Gcp practical

For creating a Vm go to service account and create a service account waith

**Step 1: Create a Service Account and Download the Key**

1. **Go to the Google Cloud Console**:
   * Open Google Cloud Console.
2. **Create a Service Account**:
   * Navigate to **IAM & Admin** > **Service Accounts**.
   * Click **Create Service Account**.
   * Provide a name (e.g., vm-creator) and a description.
   * Under **Role**, choose **Compute Admin** and any other roles that your project might need.
   * Click **Create**.
3. **Create and Download the Key**:
   * Once the service account is created, click the three dots next to it and choose **Create Key**.
   * Select **JSON** and click **Create**. This will download a JSON key file to your machine.
   * Save the file in a secure location.
4. user121@DESKTOP-JFJ21I5 MINGW64 /d/agcp
5. $ export GOOGLE\_APPLICATION\_CREDENTIALS="/c/Users/user121/Downloads/deep-gcp.json"
6. user121@DESKTOP-JFJ21I5 MINGW64 /d/agcp
7. $ echo $GOOGLE\_APPLICATION\_CREDENTIALS
8. /c/Users/user121/Downloads/deep-gcp.json
9. user121@DESKTOP-JFJ21I5 MINGW64 /d/agcp

### Step 1: Correctly Set the Environment Variable in Git Bash

Since you’re using **Git Bash** (as indicated by MINGW64), the syntax for setting environment variables is different compared to Windows Command Prompt or PowerShell.

Use the following command in **Git Bash** to set the GOOGLE\_APPLICATION\_CREDENTIALS environment variable:

bash

CopyEdit

export GOOGLE\_APPLICATION\_CREDENTIALS="/c/Users/user121/Downloads/deep-gcp.json"

Make sure to use the correct path with /c/ instead of C:\ when using Git Bash.

### Step 2: Verify the Environment Variable is Set

To check if the environment variable is correctly set, you can run:

bash

CopyEdit

echo $GOOGLE\_APPLICATION\_CREDENTIALS

This should output the path to your deep-gcp.json file. If it prints the correct path, you're good to go.

### Step 3: Run the Script Again

Once the environment variable is set, run the Python script again:

bash

CopyEdit

python create\_vm.py

or if you're using Python 3:

bash

CopyEdit

python3 create\_vm.py

import os

import google.auth

from googleapiclient.discovery import build

from googleapiclient.errors import HttpError

import time

# Set your project ID and zone here

PROJECT\_ID = 'manifest-zephyr-457115-v2'  # Updated project ID

ZONE = 'us-central1-a'  # Or another zone you prefer

INSTANCE\_NAME = 'ubuntu-vm'

IMAGE\_PROJECT = 'ubuntu-os-cloud'

IMAGE\_FAMILY = 'ubuntu-2004-lts'  # Ubuntu 20.04 LTS image

# Authenticate and create the compute client

def create\_instance():

    # Initialize the Compute Engine client

    credentials, project = google.auth.default()

    compute = build('compute', 'v1', credentials=credentials)

    # Define the machine type and image

    machine\_type = f"zones/{ZONE}/machineTypes/n1-standard-1"  # You can change the machine type here

    source\_disk\_image = f"projects/{IMAGE\_PROJECT}/global/images/family/{IMAGE\_FAMILY}"

    # Define the configuration for the VM instance

    config = {

        'name': INSTANCE\_NAME,

        'machineType': machine\_type,

        'disks': [{

            'boot': True,

            'autoDelete': True,

            'initializeParams': {

                'sourceImage': source\_disk\_image

            }

        }],

        'networkInterfaces': [{

            'network': 'global/networks/default',

            'accessConfigs': [{'type': 'ONE\_TO\_ONE\_NAT', 'name': 'External NAT'}]

        }],

        'tags': {

            'items': ['http-server', 'https-server']

        },

    }

    try:

        # Create the VM instance

        print(f"Creating VM instance {INSTANCE\_NAME}...")

        operation = compute.instances().insert(

            project=PROJECT\_ID,

            zone=ZONE,

            body=config

        ).execute()

        # Wait for the operation to complete

        wait\_for\_operation(compute, operation)

        print(f"VM instance {INSTANCE\_NAME} created successfully.")

    except HttpError as err:

        print(f"An error occurred: {err}")

        return

def wait\_for\_operation(compute, operation):

    print("Waiting for operation to complete...")

    while True:

        result = compute.zoneOperations().get(

            project=PROJECT\_ID,

            zone=ZONE,

            operation=operation['name']

        ).execute()

        if result['status'] == 'DONE':

            if 'error' in result:

                raise Exception(f"Error: {result['error']}")

            else:

                print("Operation completed successfully.")

                break

        else:

            print("Operation is still in progress...")

        time.sleep(10)

if \_\_name\_\_ == "\_\_main\_\_":

    create\_instance()

user121@DESKTOP-JFJ21I5 MINGW64 /d/agcp

$ python create\_vm.py

Creating VM instance ubuntu-vm...

Waiting for operation to complete...

Operation is still in progress...

Operation completed successfully.

VM instance ubuntu-vm created successfully.

Private VM without VPC

import os

import google.auth

from googleapiclient.discovery import build

from googleapiclient.errors import HttpError

import time

# Set your project ID and zone here

PROJECT\_ID = 'manifest-zephyr-457115-v2'  # Updated project ID

ZONE = 'us-central1-a'  # Or another zone you prefer

INSTANCE\_NAME = 'private-ubuntu-vm'

IMAGE\_PROJECT = 'ubuntu-os-cloud'

IMAGE\_FAMILY = 'ubuntu-2004-lts'  # Ubuntu 20.04 LTS image

# Authenticate and create the compute client

def create\_instance():

    # Initialize the Compute Engine client

    credentials, project = google.auth.default()

    compute = build('compute', 'v1', credentials=credentials)

    # Define the machine type and image

    machine\_type = f"zones/{ZONE}/machineTypes/n1-standard-1"  # You can change the machine type here

    source\_disk\_image = f"projects/{IMAGE\_PROJECT}/global/images/family/{IMAGE\_FAMILY}"

    # Define the configuration for the VM instance

    config = {

        'name': INSTANCE\_NAME,

        'machineType': machine\_type,

        'disks': [{

            'boot': True,

            'autoDelete': True,

            'initializeParams': {

                'sourceImage': source\_disk\_image

            }

        }],

        'networkInterfaces': [{

            'network': 'global/networks/default',

            # No accessConfigs means no public IP address

        }],

        'tags': {

            'items': ['http-server', 'https-server']

        },

    }

    try:

        # Create the VM instance

        print(f"Creating private VM instance {INSTANCE\_NAME}...")

        operation = compute.instances().insert(

            project=PROJECT\_ID,

            zone=ZONE,

            body=config

        ).execute()

        # Wait for the operation to complete

        wait\_for\_operation(compute, operation)

        print(f"Private VM instance {INSTANCE\_NAME} created successfully.")

    except HttpError as err:

        print(f"An error occurred: {err}")

        return

def wait\_for\_operation(compute, operation):

    print("Waiting for operation to complete...")

    while True:

        result = compute.zoneOperations().get(

            project=PROJECT\_ID,

            zone=ZONE,

            operation=operation['name']

        ).execute()

        if result['status'] == 'DONE':

            if 'error' in result:

                raise Exception(f"Error: {result['error']}")

            else:

                print("Operation completed successfully.")

                break

        else:

            print("Operation is still in progress...")

        time.sleep(10)

if \_\_name\_\_ == "\_\_main\_\_":

    create\_instance()

user121@DESKTOP-JFJ21I5 MINGW64 /d/agcp

$ touch private\_vm.py

user121@DESKTOP-JFJ21I5 MINGW64 /d/agcp

$ python private\_vm.py

Creating private VM instance private-ubuntu-vm...

Waiting for operation to complete...

Operation is still in progress...

Operation is still in progress...

Operation completed successfully.

Private VM instance private-ubuntu-vm created successfully.

With VPC

import os

import google.auth

from googleapiclient.discovery import build

from googleapiclient.errors import HttpError

import time

# Set your project ID, VPC, subnet, and zone here

PROJECT\_ID = 'manifest-zephyr-457115-v2'  # Your Google Cloud project ID

ZONE = 'us-central1-a'  # The zone for the VM (make sure it's within 'us-central1')

INSTANCE\_NAME = 'private-ubuntu-vm-2'  # New name for the VM to avoid conflict

IMAGE\_PROJECT = 'ubuntu-os-cloud'  # The project where the Ubuntu image resides

IMAGE\_FAMILY = 'ubuntu-2004-lts'  # The image family (Ubuntu 20.04 LTS)

VPC\_NAME = 'hu-devops-25'  # Your VPC network name

SUBNET\_NAME = 'private-subnet'  # Your private subnet name

# Authenticate and create the compute client

def create\_instance():

    # Initialize the Compute Engine client

    credentials, project = google.auth.default()

    compute = build('compute', 'v1', credentials=credentials)

    # Define the machine type and image

    machine\_type = f"zones/{ZONE}/machineTypes/n1-standard-1"  # You can change the machine type here

    source\_disk\_image = f"projects/{IMAGE\_PROJECT}/global/images/family/{IMAGE\_FAMILY}"

    # Define the network and subnet configuration

    network\_interface = {

        'network': f'global/networks/{VPC\_NAME}',  # Specify the VPC network

        'subnetwork': f'regions/us-central1/subnetworks/{SUBNET\_NAME}',  # Specify the correct region 'us-central1' and subnet

        # No accessConfigs means no public IP address (this creates a private VM)

    }

    # Define the configuration for the VM instance

    config = {

        'name': INSTANCE\_NAME,

        'machineType': machine\_type,

        'disks': [{

            'boot': True,

            'autoDelete': True,

            'initializeParams': {

                'sourceImage': source\_disk\_image

            }

        }],

        'networkInterfaces': [network\_interface],  # Attach the VM to the private network

        'tags': {

            'items': ['http-server', 'https-server']  # Tags for access control (e.g., firewall rules)

        },

    }

    try:

        # Create the VM instance

        print(f"Creating private VM instance {INSTANCE\_NAME}...")

        operation = compute.instances().insert(

            project=PROJECT\_ID,

            zone=ZONE,

            body=config

        ).execute()

        # Wait for the operation to complete

        wait\_for\_operation(compute, operation)

        print(f"Private VM instance {INSTANCE\_NAME} created successfully.")

    except HttpError as err:

        print(f"An error occurred: {err}")

        return

def wait\_for\_operation(compute, operation):

    print("Waiting for operation to complete...")

    while True:

        result = compute.zoneOperations().get(

            project=PROJECT\_ID,

            zone=ZONE,

            operation=operation['name']

        ).execute()

        if result['status'] == 'DONE':

            if 'error' in result:

                raise Exception(f"Error: {result['error']}")

            else:

                print("Operation completed successfully.")

                break

        else:

            print("Operation is still in progress...")

        time.sleep(10)

if \_\_name\_\_ == "\_\_main\_\_":

    create\_instance()

user121@DESKTOP-JFJ21I5 MINGW64 /d/agcp

$ python private\_1.py

Creating private VM instance private-ubuntu-vm-2...

Waiting for operation to complete...

Operation is still in progress...

Operation completed successfully.

Private VM instance private-ubuntu-vm-2 created successfully.

user121@DESKTOP-JFJ21I5 MINGW64 /d/agcp

CREAQTING APP ENGINE

import google.auth

from googleapiclient.discovery import build

from googleapiclient.errors import HttpError

import subprocess

import time

# Set your project ID, VPC, subnet, and region here

PROJECT\_ID = 'manifest-zephyr-457115-v2'  # Your Google Cloud project ID

REGION = 'us-central1'  # The region where your VPC and subnet reside

ZONE = 'us-central1-a'  # The zone for the VM (make sure it's within 'us-central1')

INSTANCE\_NAME = 'private-ubuntu-vm'  # Name of the VM (for reference)

VPC\_NAME = 'hu-devops-25'  # Your VPC network name

SUBNET\_NAME = 'private-subnet'  # Your private subnet name

CONNECTOR\_NAME = 'my-vpc-connector'  # The name for the VPC connector

APP\_ENGINE\_ENV = 'flex'  # The App Engine flexible environment

# Authenticate and create the vpcaccess client

def create\_vpc\_connector():

    # Initialize the VPC Access API client

    credentials, project = google.auth.default()

    vpcaccess = build('vpcaccess', 'v1', credentials=credentials)

    try:

        # Create the VPC Access Connector

        print(f"Creating VPC Access Connector '{CONNECTOR\_NAME}'...")

        connector\_config = {

            'name': f"projects/{PROJECT\_ID}/locations/{REGION}/connectors/{CONNECTOR\_NAME}",

            'network': f'projects/{PROJECT\_ID}/global/networks/{VPC\_NAME}',

            'subnet': f'projects/{PROJECT\_ID}/regions/{REGION}/subnetworks/{SUBNET\_NAME}',  # Full path for subnet

        }

        operation = vpcaccess.projects().locations().connectors().create(

            parent=f"projects/{PROJECT\_ID}/locations/{REGION}",

            body=connector\_config

        ).execute()

        # Wait for the operation to complete

        wait\_for\_operation(vpcaccess, operation)

        print(f"VPC Access Connector '{CONNECTOR\_NAME}' created successfully.")

    except HttpError as err:

        print(f"An error occurred: {err}")

        return

def wait\_for\_operation(vpcaccess, operation):

    """Wait for the operation to complete."""

    print("Waiting for operation to complete...")

    while True:

        result = vpcaccess.projects().locations().operations().get(

            name=operation['name']

        ).execute()

        if result['done']:

            if 'error' in result:

                raise Exception(f"Error: {result['error']}")

            else:

                print("Operation completed successfully.")

                break

        else:

            print("Operation is still in progress...")

        time.sleep(10)

def deploy\_app\_engine():

    """Deploy the app to App Engine."""

    try:

        print("Deploying the application to App Engine...")

        # Deploy the app using `gcloud app deploy`

        subprocess.run(['gcloud', 'app', 'deploy'], check=True)

        print("App Engine deployment completed successfully.")

    except subprocess.CalledProcessError as e:

        print(f"Deployment failed: {e}")

if \_\_name\_\_ == "\_\_main\_\_":

    # Step 1: Create the VPC Access Connector

    create\_vpc\_connector()

    # Step 2: Deploy the app to App Engine

    deploy\_app\_engine()