fd191c48d6581bb918bcd605a7cbcc76205dd6acff3260                              0.9s
 => => extracting sha256:91af2433aaf067a250607e067549a710adc587e7f5cf57c4ee096224cc5834dc                              0.1s
 => => extracting sha256:0d263a67e0dbb841a730efc09f84672f73b02427a19a57d9ef380b66eaaf0aec                              0.4s
 => => extracting sha256:34abb9d38fa3fcf8526765f370a3f2c6341456eca879f3af027c42062521e5bc                              0.0s
 => [internal] load build context                                                                                      0.0s
 => => transferring context: 78B                                                                                       0.0s
 => [2/6] COPY requirements.txt .                                                                                      0.2s
 => [3/6] RUN python -m pip install -r requirements.txt                                                                5.5s
 => [4/6] WORKDIR /app                                                                                                 0.1s
 => [5/6] COPY . /app                                                                                                  0.0s
 => [6/6] RUN adduser -u 5678 --disabled-password --gecos "" appuser && chown -R appuser /app                          0.5s
 => exporting to image                                                                                                 1.3s
 => => exporting layers                                                                                                0.9s
 => => exporting manifest sha256:56e6b9500c4e5e40c45b015a5a169266e1b8df7e2abb13f47e612bd034dd0a44                      0.0s
 => => exporting config sha256:34f0323748e062733778a153b59cce6c12cbb222725e9e3ec23260eeb046f448                        0.0s
 => => exporting attestation manifest sha256:e04106f6c1c73a5bf71498b8a169b886787fc44022eec73609203b4636f9d799          0.0s
 => => exporting manifest list sha256:2f6cd80aa5cb20af5cbd70b7fc18e1639482652090ba080e05eb64e0d97aedad                 0.0s
 => => naming to docker.io/library/simpletimeserviceapp:latest                                                         0.0s
 => => unpacking to docker.io/library/simpletimeserviceapp:latest                                                      0.3s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/2dr6eb9l6bgat022ph2p4otik

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
```

→Run the image to Create a Container

```
PS C:\Users\ktssk\Downloads\SimpleTimeService> docker run -d -p 5000:5000 --name simple-time-service simpletimeserviceapp:latest
3d1862cb3e1fcbed033e699082b771b4d5b15a071dd77d52bf4f3269cd77f45d
```

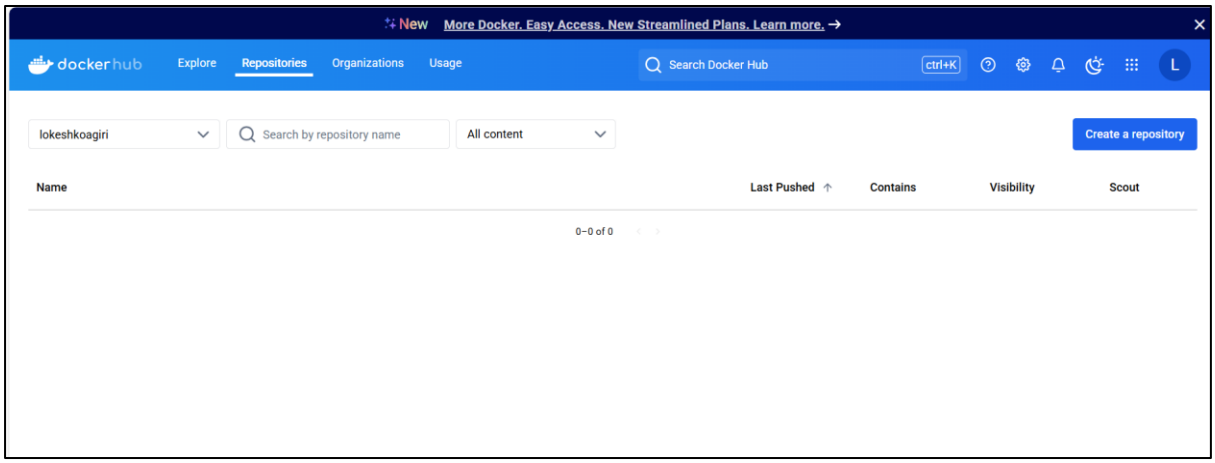
→Go to Docker Engine and verify the container.



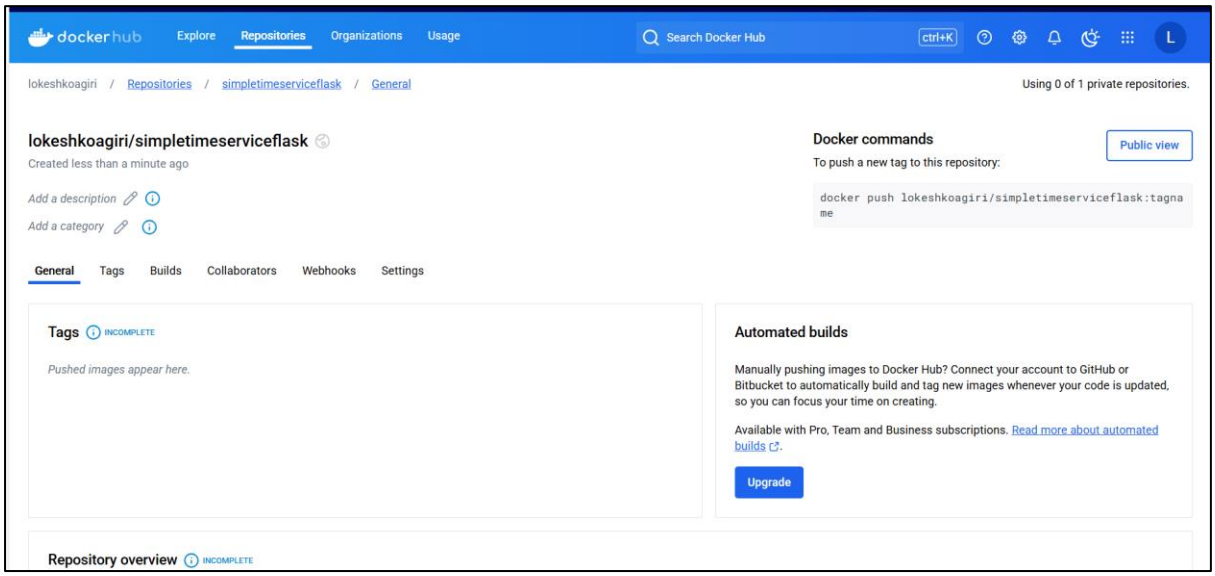
→Open the Located port to view.

→Push the Docker image to the Docker Hub. Go to hub.docker.com and login with your Credentials to store our docker image to Docker hub.

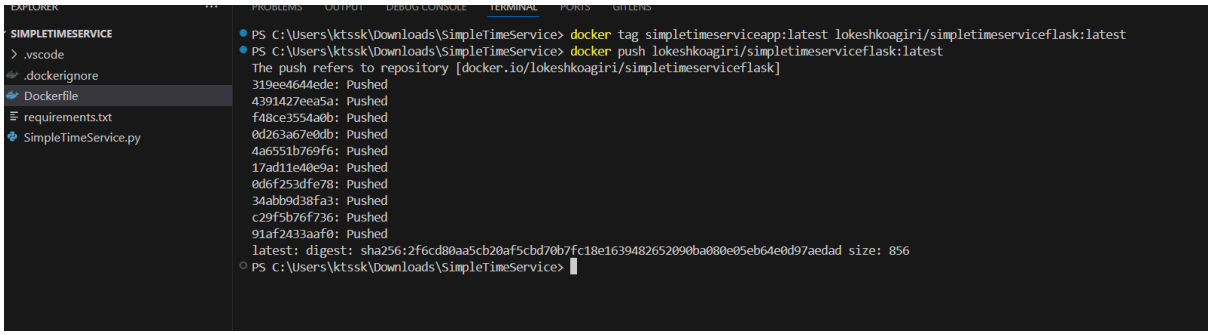
→Create a Repository and push the image to hub.



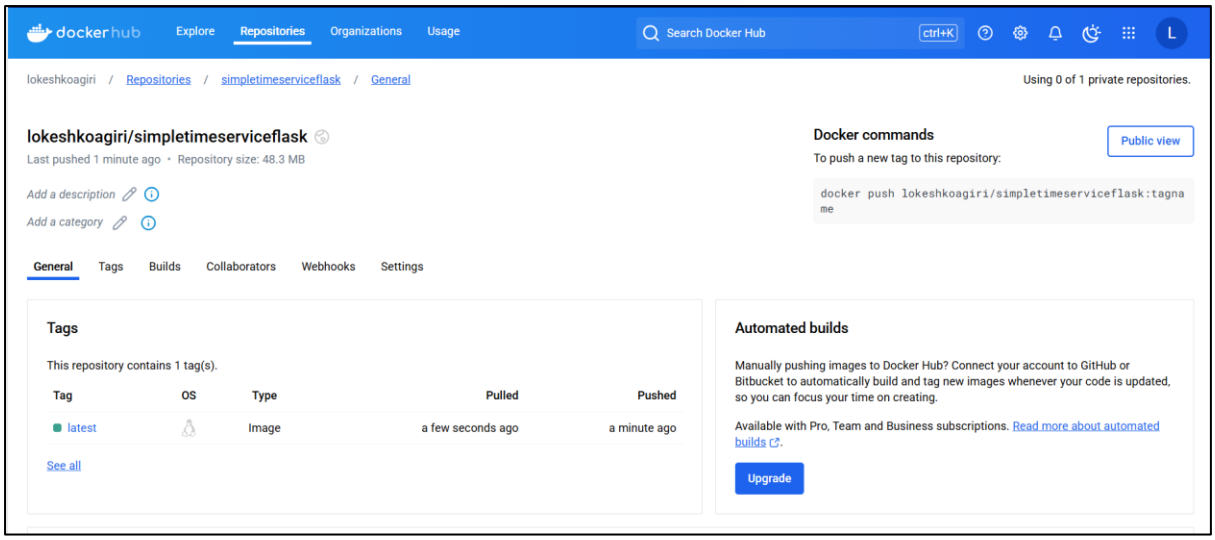
→After created the Repository look like this.



→Tag and push the Docker image to docker hub with commands.

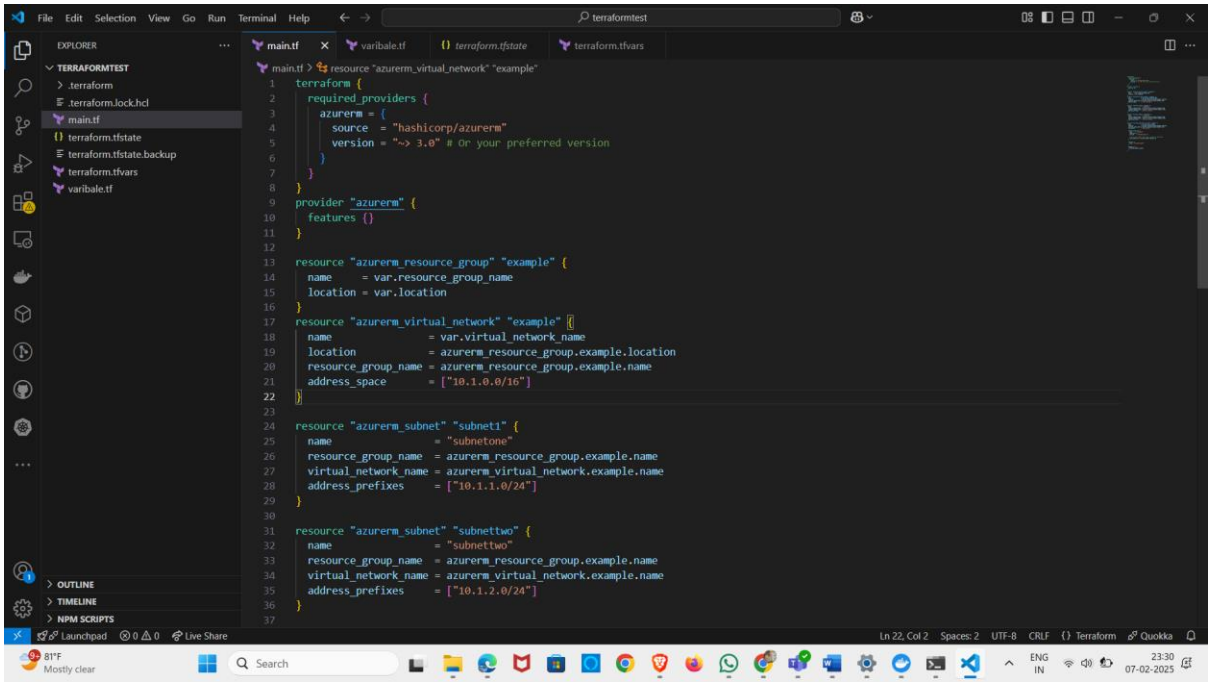


→Verify the image is stored in the Docker HUB.

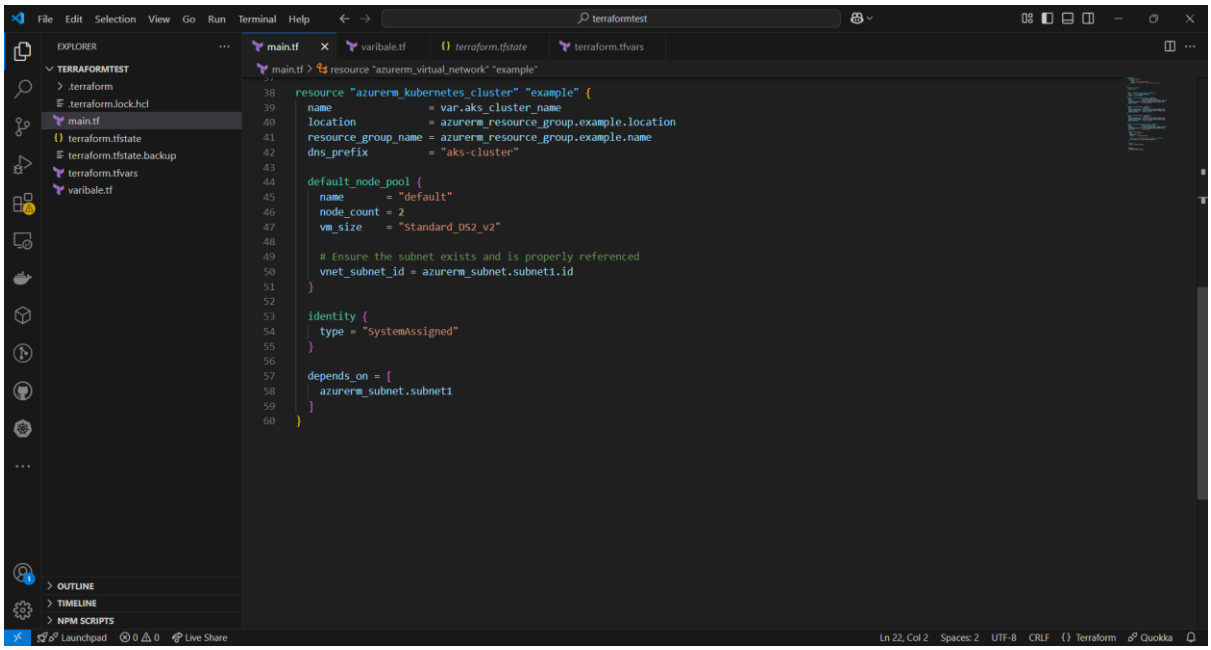


5. Infrastructure as Code (Terraform on Azure):

→Create a Terraform script with main.tf

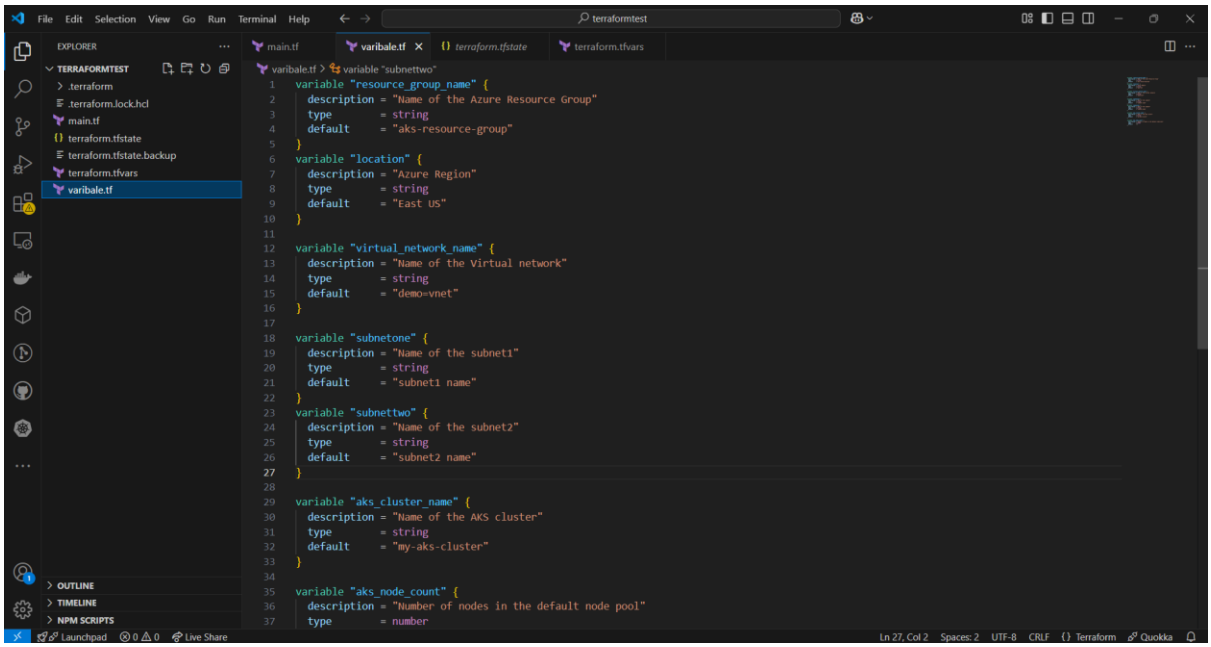


```terraform
terraform {
  required_providers {
    azurerm = {
      source  = "hashicorp/azurerm"
      version = "~> 3.0" # Or your preferred version
    }
  }
}
provider "azurerm" {
  features {}
}

resource "azurerm_resource_group" "example" {
  name     = var.resource_group_name
  location = var.location
}
resource "azurerm_virtual_network" "example" {
  name                = var.virtual_network_name
  location            = azurerm_resource_group.example.location
  resource_group_name = azurerm_resource_group.example.name
  address_space       = ["10.1.0.0/16"]
}

resource "azurerm_subnet" "subnet1" {
  name                 = "subnetone"
  resource_group_name  = azurerm_resource_group.example.name
  virtual_network_name = azurerm_virtual_network.example.name
  address_prefixes     = ["10.1.1.0/24"]
}

resource "azurerm_subnet" "subnettwo" {
  name                 = "subnettwo"
  resource_group_name  = azurerm_resource_group.example.name
  virtual_network_name = azurerm_virtual_network.example.name
  address_prefixes     = ["10.1.2.0/24"]
}
```

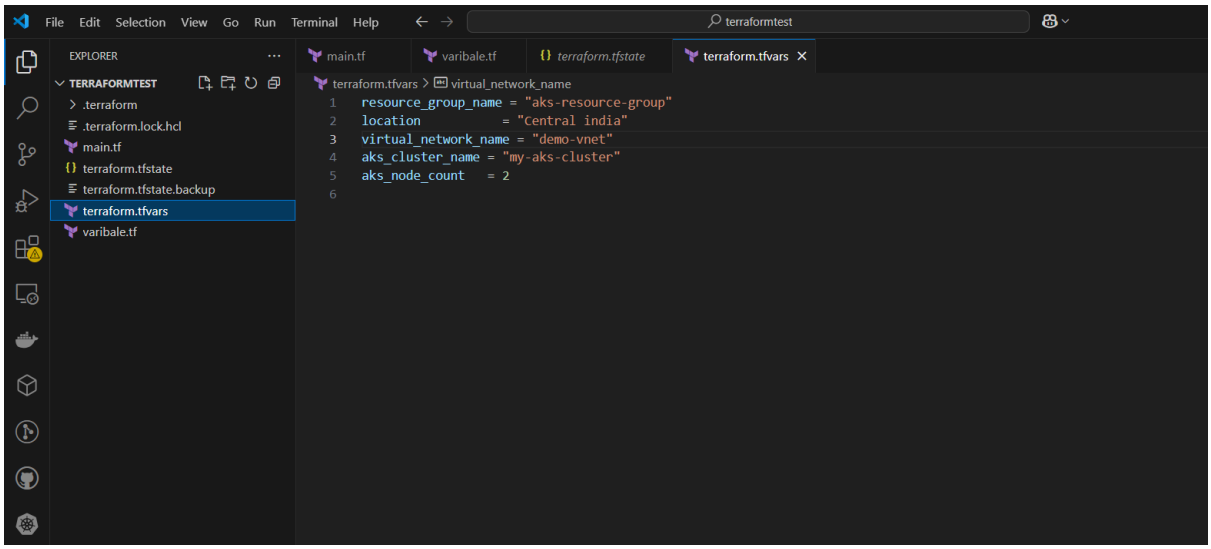


```terraform
resource "azurerm_kubernetes_cluster" "example" {
  name                = var.aks_cluster_name
  location            = azurerm_resource_group.example.location
  resource_group_name = azurerm_resource_group.example.name
  dns_prefix          = "aks-cluster"

  default_node_pool {
    name       = "default"
    node_count = 2
    vm_size    = "Standard_DS2_v2"

    # Ensure the subnet exists and is properly referenced
    vnet_subnet_id = azurerm_subnet.subnet1.id
  }

  identity {
    type = "SystemAssigned"
  }

  depends_on = [
    azurerm_subnet.subnet1
  ]
}
```

→Create a Variable file based on the main code.

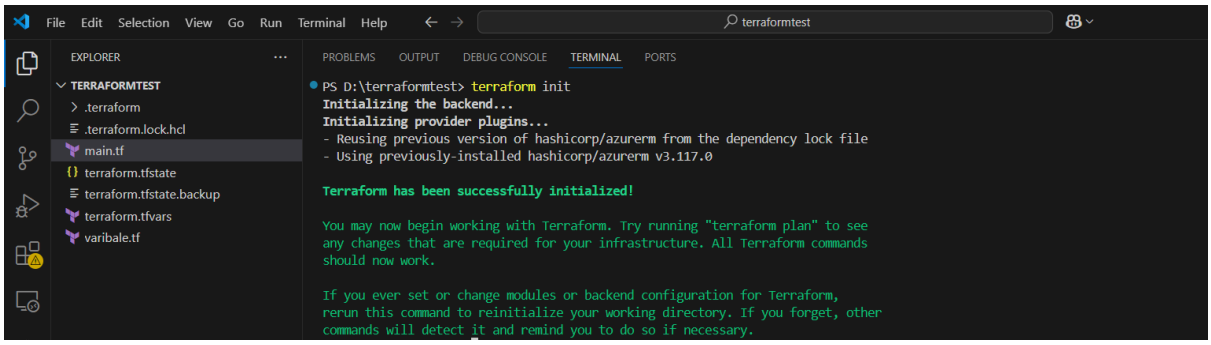


→Create a terraform var file to call the values

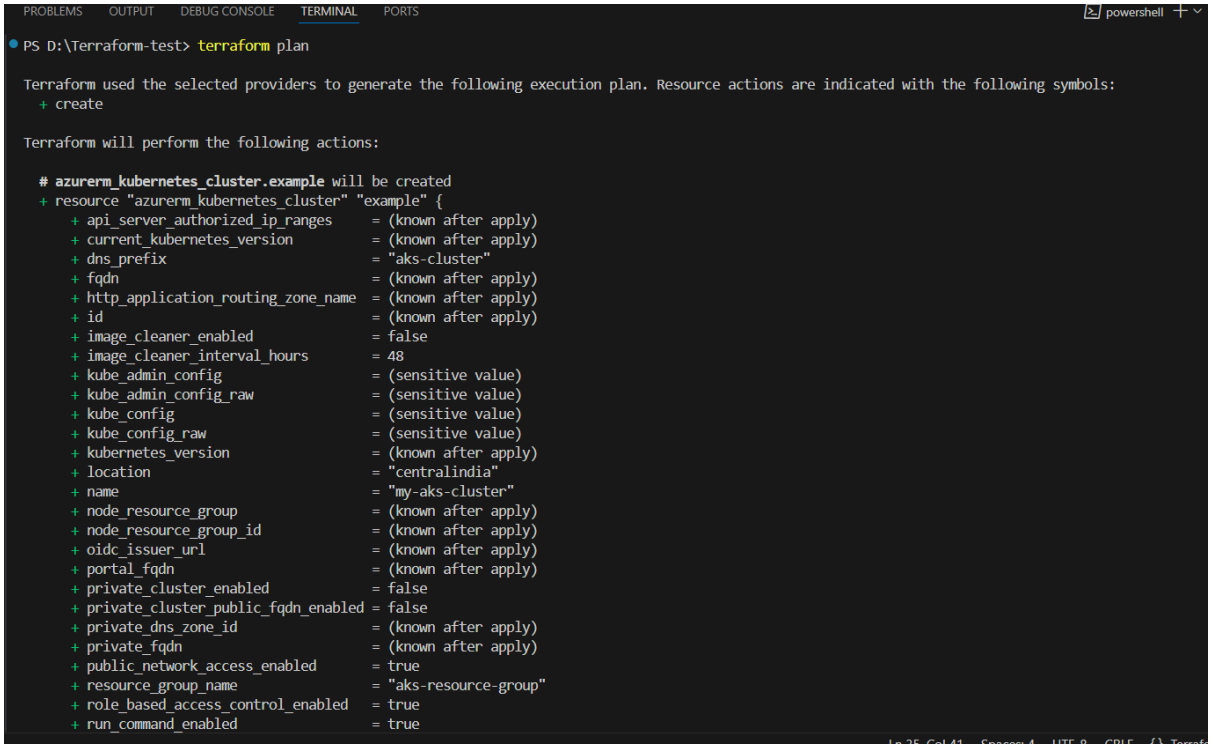


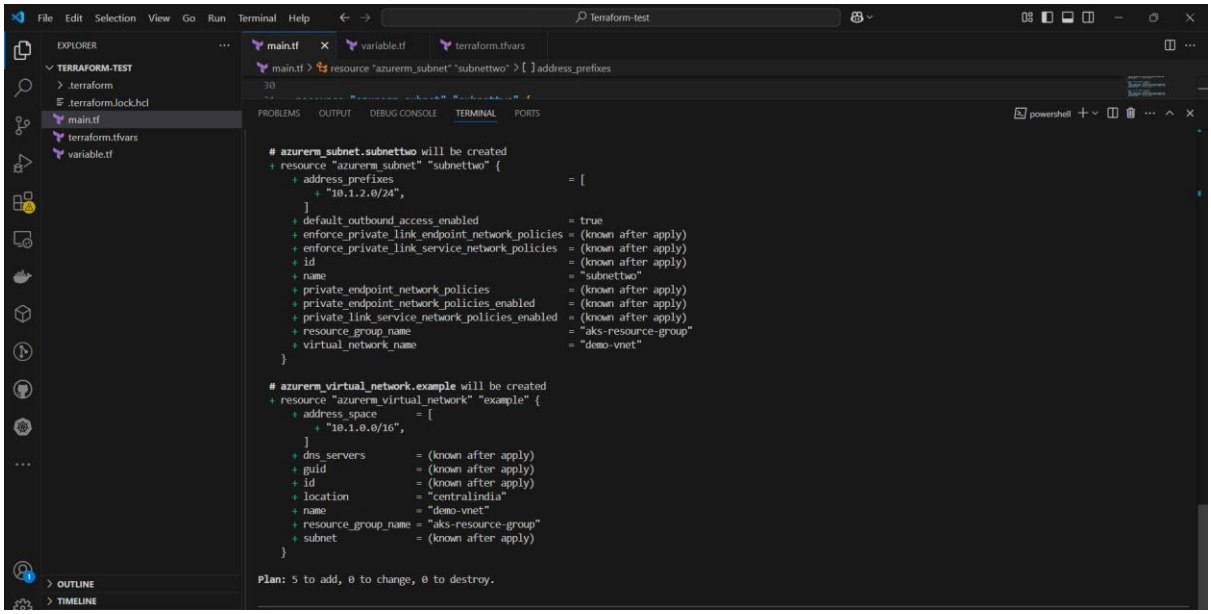
→Before Initialize the terraform commands we need to connect to azure account with **az login** command.

The **terraform init** command is used to initialize a Terraform working directory.

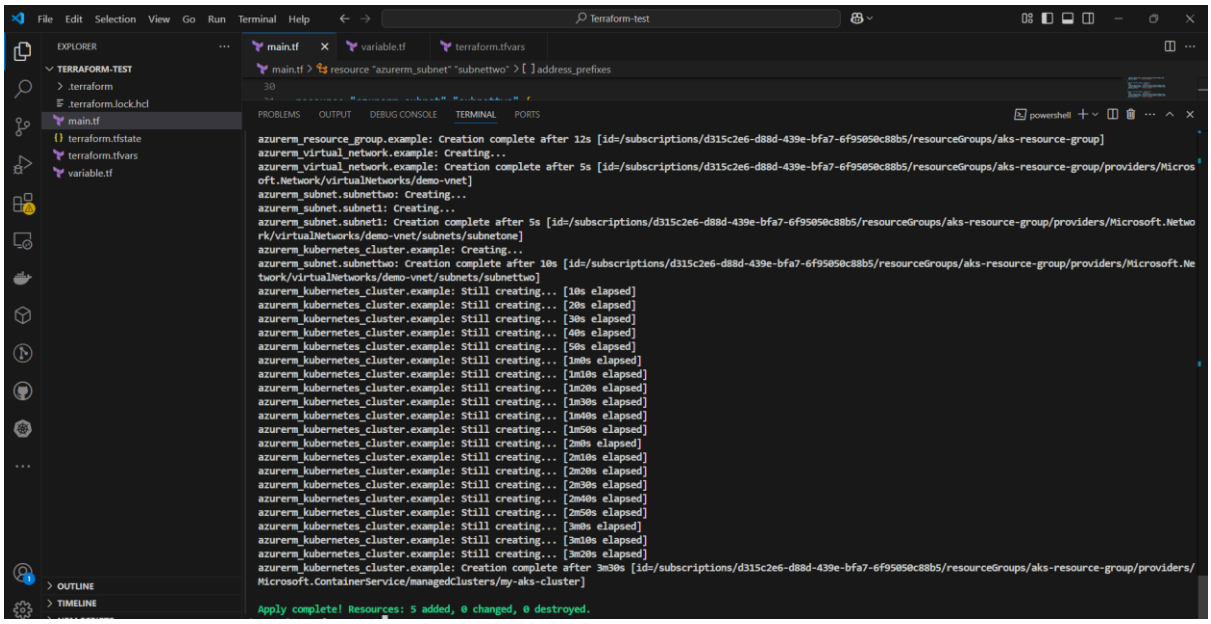


→The **terraform plan** command is used to generate an execution plan.

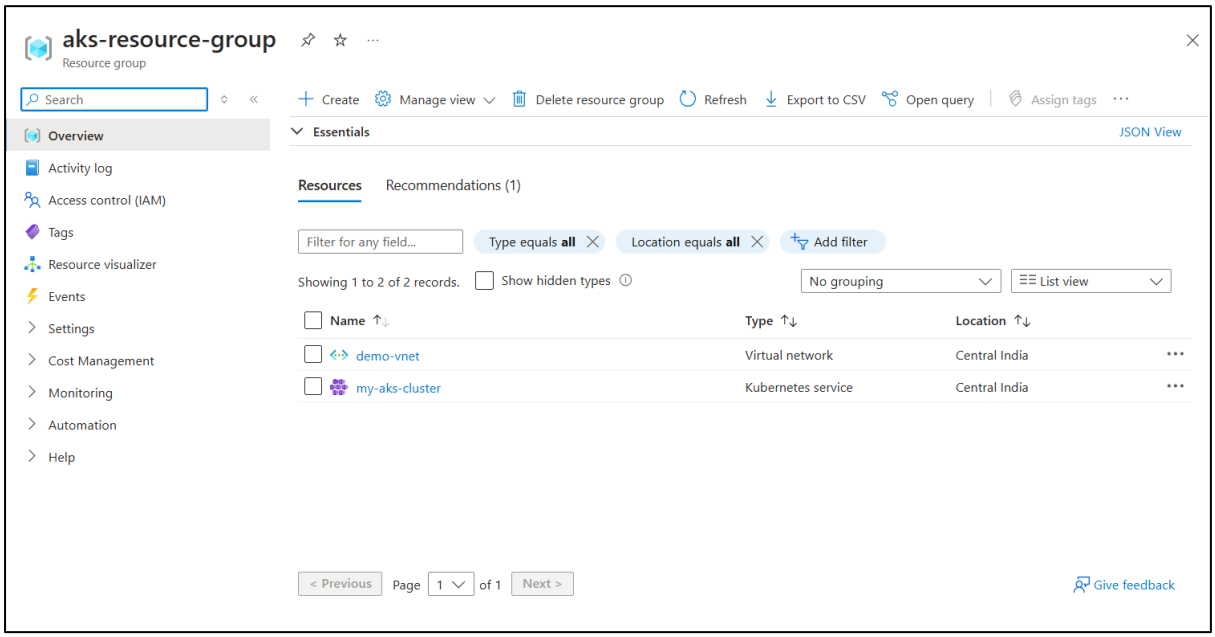




→The **terraform apply** command is used to apply changes to your infrastructure.



→verify the Resources are Created.



6. Deployment to Azure Kubernetes Service:

az account set --subscription "<SUBSCRIPTION_ID>"

az aks get-credentials --resource-group <RESOURCE_GROUP> --name <CLUSTER_NAME>

→Create a Deployment file to host the image to Azure Kubernetes service.

Deploy.yaml 1 × | service.yaml

Deploy.yaml > {} spec > {} template > {} spec > [ ] containers > {} 0 > image

io.k8s.api.apps.v1.Deployment (v1@deployment.json)

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: lok-deploy
  labels:
    app: app-dev
spec:
  replicas: 2
  selector:
    matchLabels:
      app: app-dev
  template:
    metadata:
      labels:
        app: app-dev
    spec:
      containers:
        - name: image
          image: lokeshkoagiri/simpletimeserviceflask
          imagePullPolicy: Always
          ports:
            - containerPort: 5000
```

→Run the Apply command to create a deployment to Cluster node.

```
PS C:\Users\ktssk\Desktop\aksapp> kubectl apply -f .\Deploy.yaml
deployment.apps/lok-deploy created
```

→Create a Service file to Expose the Public IP to Client with Load Balancer.

```
! Deploy.yaml 1    ! service.yaml ×
! service.yaml > {} spec > [ ] ports > {} 0 > # targetPort
    io.k8s.api.core.v1.Service (v1@service.json)
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: lok-svc
5    labels:
6      app: app-dev
7      tier: Backend
8  spec:
9    type: LoadBalancer
10   selector:
11     app: app-dev
12   ports:
13     - name: http
14       port: 5000
15       targetPort: 5000
```

→Run the Apply command for service file to Expose the Public IP.

```
PS C:\Users\ktssk\Desktop\aksapp> kubectl apply -f .\service.yaml
service/lok-svc created
```

→**kubectl get service** to view the IP address.

```
PS C:\Users\ktssk\Desktop\aksapp> kubectl get service
NAME         TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)          AGE
kubernetes   ClusterIP      10.0.0.1       <none>          443/TCP          24m
lok-svc      LoadBalancer   10.0.95.156    20.244.73.170   5000:32735/TCP   45s
```

→Browse the Public IP address to view the webpage.