DEPLOY A WEB APPLICATION ON THE CLOUD

(Write a Python Flask application and deploy it on your cloud VM. Configure the firewall to allow HTTP traffic.)

INTRODUCTION:

This Proof of Concept (PoC) demonstrates how to deploy a simple Python Flask web application on a cloud-based Virtual Machine (VM). The goal is to set up a basic web service, make it accessible over the internet, and configure necessary security settings, such as firewall rules.

The PoC involves:

1. Setting up a cloud VM (AWS, GCP, Azure, etc.).
2. Installing dependencies and writing a basic Flask web application.
3. Running the application and exposing it to external users.
4. Configuring firewall rules to allow HTTP traffic.
5. (Optional) Enhancing the deployment with a process manager (e.g., Gunicorn) and a reverse proxy (e.g., Nginx).

IMPORTANCE:

1. Cloud Deployment Basics - It helps in understanding how cloud instances work, including networking, security, and resource management.

2. Real-World Application Hosting - Deploying a Flask application on a cloud VM mimics real-world scenarios where web services are hosted on cloud platforms.

3. Scalability & Performance Testing - Once deployed, this PoC can be extended to test load balancing, auto-scaling, and high availability solutions, preparing the system for production-level traffic.

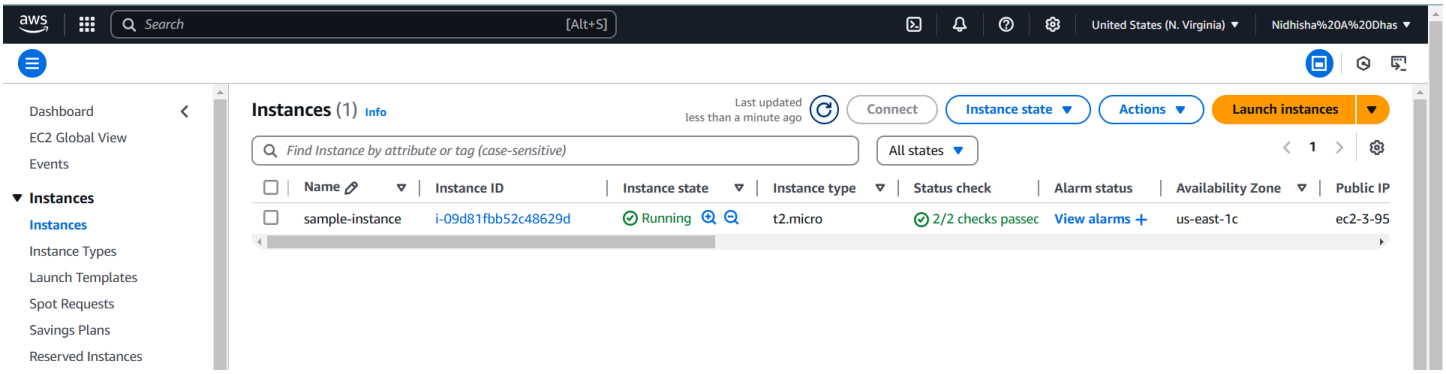
4. Security & Firewall Configuration - By configuring the firewall to allow HTTP traffic while restricting other ports, this PoC highlights the importance of cloud security practices.

5. Preparation for Advanced Deployments - This experiment sets the stage for more advanced deployment strategies, such as containerization with Docker, Kubernetes orchestration, and CI/CD automation.

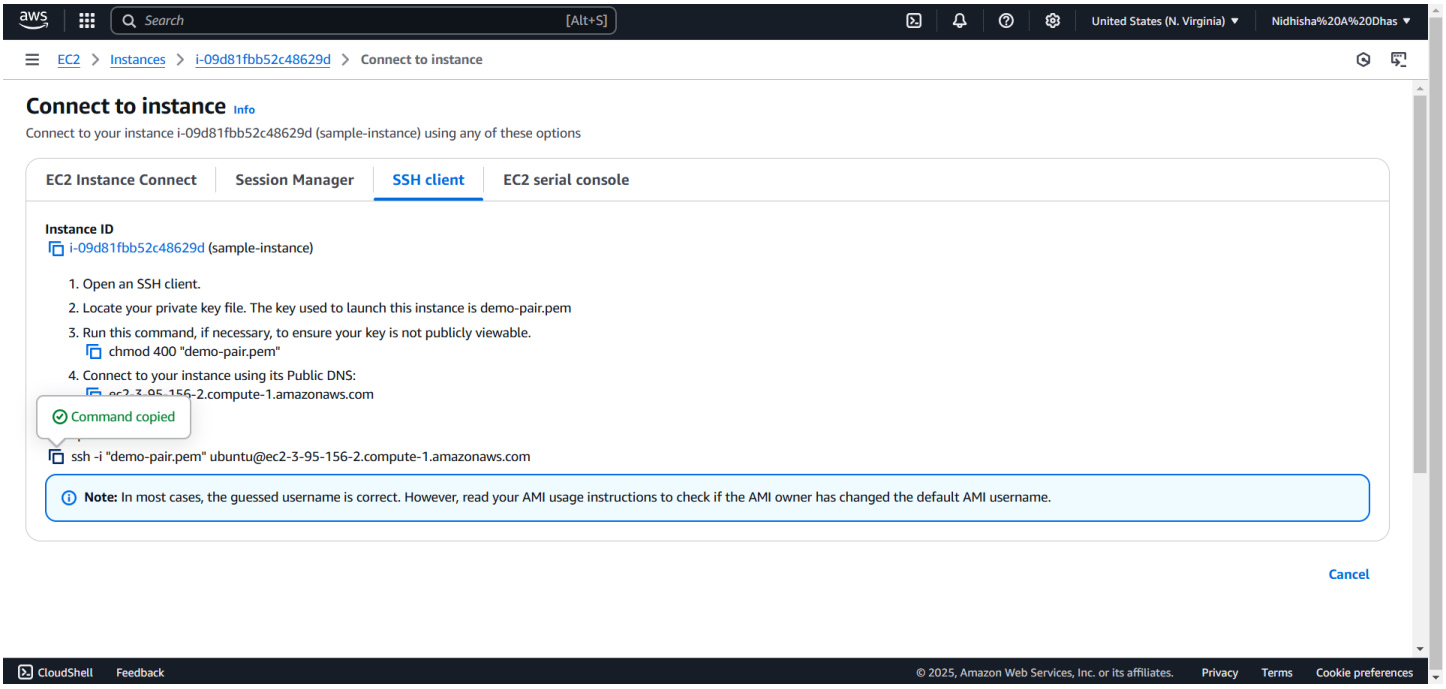
STEP BY STEP OVERVIEW:

Step 1: CREATE AN EC2 INSTANCE

- Login into your AWS console and navigate to the EC2 dashboard.
- Click on ‘launch instance’ and create your own instance. Ensure your cloud VM is running a Linux distribution (Ubuntu, CentOS, etc.).



- Connect your EC2 instance and copy the ssh command.



Step 2: INSTALLING AND SET-UP

- Open your PowerShell, change the file directory.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Aruldhas> cd downloads
PS C:\Users\Aruldhas\downloads> ssh -i "demo-pair.pem" ubuntu@ec2-3-95-156-2.compute-1.amazonaws.com
The authenticity of host 'ec2-3-95-156-2.compute-1.amazonaws.com (3.95.156.2)' can't be established.
ED25519 key fingerprint is SHA256:HThQW9xeM1SF84jEvhtQ2tc+mXvZPFutHU44KvuJ6ww.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-95-156-2.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-1021-aws x86_64)
```

- Update & upgrade the package.

```
ubuntu@ip-172-31-92-49:~$ sudo apt update
```

```
ubuntu@ip-172-31-92-49:~$ sudo apt upgrade
```

- Install python3 and pip.

```
ubuntu@ip-172-31-92-49:~$ sudo apt install python3 python3-pip -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
python3 is already the newest version (3.12.3-0ubuntu2).
python3-pip is already the newest version (24.0+dfsg-1ubuntu1.1).
0 upgraded, 0 newly installed, 0 to remove and 6 not upgraded.
ubuntu@ip-172-31-92-49:~$ sudo apt install python3-venv -y
```

- Install Flask.

```
ubuntu@ip-172-31-92-49:~$ python3 -m venv flaskenv
ubuntu@ip-172-31-92-49:~$ source flaskenv/bin/activate
(flaskenv) ubuntu@ip-172-31-92-49:~$ pip install flask
```

- Create a directory for your app and create a file called ‘app.py’.

```
(flaskenv) ubuntu@ip-172-31-92-49:~$ mkdir ~/flask_app
(flaskenv) ubuntu@ip-172-31-92-49:~$ cd ~/flask_app
(flaskenv) ubuntu@ip-172-31-92-49:~/flask_app$ nano app.py
(flaskenv) ubuntu@ip-172-31-92-49:~/flask_app$
```

- This will open an editor. Write the following code and click enter.

- Now, exit the virtual environment.
- Then, add your virtual environment's python path and run the application.

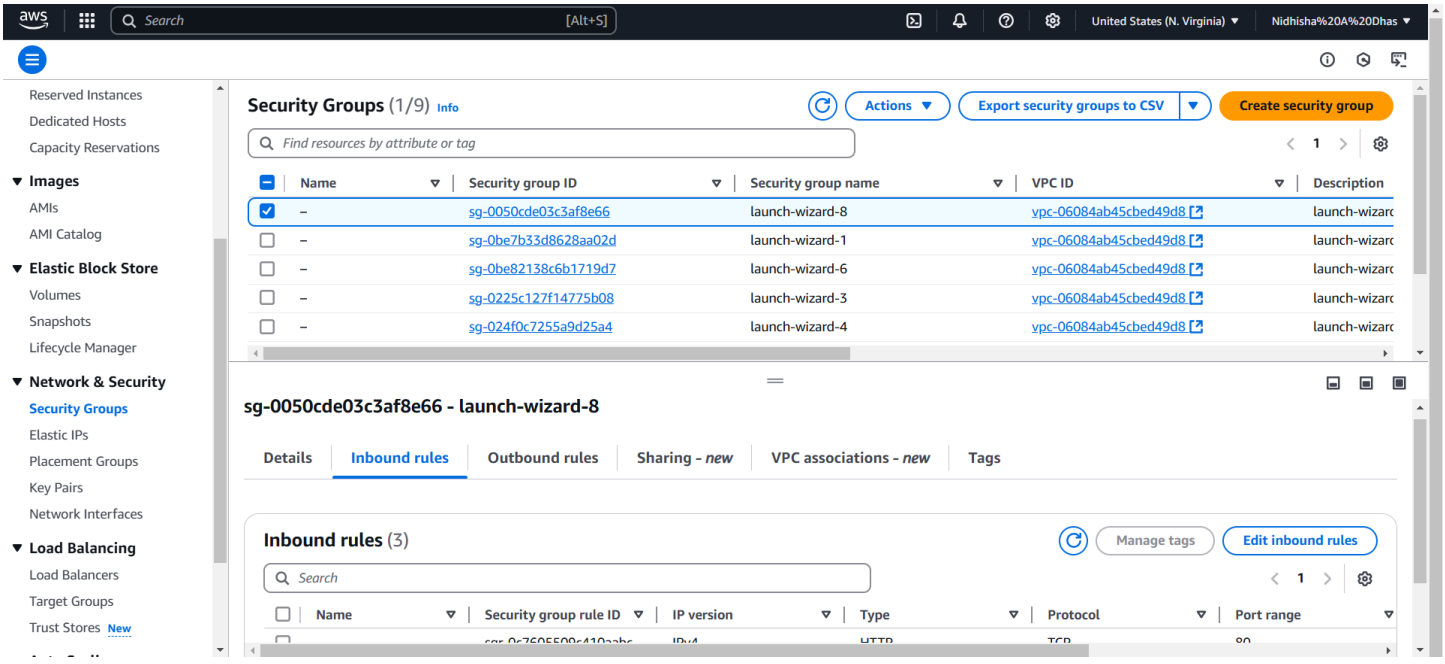
```
(flaskenv) ubuntu@ip-172-31-92-49:~$ mkdir ~/flask_app
(flaskenv) ubuntu@ip-172-31-92-49:~$ cd ~/flask_app
(flaskenv) ubuntu@ip-172-31-92-49:~/flask_app$ nano app.py
(flaskenv) ubuntu@ip-172-31-92-49:~/flask_app$ deactivate
ubuntu@ip-172-31-92-49:~/flask_app$ source~/flaskenv/bin/activate
-bash: source~/flaskenv/bin/activate: No such file or directory
ubuntu@ip-172-31-92-49:~/flask_app$ source ~/flaskenv/bin/activate
(flaskenv) ubuntu@ip-172-31-92-49:~/flask_app$ pip install flask
```

- Your Flask is now running!

```
(flaskenv) ubuntu@ip-172-31-92-49:~/flask_app$ sudo ~/flaskenv/bin/python app.py
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:80
 * Running on http://172.31.92.49:80
Press CTRL+C to quit
```

Step 3: EDIT INBOUND RULES

- Go to the EC2 dashboard, > Instances
- Find your security group attached to it in 'network and Security' section.
- Under the inbound rules, ensure that there is a rule for HTTP(port 80).



Step 4: TESTING AND ACCESSING

- Open your browser to navigate to 'http:// <instance-public-IP>/'

CONCLUSION:

By completing this PoC, you will be able to:

- Launch and configure Ec2 instance with Ubuntu as the OS.
- Install and configure Python for the flask framework.
- Write a simple flask application code.
- Host and access the flask web application on the EC2 instance.