St. JOSEPH'S
GROUP OF INSTITUTIONS
OMR, CHENNAI - 119

aws

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: CHANDRU S

Department: INFORMATION TECHNOLOGY

St. JOSEPH'S
COLLEGE OF ENGINEERING

St. JOSEPH'S
INSTITUTE OF TECHNOLOGY

AUTONOMOUS INSTITUTIONS, AFFILIATED TO ANNA UNIVERSITY

Introduction

Cloud computing has transformed application development and deployment by providing scalability, flexibility, and cost-efficiency. This Proof of Concept (PoC) focuses on deploying a Python-based Flask web application on an AWS EC2 instance. Flask, a lightweight and versatile web framework, is well-suited for building simple yet powerful applications. Through this project, you will gain hands-on experience in setting up a virtual machine in AWS, configuring the environment, and deploying a web application that is accessible worldwide.

Overview

This project involves developing and deploying a Flask application on an Amazon EC2 instance. The application runs on a cloud-hosted Linux server with a publicly accessible HTTP endpoint. The key steps include:

1. Launching an EC2 instance.
2. Configuring the instance with Python, Flask, and required dependencies.
3. Developing a Flask web application.
4. Setting up firewall rules to allow HTTP traffic.
5. Testing the application via a web browser.

This PoC provides a straightforward yet effective approach to understanding web application deployment in a cloud environment.

Objectives

- **Learn Flask Framework** – Understand the basics of Flask and develop a simple web application.
- **Deploy on AWS EC2** – Gain practical experience in hosting applications on AWS.
- **Configure Security** – Set up inbound rules to allow secure HTTP traffic.
- **Ensure Global Accessibility** – Make the application accessible via a public IP.
- **Develop Cloud Skills** – Build expertise in cloud computing and web application deployment.

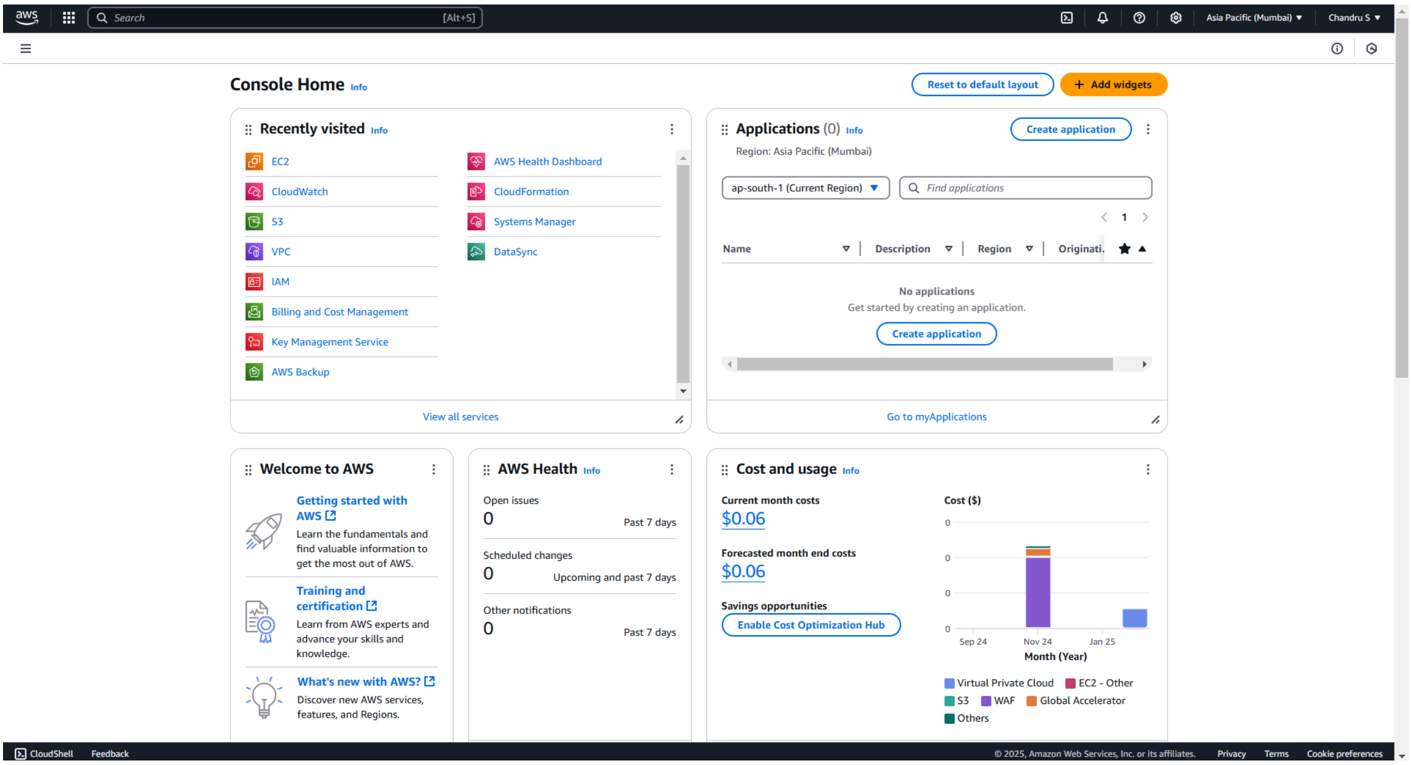
Importance

- Hands-on Experience – Provides practical exposure to deploying cloud-based applications, an essential IT skill.
- Skill Enhancement – Strengthens knowledge of cloud services, virtual machines, and web development.
- Scalability – Demonstrates how cloud infrastructure enables seamless application scaling.
- Career Growth – Enhances proficiency in cloud computing, a highly in-demand field.
- Problem-Solving – Encourages troubleshooting and environment configuration skills.

Step-by-Step Overview

Step 1:

- Go to the AWS Management Console.
- Enter your username and password to log in.



Region: Asia Pacific (Mumbai)

ap-south-1 (Current Region) ▾ | Find applications

Name	Description	Region	Originati…

No applications

Get started by creating an application.

Create application

Getting started with AWS

Learn the fundamentals and find valuable information to get the most out of AWS.

Training and certification

Learn from AWS experts and advance your skills and knowledge.

What's new with AWS?

Discover new AWS services, features, and Regions.

Open issues 0 Past 7 days

Scheduled changes 0 Upcoming and past 7 days

Other notifications 0 Past 7 days

Current month costs $0.06

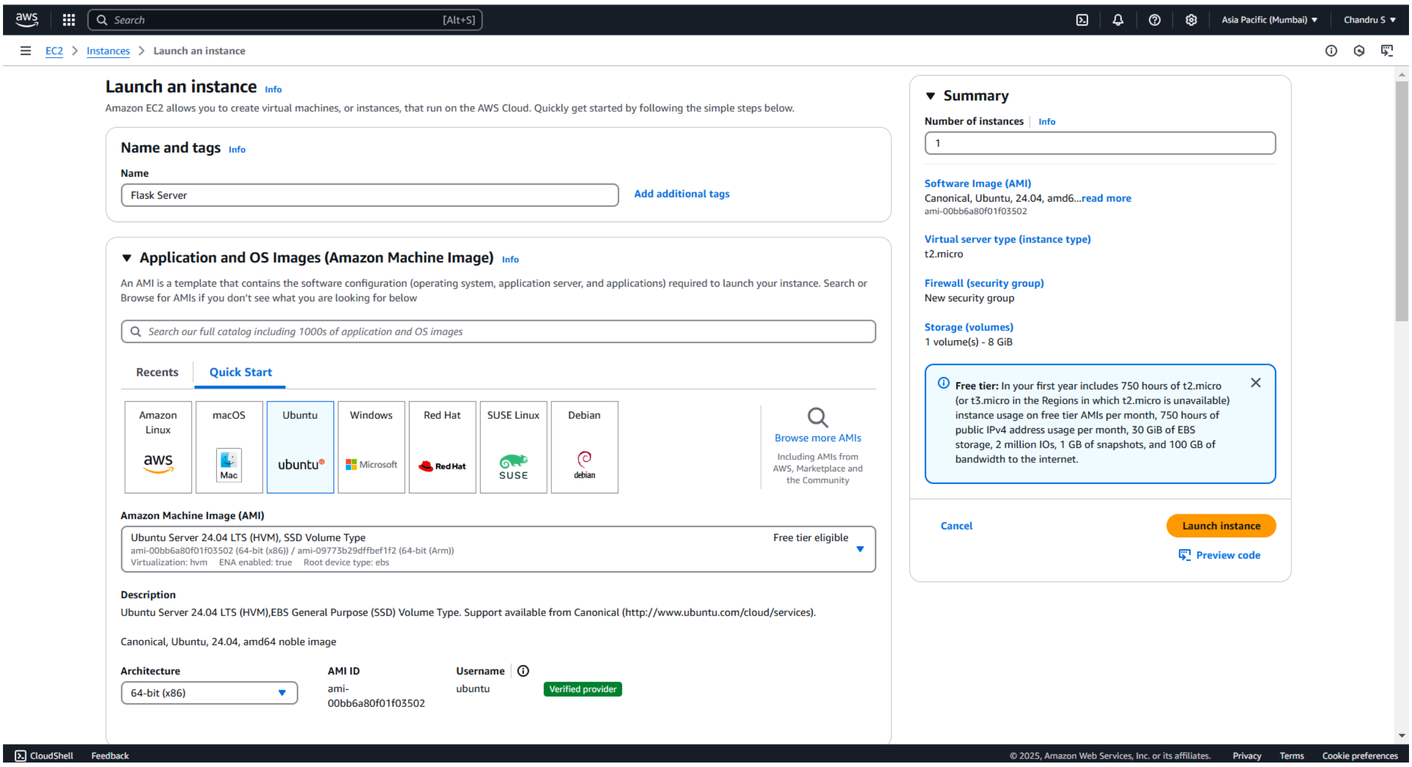
Forecasted month end costs $0.06

Savings opportunities

© 2025, Amazon Web Services, Inc. or its affiliates.

Step 2:

- On the **EC2 Dashboard**, click on **Launch Instances**.
- Enter a name for your instance (e.g., "Flask Server").
- Select **Ubuntu** as the operating system.
- Create a **key pair** (download and save it securely).
- Leave other settings as default and click **Launch Instance**.



Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name: Flask Server

Add additional tags

▼ Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images
Recents
Quick Start
Amazon Linux
macOS
Ubuntu
Windows
Red Hat
SUSE Linux
Debian
Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type — Free tier eligible
ami-00bb6a80f01f03502 (64-bit (x86)) / ami-09773b29dffbef1f2 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Description

Ubuntu Server 24.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).

Canonical, Ubuntu, 24.04, amd64 noble image

Architecture: 64-bit (x86)

AMI ID: ami-00bb6a80f01f03502

Username: ubuntu

Verified provider

▼ Summary

Number of instances Info: 1

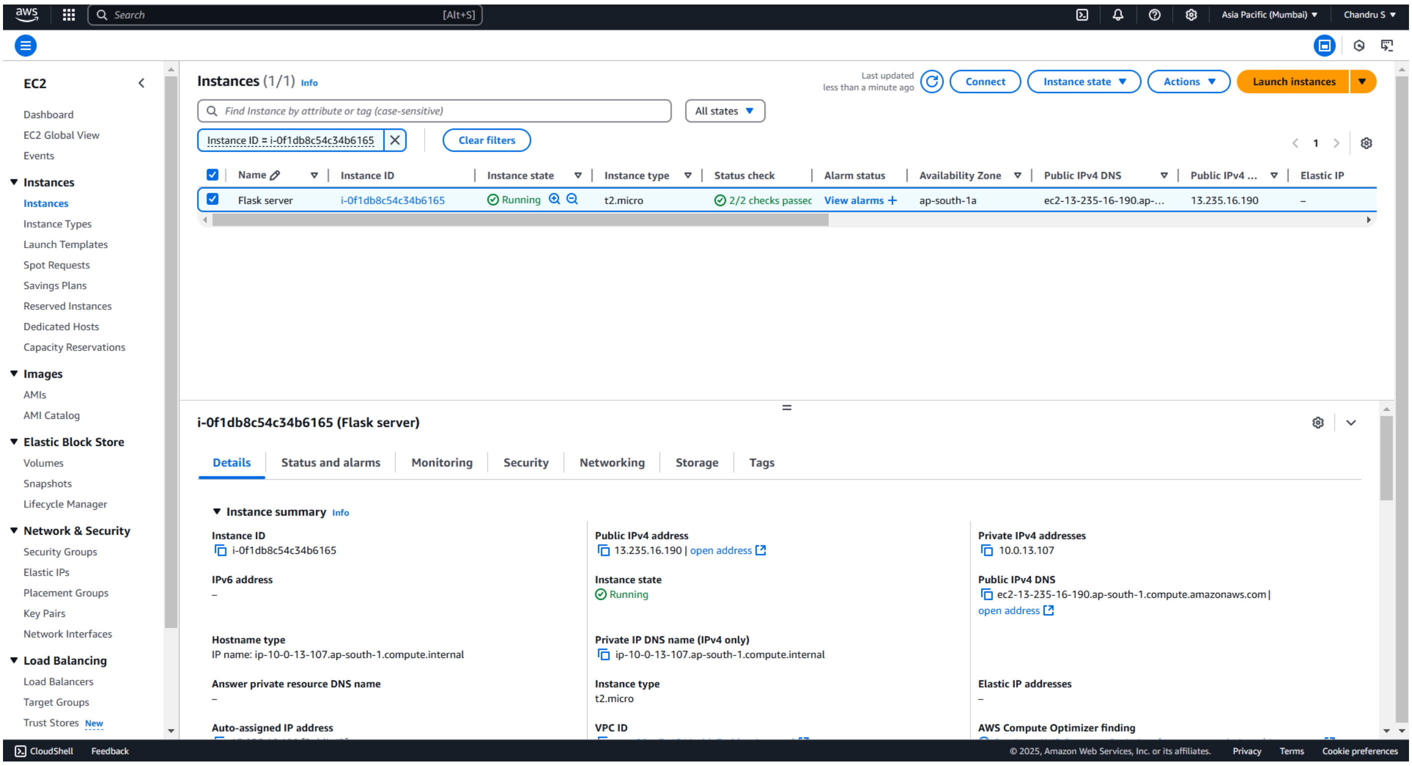
Software Image (AMI)
Canonical, Ubuntu, 24.04, amd6...read more
ami-00bb6a80f01f03502

Virtual server type (instance type)
t2.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month, 30 GiB of EBS storage, 2 million IOs, 1 GB of snapshots, and 100 GB of bandwidth to the internet.



Instances (1/1) Info

Instance ID = i-0f1db8c54c34b6165

☑	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
☑	Flask server	i-0f1db8c54c34b6165	✓ Running	t2.micro	✓ 2/2 checks passec	View alarms +	ap-south-1a	ec2-13-235-16-190.ap-...	13.235.16.190	–

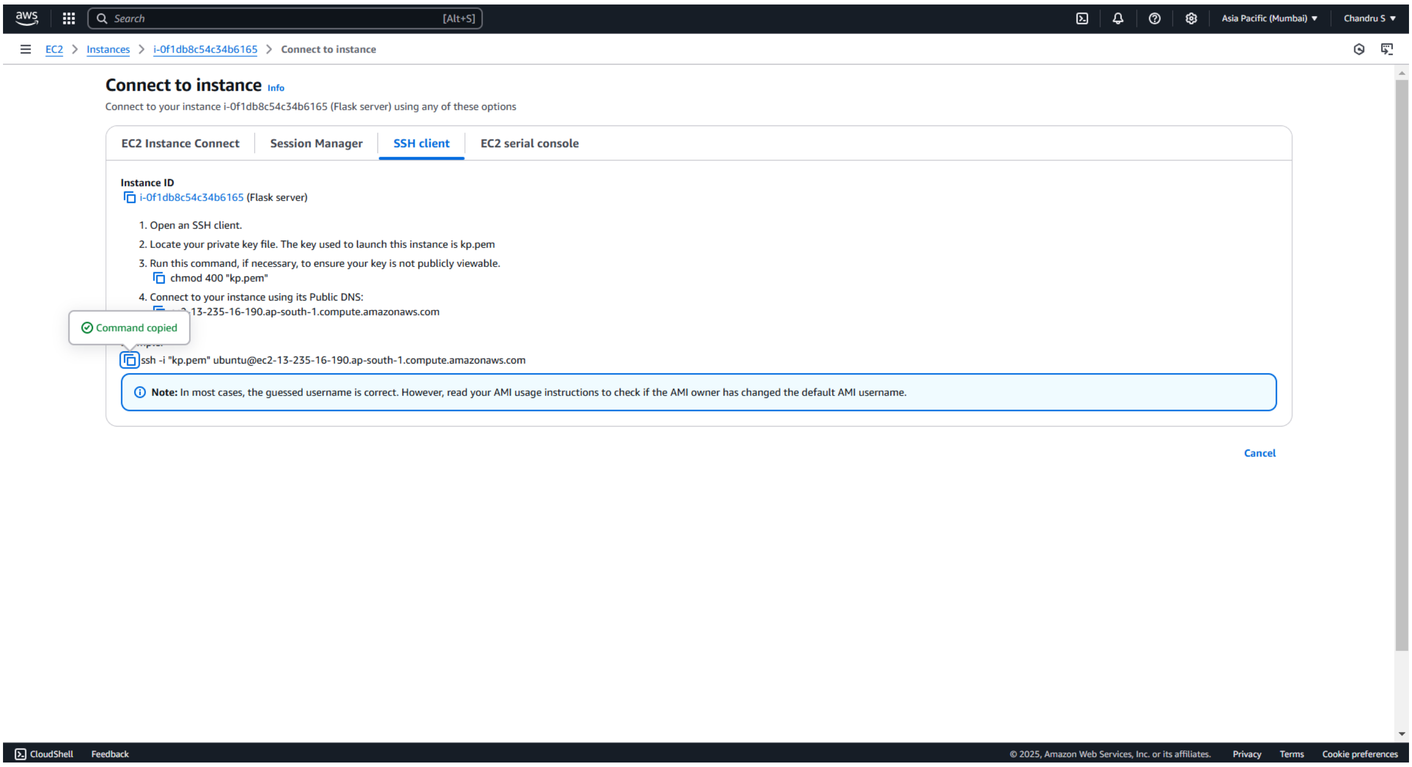
i-0f1db8c54c34b6165 (Flask server)

▼ Instance summary Info

Instance ID i-0f1db8c54c34b6165	Public IPv4 address 13.235.16.190 | open address	Private IPv4 addresses 10.0.13.107
IPv6 address –	Instance state ✓ Running	Public IPv4 DNS ec2-13-235-16-190.ap-south-1.compute.amazonaws.com | open address
Hostname type IP name: ip-10-0-13-107.ap-south-1.compute.internal	Private IP DNS name (IPv4 only) ip-10-0-13-107.ap-south-1.compute.internal	
Answer private resource DNS name –	Instance type t2.micro	Elastic IP addresses –
Auto-assigned IP address	VPC ID	AWS Compute Optimizer finding

Step 3:

- In the EC2 dashboard, click on your launched instance.
- Click **Connect**, then go to the **SSH client** section.
- Copy the command provided under the "Example" section.



Connect to instance Info

Connect to your instance i-0f1db8c54c34b6165 (Flask server) using any of these options

Instance ID

i-0f1db8c54c34b6165 (Flask server)

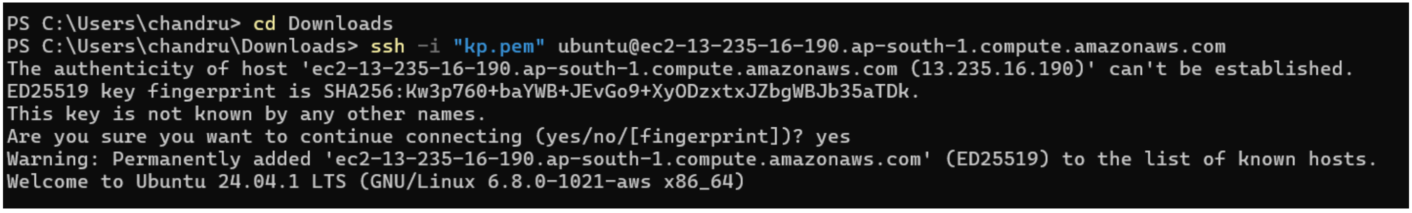
1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is kp.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
   chmod 400 "kp.pem"
4. Connect to your instance using its Public DNS:
   13-235-16-190.ap-south-1.compute.amazonaws.com

ssh -i "kp.pem" ubuntu@ec2-13-235-16-190.ap-south-1.compute.amazonaws.com

Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

Step 4:

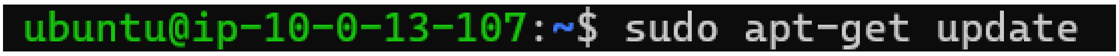
- Open PowerShell on your computer.
- Navigate to the Downloads directory (where your key pair is stored) using: **cd Downloads**
- Paste the SSH command copied from the EC2 Connect page.
- Replace the key pair name with your downloaded key (e.g., kp.pem).
- Press Enter and type yes when prompted.



```
PS C:\Users\chandru> cd Downloads
PS C:\Users\chandru\Downloads> ssh -i "kp.pem" ubuntu@ec2-13-235-16-190.ap-south-1.compute.amazonaws.com
The authenticity of host 'ec2-13-235-16-190.ap-south-1.compute.amazonaws.com (13.235.16.190)' can't be established.
ED25519 key fingerprint is SHA256:Kw3p760+baYWB+JEvGo9+XyODzxtxJZbgWBJb35aTDk.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-13-235-16-190.ap-south-1.compute.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)
```

Step 5:

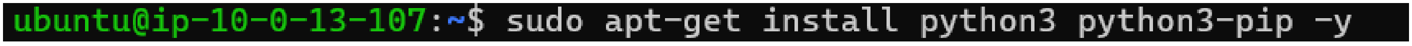
Run the following command to update the package list:



```
ubuntu@ip-10-0-13-107:~$ sudo apt-get update
```

Step 6:

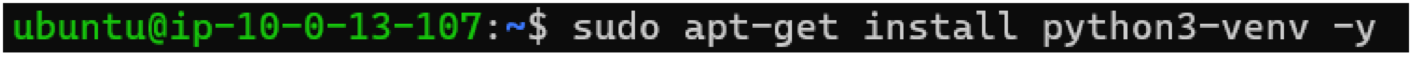
Install Python3 and pip



```
ubuntu@ip-10-0-13-107:~$ sudo apt-get install python3 python3-pip -y
```

Step 7:

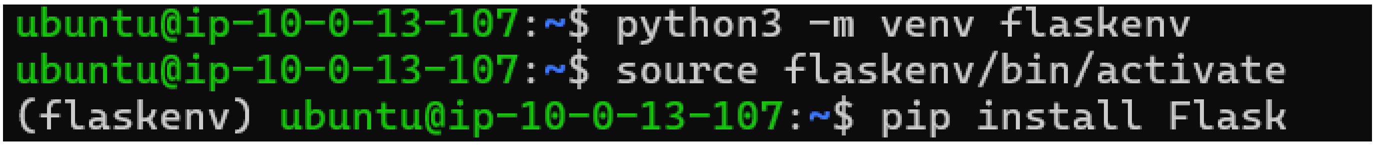
Virtual environments help manage dependencies separately. Install them using:



```
ubuntu@ip-10-0-13-107:~$ sudo apt-get install python3-venv -y
```

Step 8:

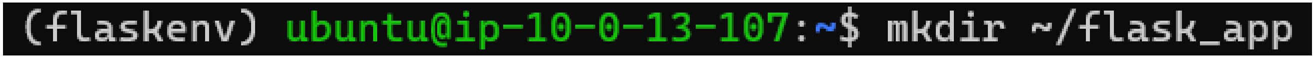
- Create a virtual environment:
  python3 -m venv flaskenv
- Activate it:
  source flaskenv/bin/activate
- Install Flask:
  pip install flask



```
ubuntu@ip-10-0-13-107:~$ python3 -m venv flaskenv
ubuntu@ip-10-0-13-107:~$ source flaskenv/bin/activate
(flaskenv) ubuntu@ip-10-0-13-107:~$ pip install Flask
```

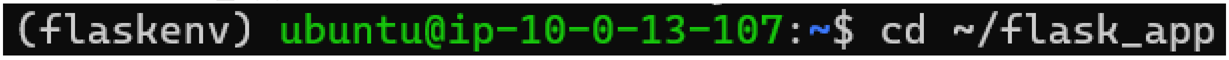
Step 9:

- Create a directory for your app:
  mkdir ~/flask_app



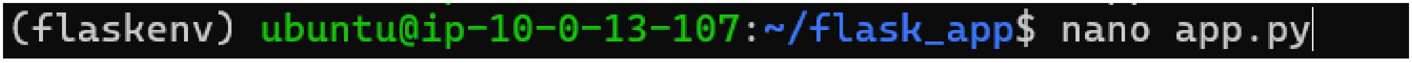
```
(flaskenv) ubuntu@ip-10-0-13-107:~$ mkdir ~/flask_app
```

cd flask_app



```
(flaskenv) ubuntu@ip-10-0-13-107:~$ cd ~/flask_app
```

- Create a file named app.py using a text editor like nano:
  nano app.py

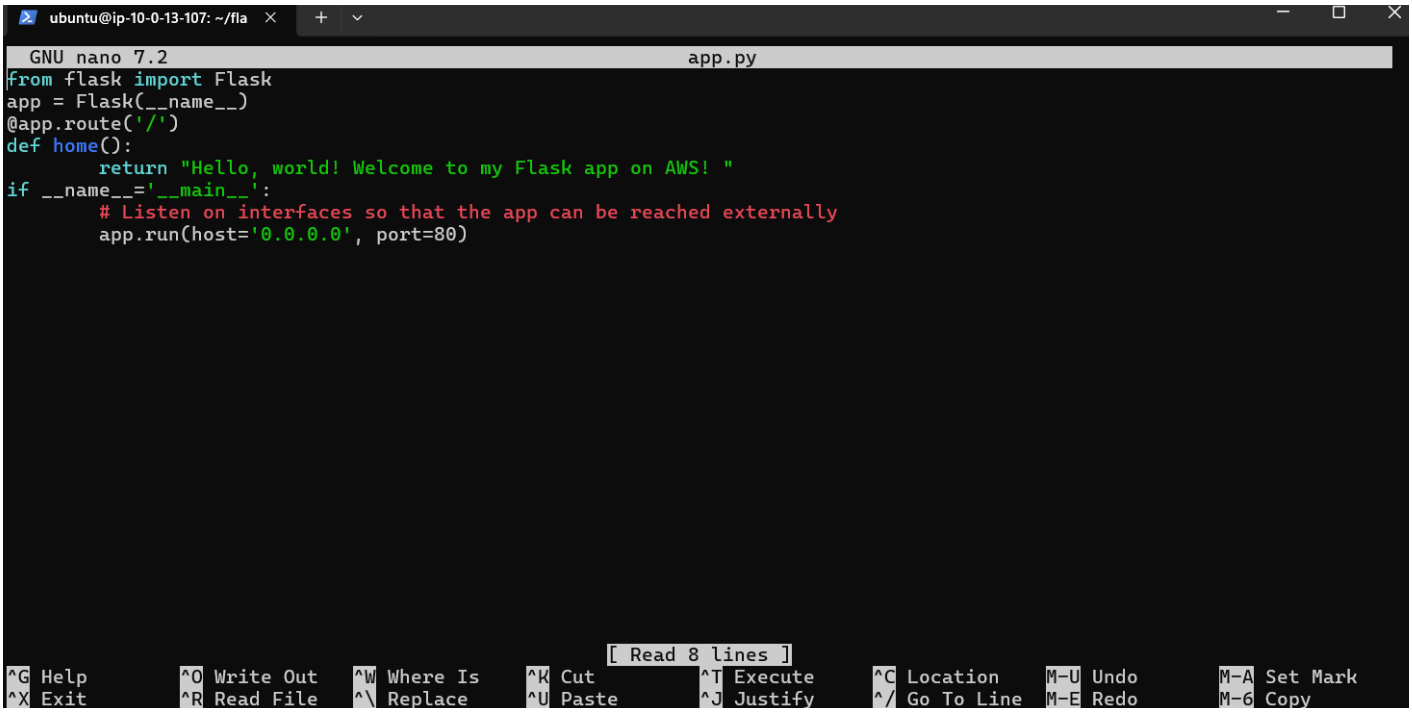


```
(flaskenv) ubuntu@ip-10-0-13-107:~/flask_app$ nano app.py
```

Step 10:

Copy and paste the following code into the editor:

- Press **Ctrl + O** to save the file, then Enter.
- Press **Ctrl + X** to exit the editor.



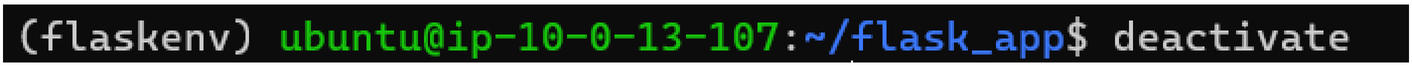
```
  GNU nano 7.2                                    app.py
from flask import Flask
app = Flask(__name__)
@app.route('/')
def home():
        return "Hello, world! Welcome to my Flask app on AWS! "
if __name__='__main__':
        # Listen on interfaces so that the app can be reached externally
        app.run(host='0.0.0.0', port=80)

                                     [ Read 8 lines ]
```

Step 11:

Exit the Virtual Environment:

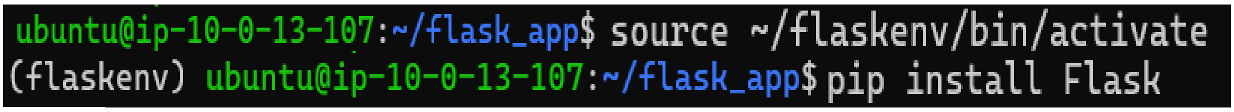
deactivate



```
(flaskenv) ubuntu@ip-10-0-13-107:~/flask_app$ deactivate
```

Step 12:

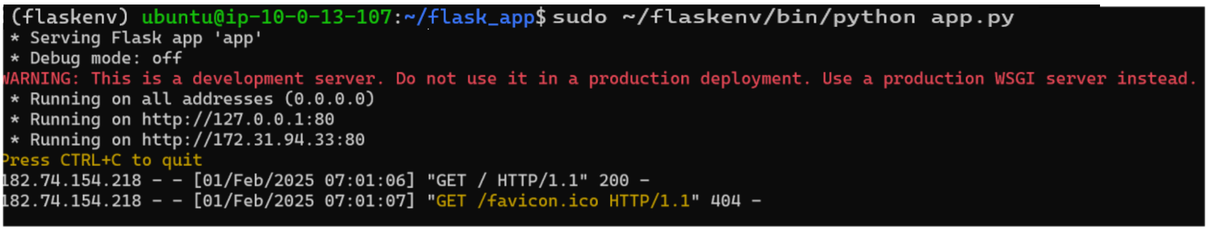
Add the virtual environment’s Python path to the sudo command:



```
ubuntu@ip-10-0-13-107:~/flask_app$ source ~/flaskenv/bin/activate
(flaskenv) ubuntu@ip-10-0-13-107:~/flask_app$ pip install Flask
```

Step 13:

Your Flask App is Now Running!

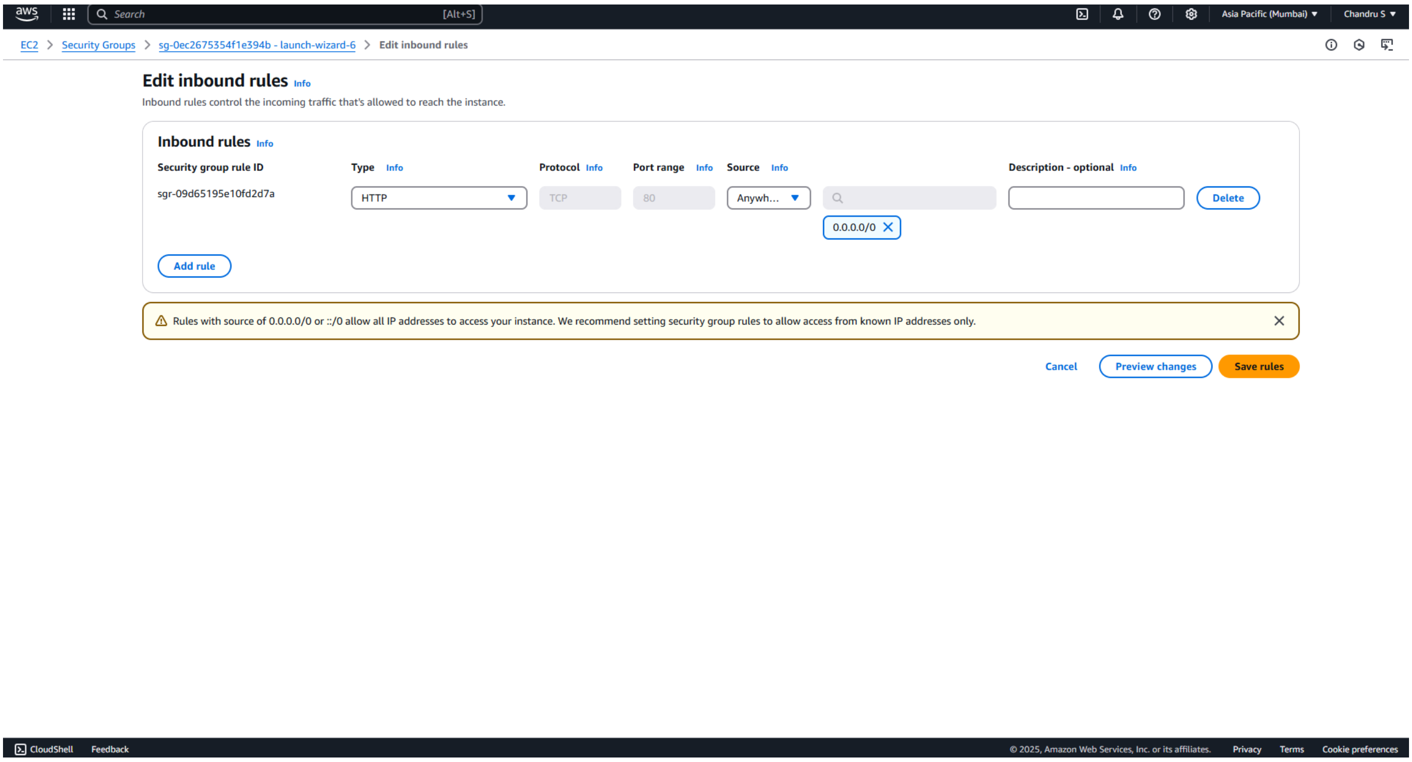


```
(flaskenv) ubuntu@ip-10-0-13-107:~/flask_app$ sudo ~/flaskenv/bin/python app.py
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:80
 * Running on http://172.31.94.33:80
Press CTRL+C to quit
182.74.154.218 - - [01/Feb/2025 07:01:06] "GET / HTTP/1.1" 200 -
182.74.154.218 - - [01/Feb/2025 07:01:07] "GET /favicon.ico HTTP/1.1" 404 -
```

Step 14:

Configure Security Group for HTTP Access

1. Go to **EC2 Dashboard** > Instances.
2. Find your instance and note the Security Group attached to it.
3. Navigate to **Security Groups** under the Network & Security section.
4. Select the Security Group associated with your EC2 instance.
5. 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)
6. If there is no HTTP rule, click **Edit inbound rules** and add it.



EC2 > Security Groups > sg-0ec2675354f1e394b - launch-wizard-6 > Edit inbound rules

Edit inbound rules Info

Inbound rules control the incoming traffic that's allowed to reach the instance.

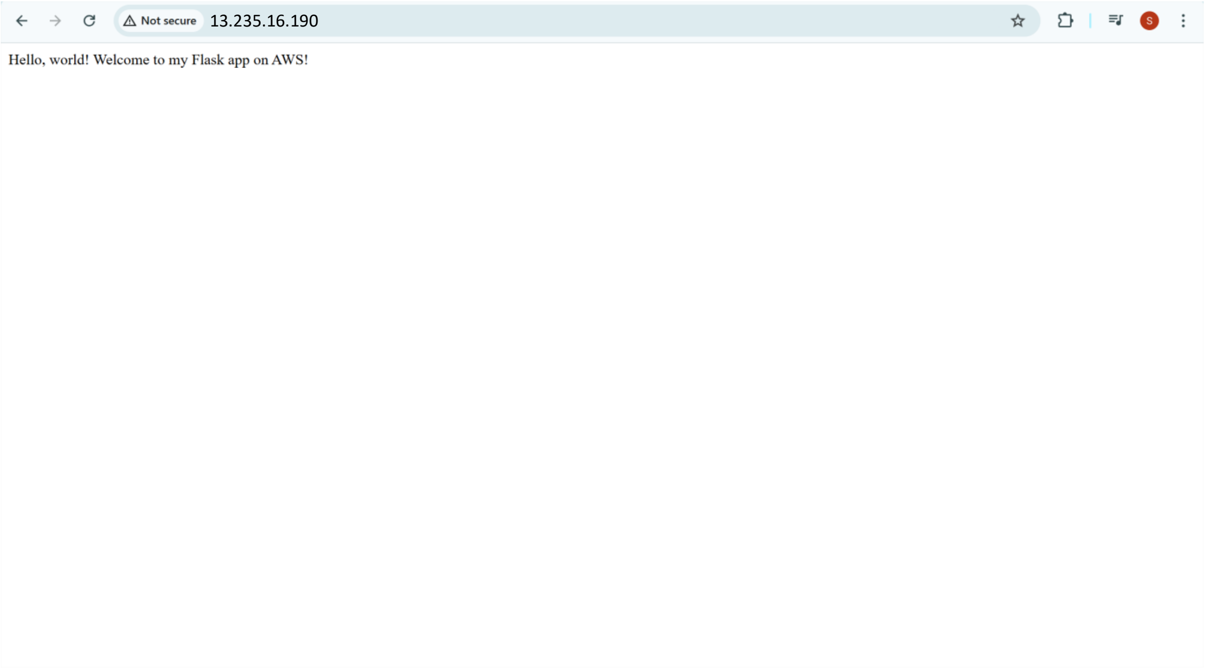
Inbound rules Info

Security group rule ID	Type	Protocol	Port range	Source		Description - optional
sgr-09d65195e10fd2d7a	HTTP	TCP	80	Anywh...	0.0.0.0/0	

⚠ Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Step 15:

- Open your web browser and navigate to: **http://<Your-Instance-Public-IP>/**
- Replace **<Your-Instance-Public-IP>** with your EC2 instance's Public IPv4 address (found in the EC2 instance dashboard).
- Your Flask web application should now be live!



Not secure 13.235.16.190

Hello, world! Welcome to my Flask app on AWS!

Outcome

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

1. **Set Up an EC2 Instance** – Launch and configure an Ubuntu-based EC2 instance.
2. **Configure the Python Environment** – Install and set up Python along with Flask and its dependencies.
3. **Develop a Flask Application** – Create a simple Flask app (app.py) that displays a message when accessed via a web browser.
4. **Deploy and Secure the Application** – Host the Flask application on the EC2 instance and configure security group rules to allow HTTP traffic.
5. **Access the Live Application** – Test and access the deployed application using the EC2 instance's Public IPv4 DNS or IP address.