

Assignment–3

Module-3 Python

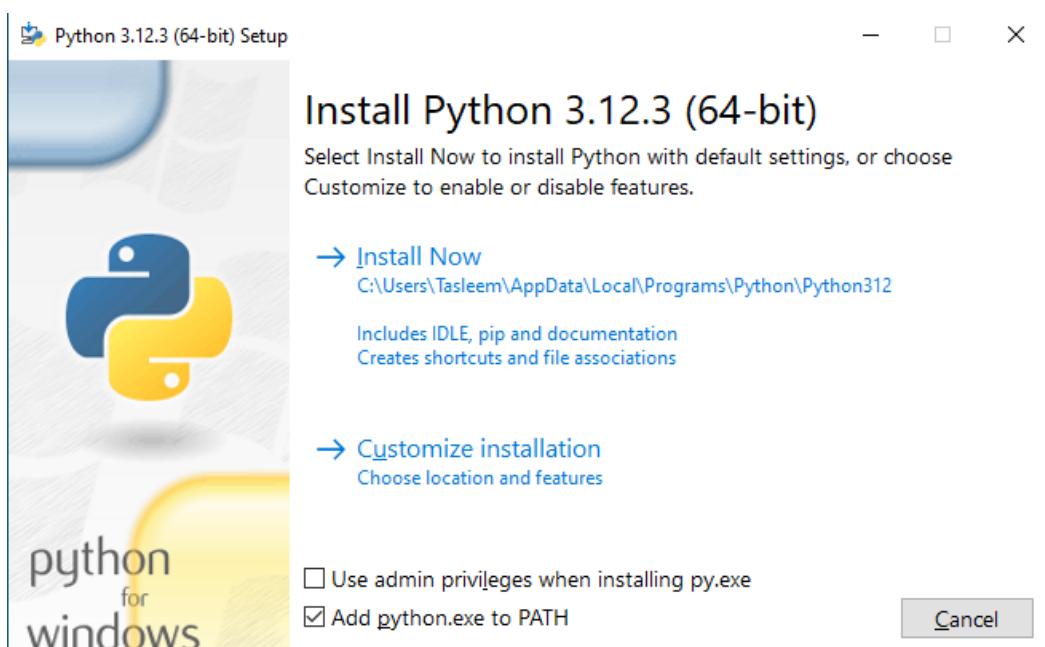
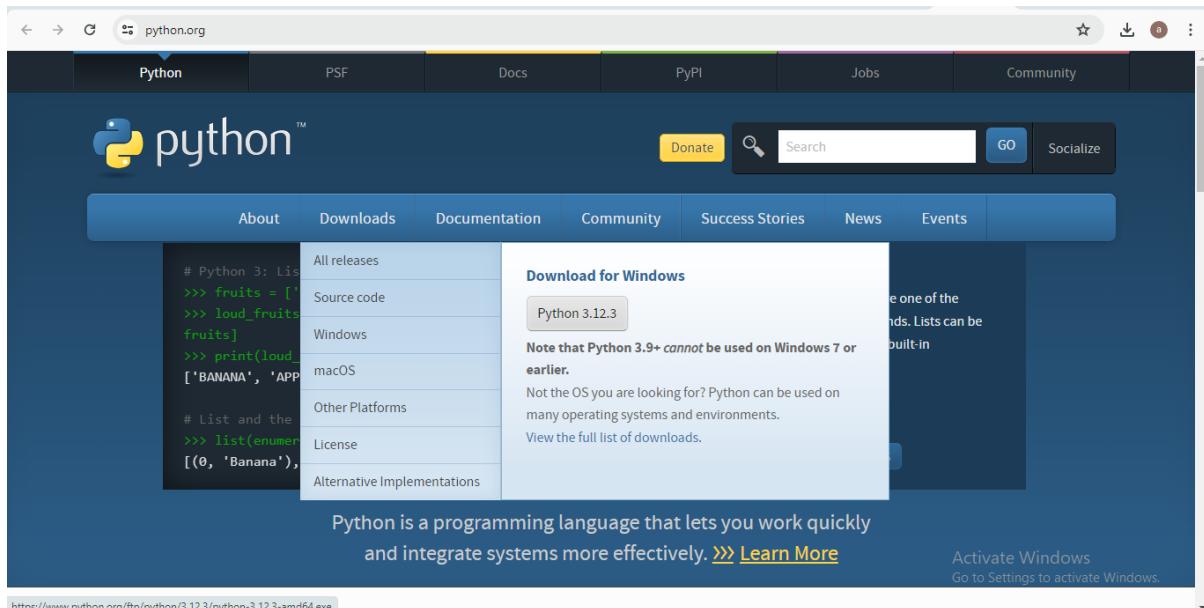
Submitted by : Shaik Junaid Adil

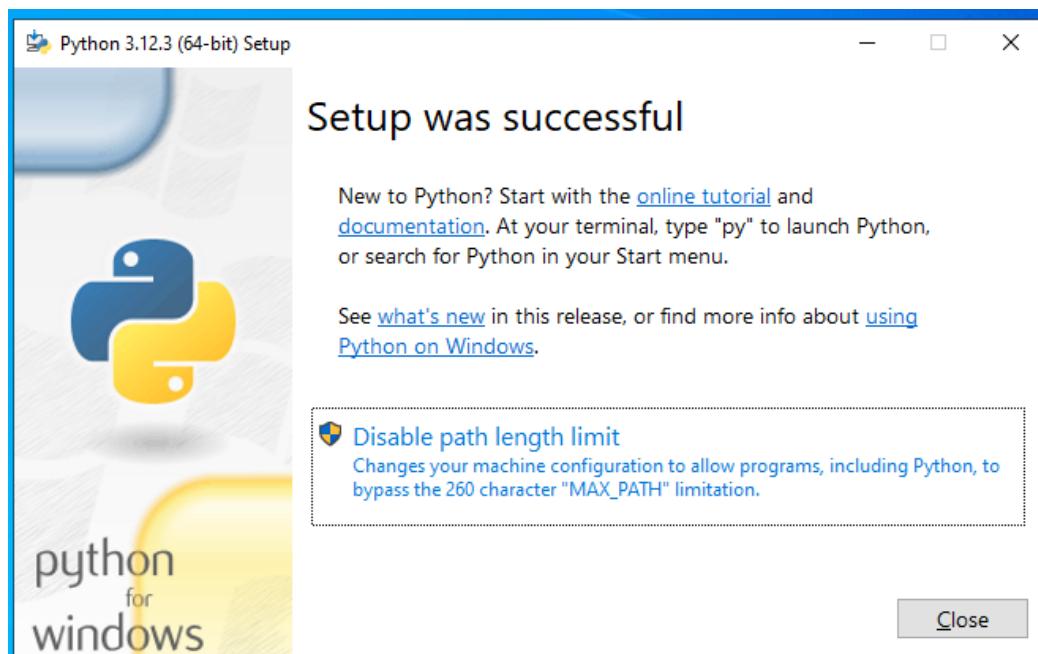
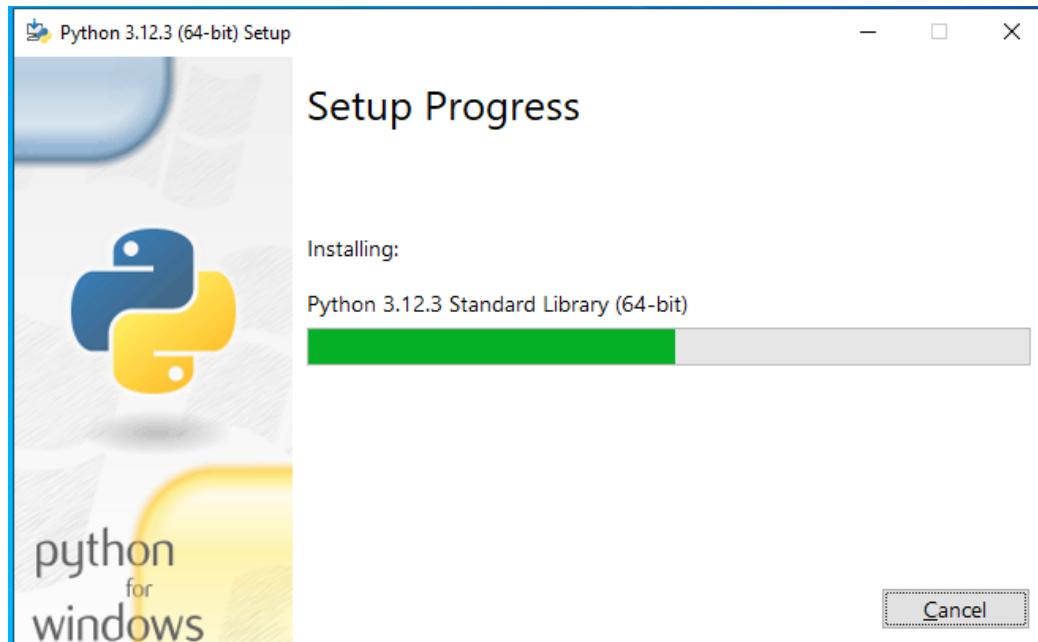
Date of Submission: 13-05-2024

Submitted to: Vikul

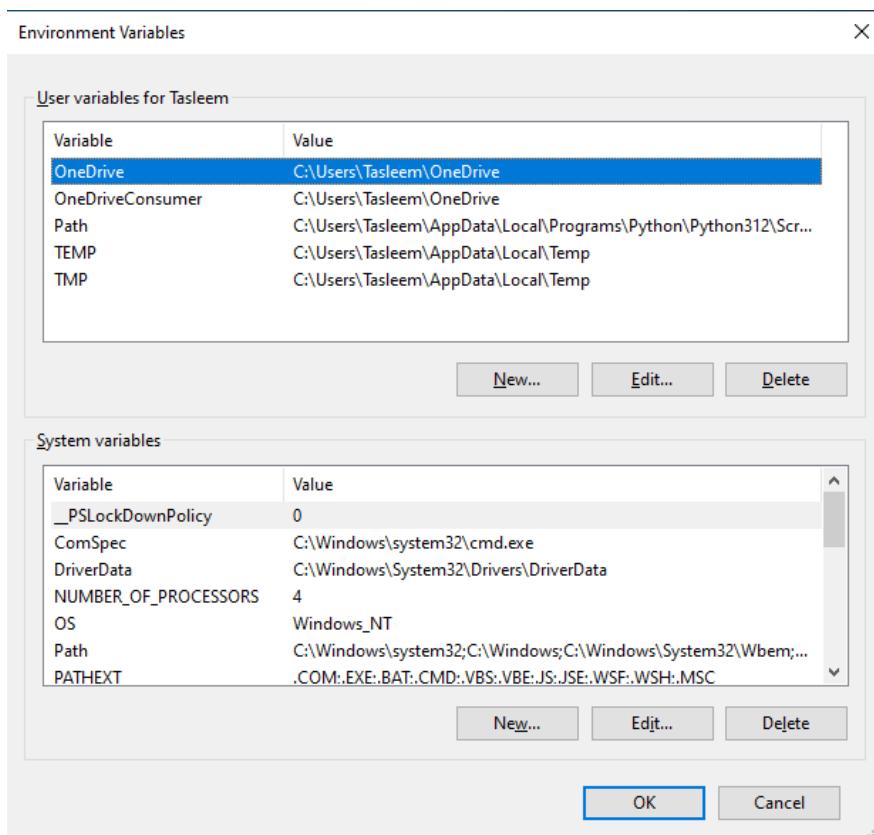
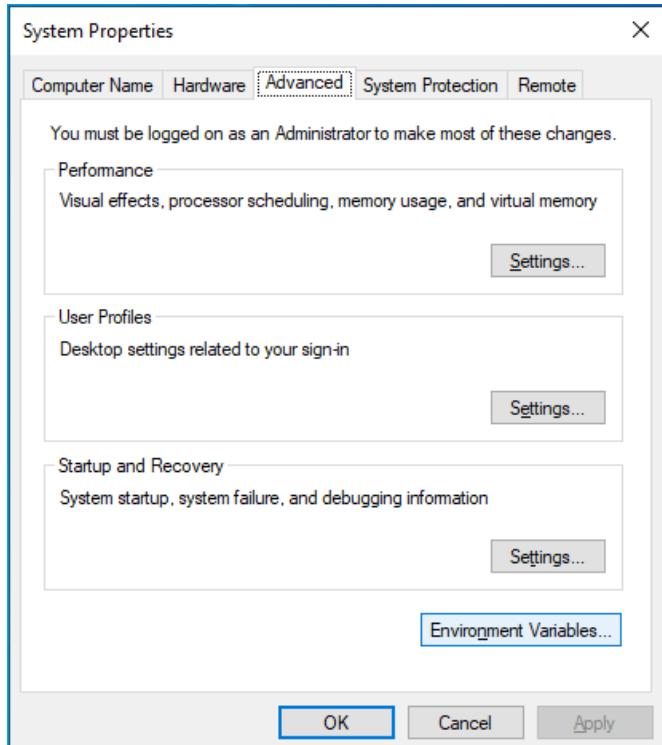
L1 - In Local Windows Machine Install and setup Environment Variable for Python

Step 1: Go to “ Python.org ” and download the latest version. During the installation ensure to check the box that says “Add Python 3.x to PATH” at the beginning of the installation process. This option sets the environment variable for Python, which allows to run Python from any command prompt or PowerShell window.

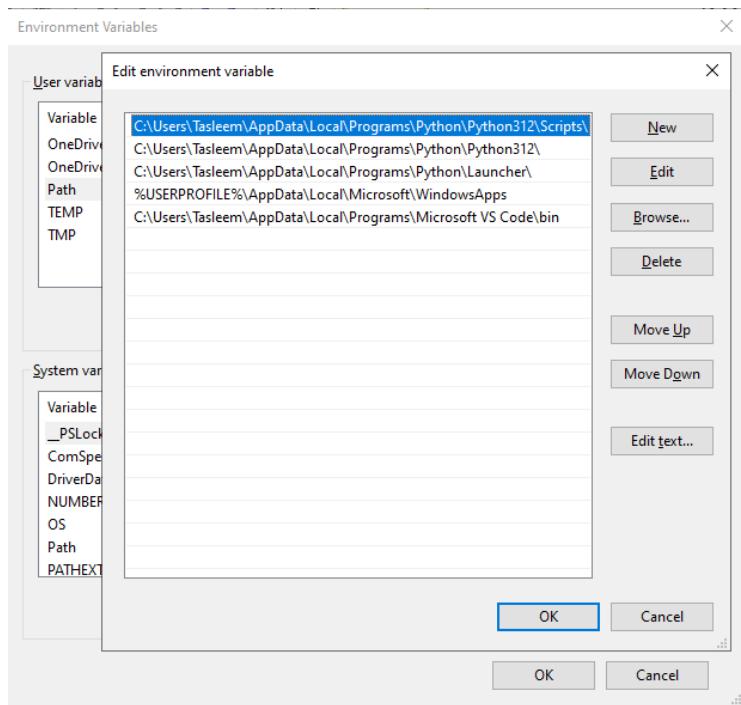




Step 2: To verify, Go to System Properties → Advanced tab → go to Environment Variables..



Go to Path,



We can see that Python is properly installed in the PATH Environment Variables..

And also we can verify using command prompt.

```
Microsoft Windows [Version 10.0.19045.4291]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Tasleem>python --version
```

The screenshot shows a Windows Command Prompt window. The title bar says 'Command Prompt'. The window displays the command 'python --version' entered at the prompt. The output shows the Python version information. At the bottom right of the window, there is a watermark that reads 'Activate Windows Go to Settings to activate Windows.'

```
Command Prompt - python
Microsoft Windows [Version 10.0.19045.4291]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Tasleem>python --version
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.

>>> 
```

Activate Windows
Go to Settings to activate Windows.

We can see that Python is successfully installed.

```
Command Prompt
Microsoft Windows [Version 10.0.19045.4291]
(c) Microsoft Corporation. All rights reserved.

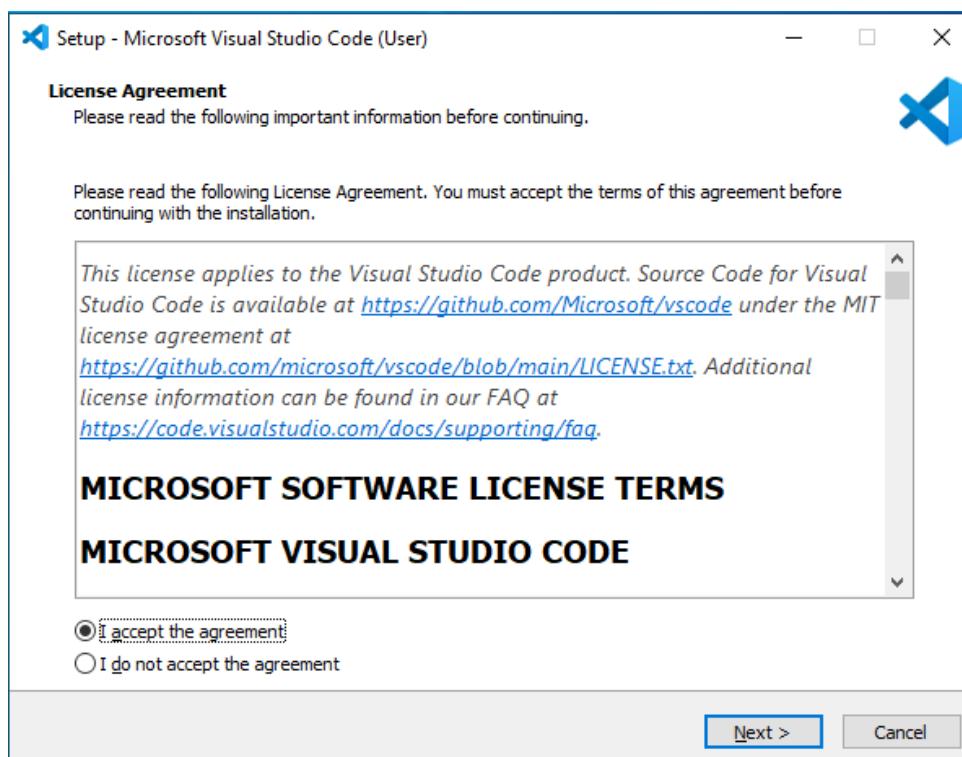
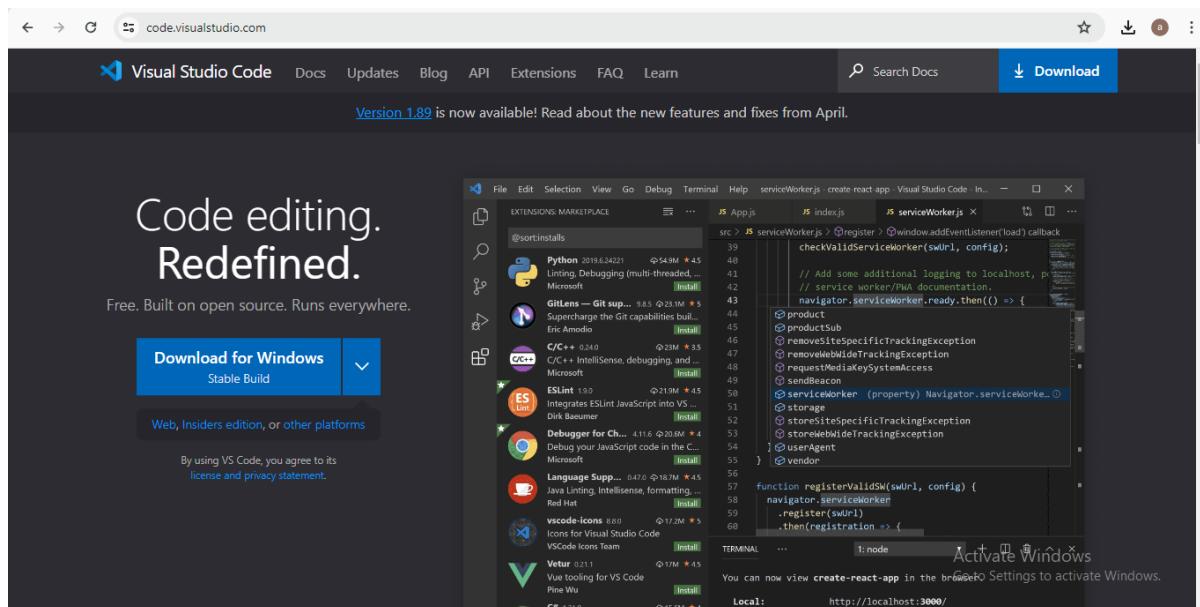
C:\Users\Tasleem>python --version
Python 3.12.3

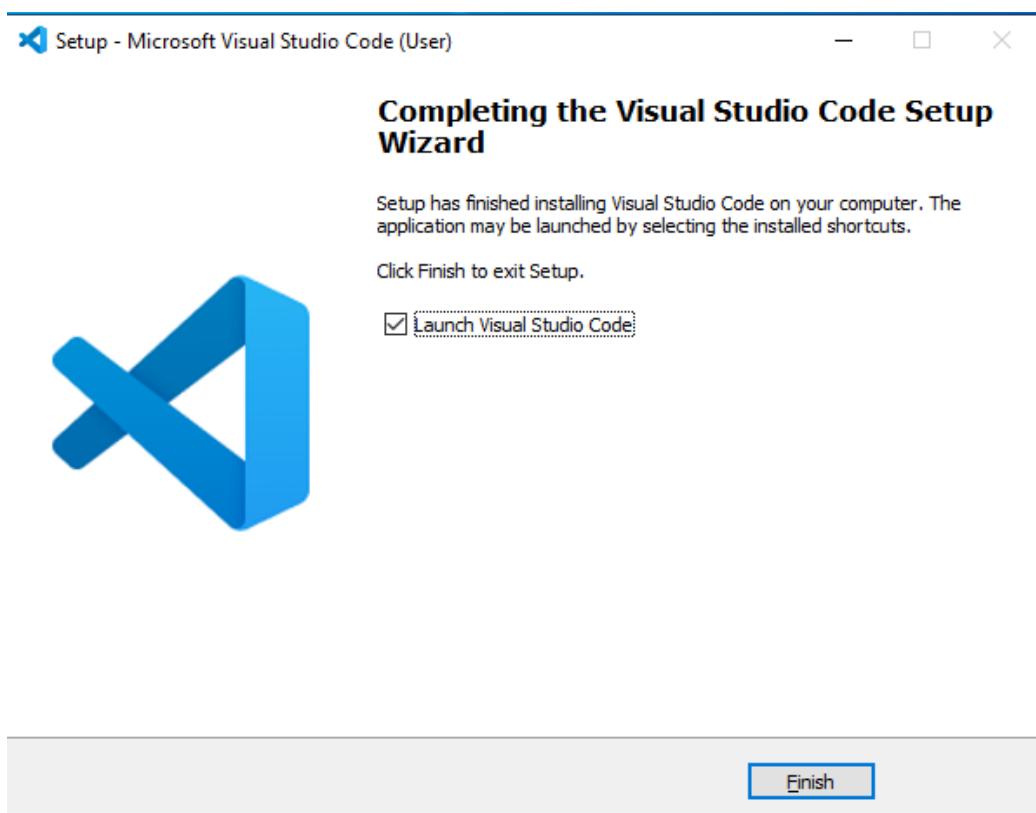
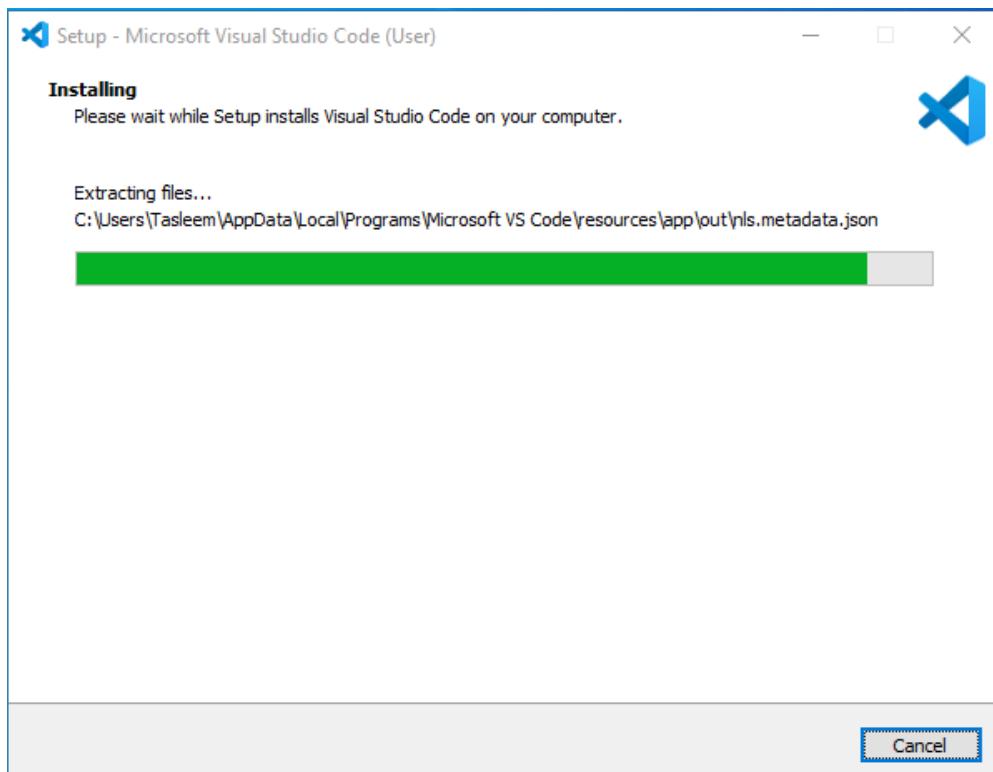
C:\Users\Tasleem>
```

Activate Windows
Go to Settings to activate Windows.

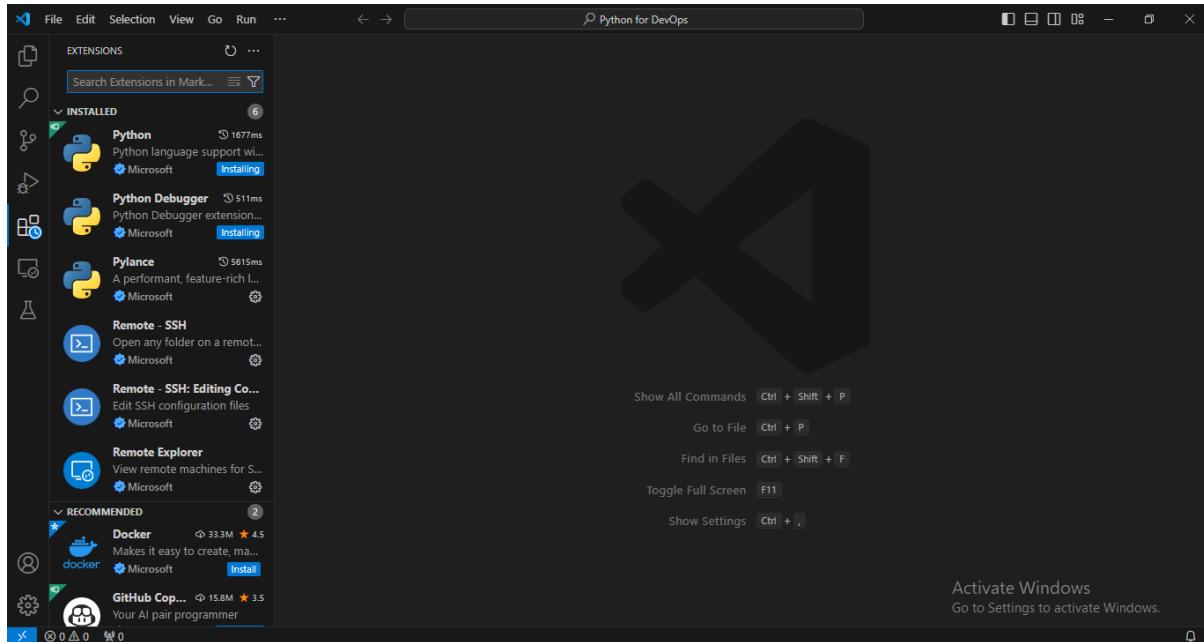
L2 - Install Visual Studio Code and Install Python and Terraform Extensions in VS Code

Step 1: Go to “code.visualstudio.com”, download & Install the VSC.





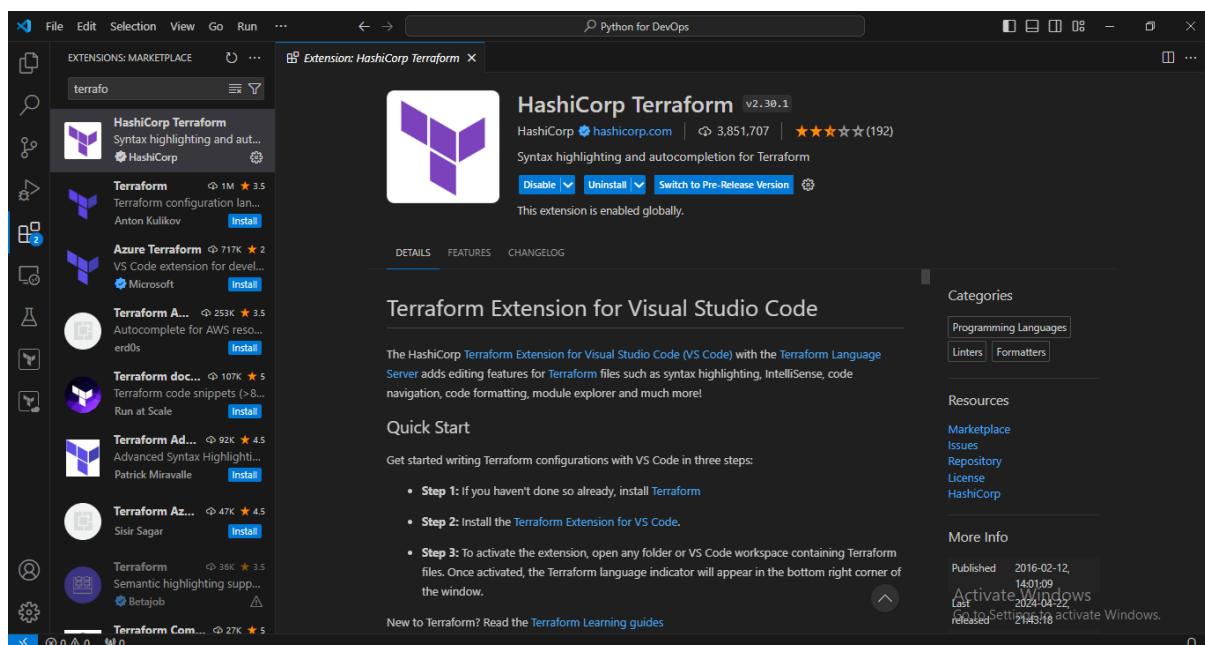
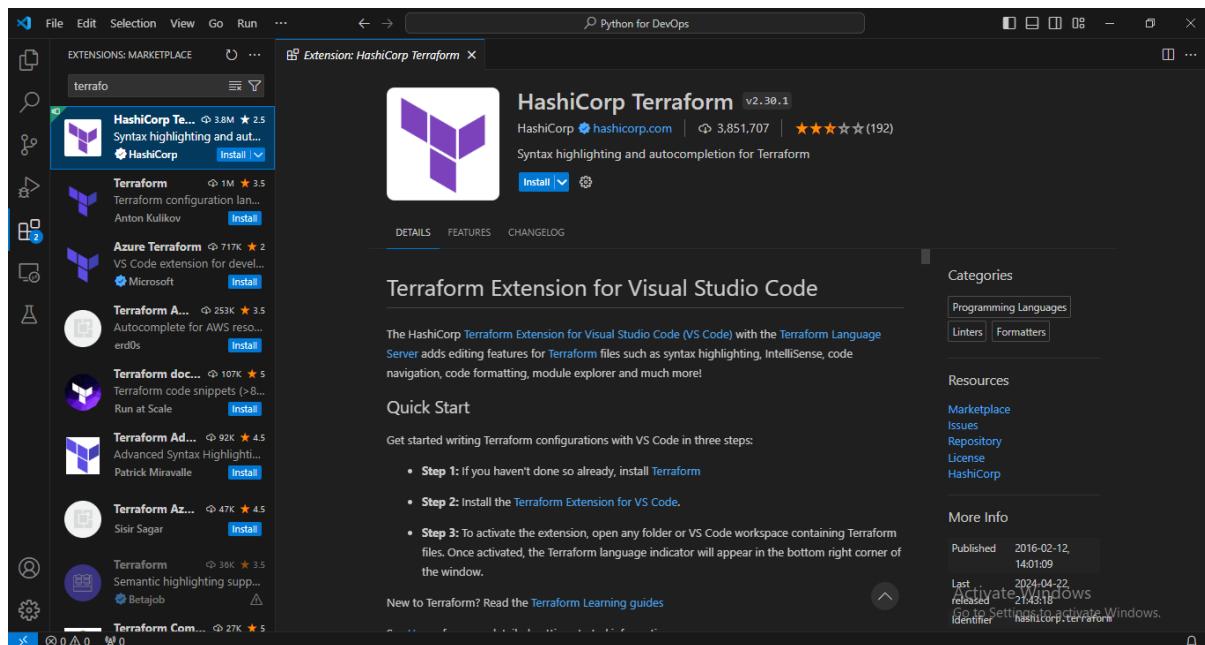
Step 2: Launch the VS Code and go to Extentions from left side panel.



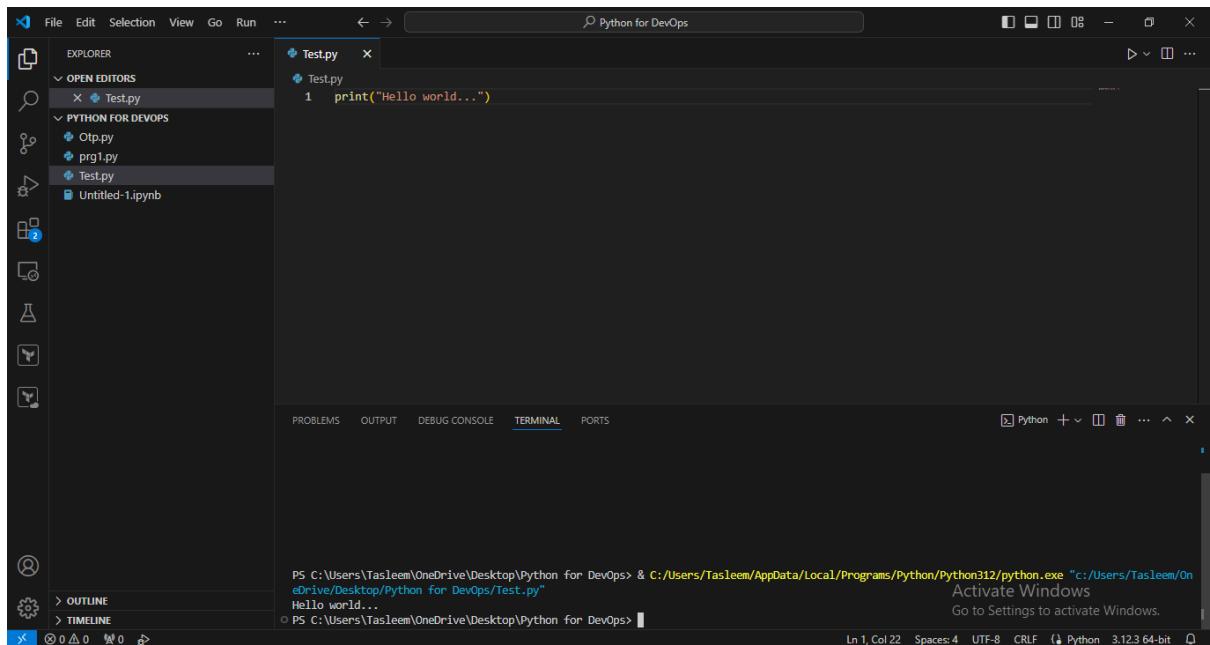
Step 3: Search for Python and click on Install to add the Python Extension to VS Code.



Step 4: Then search for Terraform and Install to add the Extension.



Step 5: To verify Python Extension, create a new file with .py extension “Test.py”. Open the new Terminal and Run the python file “print(“Hello world...”)”.



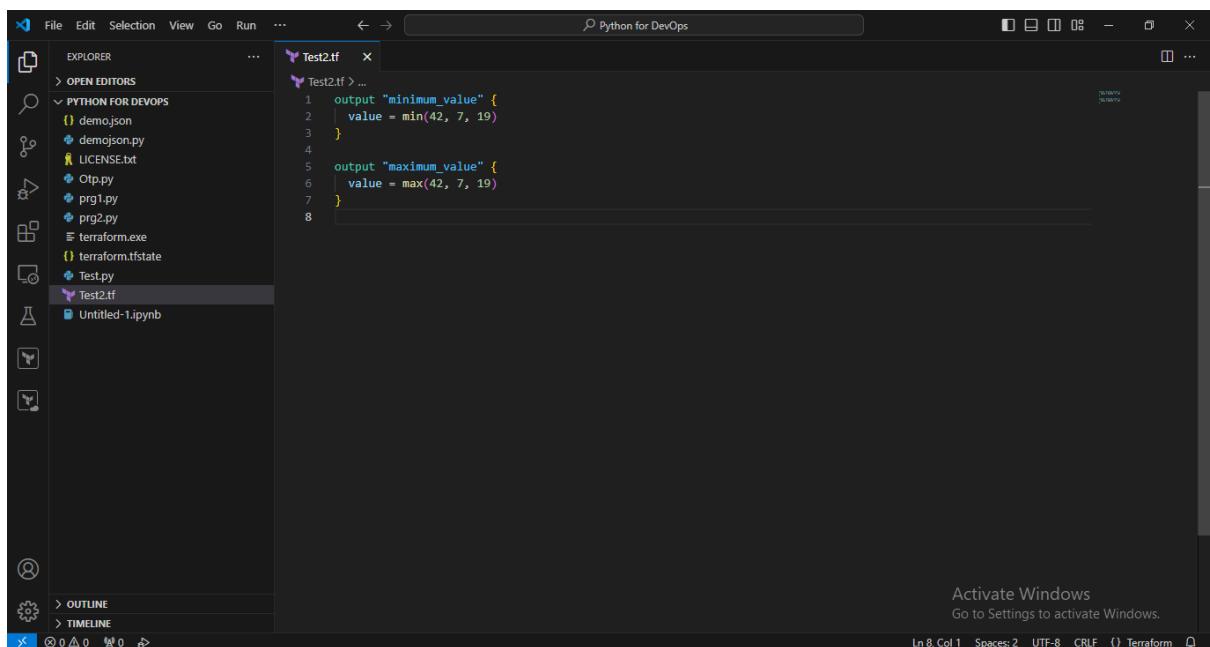
The screenshot shows the VS Code interface with the Python extension installed. The Explorer sidebar on the left lists files: Test.py, Otp.py, prg1.py, test.py, and Untitled-1.ipynb. The Test.py file is open in the editor, containing the code:

```
1 print("Hello world...")
```

. The Terminal tab at the bottom is active, showing the command PS C:\Users\Tasleem\OneDrive\Desktop\Python for DevOps & C:/Users/Tasleem/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/Tasleem/OneDrive/Desktop/Python for DevOps/Test.py" and the output Hello world... Below the terminal, status information includes Activate Windows, Go to Settings to activate Windows, Ln 1, Col 22, Spaces:4, UTF-8, CRLF, Python 3.12.3 64-bit.

We can see the output in the Terminal below.

Step 6: To verify Terraform Extension, create a new file with the name “Test2.tf” and execute the command “terraform init” to initialize Terraform



The screenshot shows the VS Code interface with the Terraform extension installed. The Explorer sidebar lists files: Test2.tf, demojson, LICENSE.txt, Otp.py, prg1.py, prg2.py, terraform.tfstate, Test.py, Test2.tf, and Untitled-1.ipynb. The Test2.tf file is open in the editor, containing the code:

```
1 output "minimum_value" {  
2   value = min(42, 7, 19)  
3 }  
4  
5 output "maximum_value" {  
6   value = max(42, 7, 19)  
7 }
```

. The Terminal tab at the bottom is active, showing the command PS C:\Users\Tasleem\OneDrive\Desktop\Python for DevOps & C:/Users/Tasleem/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/Tasleem/OneDrive/Desktop/Python for DevOps/Test2.tf" and the output 7 42 19. Below the terminal, status information includes Activate Windows, Go to Settings to activate Windows, Ln 8, Col 1, Spaces:2, UTF-8, CRLF, Terraform.

The screenshot shows the VS Code interface with the terminal tab selected. The terminal window displays the command 'terraform init' being run in the directory 'C:\Users\Tasleem\OneDrive\Desktop\Python for DevOps'. The output shows the initialization process: 'Initializing the backend...', 'Initializing provider plugins...', and 'Terraform has been successfully initialized!'. A message indicates that changes are required for the infrastructure. The status bar at the bottom right shows 'Activate Windows' and 'Ln 8, Col 1, Spaces: 2, UTF-8, CRLF'.

```
C:\Users\Tasleem\OneDrive\Desktop\Python for DevOps>terraform init
Initializing the backend...
Initializing provider plugins...
Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
Activate Windows
Go to Settings to activate Windows.
Ln 8, Col 1, Spaces: 2, UTF-8, CRLF
```

Then run the command “terraform apply”

The screenshot shows the VS Code interface with the terminal tab selected. The terminal window displays the command 'terraform apply' being run in the same directory. The output shows 'Changes to Outputs:' followed by two entries: '+ maximum_value = 42' and '+ minimum_value = 7'. It also states that the plan can be applied to save new output values to the state. A prompt asks if the user wants to perform these actions, with a note that 'Terraform will perform the actions described above.' and that 'Only 'yes'' will be accepted to approve. The status bar at the bottom right shows 'Activate Windows' and 'Ln 8, Col 1, Spaces: 2, UTF-8, CRLF'.

```
Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.

C:\Users\Tasleem\OneDrive\Desktop\Python for DevOps>terraform apply
Changes to Outputs:
+ maximum_value = 42
+ minimum_value = 7

You can apply this plan to save these new output values to the Terraform state, without changing any real infrastructure.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.
Activate Windows
Go to Settings to activate Windows.
Ln 8, Col 1, Spaces: 2, UTF-8, CRLF
```

We can see the output is correct. So we can enter yes to apply

```
1 output "minimum_value" {
2   value = min(42, 7, 19)
3 }
4
5 output "maximum_value" {
6   value = max(42, 7, 19)
7 }
```

C:\Users\Tasleem\OneDrive\Desktop\Python for DevOps>terraform apply

Changes to Outputs:

- + maximum_value = 42
- + minimum_value = 7

You can apply this plan to save these new output values to the Terraform state, without changing any real infrastructure.

Do you want to perform these actions?

Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

Activate Windows
Go to Settings to activate Windows.

In 8, Col 1 Spaces: 2 UTF-8 CRLF () Terraform

```
1 output "minimum_value" {
2   value = min(42, 7, 19)
3 }
4
5 output "maximum_value" {
6   value = max(42, 7, 19)
7 }
```

Do you want to perform these actions?

Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.

Outputs:

```
maximum_value = 42
minimum_value = 7
```

Activate Windows
Go to Settings to activate Windows.

In 8, Col 1 Spaces: 2 UTF-8 CRLF () Terraform

We can see the Output in the above snip.

Maximun_value = 42

Minimum_value = 7

We can also see the Terraform version : v1.8.3

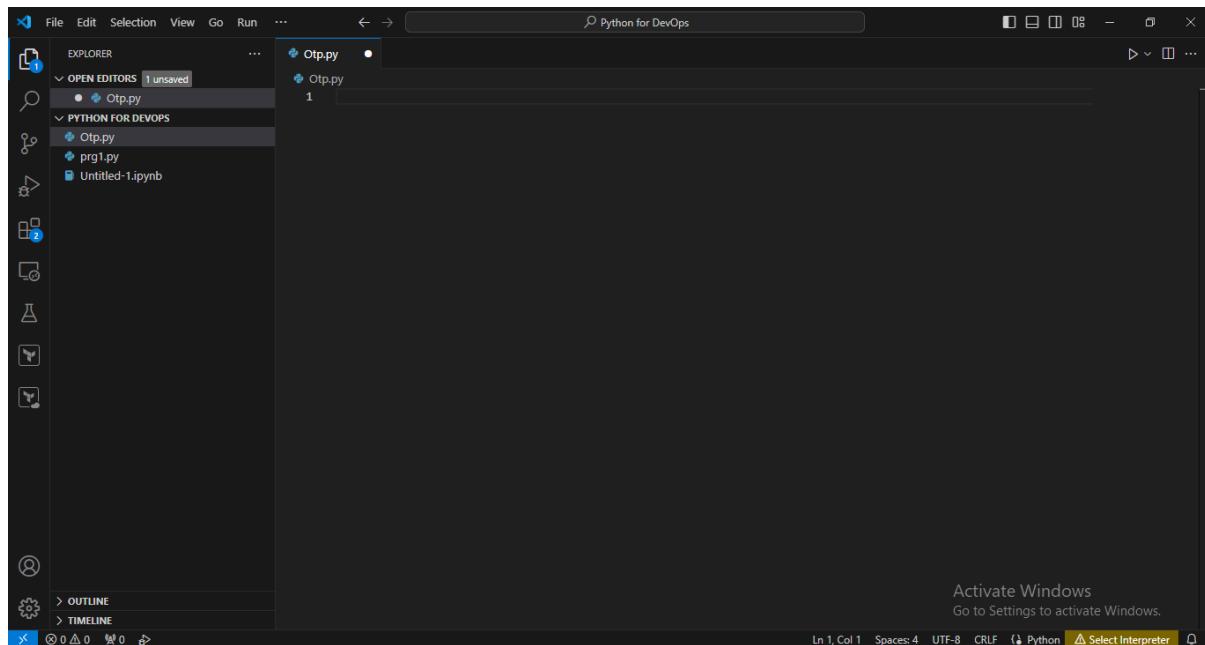
The screenshot shows the Visual Studio Code interface with the following details:

- Explorer View:** Shows files in the workspace, including `Test2.tf`, `demo.json`, `demojson.py`, `LICENSE.txt`, `Otp.py`, `prg1.py`, `prg2.py`, `terraform.exe`, `terraform.tfstate`, `Test.py`, `test2.tf`, and `Untitled-1.ipynb`.
- Terminal View:** Displays the output of the `terraform version` command:

```
C:\Users\Tasleem\OneDrive\Desktop\Python for DevOps>terraform version
Terraform v1.8.3
on windows_amd64
```
- Status Bar:** Shows the path `C:\Users\Tasleem\OneDrive\Desktop\Python for DevOps`, file `Test2.tf`, line 8, column 1, spaces: 2, encoding: UTF-8, and CRLF.

L3 - Create Python Console Application to randomly generate OTP kind of secure code.

Step 1: Go to VSC and create a new file “Otp.py”



Step 2: enter the programs as shown below.

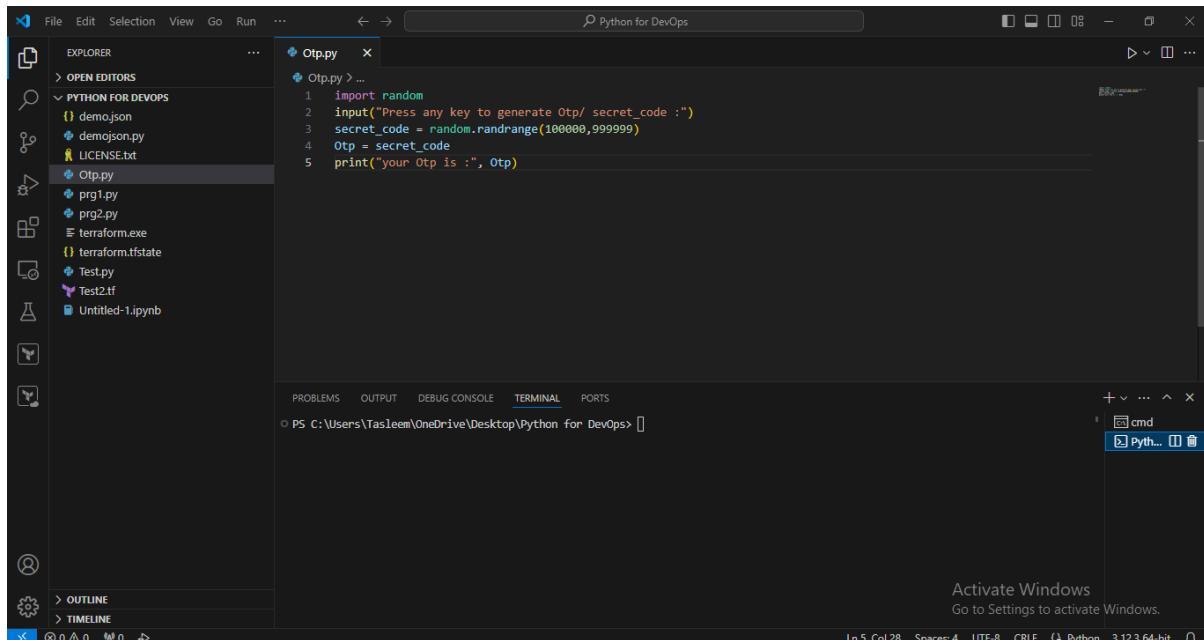
A screenshot of the Visual Studio Code interface. The left sidebar shows a tree view with 'OPEN EDITORS' containing 'Otp.py' (1 unsaved). Below it is 'PYTHON FOR DEVOPS' with 'demo.json', 'demojson.py', 'LICENSE.txt', and 'Otp.py'. The main editor area contains the following Python code:

```
import random
input("Press any key to generate Otp/ secret_code :")
secret_code = random.randrange(100000,999999)
Otp = secret_code
print("your Otp is :", Otp)
```

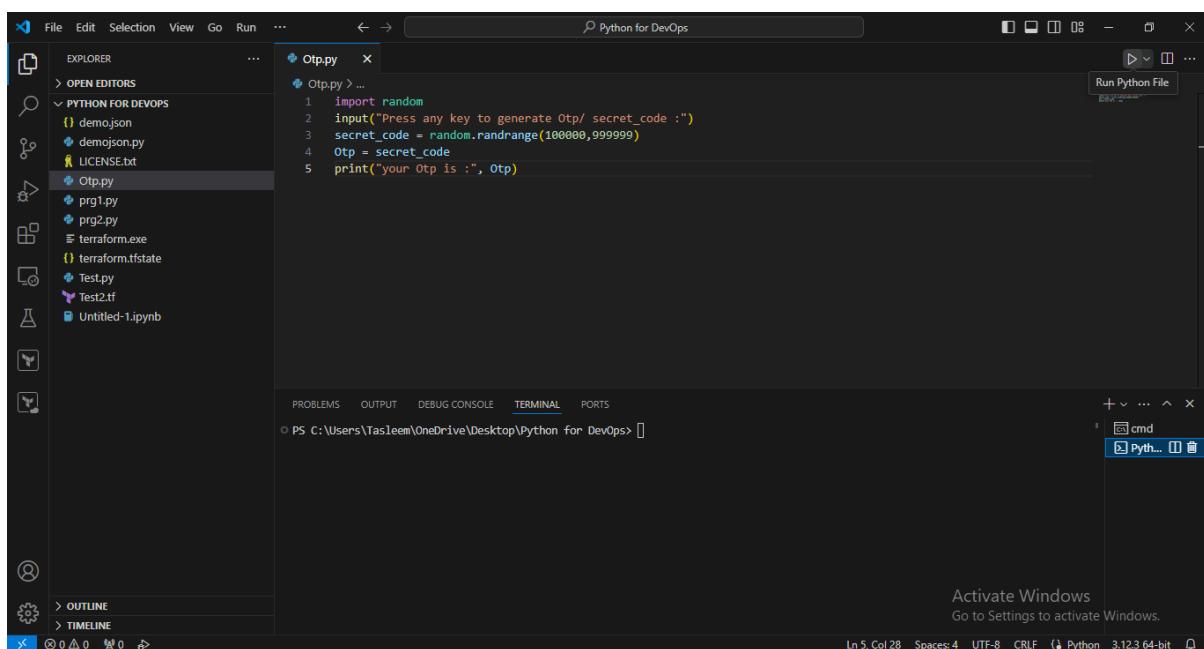
The status bar at the bottom right indicates 'Ln 5, Col 28' and 'Python 3.12.3 64-bit'.

```
import random
input("Press any key to generate Otp/ secret_code :")
secret_code = random.randrange(100000,999999)
Otp = secret_code
print("your Otp is :", Otp)
```

Step 3: Then open the new Terminal



Step 4: Run the python file



The screenshot shows the Visual Studio Code interface with a dark theme. The Explorer sidebar on the left lists files including 'Otp.py', 'demo.json', 'LICENSE.txt', 'prg1.py', 'prg2.py', 'terraform.exe', 'Test.py', 'Test2.tf', and 'Untitled-1.ipynb'. The main editor window displays the code for 'Otp.py':

```
1 import random
2 input("Press any key to generate Otp/ secret_code :")
3 secret_code = random.randrange(100000,999999)
4 Otp = secret_code
5 print("your Otp is :", Otp)
```

The Terminal tab is active, showing the command line output:

```
PS C:\Users\Tasleem\OneDrive\Desktop\Python for DevOps> & C:/Users/Tasleem/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/Tasleem/OneDrive/Desktop/Python for DevOps/Otp.py"
PS C:\Users\Tasleem\OneDrive\Desktop\Python for DevOps> & C:/Users/Tasleem/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/Tasleem/OneDrive/Desktop/Python for DevOps/Otp.py"
Press any key to generate Otp/ secret_code :[]
```

The status bar at the bottom indicates: Ln 5, Col 28, Spaces: 4, UTF-8, CRLF, Python 3.12.3 64-bit.

Press any key to generate Otp : give any key and press enter. For Example used “ Hi ”

The screenshot shows the Visual Studio Code interface with a dark theme. The Explorer sidebar on the left lists files including 'Otp.py', 'demo.json', 'LICENSE.txt', 'prg1.py', 'prg2.py', 'terraform.exe', 'Test.py', 'Test2.tf', and 'Untitled-1.ipynb'. The main editor window displays the code for 'Otp.py':

```
1 import random
2 input("Press any key to generate Otp/ secret_code :")
3 secret_code = random.randrange(100000,999999)
4 Otp = secret_code
5 print("your Otp is :", Otp)
```

The Terminal tab is active, showing the command line output:

```
PS C:\Users\Tasleem\OneDrive\Desktop\Python for DevOps> & C:/Users/Tasleem/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/Tasleem/OneDrive/Desktop/Python for DevOps/Otp.py"
PS C:\Users\Tasleem\OneDrive\Desktop\Python for DevOps> & C:/Users/Tasleem/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/Tasleem/OneDrive/Desktop/Python for DevOps/Otp.py"
Press any key to generate Otp/ secret_code :Hi
your Otp is : 597494
PS C:\Users\Tasleem\OneDrive\Desktop\Python for DevOps>
```

The status bar at the bottom indicates: Ln 5, Col 28, Spaces: 4, UTF-8, CRLF, Python 3.12.3 64-bit.

We can see the Otp has been generated.

The screenshot shows the Visual Studio Code interface with the following details:

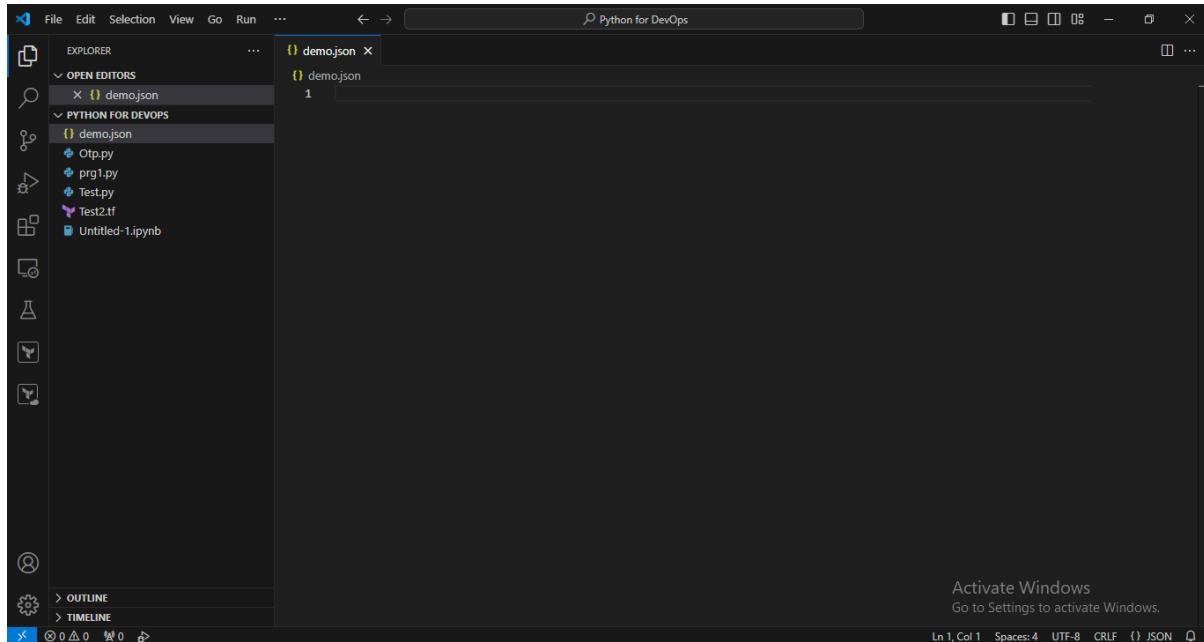
- File Explorer:** Shows a folder named "PYTHON FOR DEVOPS" containing files: demo.json, demajson.py, LICENSE.txt, Otp.py, prg1.py, prg2.py, terraform.exe, terraform.tfstate, Test.py, Test2.tf, and Untitled-1.ipynb.
- Terminal:** The active tab is "TERMINAL". It displays three command-line sessions (PS) showing the execution of the "Otp.py" script. The script generates random numbers between 100000 and 999999. The output from the first session is:

```
PS C:\Users\Tasleem\OneDrive\Desktop\Python for DevOps> & C:/Users/Tasleem/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/Tasleem/OneDrive/Desktop/Python for DevOps/Otp.py"
Press any key to generate Otp/ secret_code :Hi
your Otp is : 597494
```
- Status Bar:** Shows "Ln 5, Col 28" and "Spaces: 4" and "UTF-8" and "CRLF" and "Python 3.12.3 64-bit".

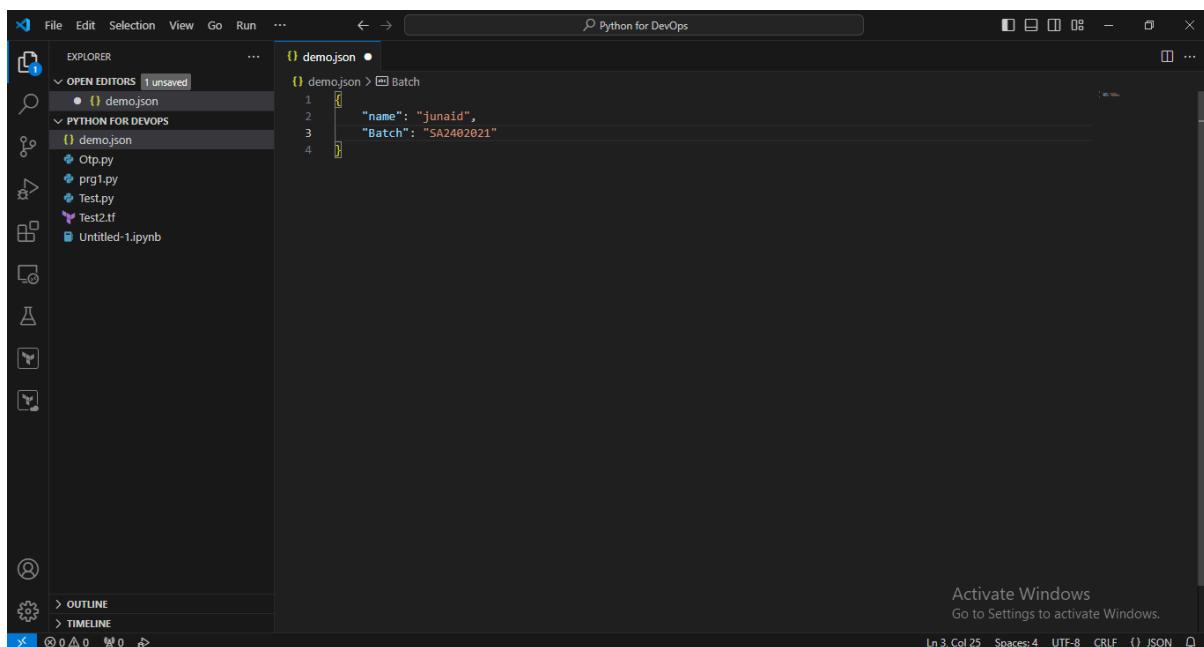
Here we can see how many times we run the file, we get random numbers as we are using the `#random` module.

L4 - Create Python Console Application to read the contents of .json file and print in the VS Code python console output

Step 1: Create a Json file “Demo.json”



The screenshot shows the VS Code interface with a dark theme. The Explorer sidebar on the left lists several files: 'demo.json' (marked with a yellow icon), 'Otp.py', 'prg1.py', 'Test.py', 'Test2.tf', and 'Untitled-1.ipynb'. The 'OPEN EDITORS' section shows 'demo.json' is open. The main editor area contains the text '1'. In the bottom right corner, there is a message: 'Activate Windows Go to Settings to activate Windows.' Below the status bar, it says 'Ln 1, Col 1 Spaces:4 UTF-8 CRLF {} JSON'.



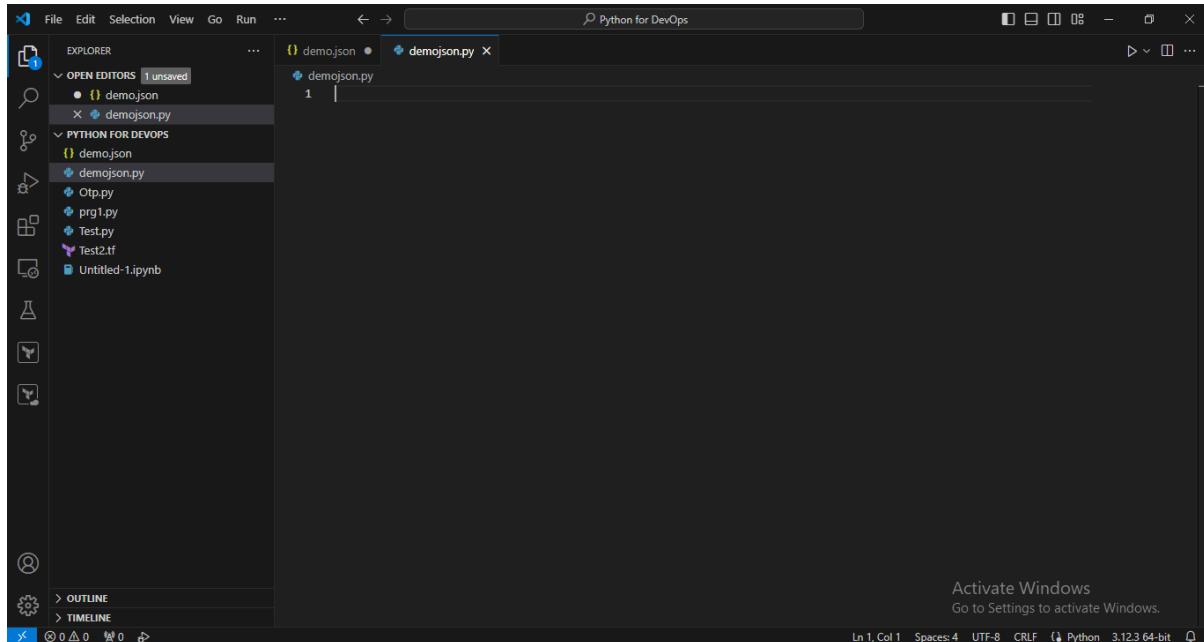
The screenshot shows the VS Code interface with a dark theme. The Explorer sidebar lists 'demo.json' (marked with a yellow icon) as an unsaved file. The main editor area displays the following JSON code:

```
[{"name": "junaid", "Batch": "SA2402021"}]
```

In the bottom right corner, there is a message: 'Activate Windows Go to Settings to activate Windows.' Below the status bar, it says 'Ln 3, Col 25 Spaces:4 UTF-8 CRLF {} JSON'.

Created a Json file with the program in it.

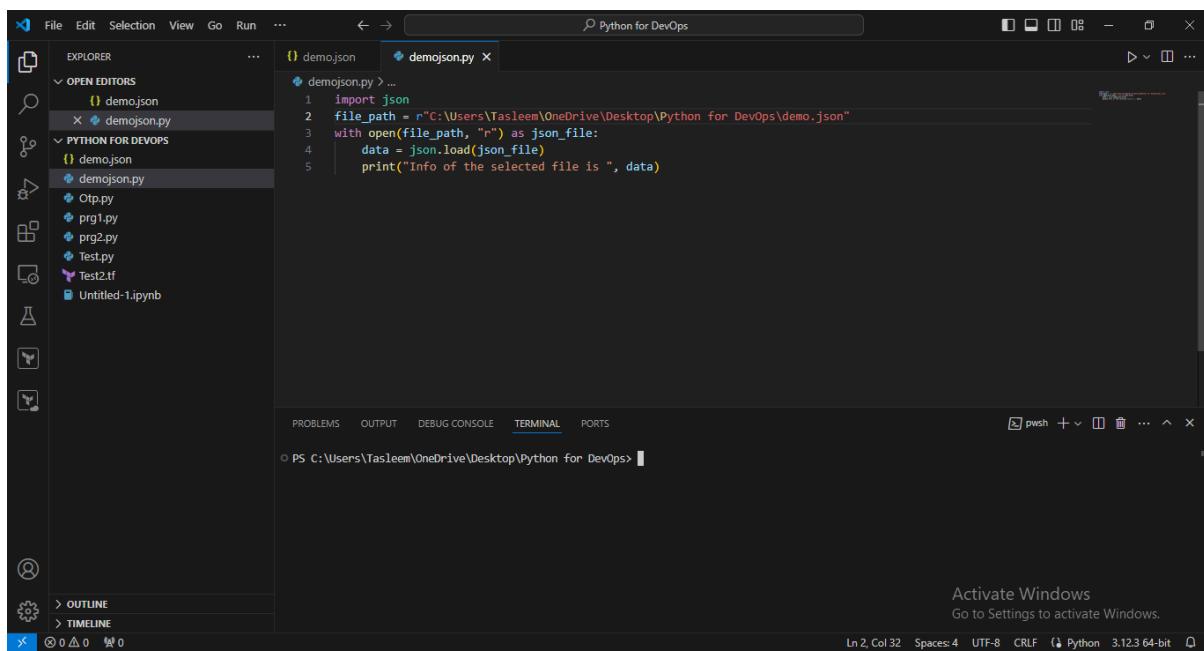
Step 2: Then create a Python file with the name “demojson.py”



The screenshot shows the Visual Studio Code interface with a dark theme. The Explorer sidebar on the left lists several files: demo.json, demojson.py (which is currently selected), demo.json, Otp.py, prg1.py, prg2.py, Test.py, Test2.tf, and Untitled-1.ipynb. The main editor area contains the code for demojson.py, which is currently empty, showing only the number '1' on the first line. The status bar at the bottom right indicates 'Ln 1, Col 1' and other settings like 'Spaces: 4', 'UTF-8', 'CRLF', 'Python 3.12.3 64-bit'.

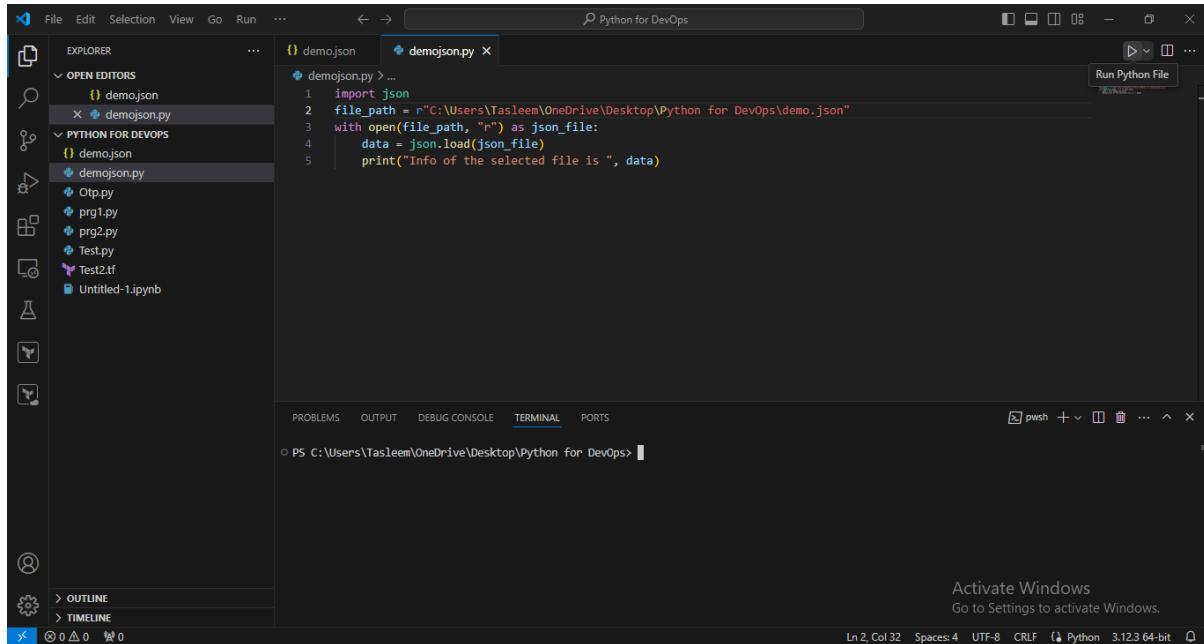
Step 3: Then enter the program as shown below

```
Import json
file_path = r"C:\Users\Tasleem\OneDrive\Desktop\Python for DevOps\demo.json"
with open(file_path,"r") as json_file:
    data = json.load(json_file)
    print("Info of the selected file is ", data)
```



The screenshot shows the Visual Studio Code interface with a dark theme. The Explorer sidebar on the left lists the same files as before. The main editor area now contains the complete Python script for demojson.py. Below the editor, a terminal window is open, showing the command 'PS C:\Users\Tasleem\OneDrive\Desktop\Python for DevOps>' followed by a blank line. The status bar at the bottom right indicates 'Ln 2, Col 32' and other settings like 'Spaces: 4', 'UTF-8', 'CRLF', 'Python 3.12.3 64-bit'.

Step 4: Run the python file using Run option on top right side



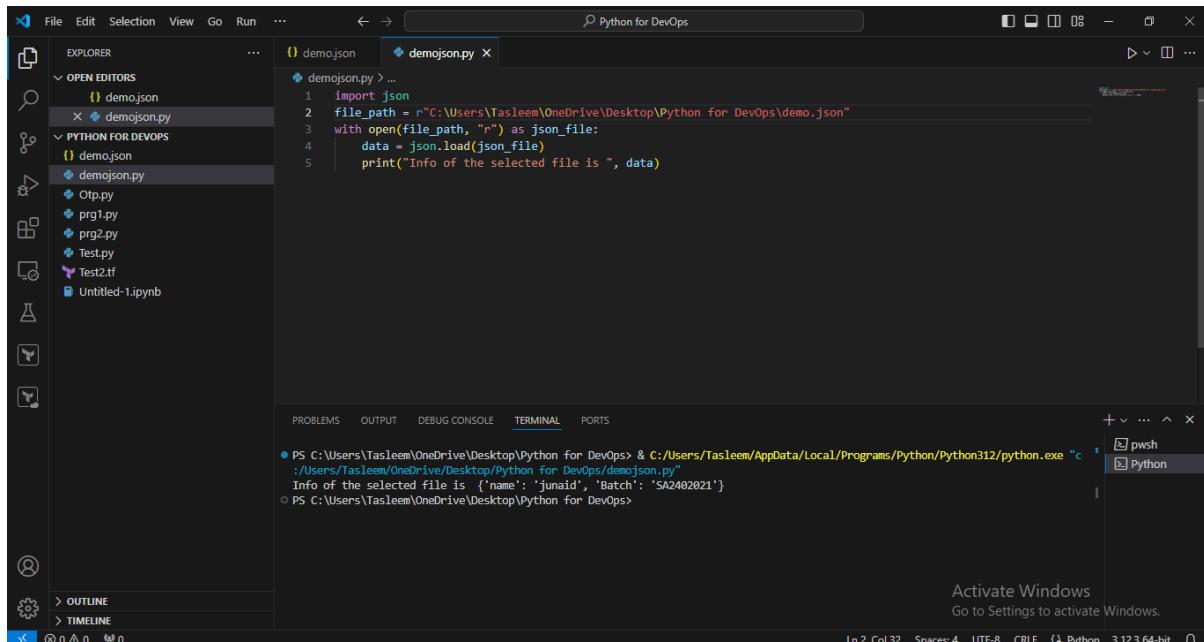
```
demo.json demojson.py
demojson.py > ...
1 import json
2 file_path = r"C:\Users\Tasleem\OneDrive\Desktop\Python for DevOps\demo.json"
3 with open(file_path, "r") as json_file:
4     data = json.load(json_file)
5     print("Info of the selected file is ", data)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Tasleem\OneDrive\Desktop\Python for DevOps>

Activate Windows
Go to Settings to activate Windows.

Ln 2, Col 32 Spaces:4 UTF-8 CRLF Python 3.12.3 64-bit
```

In Terminal we can see the Output. i.e, content of json file



```
demo.json demojson.py
demojson.py > ...
1 import json
2 file_path = r"C:\Users\Tasleem\OneDrive\Desktop\Python for DevOps\demo.json"
3 with open(file_path, "r") as json_file:
4     data = json.load(json_file)
5     print("Info of the selected file is ", data)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Tasleem\OneDrive\Desktop\Python for DevOps> & C:/Users/Tasleem/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/Tasleem/OneDrive/Desktop/Python for DevOps/demojson.py"
Info of the selected file is {"name": "junaid", "Batch": "SA2402021"}
PS C:\Users\Tasleem\OneDrive\Desktop\Python for DevOps>

Activate Windows
Go to Settings to activate Windows.

Ln 2, Col 32 Spaces:4 UTF-8 CRLF Python 3.12.3 64-bit
```

L5 - Create Python Web Application to using Flask Web Application Framework

Step 1: Go to EC2 Dashboard → Launch Instance

The screenshot shows the AWS EC2 Dashboard. On the left, there's a sidebar with options like Instances, Images, and Elastic Block Store. The main area has a 'Resources' summary with counts for various EC2 components. Below it is a 'Launch instance' section with a large orange 'Launch instance' button. To the right, there's a 'EC2 Free Tier' summary and a 'Service health' section.

Step 2: Create an Instance “Flaskwebserver” and enable “Allow HTTPS traffic from the internet” and “Allow HTTP traffic from the internet” in Network settings. Then launch the instance.

The screenshot shows the 'Launch instances' wizard. Step 1 is 'Name and tags'. It has a 'Name' input field where 'Flaskwebserver' is typed. Other tabs like 'Add additional tags' and 'Advanced details' are visible. To the right, there's a 'Summary' section with fields for 'Number of instances' (set to 1), 'Software Image (AMI)', 'Virtual server type (instance type)', 'Firewall (security group)', 'Storage (volumes)', and a large 'Launch instance' button.

ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#LaunchInstances:

aws Services Search [Alt+S]

Junaid Adil ▾

Network Info
vpc-0a146ac274a0f56f1

Subnet | Info
No preference (Default subnet in any availability zone)

Auto-assign public IP | Info
Enable

Additional charges apply when outside of free tier allowance

Firewall (security groups) | Info
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group Select existing security group

We'll create a new security group called 'launch-wizard-13' with the following rules:

Allow SSH traffic from Anywhere
Helps you connect to your instance
0.0.0.0/0

Allow HTTPS traffic from the internet
To set up an endpoint, for example when creating a web server

Allow HTTP traffic from the internet
To set up an endpoint, for example when creating a web server

▼ Summary

Number of instances | Info
1

Software Image (AMI)
Canonical, Ubuntu, 24.04 LTS, ...read more
ami-0f58b397bc5c1f2e8

Virtual server type (instance type)
t2.micro

Firewall (security group)
New security group

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

Cancel **Launch instance** Review commands

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#LaunchInstances:

aws Services Search [Alt+S]

Junaid Adil ▾

EC2 > Instances > Launch an instance

Launching instance
Creating security group rules

21%

▶ Details

Please wait while we launch your instance.
Do not close your browser while this is loading.

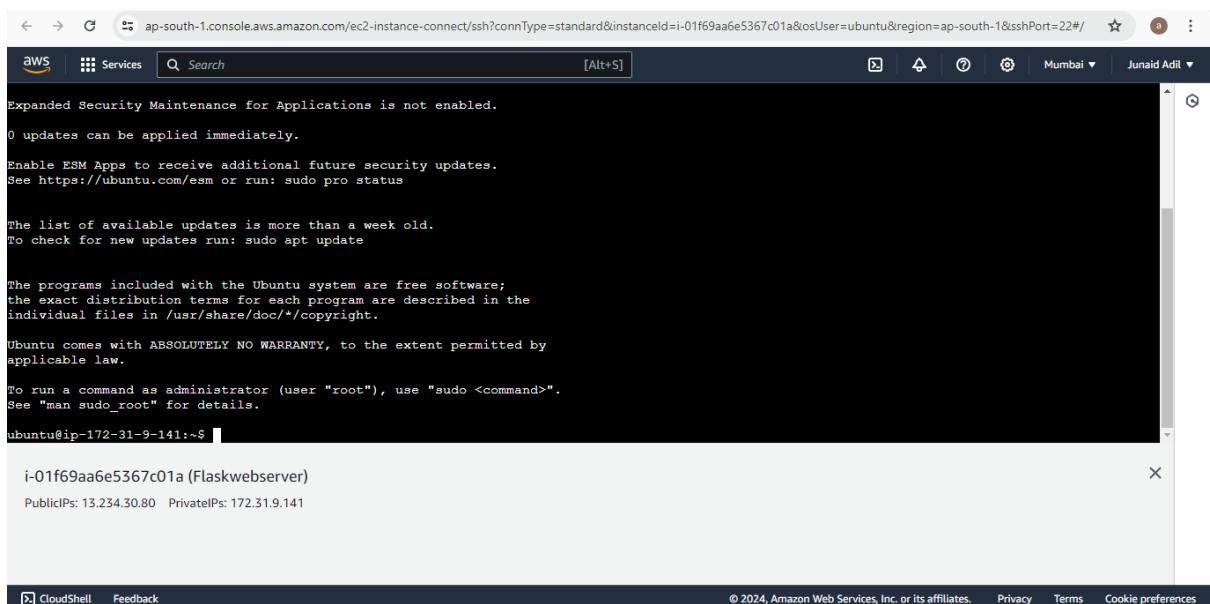
CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with options like EC2 Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity, Reservations, Images (AMIs, AMI Catalog), and Elastic Block Store. The main content area displays a table titled 'Instances (1) Info'. It shows one instance: 'Flaskwebserver' with Instance ID 'i-01f69aa6e5367c01a', which is 'Running' (indicated by a green circle), has an 'Instance type' of 't2.micro', and is in the 'Initializing' status check state. The 'View alarms' and 'Availability' buttons are also visible. Below the table, a modal window titled 'Select an instance' is open, showing the same instance information.

Step 3: Connect to Instance

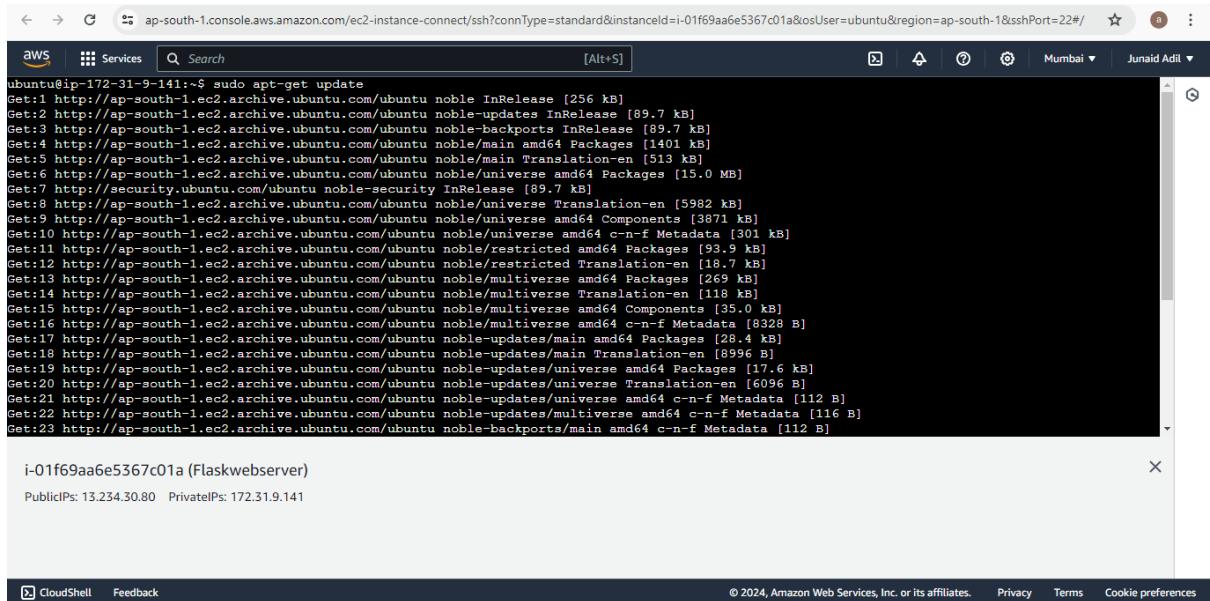
The screenshot shows the 'EC2 Instance Connect' page for the instance 'i-01f69aa6e5367c01a'. At the top, it says 'Connect to your instance i-01f69aa6e5367c01a (Flaskwebserver) using any of these options'. Below this, there are tabs for 'EC2 Instance Connect', 'Session Manager', 'SSH client', and 'EC2 serial console'. The 'EC2 Instance Connect' tab is selected. A warning message in a yellow box states: 'Port 22 (SSH) is open to all IP addresses. Port 22 (SSH) is currently open to all IP addresses, indicated by 0.0.0.0/0 in the inbound rule in [your security group](#). For increased security, consider restricting access to only the EC2 Instance Connect service IP addresses for your Region: 13.233.177.0/29. [Learn more](#)'.

Below the warning, the 'Instance ID' is listed as 'i-01f69aa6e5367c01a (Flaskwebserver)'. Under 'Connection Type', there are two options: 'Connect using EC2 Instance Connect' (selected, highlighted in blue) and 'Connect using EC2 Instance Connect Endpoint'. The 'Public IP address' field contains '13.234.30.80'. The 'Username' field is set to 'ubuntu'. At the bottom, there are links for 'CloudShell' and 'Feedback'.



```
Expanded Security Maintenance for Applications is not enabled.  
0 updates can be applied immediately.  
Enable ESM Apps to receive additional future security updates.  
See https://ubuntu.com/esm or run: sudo pro status  
  
The list of available updates is more than a week old.  
To check for new updates run: sudo apt update  
  
The programs included with the Ubuntu system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*copyright.  
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by  
applicable law.  
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.  
ubuntu@ip-172-31-9-141:~$ [REDACTED]  
  
i-01f69aa6e5367c01a (Flaskwebserver)  
PublicIPs: 13.234.30.80 PrivateIPs: 172.31.9.141
```

Step 4: Run the command “**`sudo apt-get update`**”. To install Python Virtual environment run command “**`sudo apt-get install python3-venv`**”.



```
ubuntu@ip-172-31-9-141:~$ sudo apt-get update  
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease [256 kB]  
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [89.7 kB]  
Get:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [89.7 kB]  
Get:4 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 Packages [1401 kB]  
Get:5 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/main translation-en [513 kB]  
Get:6 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]  
Get:7 http://security.ubuntu.com/ubuntu noble-security InRelease [89.7 kB]  
Get:8 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]  
Get:9 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]  
Get:10 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]  
Get:11 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/restricted amd64 Packages [93.9 kB]  
Get:12 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/restricted Translation-en [18.7 kB]  
Get:13 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]  
Get:14 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]  
Get:15 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Components [35.0 kB]  
Get:16 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 c-n-f Metadata [8328 B]  
Get:17 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [20.4 kB]  
Get:18 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [8996 B]  
Get:19 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [17.6 kB]  
Get:20 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [6096 B]  
Get:21 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 c-n-f Metadata [112 B]  
Get:22 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 c-n-f Metadata [116 B]  
Get:23 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 c-n-f Metadata [112 B]  
  
i-01f69aa6e5367c01a (Flaskwebserver)  
PublicIPs: 13.234.30.80 PrivateIPs: 172.31.9.141
```

```
← → ⚙ ap-south-1.console.aws.amazon.com/ec2-instance-connect/ssh?connType=standard&instanceId=i-01f69aa6e5367c01a&osUser=ubuntu&region=ap-south-1&sshPort=22#/ Junaid Adil ▾

aws Services Search [Alt+S] Mumbai ▾ Junaid Adil ▾

Get:14 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
Get:15 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Components [35.0 kB]
Get:16 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 c-n-f Metadata [8328 B]
Get:17 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [28.4 kB]
Get:18 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [8996 B]
Get:19 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [17.6 kB]
Get:20 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [6096 B]
Get:21 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 c-n-f Metadata [112 B]
Get:22 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 c-n-f Metadata [116 B]
Get:23 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 c-n-f Metadata [112 B]
Get:24 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Packages [3936 B]
Get:25 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe Translation-en [1392 B]
Get:26 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 c-n-f Metadata [116 B]
Get:27 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 c-n-f Metadata [116 B]
Get:28 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 c-n-f Metadata [116 B]
Get:29 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [26.7 kB]
Get:30 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [8124 B]
Get:31 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [10.6 kB]
Get:32 http://security.ubuntu.com/ubuntu noble-security/universe Translation-en [4596 B]
Get:33 http://security.ubuntu.com/ubuntu noble-security/universe amd64 c-n-f Metadata [112 B]
Get:34 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 c-n-f Metadata [116 B]
Fetched 28.3 MB in 5s (5168 kB/s)
Reading package lists... Done
ubuntu@ip-172-31-9-141:~$ sudo apt-get install python3-venv
```

i-01f69aa6e5367c01a (Flaskwebserver)

PublicIPs: 13.234.30.80 PrivateIPs: 172.31.9.141

```
← → ⚙ ap-south-1.console.aws.amazon.com/ec2-instance-connect/ssh?connType=standard&instanceId=i-01f69aa6e5367c01a&osUser=ubuntu&region=ap-south-1&sshPort=22#/ Junaid Adil ▾

aws Services Search [Alt+S] Mumbai ▾ Junaid Adil ▾

Unpacking python3-setuptools-whl (68.1.2-2ubuntu1) ...
Selecting previously unselected package python3.12-venv.
Preparing to unpack .../python3.12-venv_3.12.3-1_amd64.deb ...
Unpacking python3.12-venv (3.12.3-1) ...
Selecting previously unselected package python3-venv.
Preparing to unpack .../python3-venv_3.12.3-0ubuntu1_amd64.deb ...
Unpacking python3-venv (3.12.3-0ubuntu1) ...
Setting up python3-setuptools-whl (68.1.2-2ubuntu1) ...
Setting up python3-pip-whl (24.0+dfsg-1ubuntu1) ...
Setting up python3.12-venv (3.12.3-1) ...
Setting up python3-venv (3.12.3-0ubuntu1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-9-141:~$
```

i-01f69aa6e5367c01a (Flaskwebserver)

PublicIPs: 13.234.30.80 PrivateIPs: 172.31.9.141

Step 5: Create the Directory “helloworld” .

The screenshot shows a terminal window in the AWS CloudShell interface. The user has run the command `mkdir helloworld`. The terminal output shows the directory creation process and some system status information.

```
Unpacking python3-setuptools-whl (68.1.2-2ubuntu1) ...
Selecting previously unselected package python3.12-venv.
Preparing to unpack .../python3.12-venv_3.12.3-1_amd64.deb ...
Unpacking python3.12-venv (3.12.3-1) ...
Selecting previously unselected package python3-venv.
Preparing to unpack .../python3-venv_3.12.3-0ubuntu1_amd64.deb ...
Unpacking python3-venv (3.12.3-0ubuntu1) ...
Setting up python3-setuptools-whl (68.1.2-2ubuntu1) ...
Setting up python3-pip-whl (24.0.0+dfsg-1ubuntu1) ...
Setting up python3.12-venv (3.12.3-1) ...
Setting up python3-venv (3.12.3-0ubuntu1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.

ubuntu@ip-172-31-9-141:~$ mkdir helloworld
```

Below the terminal, the AWS CloudShell interface shows the instance details: i-01f69aa6e5367c01a (Flaskwebserver), PublicIPs: 13.234.30.80, PrivateIPs: 172.31.9.141.

The screenshot shows a terminal window in the AWS CloudShell interface. The user has run the command `cd helloworld`. The terminal output shows the directory change process and some system status information.

```
Preparing to unpack .../python3.12-venv_3.12.3-1_amd64.deb ...
Unpacking python3.12-venv (3.12.3-1) ...
Selecting previously unselected package python3-venv.
Preparing to unpack .../python3-venv_3.12.3-0ubuntu1_amd64.deb ...
Unpacking python3-venv (3.12.3-0ubuntu1) ...
Setting up python3-setuptools-whl (68.1.2-2ubuntu1) ...
Setting up python3-pip-whl (24.0.0+dfsg-1ubuntu1) ...
Setting up python3.12-venv (3.12.3-1) ...
Setting up python3-venv (3.12.3-0ubuntu1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

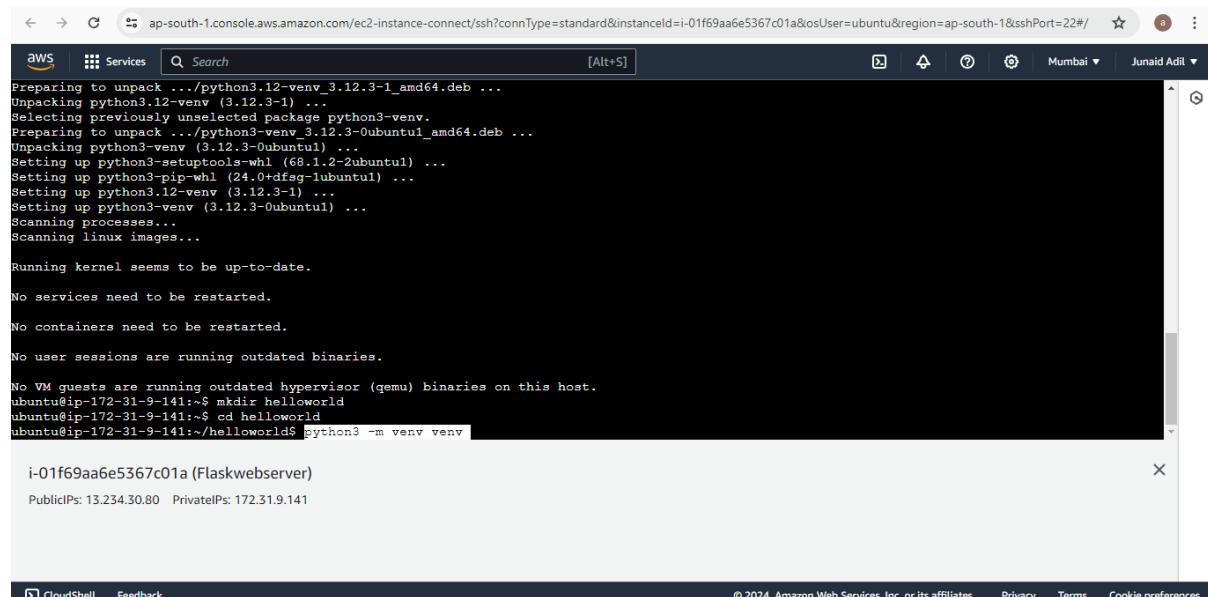
No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.

ubuntu@ip-172-31-9-141:~$ mkdir helloworld
ubuntu@ip-172-31-9-141:~$ cd helloworld
ubuntu@ip-172-31-9-141:~/helloworld$
```

Below the terminal, the AWS CloudShell interface shows the instance details: i-01f69aa6e5367c01a (Flaskwebserver), PublicIPs: 13.234.30.80, PrivateIPs: 172.31.9.141.

Step 6: Inside the directory create the Virtual Environment using command “**python3 -m venv venv**”



Preparing to unpack .../python3.12-venv_3.12.3-1_amd64.deb ...
Unpacking python3.12-venv (3.12.3-1) ...
Selecting previously unselected package python3-venv.
Preparing to unpack .../python3-venv_3.12.3-0ubuntu1_amd64.deb ...
Unpacking python3-venv (3.12.3-0ubuntu1) ...
Setting up python3-setuptools-whl (68.1.2-2ubuntu1) ...
Setting up python3-pip-whl (24.0+dfsg-1ubuntul) ...
Setting up python3.12-venv (3.12.3-1) ...
Setting up python3-venv (3.12.3-0ubuntu1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

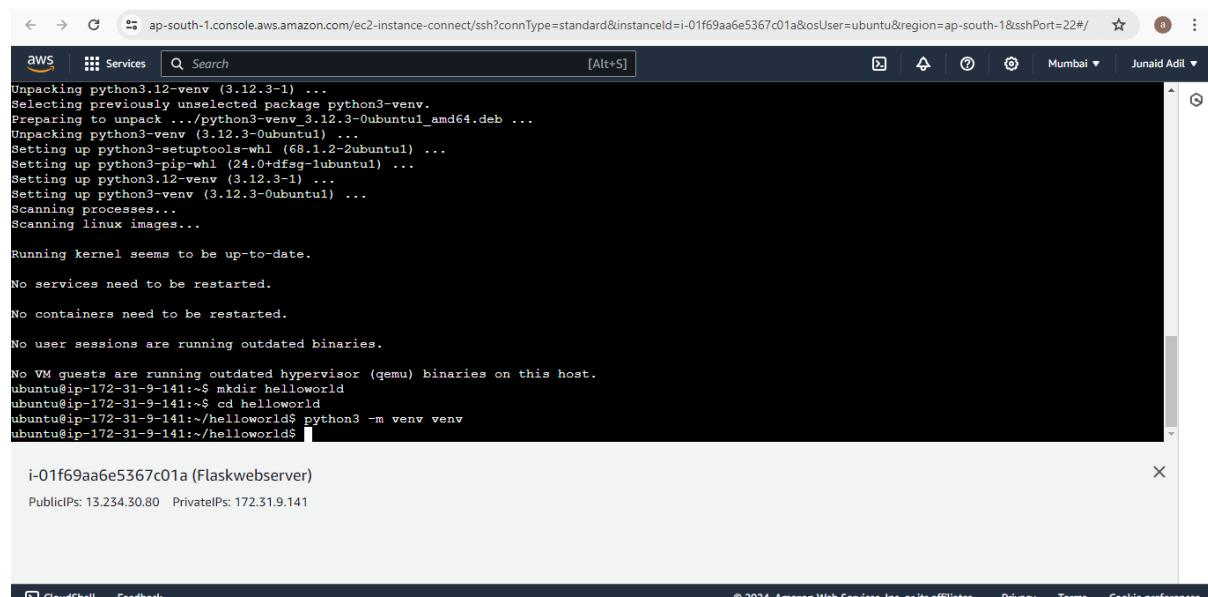
No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-9-141:~\$ mkdir helloworld
ubuntu@ip-172-31-9-141:~\$ cd helloworld
ubuntu@ip-172-31-9-141:~/helloworld\$ python3 -m venv venv

i-01f69aa6e5367c01a (Flaskwebserver)
PublicIPs: 13.234.30.80 PrivateIPs: 172.31.9.141



Unpacking python3.12-venv (3.12.3-1) ...
Selecting previously unselected package python3-venv.
Preparing to unpack .../python3-venv_3.12.3-0ubuntu1_amd64.deb ...
Unpacking python3-venv (3.12.3-0ubuntu1) ...
Setting up python3-setuptools-whl (68.1.2-2ubuntu1) ...
Setting up python3-pip-whl (24.0+dfsg-1ubuntul) ...
Setting up python3.12-venv (3.12.3-1) ...
Setting up python3-venv (3.12.3-0ubuntu1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

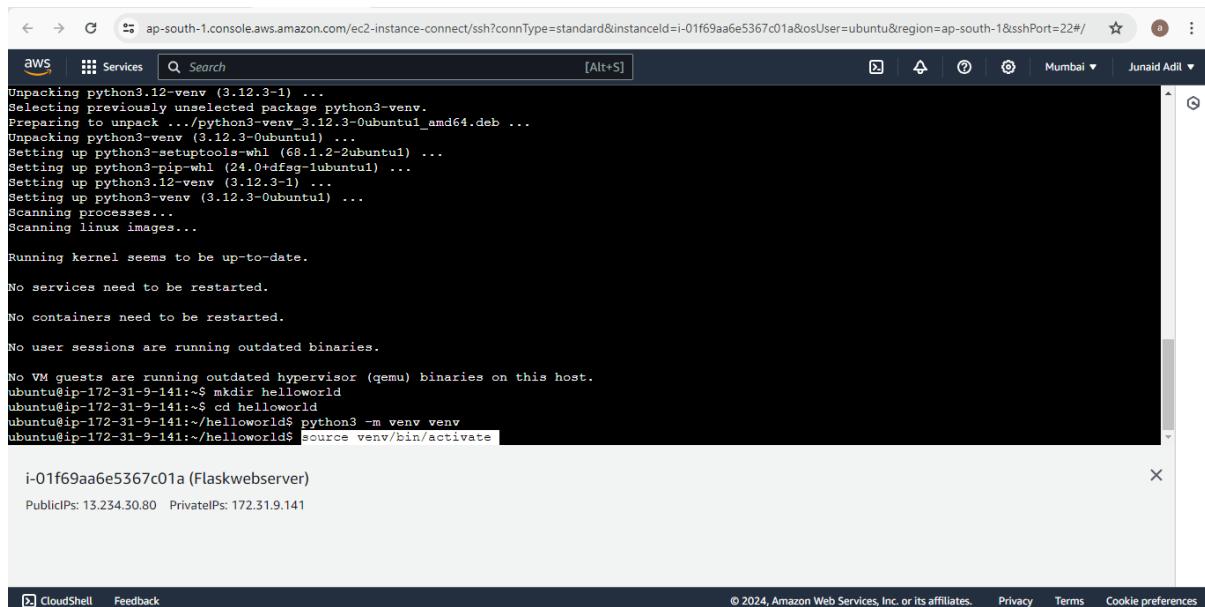
No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-9-141:~\$ mkdir helloworld
ubuntu@ip-172-31-9-141:~\$ cd helloworld
ubuntu@ip-172-31-9-141:~/helloworld\$ python3 -m venv venv
ubuntu@ip-172-31-9-141:~/helloworld\$

i-01f69aa6e5367c01a (Flaskwebserver)
PublicIPs: 13.234.30.80 PrivateIPs: 172.31.9.141

Step 7: Activate the Virtual environment using command “source venv/bin/activate”



A screenshot of the AWS CloudShell interface. The terminal window shows the following command being run:

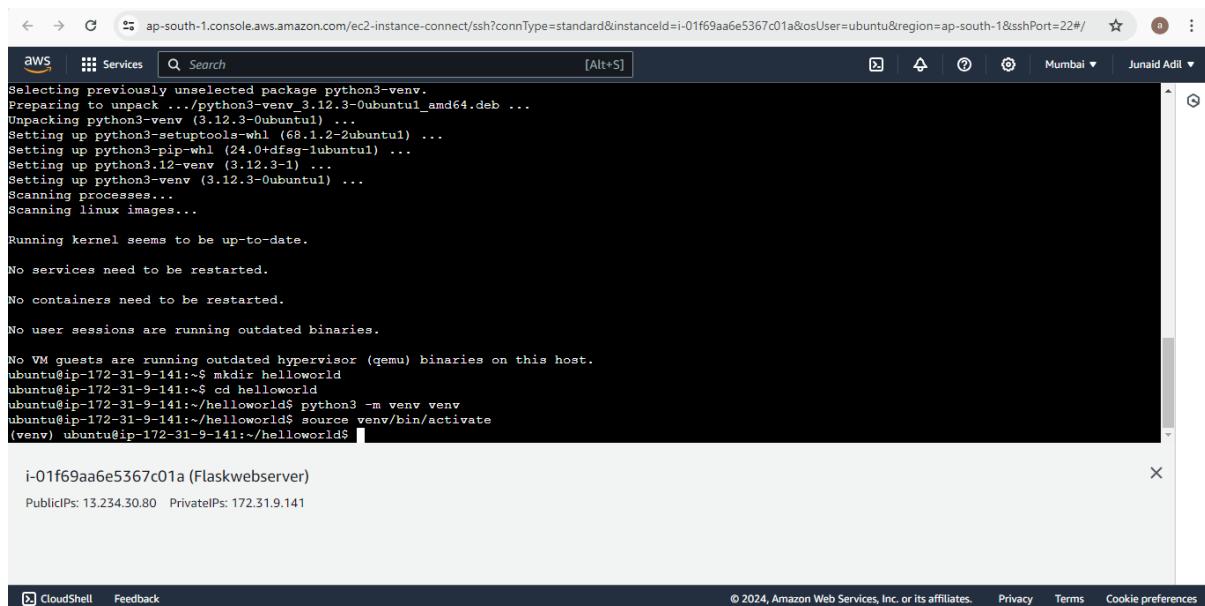
```
ubuntu@ip-172-31-9-141:~/helloworld$ source venv/bin/activate
```

The output of the command indicates that the virtual environment has been activated:

```
(venv) ubuntu@ip-172-31-9-141:~/helloworld$
```

Below the terminal, the status bar displays "i-01f69aa6e5367c01a (Flaskwebserver)" and "PublicIPs: 13.234.30.80 PrivateIPs: 172.31.9.141".

We can see the venv is activated



A screenshot of the AWS CloudShell interface. The terminal window shows the following command being run:

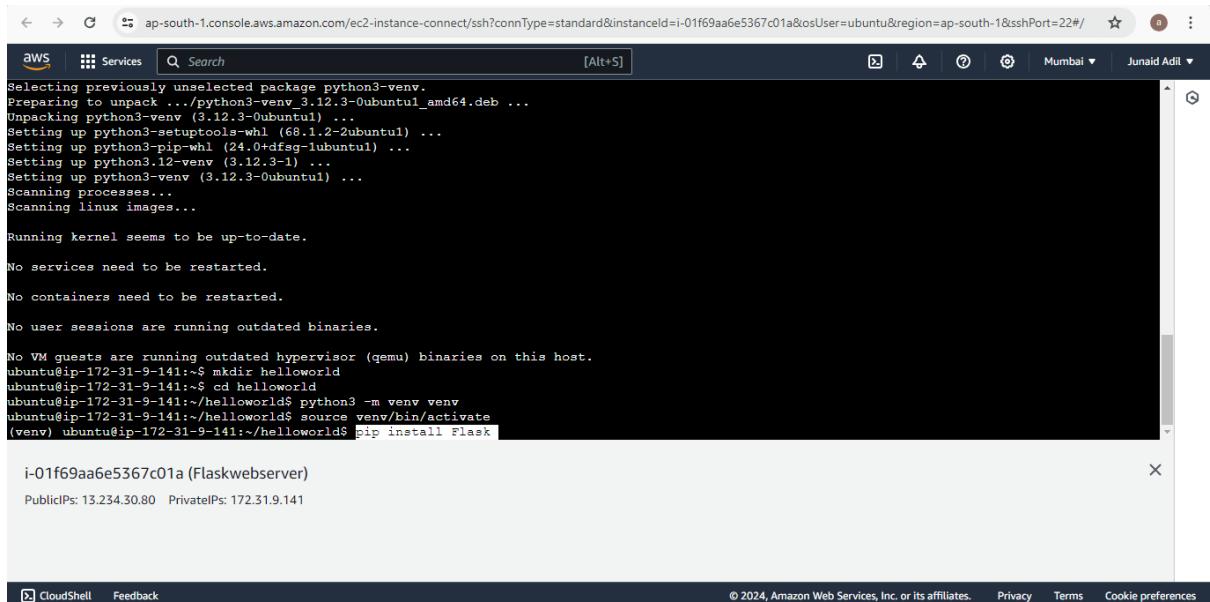
```
ubuntu@ip-172-31-9-141:~/helloworld$ source venv/bin/activate
```

The output of the command indicates that the virtual environment has been activated:

```
(venv) ubuntu@ip-172-31-9-141:~/helloworld$
```

Below the terminal, the status bar displays "i-01f69aa6e5367c01a (Flaskwebserver)" and "PublicIPs: 13.234.30.80 PrivateIPs: 172.31.9.141".

Step 8: Install the Flask using command “pip install Flask”



The screenshot shows a terminal window in the AWS CloudShell interface. The user has run the command `pip install Flask`. The output shows the package being downloaded and installed:

```
Selecting previously unselected package python3-venv.
Preparing to unpack .../python3-venv_3.12.3-0ubuntu1_amd64.deb ...
Unpacking python3-venv (3.12.3-0ubuntu1) ...
Setting up python3-setuptools-whl (68.1.2-2ubuntu1) ...
Setting up python3-pip-whl (24.0+dfsg-1ubuntu1) ...
Setting up python3.12-venv (3.12.3-1) ...
Setting up python3-venv (3.12.3-0ubuntu1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

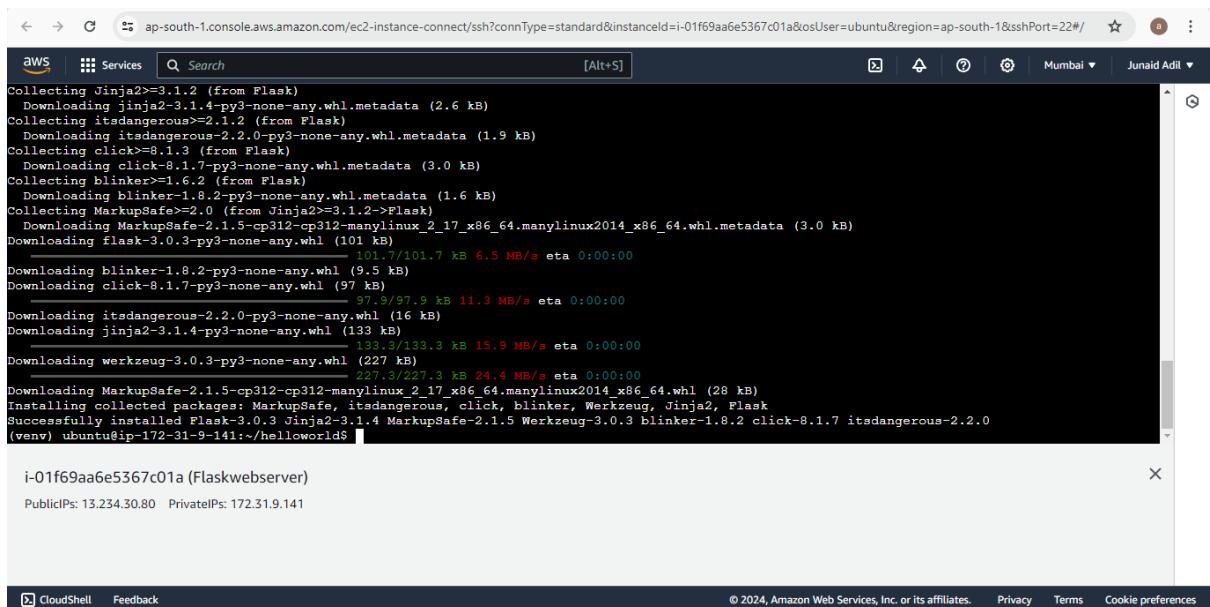
No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-9-141:~$ mkdir helloworld
ubuntu@ip-172-31-9-141:~$ cd helloworld
ubuntu@ip-172-31-9-141:~/helloworld$ python3 -m venv venv
ubuntu@ip-172-31-9-141:~/helloworld$ source venv/bin/activate
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ pip install Flask
```

At the bottom of the terminal, it shows the IP address and port information for the Flask webserver.



The screenshot shows a terminal window in the AWS CloudShell interface. The user has run the command `pip install Flask`. The output shows the progress of the package download and installation, with multiple files being transferred at once:

```
Collecting Jinja2>=2.1.2 (from Flask)
  Downloading jinja2-3.1.4-py3-none-any.whl.metadata (2.6 kB)
Collecting itsdangerous>=2.1.2 (from Flask)
  Downloading itsdangerous-2.2.0-py3-none-any.whl.metadata (1.9 kB)
Collecting click>=8.1.3 (from Flask)
  Downloading click-8.1.7-py3-none-any.whl.metadata (3.0 kB)
Collecting blinker>=1.6.2 (from Flask)
  Downloading blinker-1.8.2-py3-none-any.whl.metadata (1.6 kB)
Collecting MarkupSafe>=2.0 (from Jinja2>=2.1.2>Flask)
  Downloading MarkupSafe-2.1.5-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (3.0 kB)
  Downloading flask-3.0.3-py3-none-any.whl (101 kB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 101.7/101.7 kB 6.5 MB/s eta 0:00:00
  Downloading blinker-1.8.2-py3-none-any.whl (9.5 kB)
  Downloading click-8.1.7-py3-none-any.whl (97 kB)
    ━━━━━━━━━━━━━━━━━━━━━━━━ 97.9/97.9 kB 11.3 MB/s eta 0:00:00
  Downloading itsdangerous-2.2.0-py3-none-any.whl (16 kB)
  Downloading jinja2-3.1.4-py3-none-any.whl (133 kB)
    ━━━━━━━━━━━━━━━━ 133.3/133.3 kB 15.9 MB/s eta 0:00:00
  Downloading werkzeug-3.0.3-py3-none-any.whl (227 kB)
    ━━━━━━━━━━━━━━ 227.3/227.3 kB 24.4 MB/s eta 0:00:00
  Downloading MarkupSafe-2.1.5-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (28 kB)
Installing collected packages: MarkupSafe, itsdangerous, click, blinker, Werkzeug, Jinja2, Flask
Successfully installed Flask-3.0.3 Jinja2-3.1.4 MarkupSafe-2.1.5 Werkzeug-3.0.3 click-8.1.7 itsdangerous-2.2.0
(venv) ubuntu@ip-172-31-9-141:~/helloworld$
```

At the bottom of the terminal, it shows the IP address and port information for the Flask webserver.

Step 9: Create the application app.py using command “**sudo vi app.py**”

```
Collecting Jinja2>=2.1.2 (from Flask)
  Downloading jinja2-2.1.4-py3-none-any.whl.metadata (2.6 kB)
Collecting itsdangerous>=2.1.2 (from Flask)
  Downloading itsdangerous-2.2.0-py3-none-any.whl.metadata (1.9 kB)
Collecting click>=8.1.3 (from Flask)
  Downloading click-8.1.7-py3-none-any.whl.metadata (3.0 kB)
Collecting blinker>=1.6.2 (from Flask)
  Downloading blinker-1.8.2-py3-none-any.whl.metadata (1.6 kB)
Collecting MarkupSafe>=2.0 (from Jinja2>=2.1.2>Flask)
  Downloading MarkupSafe-2.1.5-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (3.0 kB)
  Downloading flask-2.0.3-py3-none-any.whl (10 kB)
    101.7/101.7 kB 6.5 MB/s eta 0:00:00
  Downloading blinker-1.8.2-py3-none-any.whl (9.5 kB)
  Downloading click-8.1.7-py3-none-any.whl (97 kB)
    97.9/97.9 kB 11.3 MB/s eta 0:00:00
  Downloading itsdangerous-2.2.0-py3-none-any.whl (16 kB)
  Downloading jinja2-3.1.4-py3-none-any.whl (133 kB)
    133.3/133.3 kB 15.9 MB/s eta 0:00:00
  Downloading werkzeug-2.0.3-py3-none-any.whl (227 kB)
    227.3/227.3 kB 24.4 MB/s eta 0:00:00
  Downloading MarkupSafe-2.1.5-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (28 kB)
Installing collected packages: MarkupSafe, itsdangerous, click, blinker, Werkzeug, Jinja2, Flask
Successfully installed Flask-2.0.3 Jinja2-3.1.4 MarkupSafe-2.1.5 Werkzeug-2.0.3 blinker-1.8.2 click-8.1.7 itsdangerous-2.2.0
[venv] ubuntu@ip-172-31-9-141:~/helloworld$ sudo vi app.py
```

i-01f69aa6e5367c01a (Flaskwebserver)
PublicIPs: 13.234.30.80 PrivateIPs: 172.31.9.141

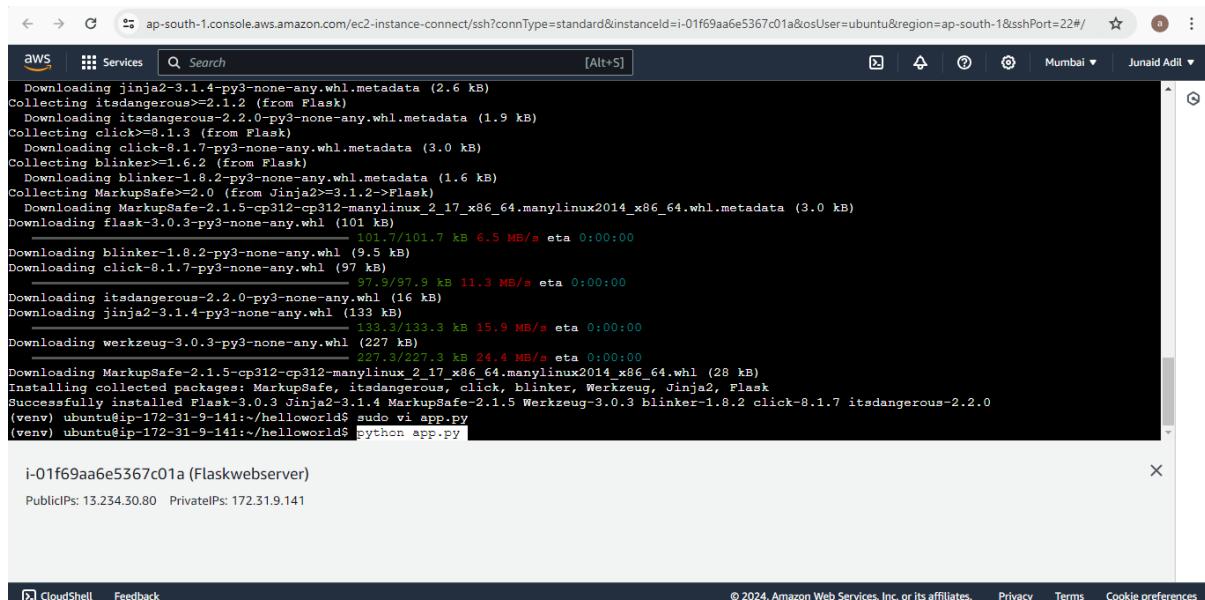
Add the commands in the file

```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def hello_world():
    return 'Hello World!'
if __name__ == "__main__":
    app.run()
```

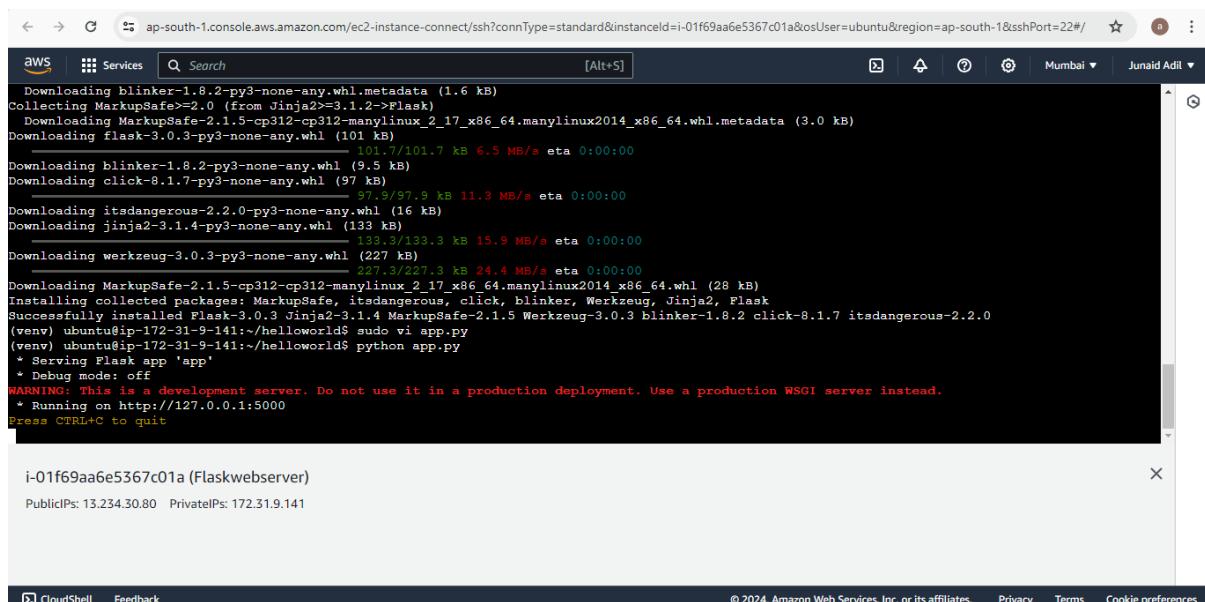
i-01f69aa6e5367c01a (Flaskwebserver)
PublicIPs: 13.234.30.80 PrivateIPs: 172.31.9.141

Then save the file

Step 10: Run the file using command “**python app.py**”



```
aws Services Search [Alt+S] Mumbai Junaid Adil
↳ ↵ ap-south-1.console.aws.amazon.com/ec2-instance-connect/ssh?connType=standard&instanceId=i-01f69aa6e5367c01a&osUser=ubuntu&region=ap-south-1&sshPort=22#/
↳ ↵ Downloading jinja2==3.1.4-py3-none-any.whl.metadata (2.6 kB)
Collecting itsdangerous>=2.1.2 (from Flask)
↳ ↵ Downloading itsdangerous-2.2.0-py3-none-any.whl.metadata (1.9 kB)
Collecting click>=8.1.3 (from Flask)
↳ ↵ Downloading click-8.1.7-py3-none-any.whl.metadata (3.0 kB)
Collecting blinker>=1.6.2 (from Flask)
↳ ↵ Downloading blinker-1.8.2-py3-none-any.whl.metadata (1.6 kB)
Collecting MarkupSafe>=2.0 (from Jinja2>=3.1.2>Flask)
↳ ↵ Downloading MarkupSafe-2.1.5-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (3.0 kB)
↳ ↵ Downloading flask==3.0.3-py3-none-any.whl (101 kB)
↳ ↵   101.7/101.7 kB 6.5 MB/s eta 0:00:00
↳ ↵ Downloading blinker-1.8.2-py3-none-any.whl (9.5 kB)
↳ ↵ Downloading click-8.1.7-py3-none-any.whl (97 kB)
↳ ↵   97.5/97.9 kB 11.3 MB/s eta 0:00:00
↳ ↵ Downloading itsdangerous-2.2.0-py3-none-any.whl (16 kB)
↳ ↵ Downloading jinja2==3.1.4-py3-none-any.whl (133 kB)
↳ ↵   133.3/133.3 kB 15.9 MB/s eta 0:00:00
↳ ↵ Downloading werkzeug==3.0.3-py3-none-any.whl (227 kB)
↳ ↵   227.3/227.3 kB 24.4 MB/s eta 0:00:00
↳ ↵ Downloading MarkupSafe-2.1.5-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (28 kB)
↳ ↵ Installing collected packages: MarkupSafe, itsdangerous, click, blinker, Werkzeug, Jinja2, Flask
↳ ↵ Successfully installed Flask-3.0.3 Jinja2-3.1.4 MarkupSafe-2.1.5 Werkzeug-3.0.3 blinker-1.8.2 click-8.1.7 itsdangerous-2.2.0
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo vi app.py
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ python app.py
i-01f69aa6e5367c01a (Flaskwebserver)
PublicIPs: 13.234.30.80 PrivateIPs: 172.31.9.141
```



```
aws Services Search [Alt+S] Mumbai Junaid Adil
↳ ↵ ap-south-1.console.aws.amazon.com/ec2-instance-connect/ssh?connType=standard&instanceId=i-01f69aa6e5367c01a&osUser=ubuntu&region=ap-south-1&sshPort=22#/
↳ ↵ Downloading blinker-1.8.2-py3-none-any.whl.metadata (1.6 kB)
Collecting MarkupSafe>=2.0 (from Jinja2>=3.1.2>Flask)
↳ ↵ Downloading MarkupSafe-2.1.5-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (3.0 kB)
↳ ↵ Downloading flask==3.0.3-py3-none-any.whl (101 kB)
↳ ↵   101.7/101.7 kB 6.5 MB/s eta 0:00:00
↳ ↵ Downloading blinker-1.8.2-py3-none-any.whl (9.5 kB)
↳ ↵ Downloading click-8.1.7-py3-none-any.whl (97 kB)
↳ ↵   97.5/97.9 kB 11.3 MB/s eta 0:00:00
↳ ↵ Downloading itsdangerous-2.2.0-py3-none-any.whl (16 kB)
↳ ↵ Downloading jinja2==3.1.4-py3-none-any.whl (133 kB)
↳ ↵   133.3/133.3 kB 15.9 MB/s eta 0:00:00
↳ ↵ Downloading werkzeug==3.0.3-py3-none-any.whl (227 kB)
↳ ↵   227.3/227.3 kB 24.4 MB/s eta 0:00:00
↳ ↵ Downloading MarkupSafe-2.1.5-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (28 kB)
↳ ↵ Installing collected packages: MarkupSafe, itsdangerous, click, blinker, Werkzeug, Jinja2, Flask
↳ ↵ Successfully installed Flask-3.0.3 Jinja2-3.1.4 MarkupSafe-2.1.5 Werkzeug-3.0.3 blinker-1.8.2 click-8.1.7 itsdangerous-2.2.0
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo vi 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 http://127.0.0.1:5000
Press CTRL+C to quit
i-01f69aa6e5367c01a (Flaskwebserver)
PublicIPs: 13.234.30.80 PrivateIPs: 172.31.9.141
```

There we can see the port on which it is running Localhost:5000

Step 11: To stop it use Ctrl+c

Step 12: Install gunicorn server using command “**pip install gunicorn**”

```
< → G ap-south-1.console.aws.amazon.com/ec2-instance-connect/ssh?connType=standard&instanceId=i-01f69aa6e5367c01a&osUser=ubuntu&region=ap-south-1&sshPort=22#/ Junaid Adil ▾

aws Services Search [Alt+S] Mumbai ▾ Junaid Adil ▾
Downloaded werkzeug-3.0.3-py3-none-any.whl (227 kB) 227.3/227.3 kB 24.4 MB/s eta 0:00:00
Downloading MarkupSafe-2.1.5-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (28 kB)
Installing collected packages: MarkupSafe, itsdangerous, click, blinker, Werkzeug, Jinja2, Flask
Successfully installed Flask-3.0.3 Jinja2-3.1.4 MarkupSafe-2.1.5 Werkzeug-3.0.3 blinker-1.8.2 click-8.1.7 itsdangerous-2.2.0
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo vi app.py
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ 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 http://127.0.0.1:5000
Press CTRL+C to quit
^C(venv) ubuntu@ip-172-31-9-141:~/helloworld$ pip install gunicorn
Collecting gunicorn
  Downloading gunicorn-22.0.0-py3-none-any.whl.metadata (4.4 kB)
Collecting packaging (from gunicorn)
  Downloading packaging-24.0-py3-none-any.whl.metadata (3.2 kB)
Downloaded gunicorn-22.0.0-py3-none-any.whl (84 kB) 84.4/84.4 kB 4.0 MB/s eta 0:00:00
Downloaded packaging-24.0-py3-none-any.whl (53 kB) 53.5/53.5 kB 5.5 MB/s eta 0:00:00
Installing collected packages: packaging, gunicorn
Successfully installed gunicorn-22.0.0 packaging-24.0
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ 

i-01f69aa6e5367c01a (Flaskwebserver)
PublicIPs: 13.234.30.80 PrivateIPs: 172.31.9.141
```

Step 13: Then run the server using command “**gunicorn -b 0.0.0.0:8000 app:app**”

```
< → G ap-south-1.console.aws.amazon.com/ec2-instance-connect/ssh?connType=standard&instanceId=i-01f69aa6e5367c01a&osUser=ubuntu&region=ap-south-1&sshPort=22#/ Junaid Adil ▾

aws Services Search [Alt+S] Mumbai ▾ Junaid Adil ▾
Downloaded werkzeug-3.0.3-py3-none-any.whl (227 kB) 227.3/227.3 kB 24.4 MB/s eta 0:00:00
Downloading MarkupSafe-2.1.5-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (28 kB)
Installing collected packages: MarkupSafe, itsdangerous, click, blinker, Werkzeug, Jinja2, Flask
Successfully installed Flask-3.0.3 Jinja2-3.1.4 MarkupSafe-2.1.5 Werkzeug-3.0.3 blinker-1.8.2 click-8.1.7 itsdangerous-2.2.0
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo vi app.py
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ 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 http://127.0.0.1:5000
Press CTRL+C to quit
^C(venv) ubuntu@ip-172-31-9-141:~/helloworld$ pip install gunicorn
Collecting gunicorn
  Downloading gunicorn-22.0.0-py3-none-any.whl.metadata (4.4 kB)
Collecting packaging (from gunicorn)
  Downloading packaging-24.0-py3-none-any.whl.metadata (3.2 kB)
Downloaded gunicorn-22.0.0-py3-none-any.whl (84 kB) 84.4/84.4 kB 4.0 MB/s eta 0:00:00
Downloaded packaging-24.0-py3-none-any.whl (53 kB) 53.5/53.5 kB 5.5 MB/s eta 0:00:00
Installing collected packages: packaging, gunicorn
Successfully installed gunicorn-22.0.0 packaging-24.0
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ gunicorn -b 0.0.0.0:8000 app:app

i-01f69aa6e5367c01a (Flaskwebserver)
PublicIPs: 13.234.30.80 PrivateIPs: 172.31.9.141
```

```
aws Services Search [Alt+S] Mumbai Junaid Adil
< > ap-south-1.console.aws.amazon.com/ec2-instance-connect/ssh?connType=standard&instanceId=i-01f69aa6e5367c01a&osUser=ubuntu&region=ap-south-1&sshPort=22#/
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo vi app.py
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ 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 http://127.0.0.1:5000
Press CTRL+C to quit
^C(venv) ubuntu@ip-172-31-9-141:~/helloworld$ pip install gunicorn
Collecting gunicorn
  Downloading gunicorn-22.0.0-py3-none-any.whl.metadata (4.4 kB)
Collecting packaging (from gunicorn)
  Downloading packaging-24.0-py3-none-any.whl.metadata (3.2 kB)
Downloading gunicorn-22.0.0-py3-none-any.whl (84 kB)
   84.4/84.4 kB 4.0 MB/s eta 0:00:00
Downloading packaging-24.0-py3-none-any.whl (53 kB)
   53.5/53.5 kB 5.5 MB/s eta 0:00:00
Installing collected packages: packaging, gunicorn
Successfully installed gunicorn-22.0.0 packaging-24.0
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ gunicorn -b 0.0.0.0:8000 app:app
[2024-05-12 13:08:13 +0000] [2074] [INFO] Starting gunicorn 22.0.0
[2024-05-12 13:08:13 +0000] [2074] [INFO] Listening at: http://0.0.0.0:8000 (2074)
[2024-05-12 13:08:13 +0000] [2074] [INFO] Using worker: sync
[2024-05-12 13:08:13 +0000] [2075] [INFO] Booting worker with pid: 2075
i-01f69aa6e5367c01a (Flaskwebserver)
PublicIPs: 13.234.30.80 PrivateIPs: 172.31.9.141
```

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

```
aws Services Search [Alt+S] Mumbai Junaid Adil
< > ap-south-1.console.aws.amazon.com/ec2-instance-connect/ssh?connType=standard&instanceId=i-01f69aa6e5367c01a&osUser=ubuntu&region=ap-south-1&sshPort=22#/
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo vi app.py
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ 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 http://127.0.0.1:5000
Press CTRL+C to quit
^C(venv) ubuntu@ip-172-31-9-141:~/helloworld$ pip install gunicorn
Collecting gunicorn
  Downloading gunicorn-22.0.0-py3-none-any.whl.metadata (4.4 kB)
Collecting packaging (from gunicorn)
  Downloading packaging-24.0-py3-none-any.whl.metadata (3.2 kB)
Downloading gunicorn-22.0.0-py3-none-any.whl (84 kB)
   84.4/84.4 kB 4.0 MB/s eta 0:00:00
Downloading packaging-24.0-py3-none-any.whl (53 kB)
   53.5/53.5 kB 5.5 MB/s eta 0:00:00
Installing collected packages: packaging, gunicorn
Successfully installed gunicorn-22.0.0 packaging-24.0
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ gunicorn -b 0.0.0.0:8000 app:app
[2024-05-12 13:08:13 +0000] [2074] [INFO] Starting gunicorn 22.0.0
[2024-05-12 13:08:13 +0000] [2074] [INFO] Listening at: http://0.0.0.0:8000 (2074)
[2024-05-12 13:08:13 +0000] [2074] [INFO] Using worker: sync
[2024-05-12 13:08:13 +0000] [2075] [INFO] Booting worker with pid: 2075
^C[2024-05-12 13:08:37 +0000] [2074] [INFO] Handling signal: int
[2024-05-12 13:08:37 +0000] [2075] [INFO] Worker exiting (pid: 2075)
[2024-05-12 13:08:37 +0000] [2074] [INFO] Shutting down: Master
(venv) ubuntu@ip-172-31-9-141:~/helloworld$
```

i-01f69aa6e5367c01a (Flaskwebserver)
PublicIPs: 13.234.30.80 PrivateIPs: 172.31.9.141

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Step 14: We have to create the helloworld.service file for the server using command “**sudo vi /etc/systemd/system/helloworld.service**”

```
* 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 http://127.0.0.1:5000
Press CTRL+C to quit
^C(venv) ubuntu@ip-172-31-9-141:~/helloworld$ pip install gunicorn
Collecting gunicorn
  Downloading gunicorn-22.0.0-py3-none-any.whl.metadata (4.4 kB)
Collecting packaging (from gunicorn)
  Downloading packaging-24.0-py3-none-any.whl.metadata (3.2 kB)
Downloaded gunicorn-22.0.0-py3-none-any.whl (84 kB)
   0.0%    0.0/84.4 kB 4.0 MB/s eta 0:00:00
Downloaded packaging-24.0-py3-none-any.whl (53 kB)
   100.0% 53.5/53.5 kB 5.5 MB/s eta 0:00:00
Installing collected packages: packaging, gunicorn
Successfully installed gunicorn-22.0.0 packaging-24.0
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ gunicorn -b 0.0.0.0:8000 app:app
[2024-05-12 13:08:13 +0000] [2074] [INFO] Starting gunicorn 22.0.0
[2024-05-12 13:08:13 +0000] [2074] [INFO] Listening at: http://0.0.0.0:8000 (2074)
[2024-05-12 13:08:13 +0000] [2074] [INFO] Using worker: sync
[2024-05-12 13:08:13 +0000] [2075] [INFO] Booting worker with pid: 2075
^C[2024-05-12 13:08:37 +0000] [2074] [INFO] Handling signal: int
[2024-05-12 13:08:37 +0000] [2075] [INFO] Worker exiting (pid: 2075)
[2024-05-12 13:08:37 +0000] [2074] [INFO] Shutting down: Master
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo vi /etc/systemd/system/helloworld.service
```

i-01f69aa6e5367c01a (Flaskwebserver)
PublicIPs: 13.234.30.80 PrivateIPs: 172.31.9.141

Then add the commands in the file

```
[Unit]
Description=Gunicorn instance for a simple hello world app
After=network.target
[Service]
User=ubuntu
Group=www-data
WorkingDirectory=/home/ubuntu/helloworld
ExecStart=/home/ubuntu/helloworld/venv/bin/gunicorn -b localhost:8000 app:app
Restart=always
[Install]
WantedBy=multi-user.target
```

-- INSERT --

i-01f69aa6e5367c01a (Flaskwebserver)
PublicIPs: 13.234.30.80 PrivateIPs: 172.31.9.141

Save the file :wq!

The screenshot shows a terminal session in the AWS CloudShell interface. The user has run several commands to set up a Flask webserver:

```
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
^C(venv) ubuntu@ip-172-31-9-141:~/helloworld$ pip install gunicorn
Collecting gunicorn
  Downloading gunicorn-22.0.0-py3-none-any.whl.metadata (4.4 kB)
Collecting packaging (from gunicorn)
  Downloading packaging-24.0-py3-none-any.whl.metadata (3.2 kB)
Downloading gunicorn-22.0.0-py3-none-any.whl (84 kB)
   84.4/84.4 kB 4.0 MB/s eta 0:00:00
Downloading packaging-24.0-py3-none-any.whl (53 kB)
   53.5/53.5 kB 5.5 MB/s eta 0:00:00
Installing collected packages: packaging, gunicorn
Successfully installed gunicorn-22.0.0 packaging-24.0
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ gunicorn -b 0.0.0.0:8000 app:app
[2024-05-12 13:08:13 +0000] [2074] [INFO] Starting gunicorn 22.0
[2024-05-12 13:08:13 +0000] [2074] [INFO] Listening at: http://0.0.0.0:8000 (2074)
[2024-05-12 13:08:13 +0000] [2074] [INFO] Using worker: sync
[2024-05-12 13:08:13 +0000] [2075] [INFO] Booting worker with pid: 2075
^C[2024-05-12 13:08:37 +0000] [2074] [INFO] Handling signal: int
[2024-05-12 13:08:37 +0000] [2075] [INFO] Worker exiting (pid: 2075)
[2024-05-12 13:08:37 +0000] [2074] [INFO] Shutting down: Master
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo vi /etc/systemd/system/helloworld.service
(venv) ubuntu@ip-172-31-9-141:~/helloworld$
```

At the bottom of the terminal, the user is prompted to enter their AWS CloudShell command.

Step 15: We have to start and enable the server. So run the commands “**`sudo systemctl daemon-reload`**”, “**`sudo systemctl start helloworld`**”, “**`sudo systemctl enable helloworld`**”

Run command : “**`sudo systemctl daemon-reload`**”.

The screenshot shows the continuation of the AWS CloudShell session. The user has run the command `sudo systemctl daemon-reload`:

```
i-01f69aa6e5367c01a (Flaskwebserver)
```

At the bottom of the terminal, the user is prompted to enter their AWS CloudShell command.

The screenshot shows the AWS CloudShell interface. The terminal window displays the deployment process of a Flask application named 'helloworld'. The user runs 'pip install gunicorn' and then starts the application with 'gunicorn -b 0.0.0.0:8000 app:app'. The logs show the application listening on port 8000 and booting workers. A signal is received, and the worker exits. The master process then shuts down. Finally, the user reloads the service with 'sudo systemctl daemon-reload'.

```
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
^C(venv) ubuntu@ip-172-31-9-141:~/helloworld$ pip install gunicorn
Collecting gunicorn
  Downloading gunicorn-22.0.0-py3-none-any.whl.metadata (4.4 kB)
Collecting packaging (from gunicorn)
  Downloading packaging-24.0-py3-none-any.whl.metadata (3.2 kB)
Downloaded gunicorn-22.0.0-py3-none-any.whl (84 kB)
   84.4/84.4 kB 4.0 MB/s eta 0:00:00
Downloaded packaging-24.0-py3-none-any.whl (53 kB)
   53.5/53.5 kB 5.5 MB/s eta 0:00:00
Installing collected packages: packaging, gunicorn
Successfully installed gunicorn-22.0.0 packaging-24.0
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ gunicorn -b 0.0.0.0:8000 app:app
[2024-05-12 13:08:13 +0000] [2074] [INFO] Starting gunicorn 22.0.0
[2024-05-12 13:08:13 +0000] [2074] [INFO] Listening at: http://0.0.0.0:8000 (2074)
[2024-05-12 13:08:13 +0000] [2074] [INFO] Using worker: sync
[2024-05-12 13:08:13 +0000] [2075] [INFO] Booting worker with pid: 2075
^C[2024-05-12 13:08:37 +0000] [2074] [INFO] Handling signal: int
[2024-05-12 13:08:37 +0000] [2075] [INFO] Worker exiting (pid: 2075)
[2024-05-12 13:08:37 +0000] [2074] [INFO] Shutting down: Master
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo vi /etc/systemd/system/helloworld.service
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo systemctl daemon-reload
(venv) ubuntu@ip-172-31-9-141:~/helloworld$
```

i-01f69aa6e5367c01a (Flaskwebserver)

PublicIPs: 13.234.30.80 PrivateIPs: 172.31.9.141

Run the command : “**sudo systemctl start helloworld**”

The screenshot shows the AWS CloudShell interface. The user runs 'sudo systemctl start helloworld'. The logs indicate that the service is starting and listening on port 8000.

```
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
^C(venv) ubuntu@ip-172-31-9-141:~/helloworld$ pip install gunicorn
Collecting gunicorn
  Downloading gunicorn-22.0.0-py3-none-any.whl.metadata (4.4 kB)
Collecting packaging (from gunicorn)
  Downloading packaging-24.0-py3-none-any.whl.metadata (3.2 kB)
Downloaded gunicorn-22.0.0-py3-none-any.whl (84 kB)
   84.4/84.4 kB 4.0 MB/s eta 0:00:00
Downloaded packaging-24.0-py3-none-any.whl (53 kB)
   53.5/53.5 kB 5.5 MB/s eta 0:00:00
Installing collected packages: packaging, gunicorn
Successfully installed gunicorn-22.0.0 packaging-24.0
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ gunicorn -b 0.0.0.0:8000 app:app
[2024-05-12 13:08:13 +0000] [2074] [INFO] Starting gunicorn 22.0.0
[2024-05-12 13:08:13 +0000] [2074] [INFO] Listening at: http://0.0.0.0:8000 (2074)
[2024-05-12 13:08:13 +0000] [2074] [INFO] Using worker: sync
[2024-05-12 13:08:13 +0000] [2075] [INFO] Booting worker with pid: 2075
^C[2024-05-12 13:08:37 +0000] [2074] [INFO] Handling signal: int
[2024-05-12 13:08:37 +0000] [2075] [INFO] Worker exiting (pid: 2075)
[2024-05-12 13:08:37 +0000] [2074] [INFO] Shutting down: Master
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo vi /etc/systemd/system/helloworld.service
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo systemctl daemon-reload
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo systemctl start helloworld
```

i-01f69aa6e5367c01a (Flaskwebserver)

PublicIPs: 13.234.30.80 PrivateIPs: 172.31.9.141

The screenshot shows a terminal window in the AWS CloudShell interface. The user has run the command `sudo systemctl enable helloworld`. The terminal output shows the process of installing unicorn and packaging dependencies, starting the unicorn service, and enabling it via systemctl. The user's session ID is i-01f69aa6e5367c01a.

```
Press CTRL+C to quit
^C(venv) ubuntu@ip-172-31-9-141:~/helloworld$ pip install unicorn
Collecting unicorn
  Downloading unicorn-22.0.0-py3-none-any.whl.metadata (4.4 kB)
Collecting packaging (from unicorn)
  Downloading packaging-24.0-py3-none-any.whl.metadata (3.2 kB)
Downloading unicorn-22.0.0-py3-none-any.whl (84 kB)
   84.4/84.4 kB 4.0 MB/s eta 0:00:00
Downloaded packaging-24.0-py3-none-any.whl (53 kB)
   53.5/53.5 kB 5.5 MB/s eta 0:00:00
Installing collected packages: packaging, unicorn
Successfully installed unicorn-22.0.0 packaging-24.0
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ unicorn -b 0.0.0.0:8000 app:app
[2024-05-12 13:08:13 +0000] [2074] [INFO] Starting unicorn 22.0.0
[2024-05-12 13:08:13 +0000] [2074] [INFO] Listening at: http://0.0.0.0:8000 (2074)
[2024-05-12 13:08:13 +0000] [2074] [INFO] Using worker: sync
[2024-05-12 13:08:13 +0000] [2075] [INFO] Booting worker with pid: 2075
^C[2024-05-12 13:08:37 +0000] [2074] [INFO] Handling signal: int
[2024-05-12 13:08:37 +0000] [2075] [INFO] Worker exiting (pid: 2075)
[2024-05-12 13:08:37 +0000] [2074] [INFO] Shutting down: Master
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo vi /etc/systemd/system/helloworld.service
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo systemctl daemon-reload
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo systemctl start helloworld
(venv) ubuntu@ip-172-31-9-141:~/helloworld$
```

Run the command: “**`sudo systemctl enable helloworld`**”

The screenshot shows a terminal window in the AWS CloudShell interface. The user has run the command `sudo systemctl enable helloworld`. The terminal output shows the process of installing unicorn and packaging dependencies, starting the unicorn service, and enabling it via systemctl. The user's session ID is i-01f69aa6e5367c01a.

```
Press CTRL+C to quit
^C(venv) ubuntu@ip-172-31-9-141:~/helloworld$ pip install unicorn
Collecting unicorn
  Downloading unicorn-22.0.0-py3-none-any.whl.metadata (4.4 kB)
Collecting packaging (from unicorn)
  Downloading packaging-24.0-py3-none-any.whl.metadata (3.2 kB)
Downloading unicorn-22.0.0-py3-none-any.whl (84 kB)
   84.4/84.4 kB 4.0 MB/s eta 0:00:00
Downloaded packaging-24.0-py3-none-any.whl (53 kB)
   53.5/53.5 kB 5.5 MB/s eta 0:00:00
Installing collected packages: packaging, unicorn
Successfully installed unicorn-22.0.0 packaging-24.0
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ unicorn -b 0.0.0.0:8000 app:app
[2024-05-12 13:08:13 +0000] [2074] [INFO] Starting unicorn 22.0.0
[2024-05-12 13:08:13 +0000] [2074] [INFO] Listening at: http://0.0.0.0:8000 (2074)
[2024-05-12 13:08:13 +0000] [2074] [INFO] Using worker: sync
[2024-05-12 13:08:13 +0000] [2075] [INFO] Booting worker with pid: 2075
^C[2024-05-12 13:08:37 +0000] [2074] [INFO] Handling signal: int
[2024-05-12 13:08:37 +0000] [2075] [INFO] Worker exiting (pid: 2075)
[2024-05-12 13:08:37 +0000] [2074] [INFO] Shutting down: Master
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo vi /etc/systemd/system/helloworld.service
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo systemctl daemon-reload
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo systemctl start helloworld
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo systemctl enable helloworld
```

The screenshot shows the AWS CloudShell interface. At the top, the URL is `ap-south-1.console.aws.amazon.com/ec2-instance-connect/ssh?connType=standard&instanceId=i-01f69aa6e5367c01a&osUser=ubuntu®ion=ap-south-1&sshPort=22#/`. The AWS logo and services menu are visible. A search bar contains "Search". The main terminal window displays the deployment process:

```
Collecting gunicorn
  Downloading gunicorn-22.0.0-py3-none-any.whl.metadata (4.4 kB)
Collecting packaging (from gunicorn)
  Downloading packaging-24.0-py3-none-any.whl.metadata (3.2 kB)
Downloading gunicorn-22.0.0-py3-none-any.whl (84 kB)
   └── packaging-24.0-py3-none-any.whl (53 kB)  84.4/84.4 kB 4.0 MB/s eta 0:00:00
  53.5/53.5 kB 5.5 MB/s eta 0:00:00
Installing collected packages: packaging, gunicorn
Successfully installed gunicorn-22.0.0 packaging-24.0
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ gunicorn -b 0.0.0.0:8000 app:app
[2024-05-12 13:08:13 +0000] [2074] [INFO] Starting gunicorn 22.0.0
[2024-05-12 13:08:13 +0000] [2074] [INFO] Listening at: http://0.0.0.0:8000 (2074)
[2024-05-12 13:08:13 +0000] [2074] [INFO] Using worker: sync
[2024-05-12 13:08:13 +0000] [2075] [INFO] Booting worker with pid: 2075
^C[2024-05-12 13:08:37 +0000] [2074] [INFO] Handling signal: int
[2024-05-12 13:08:37 +0000] [2075] [INFO] Worker exiting (pid: 2075)
[2024-05-12 13:08:37 +0000] [2074] [INFO] Shutting down: Master
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo vi /etc/systemd/system/helloworld.service
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo systemctl daemon-reload
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo systemctl start helloworld
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo systemctl enable helloworld
Created symlink /etc/systemd/system/multi-user.target.wants/helloworld.service → /etc/systemd/system/helloworld.service.
(venv) ubuntu@ip-172-31-9-141:~/helloworld$
```

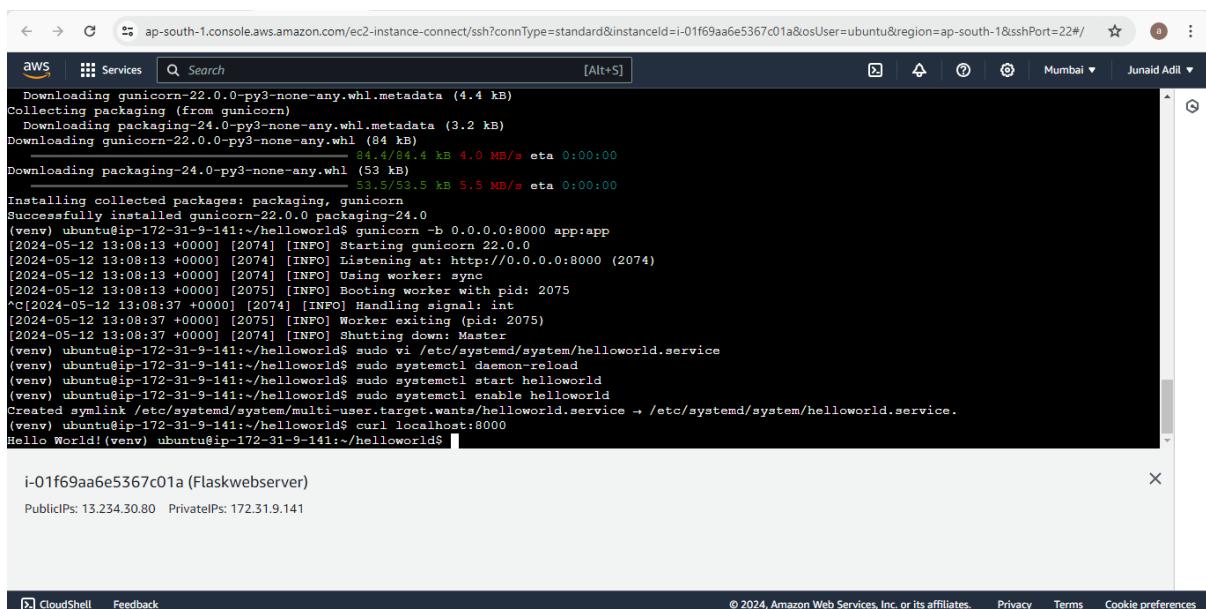
At the bottom, it says "i-01f69aa6e5367c01a (Flaskwebserver)" and "Public IPs: 13.234.30.80 Private IPs: 172.31.9.141".

Step 16: To check the server is running, use command “curl localhost:8000”

The screenshot shows the AWS CloudShell interface. The terminal window displays the deployment process and then the command `curl localhost:8000` being run:

```
Collecting gunicorn
  Downloading gunicorn-22.0.0-py3-none-any.whl.metadata (4.4 kB)
Collecting packaging (from gunicorn)
  Downloading packaging-24.0-py3-none-any.whl.metadata (3.2 kB)
Downloading gunicorn-22.0.0-py3-none-any.whl (84 kB)
   └── packaging-24.0-py3-none-any.whl (53 kB)  84.4/84.4 kB 4.0 MB/s eta 0:00:00
  53.5/53.5 kB 5.5 MB/s eta 0:00:00
Installing collected packages: packaging, gunicorn
Successfully installed gunicorn-22.0.0 packaging-24.0
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ gunicorn -b 0.0.0.0:8000 app:app
[2024-05-12 13:08:13 +0000] [2074] [INFO] Starting gunicorn 22.0.0
[2024-05-12 13:08:13 +0000] [2074] [INFO] Listening at: http://0.0.0.0:8000 (2074)
[2024-05-12 13:08:13 +0000] [2074] [INFO] Using worker: sync
[2024-05-12 13:08:13 +0000] [2075] [INFO] Booting worker with pid: 2075
^C[2024-05-12 13:08:37 +0000] [2074] [INFO] Handling signal: int
[2024-05-12 13:08:37 +0000] [2075] [INFO] Worker exiting (pid: 2075)
[2024-05-12 13:08:37 +0000] [2074] [INFO] Shutting down: Master
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo vi /etc/systemd/system/helloworld.service
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo systemctl daemon-reload
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo systemctl start helloworld
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo systemctl enable helloworld
Created symlink /etc/systemd/system/multi-user.target.wants/helloworld.service → /etc/systemd/system/helloworld.service.
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ curl localhost:8000
```

At the bottom, it says "i-01f69aa6e5367c01a (Flaskwebserver)" and "Public IPs: 13.234.30.80 Private IPs: 172.31.9.141".



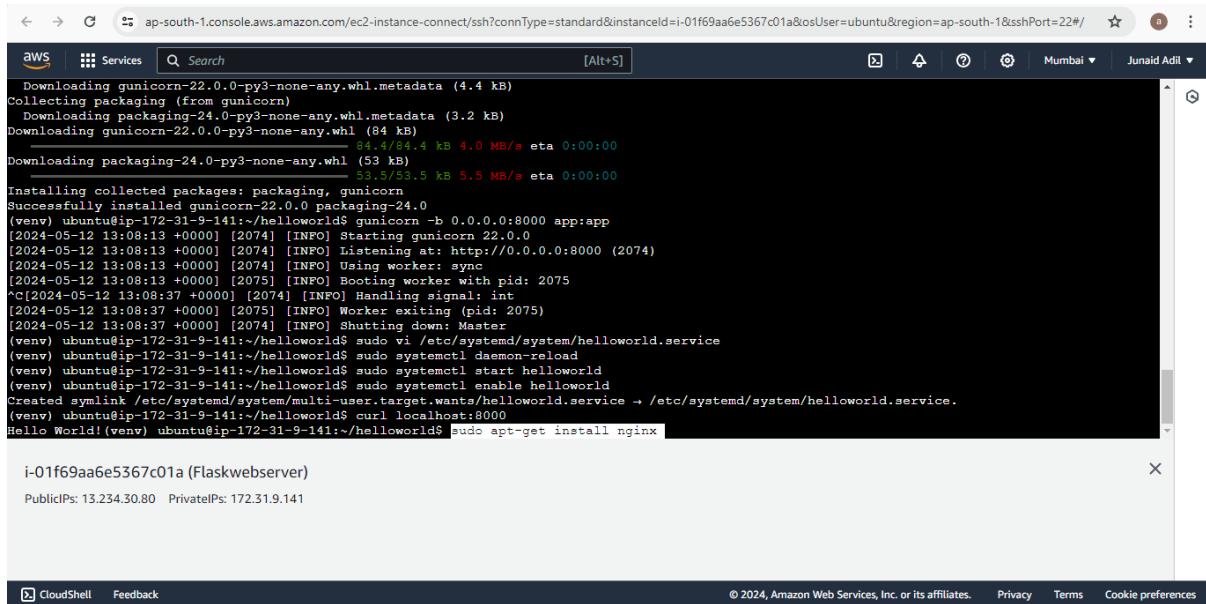
```
aws Services Search [Alt+S] Junaid Adil ▾
Download gunicorn-22.0.0-py3-none-any.whl.metadata (4.4 kB)
Collecting packaging (from gunicorn)
  Downloading packaging-24.0-py3-none-any.whl.metadata (3.2 kB)
Download gunicorn-22.0.0-py3-none-any.whl (84 kB)
  84.4/84.4 kB 4.0 MB/s eta 0:00:00
Download gunicorn-24.0-py3-none-any.whl (53 kB)
  53.5/53.5 kB 5.5 MB/s eta 0:00:00
Installing collected packages: packaging, gunicorn
Successfully installed gunicorn-22.0.0 packaging-24.0
[venv] ubuntu@ip-172-31-9-141:~/helloworld$ gunicorn -b 0.0.0.0:8000 app:app
[2024-05-12 13:08:13 +0000] [2074] [INFO] Starting gunicorn 22.0
[2024-05-12 13:08:13 +0000] [2074] [INFO] Listening at: http://0.0.0.0:8000 (2074)
[2024-05-12 13:08:13 +0000] [2074] [INFO] Using worker: sync
[2024-05-12 13:08:13 +0000] [2075] [INFO] Booting worker with pid: 2075
^C[2024-05-12 13:08:37 +0000] [2074] [INFO] Handling signal: int
[2024-05-12 13:08:37 +0000] [2075] [INFO] Worker exiting (pid: 2075)
[2024-05-12 13:08:37 +0000] [2074] [INFO] Shutting down: Master
[venv] ubuntu@ip-172-31-9-141:~/helloworld$ sudo vi /etc/systemd/system/helloworld.service
[venv] ubuntu@ip-172-31-9-141:~/helloworld$ sudo systemctl daemon-reload
[venv] ubuntu@ip-172-31-9-141:~/helloworld$ sudo systemctl start helloworld
[venv] ubuntu@ip-172-31-9-141:~/helloworld$ sudo systemctl enable helloworld
Created symlink /etc/systemd/system/multi-user.target.wants/helloworld.service → /etc/systemd/system/helloworld.service.
[venv] ubuntu@ip-172-31-9-141:~/helloworld$ curl localhost:8000
Hello World!(venv) ubuntu@ip-172-31-9-141:~/helloworld$
```

i-01f69aa6e5367c01a (Flaskwebserver)

PublicIPs: 13.234.30.80 PrivateIPs: 172.31.9.141

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Step 17: Install the Nginx server using command “**sudo apt-get install nginx**”

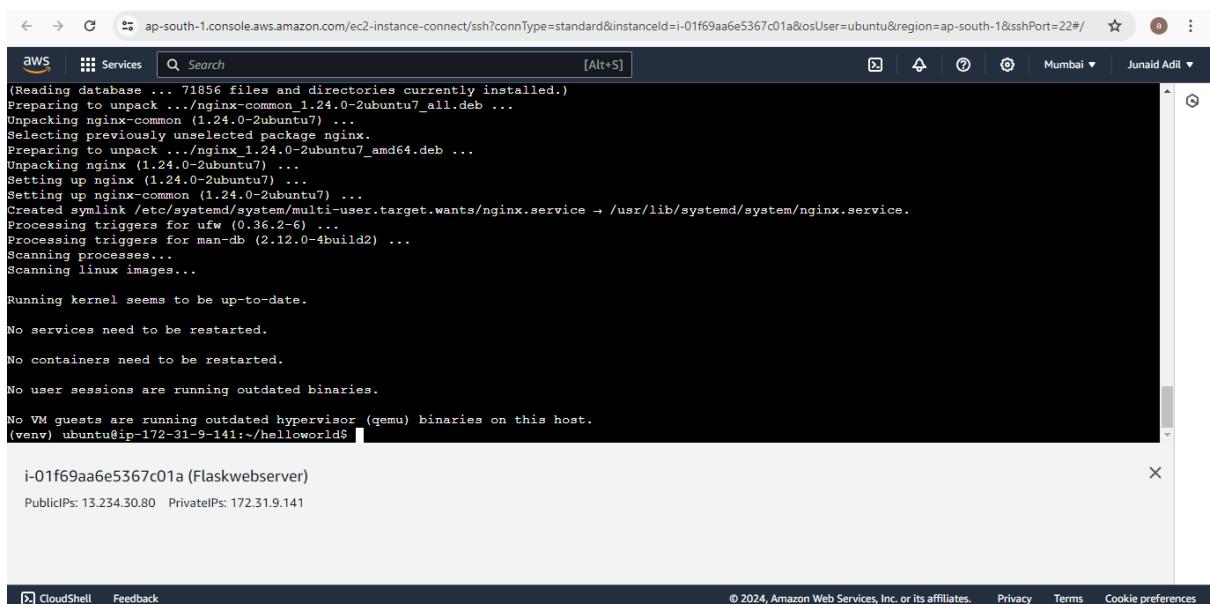


```
aws Services Search [Alt+S] Junaid Adil ▾
Download gunicorn-22.0.0-py3-none-any.whl.metadata (4.4 kB)
Collecting packaging (from gunicorn)
  Downloading packaging-24.0-py3-none-any.whl.metadata (3.2 kB)
Download gunicorn-22.0.0-py3-none-any.whl (84 kB)
  84.4/84.4 kB 4.0 MB/s eta 0:00:00
Download gunicorn-24.0-py3-none-any.whl (53 kB)
  53.5/53.5 kB 5.5 MB/s eta 0:00:00
Installing collected packages: packaging, gunicorn
Successfully installed gunicorn-22.0.0 packaging-24.0
[venv] ubuntu@ip-172-31-9-141:~/helloworld$ gunicorn -b 0.0.0.0:8000 app:app
[2024-05-12 13:08:13 +0000] [2074] [INFO] Starting gunicorn 22.0
[2024-05-12 13:08:13 +0000] [2074] [INFO] Listening at: http://0.0.0.0:8000 (2074)
[2024-05-12 13:08:13 +0000] [2075] [INFO] Using worker: sync
[2024-05-12 13:08:13 +0000] [2075] [INFO] Booting worker with pid: 2075
^C[2024-05-12 13:08:37 +0000] [2074] [INFO] Handling signal: int
[2024-05-12 13:08:37 +0000] [2075] [INFO] Worker exiting (pid: 2075)
[2024-05-12 13:08:37 +0000] [2074] [INFO] Shutting down: Master
[venv] ubuntu@ip-172-31-9-141:~/helloworld$ sudo vi /etc/systemd/system/helloworld.service
[venv] ubuntu@ip-172-31-9-141:~/helloworld$ sudo systemctl daemon-reload
[venv] ubuntu@ip-172-31-9-141:~/helloworld$ sudo systemctl start helloworld
[venv] ubuntu@ip-172-31-9-141:~/helloworld$ sudo systemctl enable helloworld
Created symlink /etc/systemd/system/multi-user.target.wants/helloworld.service → /etc/systemd/system/helloworld.service.
[venv] ubuntu@ip-172-31-9-141:~/helloworld$ sudo apt-get install nginx
Hello World!(venv) ubuntu@ip-172-31-9-141:~/helloworld$
```

i-01f69aa6e5367c01a (Flaskwebserver)

PublicIPs: 13.234.30.80 PrivateIPs: 172.31.9.141

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



```
(Reading database ... 71856 files and directories currently installed.)
Preparing to unpack .../nginx-common_1.24.0-2ubuntu7_all.deb ...
Unpacking nginx-common (1.24.0-2ubuntu7) ...
Selecting previously unselected package nginx.
Preparing to unpack .../nginx_1.24.0-2ubuntu7_amd64.deb ...
Unpacking nginx (1.24.0-2ubuntu7) ...
Setting up nginx (1.24.0-2ubuntu7) ...
Setting up nginx-common (1.24.0-2ubuntu7) ...
Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service → /usr/lib/systemd/system/nginx.service.
Processing triggers for ufw (0.36.2-6) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

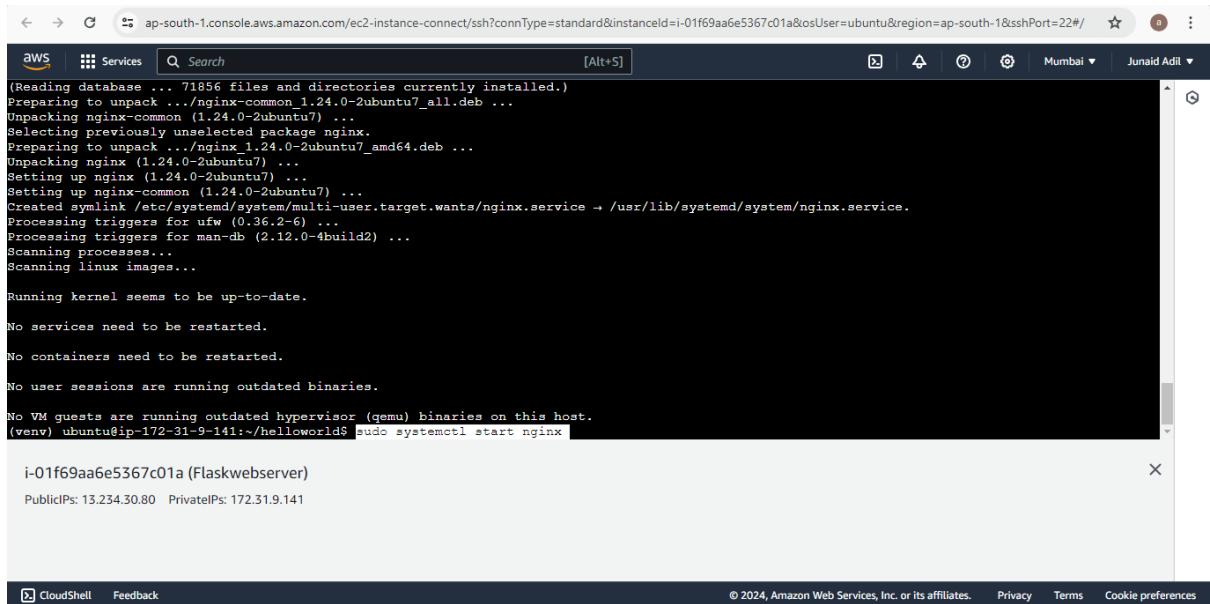
No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
(venv) ubuntu@ip-172-31-9-141:~/helloworld$
```

i-01f69aa6e5367c01a (Flaskwebserver)
Public IPs: 13.234.30.80 Private IPs: 172.31.9.141

Step 18: To start and enable the Nginx server, run the commands “**sudo systemctl start nginx**” and to enable “**sudo systemctl enable nginx**”



```
(Reading database ... 71856 files and directories currently installed.)
Preparing to unpack .../nginx-common_1.24.0-2ubuntu7_all.deb ...
Unpacking nginx-common (1.24.0-2ubuntu7) ...
Selecting previously unselected package nginx.
Preparing to unpack .../nginx_1.24.0-2ubuntu7_amd64.deb ...
Unpacking nginx (1.24.0-2ubuntu7) ...
Setting up nginx (1.24.0-2ubuntu7) ...
Setting up nginx-common (1.24.0-2ubuntu7) ...
Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service → /usr/lib/systemd/system/nginx.service.
Processing triggers for ufw (0.36.2-6) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo systemctl start nginx
```

i-01f69aa6e5367c01a (Flaskwebserver)
Public IPs: 13.234.30.80 Private IPs: 172.31.9.141

The screenshot shows the AWS CloudShell interface. At the top, the URL is `ap-south-1.console.aws.amazon.com/ec2-instance-connect/ssh?connType=standard&instanceId=i-01f69aa6e5367c01a&osUser=ubuntu®ion=ap-south-1&sshPort=22#/`. The AWS logo is on the left, followed by a "Services" button and a search bar. On the right, there are icons for "Mumbai" and "Junaid Adil". Below the header, the terminal window displays the following output:

```
Preparing to unpack .../nginx-common_1.24.0-2ubuntu7_all.deb ...
Unpacking nginx-common (1.24.0-2ubuntu7) ...
Selecting previously unselected package nginx.
Preparing to unpack .../nginx_1.24.0-2ubuntu7_amd64.deb ...
Unpacking nginx (1.24.0-2ubuntu7) ...
Setting up nginx (1.24.0-2ubuntu7) ...
Setting up nginx-common (1.24.0-2ubuntu7) ...
Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service → /usr/lib/systemd/system/nginx.service.
Processing triggers for ufw (0.36.2-6) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo systemctl start nginx
(venv) ubuntu@ip-172-31-9-141:~/helloworld$
```

At the bottom of the terminal window, it says "i-01f69aa6e5367c01a (Flaskwebserver)". Below the terminal, it lists "PublicIPs: 13.234.30.80 PrivateIPs: 172.31.9.141". The footer contains links for "CloudShell", "Feedback", "© 2024, Amazon Web Services, Inc. or its affiliates.", "Privacy", "Terms", and "Cookie preferences".

The screenshot shows the AWS CloudShell interface. At the top, the URL is `ap-south-1.console.aws.amazon.com/ec2-instance-connect/ssh?connType=standard&instanceId=i-01f69aa6e5367c01a&osUser=ubuntu®ion=ap-south-1&sshPort=22#/`. The AWS logo is on the left, followed by a "Services" button and a search bar. On the right, there are icons for "Mumbai" and "Junaid Adil". Below the header, the terminal window displays the following output:

```
Preparing to unpack .../nginx-common_1.24.0-2ubuntu7_all.deb ...
Unpacking nginx-common (1.24.0-2ubuntu7) ...
Selecting previously unselected package nginx.
Preparing to unpack .../nginx_1.24.0-2ubuntu7_amd64.deb ...
Unpacking nginx (1.24.0-2ubuntu7) ...
Setting up nginx (1.24.0-2ubuntu7) ...
Setting up nginx-common (1.24.0-2ubuntu7) ...
Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service → /usr/lib/systemd/system/nginx.service.
Processing triggers for ufw (0.36.2-6) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo systemctl start nginx
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo systemctl enable nginx
```

At the bottom of the terminal window, it says "i-01f69aa6e5367c01a (Flaskwebserver)". Below the terminal, it lists "PublicIPs: 13.234.30.80 PrivateIPs: 172.31.9.141". The footer contains links for "CloudShell", "Feedback", "© 2024, Amazon Web Services, Inc. or its affiliates.", "Privacy", "Terms", and "Cookie preferences".

The screenshot shows the AWS CloudShell interface. The terminal window displays the deployment of Nginx on an EC2 instance. The output includes:

```
Preparing to unpack .../nginx_1.24.0-2ubuntu7_amd64.deb ...
Unpacking nginx (1.24.0-2ubuntu7) ...
Setting up nginx (1.24.0-2ubuntu7) ...
Setting up nginx-common (1.24.0-2ubuntu7) ...
Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service → /usr/lib/systemd/system/nginx.service.
Processing triggers for ufw (0.36.2-6) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo systemctl start nginx
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo systemctl enable nginx
Synchronizing state of nginx.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable nginx
(venv) ubuntu@ip-172-31-9-141:~/helloworld$
```

At the bottom of the terminal, it shows the instance ID and IP address:

i-01f69aa6e5367c01a (Flaskwebserver)
PublicIPs: 13.234.30.80 PrivateIPs: 172.31.9.141

Below the terminal, there are links for CloudShell and Feedback, and a footer with copyright information and links to Privacy, Terms, and Cookie preferences.

Step 19: To make some changes in the Nginx Server we need to edit the default file using the command “**sudo vi /etc/nginx/sites-available/default**” and add the code at the top of the file and add a proxy pass to flaskhelloworld at location below

The screenshot shows the AWS CloudShell interface. The terminal window displays the deployment of Nginx on an EC2 instance. The output is identical to the previous screenshot, showing the Nginx package being installed and configured. At the bottom of the terminal, it shows the instance ID and IP address:

i-01f69aa6e5367c01a (Flaskwebserver)
PublicIPs: 13.234.30.80 PrivateIPs: 172.31.9.141

Below the terminal, there are links for CloudShell and Feedback, and a footer with copyright information and links to Privacy, Terms, and Cookie preferences.

The screenshot shows a terminal window in the AWS CloudShell interface. The user is editing an Nginx configuration file using the vi editor. The configuration includes an upstream block for 'flaskelloworld' and a server block with SSL support. The terminal also displays the instance ID, public and private IP addresses, and the current memory usage.

```
# https://wiki.debian.org/Nginx/DirectoryStructure
#
# In most cases, administrators will remove this file from sites-enabled/ and
# leave it as reference inside of sites-available where it will continue to be
# updated by the nginx packaging team.
#
# This file will automatically load configuration files provided by other
# applications, such as Drupal or Wordpress. These applications will be made
# available underneath a path with that package name, such as /drupal8.
#
# Please see /usr/share/doc/nginx-doc/examples/ for more detailed examples.
##

# Default server configuration
#
upstream flaskelloworld {
    server 127.0.0.1:8000;
}

server {
    listen 80 default_server;
    listen [::]:80 default_server;

    # SSL configuration
-- INSERT --

```

i-01f69aa6e5367c01a (Flaskwebserver)
PublicIPs: 13.234.30.80 PrivateIPs: 172.31.9.141

The screenshot shows the completed Nginx configuration. It includes a self-signed certificate snippet and a location block for PHP scripts. The configuration file is now saved and ready to be used.

```
#
# Self signed certs generated by the ssl-cert package
# Don't use them in a production server!
#
# include snippets/snakeoil.conf;

root /var/www/html;

# Add index.php to the list if you are using PHP
index index.html index.htm index.nginx-debian.html;

server_name _;

location / {
    # First attempt to serve request as file, then
    # as directory, then fall back to displaying a 404.
    proxy_pass http://flaskelloworld;#
    try_files $uri $uri/ =404;
}

# pass PHP scripts to FastCGI server
#
#location ~ \.php$ {
-- INSERT --

```

i-01f69aa6e5367c01a (Flaskwebserver)
PublicIPs: 13.234.30.80 PrivateIPs: 172.31.9.141

Then save the file :wq!

The screenshot shows the AWS CloudShell interface. The terminal window displays the deployment process of a Flask application. It starts with the unpacking and configuration of Nginx, followed by the creation of symlinks and the enablement of the Nginx service. The user then edits the Nginx configuration file to point to the Flask application's static files. Finally, the Nginx service is restarted.

```
Unpacking nginx (1.24.0-2ubuntu7) ...
Setting up nginx (1.24.0-2ubuntu7) ...
Setting up nginx-common (1.24.0-2ubuntu7) ...
Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service → /usr/lib/systemd/system/nginx.service.
Processing triggers for ufw (0.36.2-6) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo systemctl start nginx
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo systemctl enable nginx
Synchronizing state of nginx.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable nginx
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo vi /etc/nginx/sites-available/default
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ 
```

i-01f69aa6e5367c01a (Flaskwebserver)

PublicIPs: 13.234.30.80 PrivateIPs: 172.31.9.141

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Step 20: To restart the Nginx server run the command “**sudo systemctl restart nginx**”

The screenshot shows the AWS CloudShell interface. The terminal window displays the deployment process of a Flask application. It starts with the unpacking and configuration of Nginx, followed by the creation of symlinks and the enablement of the Nginx service. The user then edits the Nginx configuration file to point to the Flask application's static files. Finally, the Nginx service is restarted.

```
Unpacking nginx (1.24.0-2ubuntu7) ...
Setting up nginx (1.24.0-2ubuntu7) ...
Setting up nginx-common (1.24.0-2ubuntu7) ...
Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service → /usr/lib/systemd/system/nginx.service.
Processing triggers for ufw (0.36.2-6) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

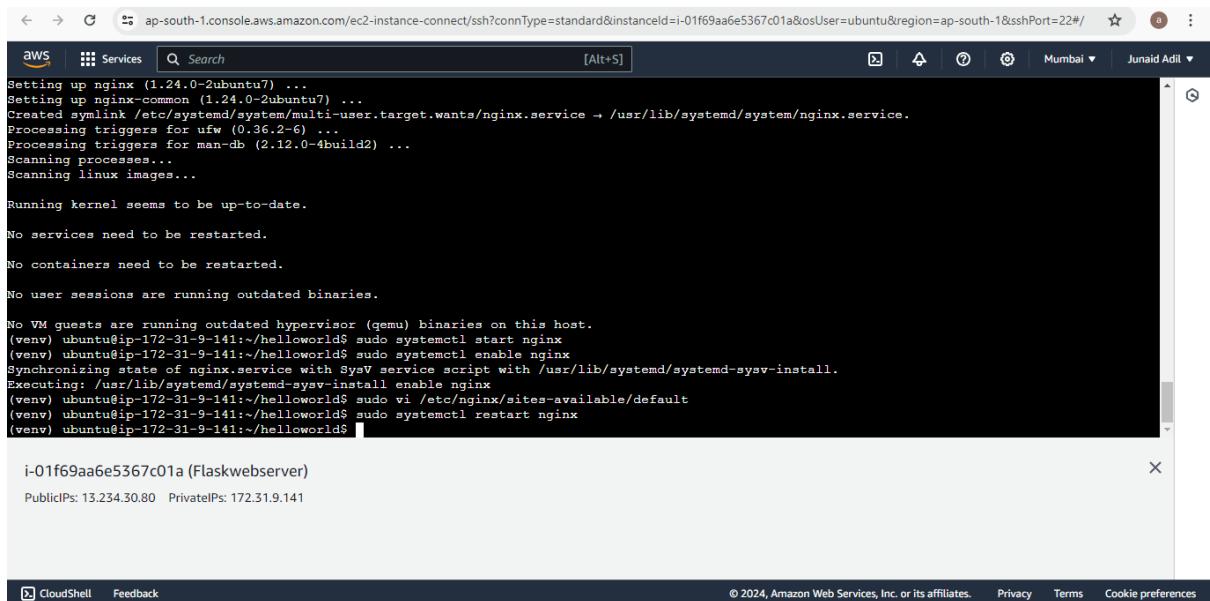
No VM guests are running outdated hypervisor (qemu) binaries on this host.
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo systemctl start nginx
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo systemctl enable nginx
Synchronizing state of nginx.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable nginx
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo vi /etc/nginx/sites-available/default
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo systemctl restart nginx 
```

i-01f69aa6e5367c01a (Flaskwebserver)

PublicIPs: 13.234.30.80 PrivateIPs: 172.31.9.141

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

We can see we didn't get any error, so the server is working fine.



```
Setting up nginx (1.24.0-2ubuntu7) ...
Setting up nginx-common (1.24.0-2ubuntu7) ...
Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service → /usr/lib/systemd/system/nginx.service.
Processing triggers for ufw (0.36.2-6) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

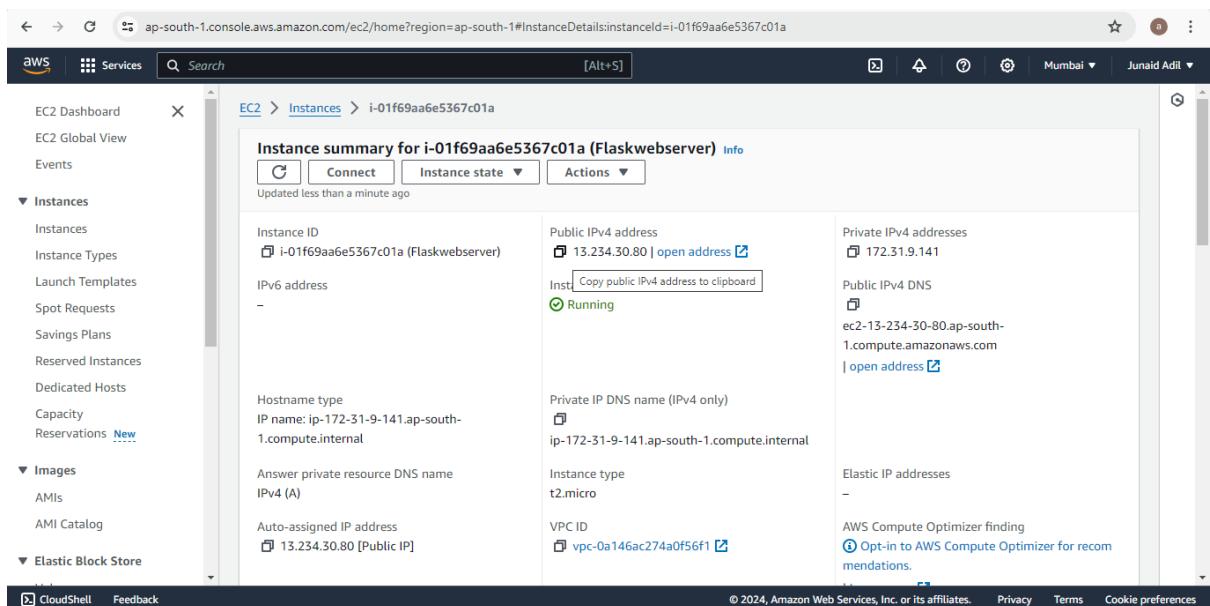
No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo systemctl start nginx
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo systemctl enable nginx
Synchronizing state of nginx.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable nginx
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo vi /etc/nginx/sites-available/default
(venv) ubuntu@ip-172-31-9-141:~/helloworld$ sudo systemctl restart nginx
(venv) ubuntu@ip-172-31-9-141:~/helloworld$
```

i-01f69aa6e5367c01a (Flaskwebserver)
PublicIPs: 13.234.30.80 PrivateIPs: 172.31.9.141

Step 21: To check go to the instance and copy the Public IP and paste it in the browser and press enter



EC2 Dashboard EC2 Global View Events

▼ Instances

- Instances
- Instance Types
- Launch Templates
- Spot Requests
- Savings Plans
- Reserved Instances
- Dedicated Hosts
- Capacity
- Reservations [New](#)

▼ Images

- AMIs
- AMI Catalog

▼ Elastic Block Store

CloudShell Feedback

Instance summary for i-01f69aa6e5367c01a (Flaskwebserver) [Info](#)

Updated less than a minute ago	Actions
Instance ID i-01f69aa6e5367c01a (Flaskwebserver)	Public IPv4 address 13.234.30.80 open address
IPv6 address -	Private IPv4 addresses 172.31.9.141
Hostname type IP name: ip-172-31-9-141.ap-south-1.compute.internal	Public IPv4 DNS ec2-13-234-30-80.ap-south-1.compute.amazonaws.com open address
Answer private resource DNS name IPv4 (A)	Private IP DNS name (IPv4 only) ip-172-31-9-141.ap-south-1.compute.internal
Auto-assigned IP address 13.234.30.80 [Public IP]	Instance type t2.micro
VPC ID vpc-0a146ac274a0f56f1	Elastic IP addresses -
AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations.	

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

