Placement Empowerment Program

Cloud Computing and DevOps Centre

Deploy a Web Application on the CloudWrite a Python Flask application and deploy it on your cloud VM. Configure the firewall to allow HTTP traffic.

Name: Mary Edlyn A

Department: CSE

Introduction

Cloud computing has revolutionized the way applications are developed and deployed, offering scalability, flexibility, and cost- effectiveness. This PoC focuses on deploying a Python-based Flask web application on an AWS EC2 instance. Flask, a lightweight web framework, is ideal for building simple yet powerful web applications. Through this project, you will learn how to set up a virtual machine in AWS, configure it, and deploy a web application, making it accessible to users globally.

Overview

In this project, a Flask application is developed and deployed on an Amazon EC2 instance. The application runs on a cloud-hosted Linux server with an accessible HTTP endpoint. The steps include:

1. Launching an EC2 instance.
2. Configuring the instance environment (Python, Flask, and dependencies).
3. Writing a Flask web application.
4. Setting up the firewall to allow HTTP traffic.
5. Testing the application on a browser.

The PoC demonstrates a simple yet effective way to understand deploying web applications in a cloud environment.

Objectives

1. **Understand Flask Framework**: Learn the basics of Flask and how to write a simple web application.

2. **Cloud Deployment**: Gain hands-on experience deploying an application on AWS EC2.

3. **Security Configuration**: Configure inbound rules in AWS to allow HTTP traffic securely.

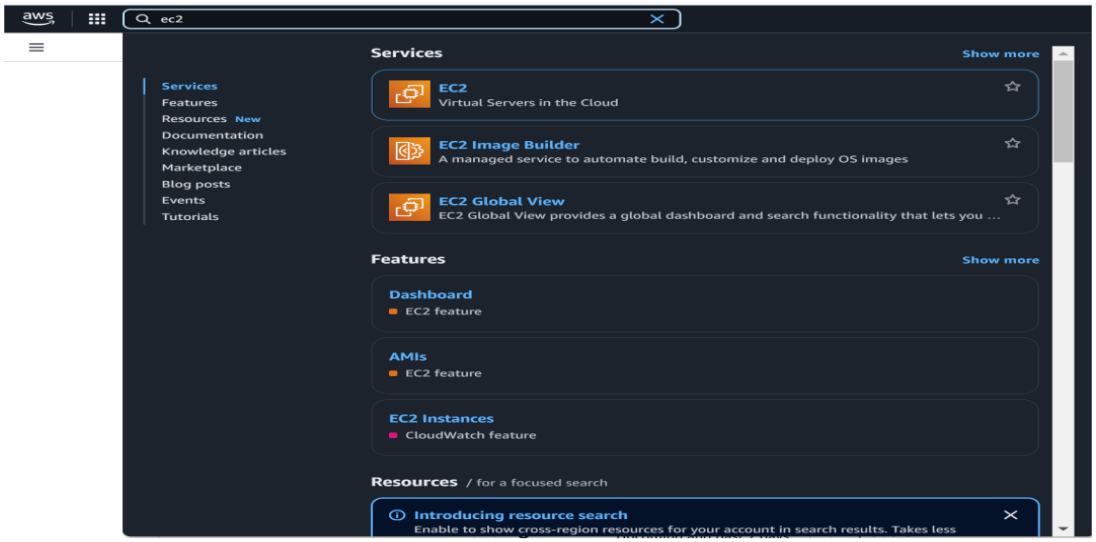
4. **Application Accessibility**: Ensure the application is accessible globally via a public IP.

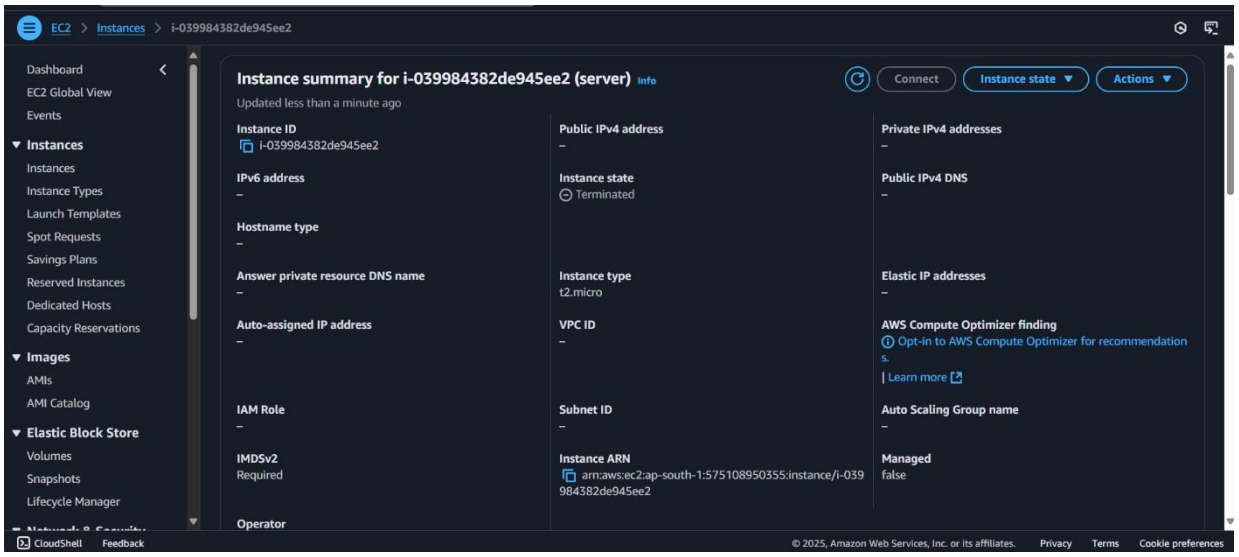
5. **Real-World Skills**: Develop skills in cloud computing and web application deployment

Step-by-Step Overview

1. Create a EC2 instance

- Go to AWS Management Console.
- On the EC2 Dashboard, click on **Launch Instances** and enter a name for your instance (e.g., "Flask Server") and select Ubuntu as OS and create a key pair. Leave other settings as default and Click **Launch Instance**.





2. Connect into the instance

~ Click the 'Connect' option on your launched instance, go to the SSH client section, and copy the command provided under the 'Example' section.

~ Open PowerShell, navigate to the 'Downloads' directory where the downloaded key pair is located using the **cd Downloads** command

~Paste the command copied from the EC2 Connect's SSH client section, replace the key pair name with your downloaded key (e.g., new.pem), press Enter, and type 'yes' when prompted.

```
PS C:\Users\nandh\Downloads> ssh -i "keypair.pem" ubuntu@ec2-13-201-226-24.ap-south-1.compute.amazonaws.com
The authenticity of host 'ec2-13-201-226-24.ap-south-1.compute.amazonaws.com (13.201.226.24)' can't be established.
ED25519 key fingerprint is SHA256:emWKmasVI9JnicMX+6IwmVq86BbA+jH4BAwXFTsUZ1M.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
```

3. Update the Package List :

```
ubuntu@ip-172-31-2-41:~$ sudo apt-get update
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
```

4. Install Python3 and pip

```
ubuntu@ip-172-31-2-41:~$ sudo apt-get install python3 python3-pip -y
```

5. Install Virtual Environment Tools : This helps keep your app's dependencies separate.

```
ubuntu@ip-172-31-2-41:~$ sudo apt-get install python3-venv -y
```

6. Create and Activate a Virtual Environment and install Flask

```
ubuntu@ip-172-31-2-41:~$ python3 -m venv flaskenv
ubuntu@ip-172-31-2-41:~$ source/flaskenv/bin/activate
-bash: source/flaskenv/bin/activate: No such file or directory
ubuntu@ip-172-31-2-41:~$ python3 -m venv flaskenv
ubuntu@ip-172-31-2-41:~$ source flaskenv/bin/activate
(flaskenv) ubuntu@ip-172-31-2-41:~$ pip install Flask
```

7. Create a Directory for Your App and Create a file called app.py using a text editor (like nano).

```
(flaskenv) ubuntu@ip-172-31-2-41:~$ mkdir ~/flask_app
(flaskenv) ubuntu@ip-172-31-2-41:~$ cd ~/flask_app
(flaskenv) ubuntu@ip-172-31-2-41:~/flask_app$ nano app.py
```

8. Write this code into the editor and press **Ctrl + O** (to write out) and then **Enter**, then **Ctrl + X** to exit.


```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def home():
        return "Hello, Cloud!"
if __name__ == '__main__':
        app.run(host='0.0.0.0', port=80)
```

9. Exit the virtual environment

```
(flaskenv) ubuntu@ip-172-31-2-41:~/flask_app$ deactivate
```

10. Add your virtual environment's Python path to the sudo command and Run the application using the virtual environment's Python.

```
ubuntu@ip-172-31-2-41:~/flask_app$ source ~/flaskenv/bin/activate
(flaskenv) ubuntu@ip-172-31-2-41:~/flask_app$ pip install Flask
```

11. Your Flask app is now running

```
(flaskenv) ubuntu@ip-172-31-2-41:~/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 producti
on WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:80
 * Running on http://172.31.2.41:80
Press CTRL+C to quit
```

12. Go to the **EC2 Dashboard** > **Instances**.

- Find your instance and note the **Security Group** attached to it. Navigate to **Security** -

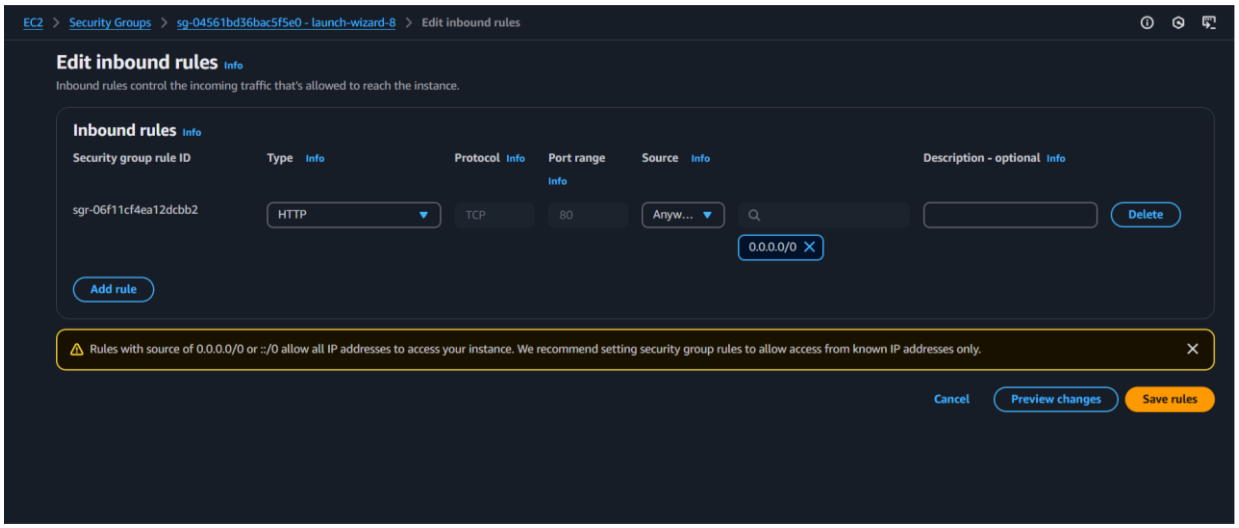
- **Groups** under the **Network & Security** section.

- Select the Security Group associated with your EC2 instance.

- Under the **Inbound Rules** tab, ensure there is a rule for **HTTP (port 80)**
- **Type:** HTTP **Protocol:**

- **Port Range:** 80

- **Source:** Anywhere (0.0.0.0/0, ::/0)

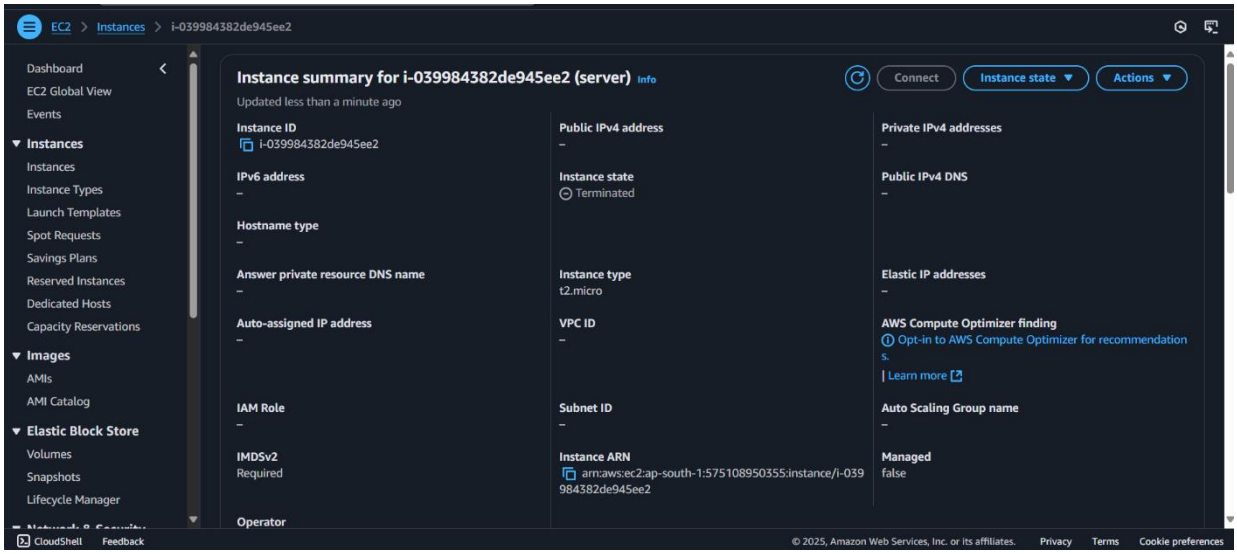


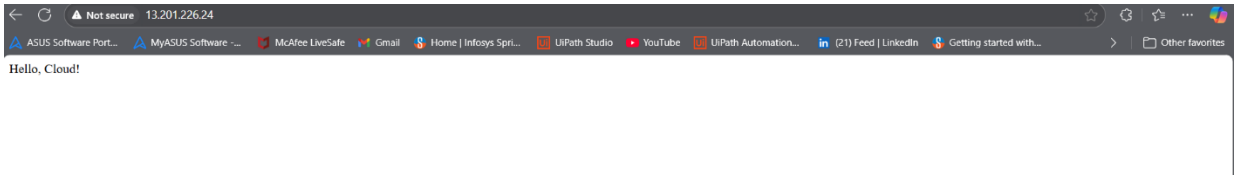
13. Open your browser and navigate to:

http://<Your-Instance-Public-IP>/

Replace <Your-Instance-Public-IP> with the Public IPv4 address of your EC2 instance (e.g., http://54.123.45.67/).

Public IPv4 address can be found in your Ec2 instance dashboard.





Outcome

By completing this PoC of deploying a Flask web application using an EC2 instance, you will:

1. Write a simple Flask application (app.py) that displays a message when accessed through a web browser.

2. Host the Flask web application on the EC2 instance and configure it to allow HTTP traffic by updating the security group rules.

3. Access your Flask web application live on the web using the EC2 instance's Public IPv4 DNS or IP address.