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 : Shalini D　　　　　　　　　　Department: IT

Introduction

Deploying a web application on the cloud involves several steps, including setting up the application, configuring the virtual machine (VM), and managing the firewall settings. Below, I'll guide you through the process of creating a simple Python Flask application and deploying it on a cloud VM.

Objectives

The objective of this process is to deploy a Python Flask web application on a cloud virtual machine (VM) and configure the firewall to allow HTTP traffic, making the application accessible over the internet.

Importance

Deploying a web application on the cloud and configuring the firewall to allow HTTP traffic is crucial for several reasons:

Accessibility: It makes your application accessible to users around the world, allowing them to interact with it through a web browser.

Scalability: Cloud infrastructure can easily handle increased traffic and load, ensuring that your application remains responsive and available.

Security: Properly configuring the firewall protects your application from unauthorized access and potential security threats.

Reliability: Cloud providers offer robust and reliable services with high uptime, ensuring that your application is available 24/7.

Cost Efficiency: Using cloud resources can be more cost-effective than maintaining on-premises servers, as you only pay for what you use.

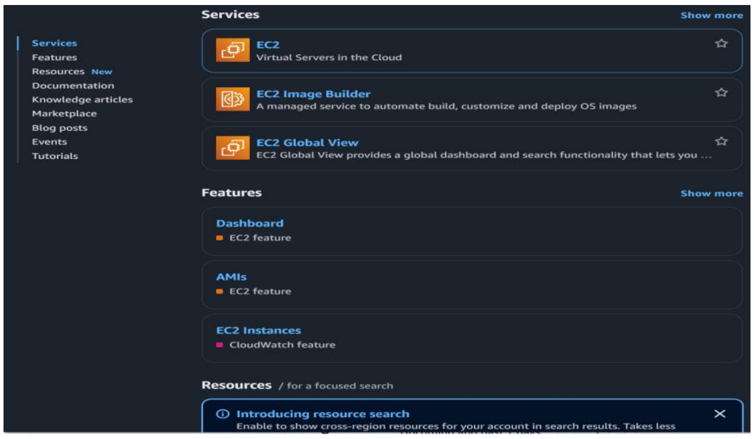
Flexibility: Cloud environments provide flexibility in terms of development, testing, and deployment, making it easier to manage and update your application.

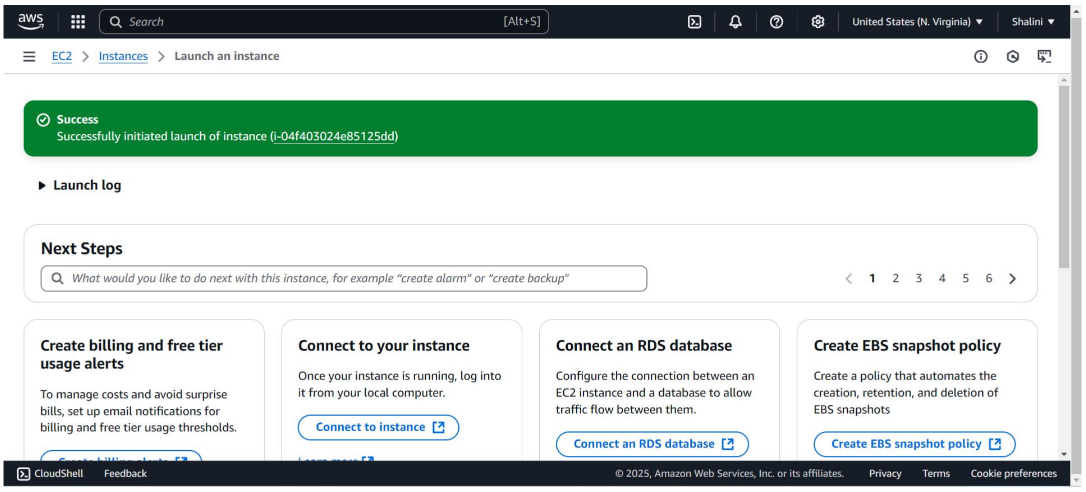
1.

Step-by-Step Overview

Step 1:

On the EC2 Dashboard, and launch an **Instances** and enter a name for your instance and select Ubuntu as OS and create a key pair.


Services
Show more
Services
Features
Resources New
Documentation
Knowledge articles
Marketplace
Blog posts
Events
Tutorials
EC2
Virtual Servers in the Cloud
EC2 Image Builder
A managed service to automate build, customize and deploy OS images
EC2 Global View
EC2 Global View provides a global dashboard and search functionality that lets you ...
Features
Show more
Dashboard
EC2 feature
AMIs
EC2 feature
EC2 Instances
CloudWatch feature
Resources / for a focused search
Introducing resource search
Enable to show cross-region resources for your account in search results. Takes less


aws
Search [Alt+S]
United States (N. Virginia)
Shalini
EC2 > Instances > Launch an instance
Launch log
Next Steps
What would you like to do next with this instance, for example "create alarm" or "create backup"
1 2 3 4 5 6
Connect to instance
Connect an RDS database
Create EBS snapshot policy
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Success
Successfully initiated launch of instance (i-04f403024e85125dd)

Create billing and free tier usage alerts
To manage costs and avoid surprise bills, set up email notifications for billing and free tier usage thresholds.

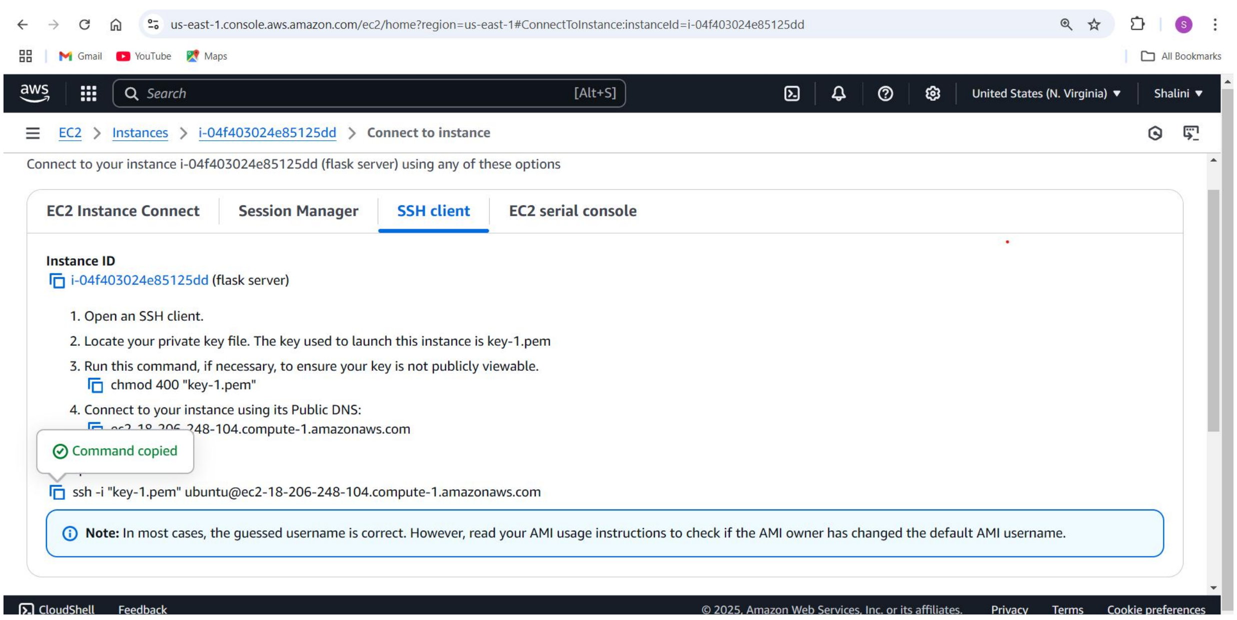
Connect to your instance
Once your instance is running, log into it from your local computer.

Connect an RDS database
Configure the connection between an EC2 instance and a database to allow traffic flow between them.

Create EBS snapshot policy
Create a policy that automates the creation, retention, and deletion of EBS snapshots

Step 2:

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



us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#ConnectToInstance:instanceId=i-04f403024e85125dd

EC2 > Instances > i-04f403024e85125dd > Connect to instance

Connect to your instance i-04f403024e85125dd (flask server) using any of these options

EC2 Instance Connect | Session Manager | SSH client | EC2 serial console

Instance ID
i-04f403024e85125dd (flask server)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is key-1.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
   ```
   chmod 400 "key-1.pem"
   ```
4. Connect to your instance using its Public DNS:
   ec2-18-206-248-104.compute-1.amazonaws.com

Command copied

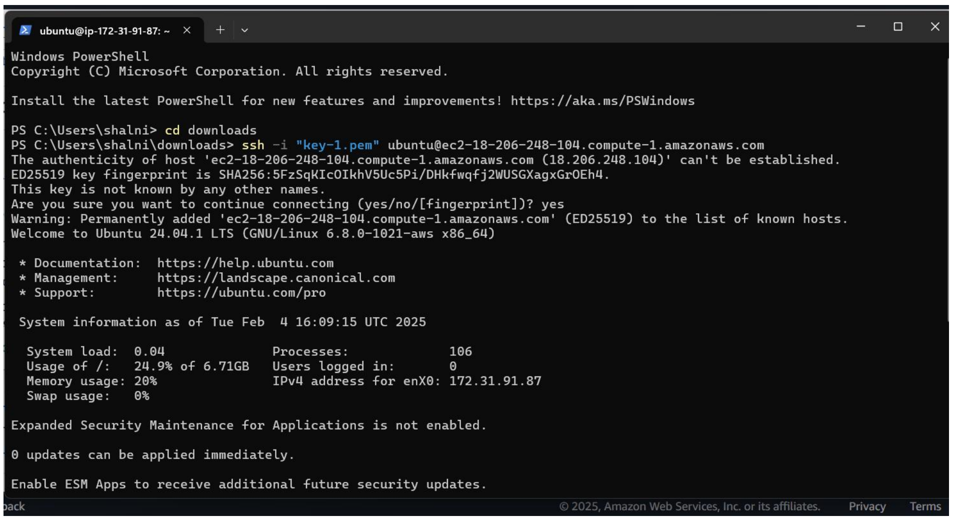
```
ssh -i "key-1.pem" ubuntu@ec2-18-206-248-104.compute-1.amazonaws.com
```

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 3:

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.

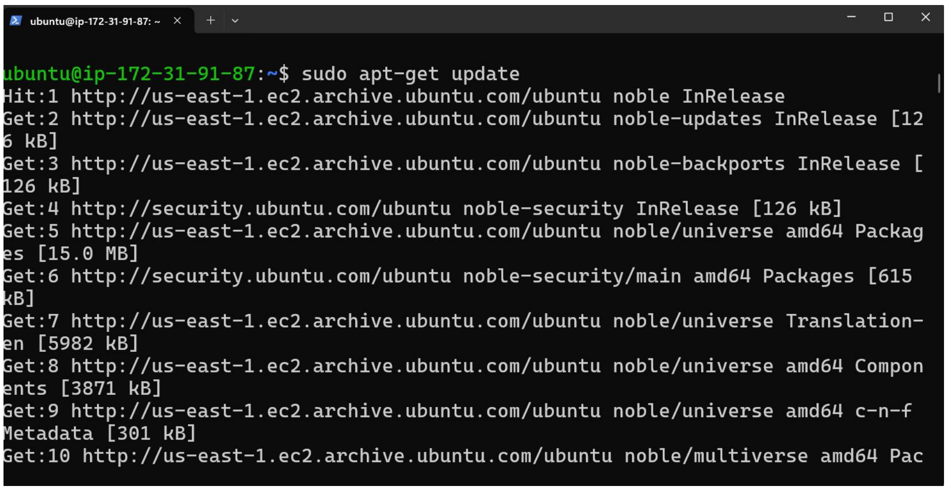


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

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

PS C:\Users\shalni> cd downloads
PS C:\Users\shalni\downloads> ssh -i "key-1.pem" ubuntu@ec2-18-206-248-104.compute-1.amazonaws.com
The authenticity of host 'ec2-18-206-248-104.compute-1.amazonaws.com (18.206.248.104)' can't be established.
ED25519 key fingerprint is SHA256:5FzSqKIcOIkhV5Uc5Pi/DHkfwqfj2WUSGXagxGrOEh4.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-18-206-248-104.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)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Tue Feb  4 16:09:15 UTC 2025

  System load:  0.04              Processes:             106
  Usage of /:   24.9% of 6.71GB   Users logged in:       0
  Memory usage: 20%               IPv4 address for enX0: 172.31.91.87
  Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
```

ubuntu@ip-172-31-91-87: ~
ack
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms

Step 4:

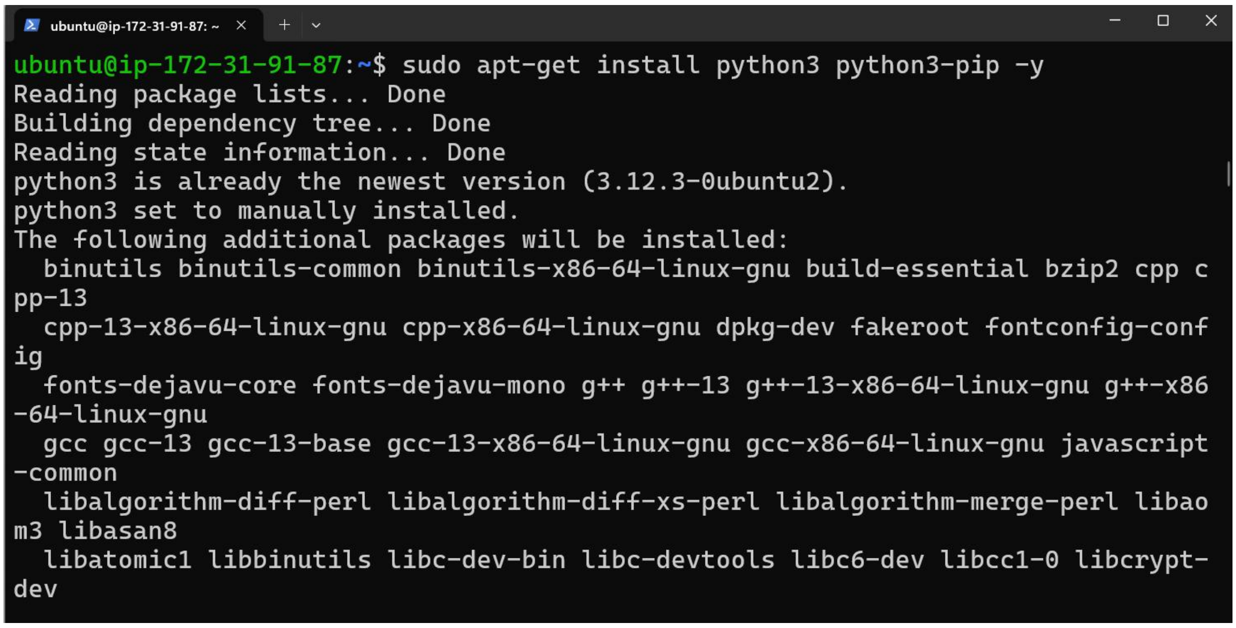
Update the Package List :



```
ubuntu@ip-172-31-91-87:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [12
6 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [
126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packag
es [15.0 MB]
Get:6 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [615
kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-
en [5982 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Compon
ents [3871 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f
Metadata [301 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Pac
```

Step 5:

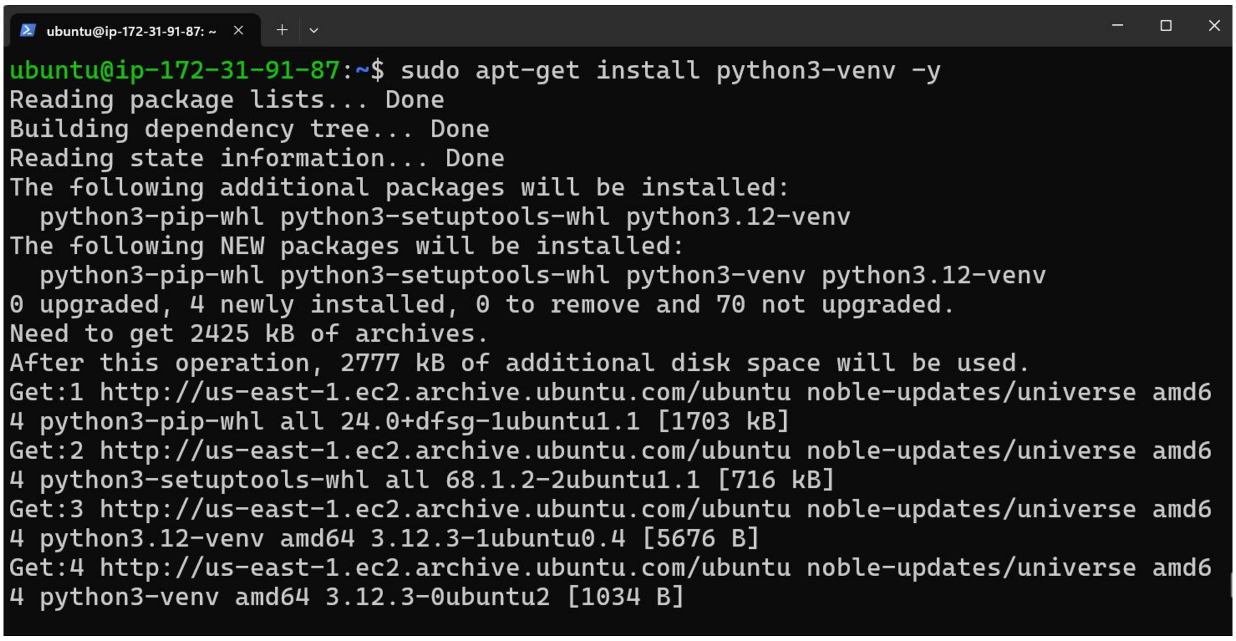
Install Python3 and pip



```
ubuntu@ip-172-31-91-87:~$ sudo apt-get 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 set to manually installed.
The following additional packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu build-essential bzip2 cpp c
pp-13
  cpp-13-x86-64-linux-gnu cpp-x86-64-linux-gnu dpkg-dev fakeroot fontconfig-conf
ig
  fonts-dejavu-core fonts-dejavu-mono g++ g++-13 g++-13-x86-64-linux-gnu g++-x86
-64-linux-gnu
  gcc gcc-13 gcc-13-base gcc-13-x86-64-linux-gnu gcc-x86-64-linux-gnu javascript
-common
  libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl libao
m3 libasan8
  libatomic1 libbinutils libc-dev-bin libc-devtools libc6-dev libcc1-0 libcrypt-
dev
```

Step 6:

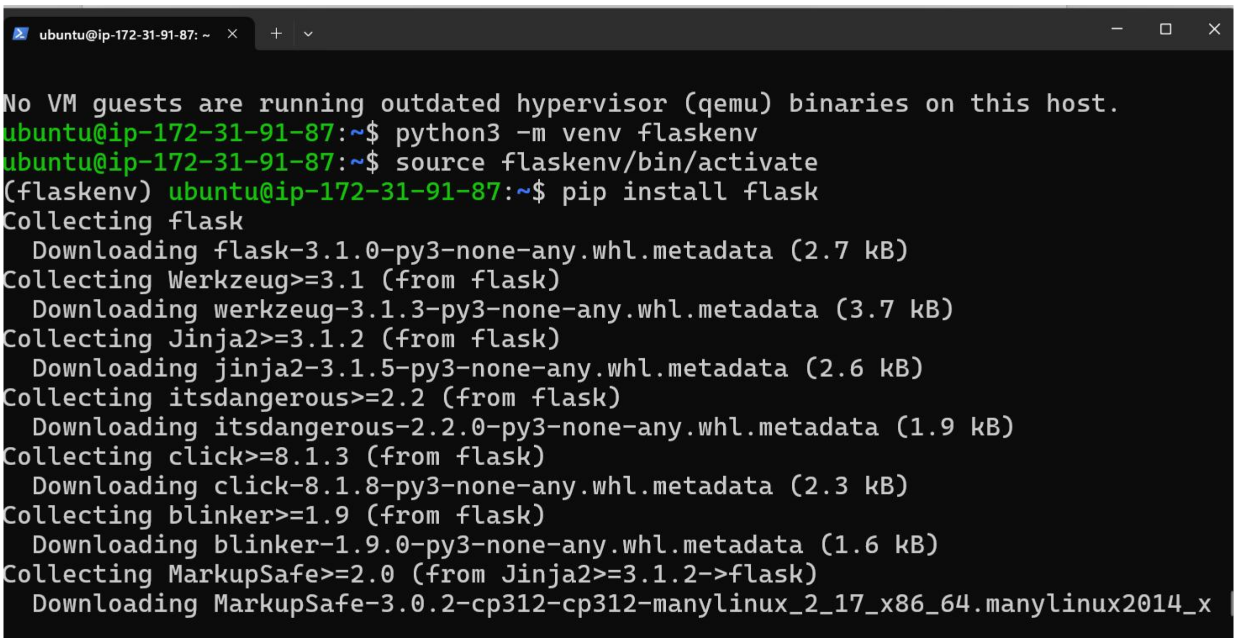
Install Virtual Environment Tools : This helps keep your app’s dependencies separate.



```
ubuntu@ip-172-31-91-87:~$ sudo apt-get install python3-venv -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  python3-pip-whl python3-setuptools-whl python3.12-venv
The following NEW packages will be installed:
  python3-pip-whl python3-setuptools-whl python3-venv python3.12-venv
0 upgraded, 4 newly installed, 0 to remove and 70 not upgraded.
Need to get 2425 kB of archives.
After this operation, 2777 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd6
4 python3-pip-whl all 24.0+dfsg-1ubuntu1.1 [1703 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd6
4 python3-setuptools-whl all 68.1.2-2ubuntu1.1 [716 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd6
4 python3.12-venv amd64 3.12.3-1ubuntu0.4 [5676 B]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd6
4 python3-venv amd64 3.12.3-0ubuntu2 [1034 B]
```

Step 7:

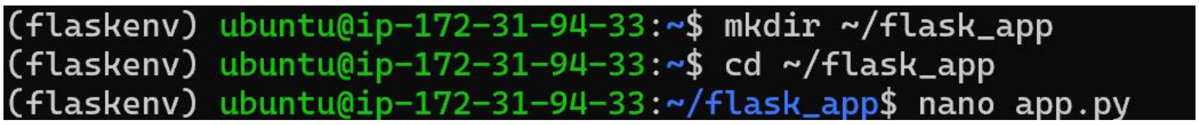
Create and Activate a Virtual Environment and install Flask



```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-91-87:~$ python3 -m venv flaskenv
ubuntu@ip-172-31-91-87:~$ source flaskenv/bin/activate
(flaskenv) ubuntu@ip-172-31-91-87:~$ pip install flask
Collecting flask
  Downloading flask-3.1.0-py3-none-any.whl.metadata (2.7 kB)
Collecting Werkzeug>=3.1 (from flask)
  Downloading werkzeug-3.1.3-py3-none-any.whl.metadata (3.7 kB)
Collecting Jinja2>=3.1.2 (from flask)
  Downloading jinja2-3.1.5-py3-none-any.whl.metadata (2.6 kB)
Collecting itsdangerous>=2.2 (from flask)
  Downloading itsdangerous-2.2.0-py3-none-any.whl.metadata (1.9 kB)
Collecting click>=8.1.3 (from flask)
  Downloading click-8.1.8-py3-none-any.whl.metadata (2.3 kB)
Collecting blinker>=1.9 (from flask)
  Downloading blinker-1.9.0-py3-none-any.whl.metadata (1.6 kB)
Collecting MarkupSafe>=2.0 (from Jinja2>=3.1.2->flask)
  Downloading MarkupSafe-3.0.2-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x
```

Step 8:

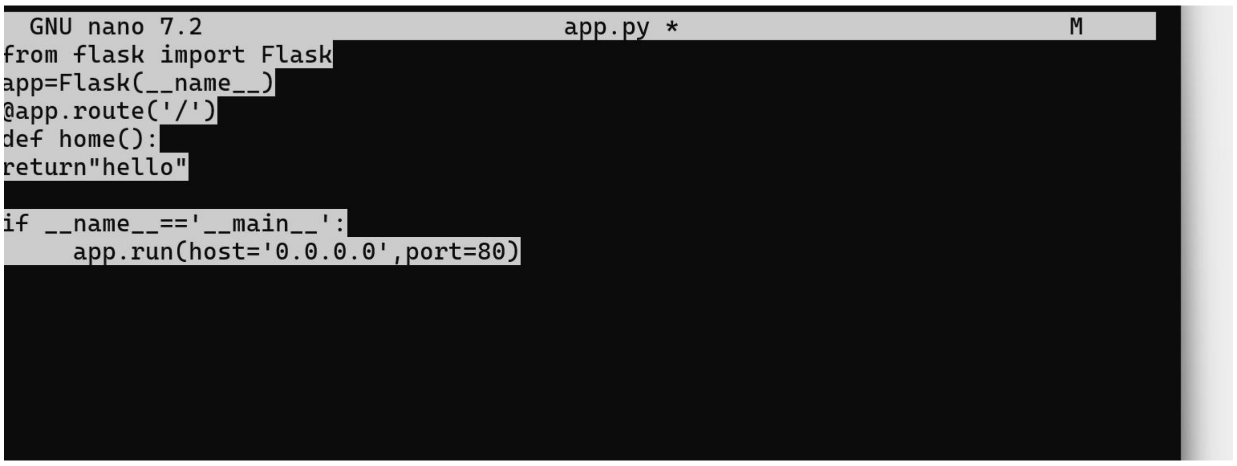
Create a Directory for Your App and Create a file called app.py using a text editor (like nano).



```
(flaskenv) ubuntu@ip-172-31-94-33:~$ mkdir ~/flask_app
(flaskenv) ubuntu@ip-172-31-94-33:~$ cd ~/flask_app
(flaskenv) ubuntu@ip-172-31-94-33:~/flask_app$ nano app.py
```

Step 10:

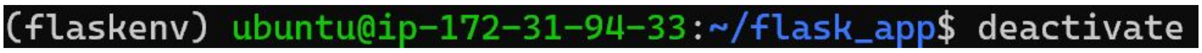
Write this code into the editor and press **Ctrl + O** (to write out) and then **Enter**, then **Ctrl + X** to exit.



```
  GNU nano 7.2                          app.py *                          M
from flask import Flask
app=Flask(__name__)
@app.route('/')
def home():
return"hello"

if __name__=='__main__':
     app.run(host='0.0.0.0',port=80)
```

Step 11:

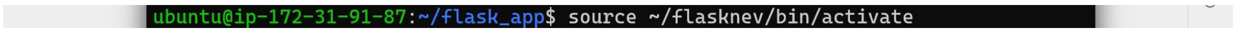
Exit the virtual environment:



```
(flaskenv) ubuntu@ip-172-31-94-33:~/flask_app$ deactivate
```

Step 12:

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

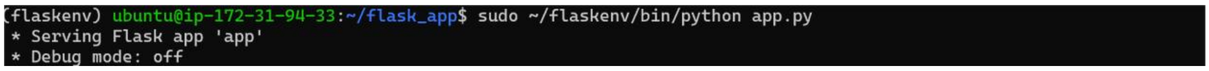


```
ubuntu@ip-172-31-91-87:~/flask_app$ source ~/flasknev/bin/activate
```

Step 13:

Your Flask app is now running!

/6



```
(flaskenv) ubuntu@ip-172-31-94-33:~/flask_app$ sudo ~/flaskenv/bin/python app.py
 * Serving Flask app 'app'
 * Debug mode: off
```

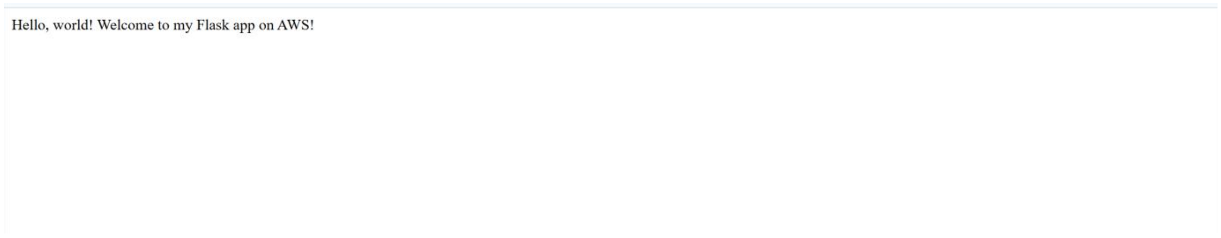
Step 15:

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.



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

Outcome

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.