# **Review and finalize**

**Multi-Cloud Deployment Guide: AWS, Azure, and GCP**

**1. AWS: Deploying an EC2 Instance Using Python**

**Prerequisites**

* AWS account with IAM permissions
* AWS CLI installed & configured (aws configure)
* Boto3 library installed (pip install boto3)

**Python Script: Deploy EC2 Instance**

import boto3

# Initialize EC2 client

ec2 = boto3.client('ec2', region\_name='us-east-1') # Change the region as needed

# Define instance parameters

instance\_params = {

"ImageId": "ami-0c55b159cbfafe1f0", # Replace with a valid AMI ID

"InstanceType": "t2.micro", # Free-tier eligible

"MinCount": 1,

"MaxCount": 1,

"KeyName": "my-key-pair", # Replace with your key pair name

"SecurityGroups": ["default"],

}

# Launch EC2 instance

response = ec2.run\_instances(\*\*instance\_params)

instance\_id = response['Instances'][0]['InstanceId']

print(f"EC2 Instance {instance\_id} created successfully!")

**Connect to EC2**

ssh -i my-key-pair.pem ec2-user@<PUBLIC\_IP>

**2. Azure: Deploying a Virtual Machine Using Python**

**Prerequisites**

* Azure account with IAM permissions
* Azure CLI installed (az login)
* azure-mgmt-compute Python SDK installed (pip install azure-mgmt-compute)

**Python Script: Deploy Azure VM**

from azure.identity import DefaultAzureCredential

from azure.mgmt.compute import ComputeManagementClient

subscription\_id = "your-subscription-id"

resource\_group = "myResourceGroup"

vm\_name = "myVM"

location = "eastus"

# Authenticate

credential = DefaultAzureCredential()

compute\_client = ComputeManagementClient(credential, subscription\_id)

# Create VM (Simplified example)

vm\_parameters = {

"location": location,

"hardware\_profile": {"vm\_size": "Standard\_DS1\_v2"},

"storage\_profile": {

"image\_reference": {"publisher": "Canonical", "offer": "UbuntuServer", "sku": "18.04-LTS", "version": "latest"}

},

"os\_profile": {

"computer\_name": vm\_name,

"admin\_username": "azureuser",

"admin\_password": "SecurePassword123!"

},

}

compute\_client.virtual\_machines.begin\_create\_or\_update(resource\_group, vm\_name, vm\_parameters)

print("Azure VM deployed successfully!")

**Connect to Azure VM**

ssh azureuser@<PUBLIC\_IP>

**3. GCP: Deploying a Compute Engine Instance Using Python**

**Prerequisites**

* GCP account with IAM permissions
* Google Cloud SDK installed (gcloud auth login)
* google-cloud-compute library installed (pip install google-cloud-compute)

**Python Script: Deploy GCP Compute Engine Instance**

from google.cloud import compute\_v1

project\_id = "your-gcp-project"

zone = "us-central1-a"

instance\_name = "my-gcp-instance"

# Initialize client

instance\_client = compute\_v1.InstancesClient()

# Define instance config

instance = compute\_v1.Instance(

name=instance\_name,

machine\_type=f"zones/{zone}/machineTypes/e2-micro",

disks=[

compute\_v1.AttachedDisk(

auto\_delete=True,

boot=True,

initialize\_params=compute\_v1.AttachedDiskInitializeParams(

source\_image="projects/debian-cloud/global/images/family/debian-11"

),

)

],

network\_interfaces=[compute\_v1.NetworkInterface(name="global/networks/default")],

)

# Create instance

operation = instance\_client.insert(project=project\_id, zone=zone, instance\_resource=instance)

print("GCP Compute Engine instance created successfully!")

**Connect to GCP VM**

gcloud compute ssh my-gcp-instance --zone=us-central1-a

**Final Thoughts & Best Practices**

* Use IAM roles and security groups to manage access.
* Always use SSH keys or managed authentication.
* Automate deployments using Terraform or Ansible for scalability.
* Monitor and optimize cloud usage to reduce costs.