Placement Empowerment Program

Cloud Computing and DevOps Centre

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.

Name: Nantha Krishnan.G.K.

Department: AML

St. JOSEPH'S COLLEGE OF ENGINEERING | St. JOSEPH'S INSTITUTE OF TECHNOLOGY

AUTONOMOUS INSTITUTIONS, AFFILIATED TO ANNA UNIVERSITY

Introduction and Overview

This document provides a step-by-step guide to deploying a Python Flask web application on an AWS EC2 instance. It covers:

- Setting up a Flask application
- Transferring files to the EC2 instance
- Configuring security groups to allow HTTP traffic
- Running and accessing the application online

By completing this guide, you will successfully deploy a web application on the cloud.

Objectives

- To develop a basic Python Flask web application.
- To deploy the Flask app on an AWS EC2 instance.
- To configure security settings to allow HTTP traffic on port 5000.
- To verify the app's accessibility over the internet.

Importance

- **Cloud Deployment Skills:** Demonstrates the ability to deploy applications in a real cloud environment.
- **Web Accessibility:** Ensures the app can be accessed from anywhere globally.
- **Security Configuration:** Provides hands-on experience with configuring security groups and firewall rules.
- **Practical AWS Knowledge:** Enhances practical understanding of AWS EC2, SSH, and server management.

STEPS:

Step 1: Write a Simple Flask Application

Create a Python file named app.py and add the following code:

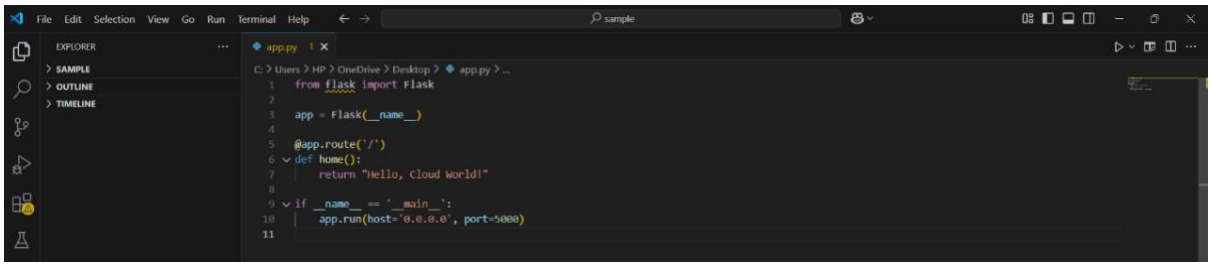
```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def home():
    return "Hello, Cloud World!"

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

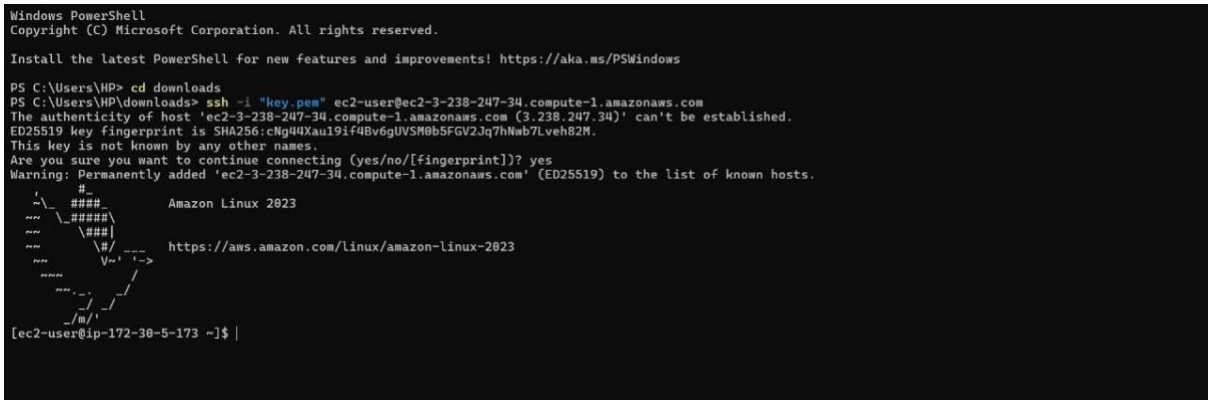
This script initializes a Flask web application and runs it on port 5000, making it accessible from any IP



STEP 2: Prepare Your EC2 Instance:

- Connect to your EC2 instance:

ssh ec2-user@your-ec2-public-ip

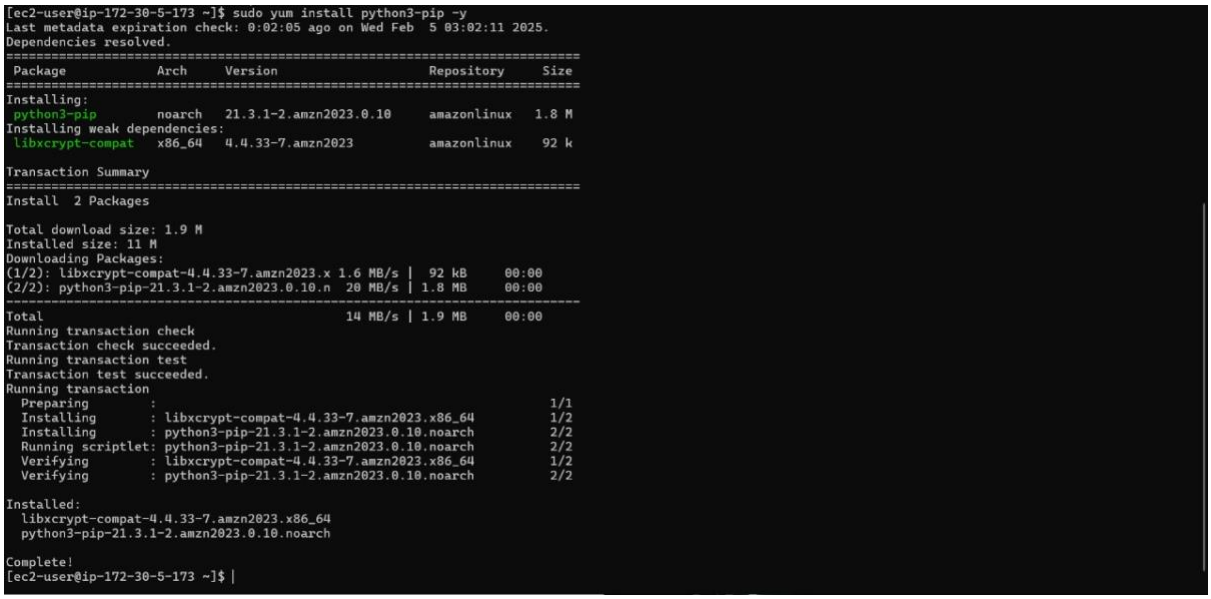


- Update packages:

sudo yum update -y

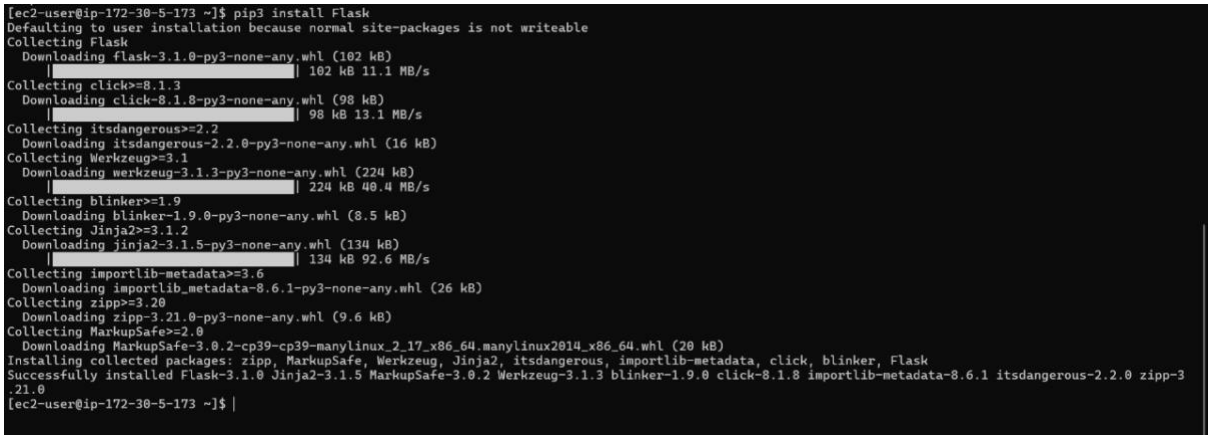
- Install Python and pip:

sudo yum install python3 python3-pip -y



- Install Flask:

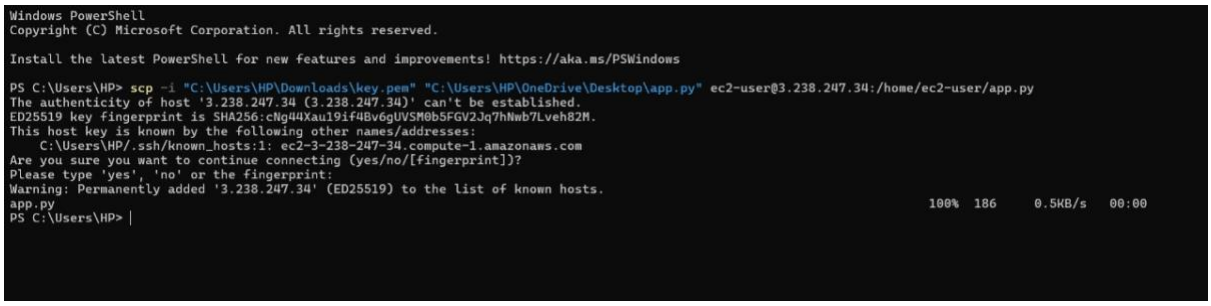
pip3 install Flask



STEP 3: Transfer the Flask App to the EC2 Instance:

Open a terminal From your local machine and run the command:

scp -i "path/to/key.pem" "D:/path/to/app.py" ec2-user@your-ec2-public-ip:/home/ec2-user/



- Change the directories with your key pair and flask app location and your EC2 public IP

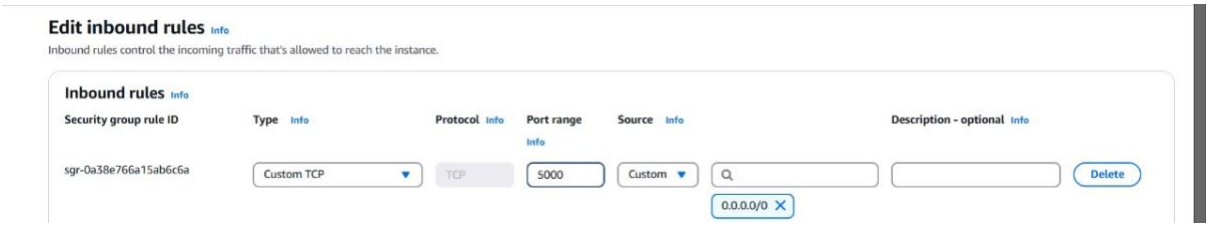
STEP 4: Run the Flask App on the EC2 Instance:

python3 app.py

```
[ec2-user@ip-172-30-5-173 ~]$ python3 app.py
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deploym
ent. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.30.5.173:5000
Press CTRL+C to quit
```

STEP 5: Configure the Firewall (Security Group Settings):

- Go to AWS Console → EC2 Dashboard → Security Groups → Inbound Rules → Edit.
- Add a rule:
    - Type: Custom TCP
    - Protocol: TCP
    - Port Range: 5000
    - Source: 0.0.0.0/0 (or specific IP for more security)



- Save the rules.

STEP 6: Access the Application:

Open a browser and go to http://your-ec2-public-ip:5000; you should see **"Hello, Flask on Cloud VM!"** displayed. If not, ensure port **5000** is open in your **EC2 Security Group**, verify the Flask app is running

using ps aux | grep flask, and restart it with python3 app.py & if necessary.

Expected Outcome

By completing this POC, you will:

1. **Successfully Deploy a Flask App** – A Python Flask web application will be deployed on a cloud-based virtual machine (VM) and accessible via a public IP.
2. **Flask App Accessibility** – The application will be accessible through http://<your_vm_public_ip>:5000, returning a simple "Hello, Flask on Cloud VM!" message.
3. **Firewall Configuration & HTTP Traffic Enablement** – The VM's firewall settings will be updated to allow incoming HTTP traffic on port 5000, ensuring external access to the application.
4. **Process Management & Background Execution** – The Flask application will be configured to run persistently using Gunicorn, nohup, or tmux, preventing downtime due to session termination.
5. **Cloud Networking & Security Understanding** – You will gain hands-on experience in configuring security groups, firewall rules, and network access controls to enable safe and efficient cloud deployments.