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:Vishnu Vardan E 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.

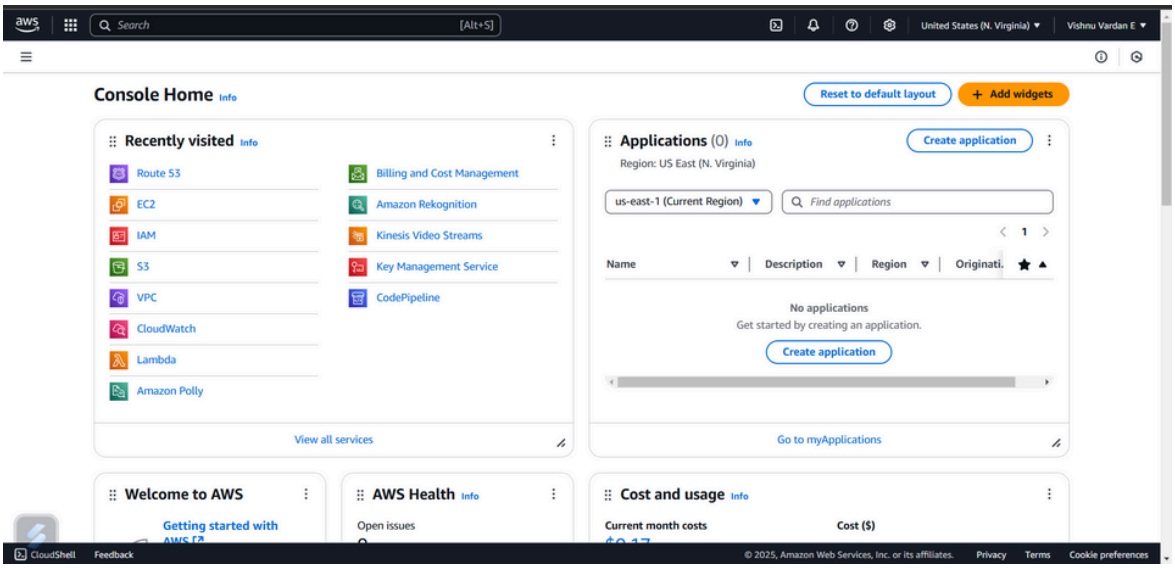
Importance

1. Practical Exposure: Provides real-world experience in deploying applications to the cloud, an essential skill in modern IT infrastructure.
2. Skill Development: Improves your understanding of cloud services, virtual machines, and web development.
3. Scalability: Demonstrates how applications can be deployed and scaled easily using cloud infrastructure.
4. Career Advancement: Builds foundational knowledge in cloud computing, a highly sought-after skill in the tech industry.
5. Problem-Solving: Encourages troubleshooting skills by resolving deployment issues and configuring environments.

Step-by-Step Overview

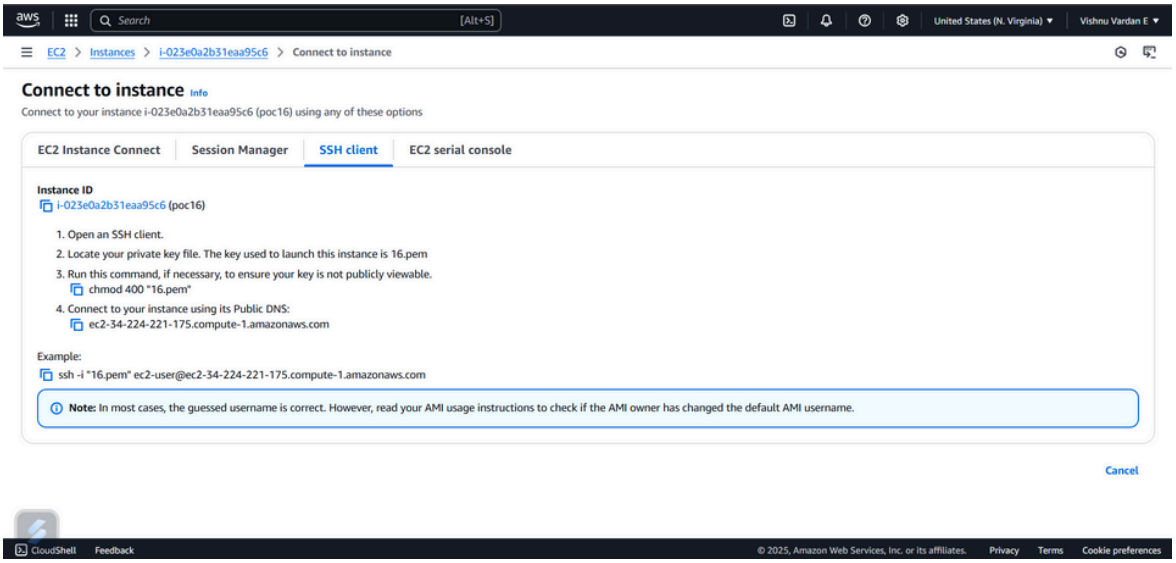
Step 1:

1. Go to AWS Management Console.
2. Enter your username and password to log in.



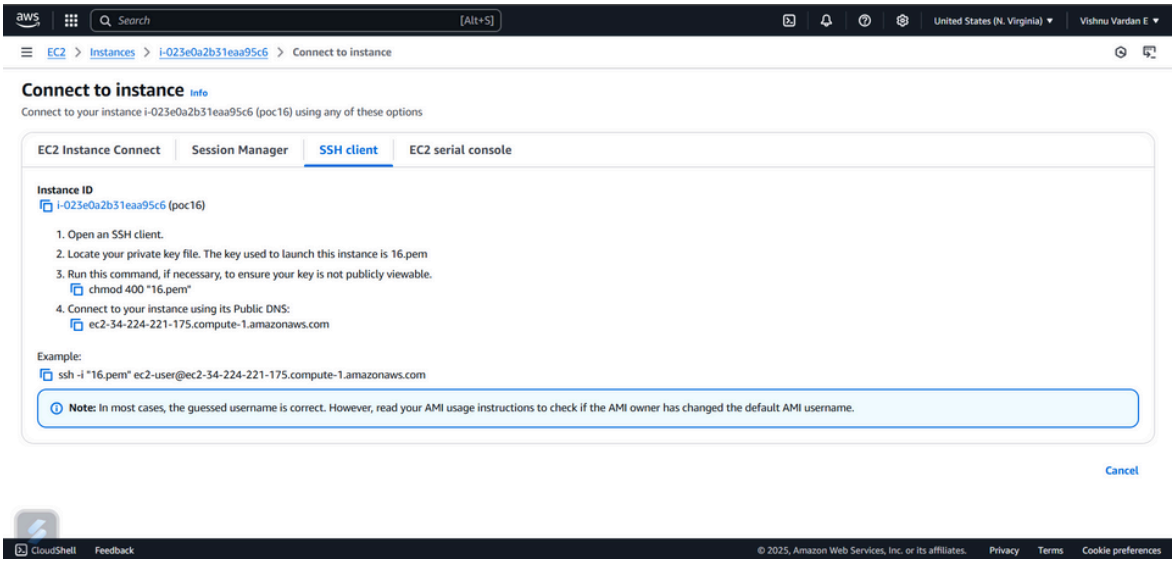
Step 2:

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.



Step 3:

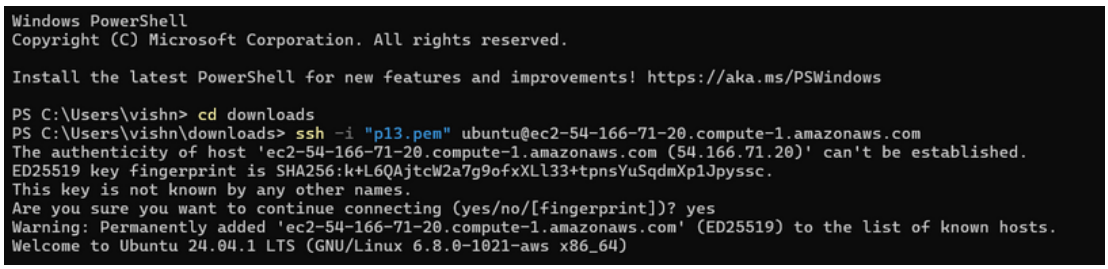
Click the 'Connect' option on your launched instance, go to the SSH client section, and copy the command provided under the 'Example' section.



Step 4:

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.

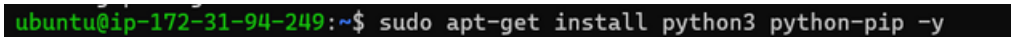


Step 5:

Update the Package List :

Step 6:

Install Python3 and pip



Step 7:

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

Step 8:

Create and Activate a Virtual Environment and install Flask

```
ubuntu@ip-172-31-94-249:~$ python3 -m venv flaskenv
ubuntu@ip-172-31-94-249:~$ source flaskenv/bin/activate
(flaskenv) ubuntu@ip-172-31-94-249:~$ pip install Flask
```

Step 9:

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

```
(flaskenv) ubuntu@ip-172-31-94-249:~$ mkdir ~/falsk_app
(flaskenv) ubuntu@ip-172-31-94-249:~$ mkdir ~/flask_app
(flaskenv) ubuntu@ip-172-31-94-249:~$ cd~/flask_app
```

Step 10:

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

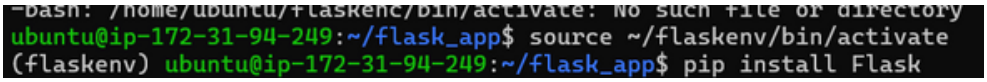
```
 GNU nano 7.2                                  app.py *
from flask import Flask
app = Flask(__name__)

@app.route('/')
def home():
    return "Hello, world! Welcome to my Flask app on AWS!"

if __name__ == '__main__':
    # Listen on all interfaces so that the app can be reached externally
    app.run(host='0.0.0.0', port=80)

Save modified buffer?
 Y Yes
 N No           ^C Cancel
```

Step 11:

Exit the virtual environment:

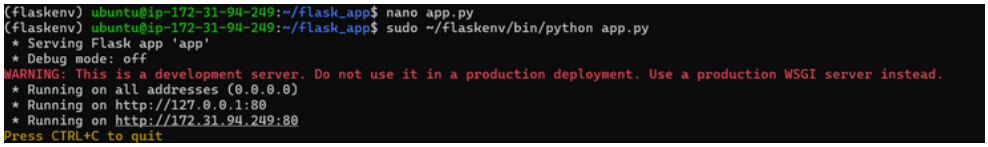
Step 12:

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



Step 13:

Your Flask app is now running!



Step 14:

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):

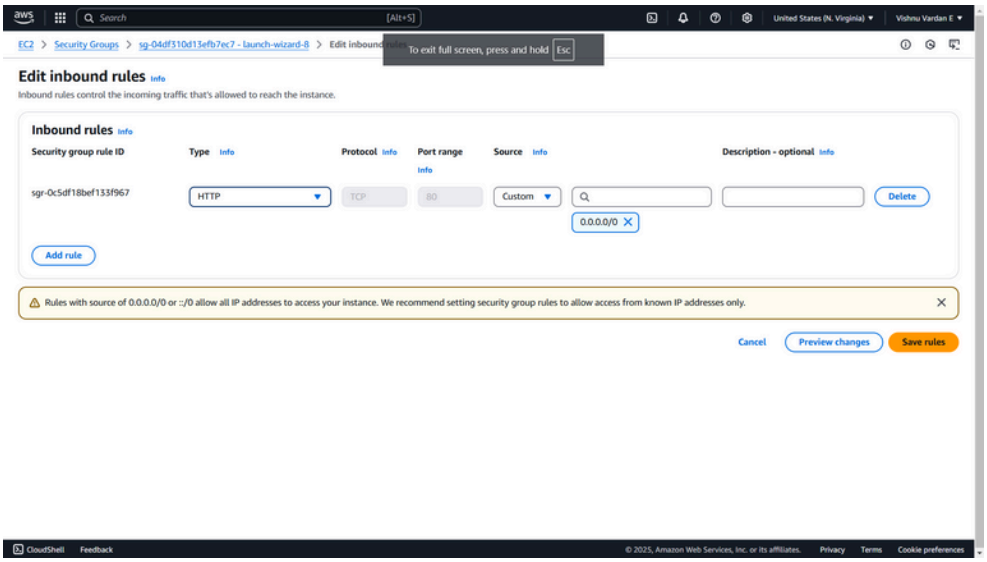
Type: HTTP

Protocol: TCP

Port Range: 80

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

If there isn't an HTTP rule, click Edit inbound rules and add it.



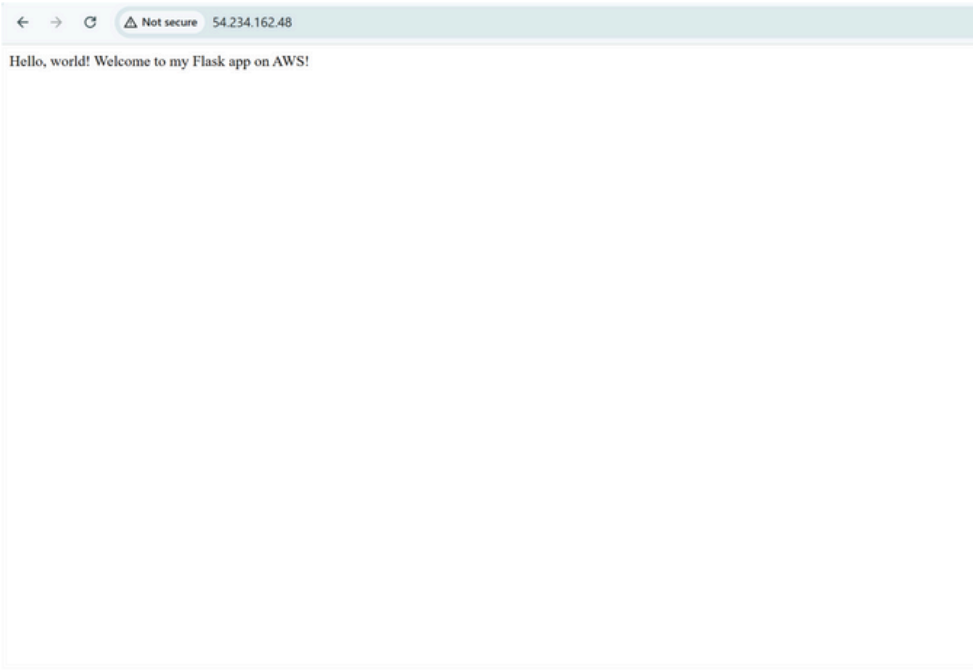
Step 15:

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. Launch and configure an EC2 instance with Ubuntu as the operating system.
2. Install and configure the necessary Python environment and dependencies for the Flask framework.
3. Write a simple Flask application (app.py) that displays a message when accessed through a web browser.
4. Host the Flask web application on the EC2 instance and configure it to allow HTTP traffic by updating the security group rules.
5. Access your Flask web application live on the web using the EC2 instance's Public IPv4 DNS or IP address.