

## **Placement Empowerment Program**

### ***Cloud Computing and DevOps Centre***

#### **Deploy a Web Application on the Cloud**

Write a Python Flask application and deploy it on your cloud VM. Configure the firewall to allow HTTP traffic.

Name: Nanthan Krishnan.G.K.

Department: AML

# Introduction and Overview

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

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

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

## Objectives

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

## Importance

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

# STEPS:

## Step 1: Write a Simple Flask Application

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

```
from flask import Flask
```

```
app = Flask(__name__)
```

```
@app.route('/')
```

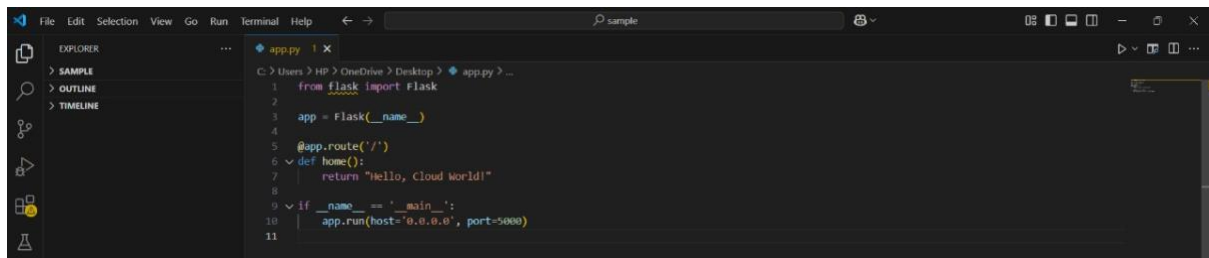
```
def home():
```

```
    return "Hello, Cloud World!"
```

```
if __name__ == '__main__':
```

```
    app.run(host='0.0.0.0', port=5000)
```

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



## STEP 2: Prepare Your EC2 Instance:

- Connect to your EC2 instance:

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

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

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

PS C:\Users\WDP> cd downloads
PS C:\Users\WDP\downloads> ssh -i "key.pem" ec2-user@ec2-3-238-247-34.compute-1.amazonaws.com
The authenticity of host 'ec2-3-238-247-34.compute-1.amazonaws.com (3.238.247.34)' can't be established.
ED25519 key fingerprint is SHA256:cNg44Xau19if4Bv6gUVSf0bsFQVzJq7hNab7Lveh82M.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-238-247-34.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
```

```
Amazon Linux 2023

#
#####
###
#/# https://aws.amazon.com/linux/amazon-linux-2023
V-->

---
--
--
--
[ec2-user@ip-172-30-5-173 ~]$
```

- Update packages:

```
sudo yum update -y
```

- Install Python and pip:

```
sudo yum install python3 python3-pip -y
```

```
[ec2-user@ip-172-30-5-173 ~]$ sudo yum install python3-pip -y
Last metadata expiration check: 0:02:05 ago on Wed Feb 5 03:02:11 2025.
Dependencies resolved.

=====
Package                Arch      Version                               Repository      Size
=====
Installing:
python3-pip            noarch   21.3.1-2.amzn2023.0.10             amazonlinux     1.8 M
Installing weak dependencies:
libxcrypt-compat       x86_64   4.4.33-7.amzn2023                 amazonlinux     92 k
=====
Transaction Summary
=====
Install 2 Packages

Total download size: 1.9 M
Installed size: 11 M
Downloading Packages:
(1/2): libxcrypt-compat-4.4.33-7.amzn2023.x.1.6 MB/s | 92 kB 00:00
(2/2): python3-pip-21.3.1-2.amzn2023.0.10.n 20 MB/s | 1.8 MB 00:00
-----
Total                               14 MB/s | 1.9 MB 00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
Preparing : 1/1
Installing : libxcrypt-compat-4.4.33-7.amzn2023.x86_64 1/1
Installing : python3-pip-21.3.1-2.amzn2023.0.10.noarch 2/2
Running scriptlet: python3-pip-21.3.1-2.amzn2023.0.10.noarch 2/2
Verifying : libxcrypt-compat-4.4.33-7.amzn2023.x86_64 1/2
Verifying : python3-pip-21.3.1-2.amzn2023.0.10.noarch 2/2

Installed:
libxcrypt-compat-4.4.33-7.amzn2023.x86_64
python3-pip-21.3.1-2.amzn2023.0.10.noarch

Complete!
[ec2-user@ip-172-30-5-173 ~]$
```

- Install Flask:

`pip3 install Flask`

```
[ec2-user@ip-172-30-5-173 ~]$ pip3 install Flask
Defaulting to user installation because normal site-packages is not writeable
Collecting Flask
  Downloading flask-3.1.0-py3-none-any.whl (102 kB)
    |#####| 102 kB 11.1 MB/s
Collecting click>=8.1.3
  Downloading click-8.1.8-py3-none-any.whl (98 kB)
    |#####| 98 kB 13.1 MB/s
Collecting itsdangerous>=2.2
  Downloading itsdangerous-2.2.0-py3-none-any.whl (16 kB)
Collecting Werkzeug>=3.1
  Downloading werkzeug-3.1.3-py3-none-any.whl (224 kB)
    |#####| 224 kB 40.4 MB/s
Collecting blinker>=1.9
  Downloading blinker-1.9.0-py3-none-any.whl (8.5 kB)
Collecting Jinja2>=3.1.2
  Downloading jinja2-3.1.5-py3-none-any.whl (134 kB)
    |#####| 134 kB 92.6 MB/s
Collecting importlib-metadata>=3.6
  Downloading importlib_metadata-8.6.1-py3-none-any.whl (26 kB)
Collecting zipp>=3.20
  Downloading zipp-3.21.0-py3-none-any.whl (9.6 kB)
Collecting MarkupSafe>=2.0
  Downloading MarkupSafe-3.0.2-cp39-cp39-manylinux_2_17_x86_64_manylinux2014_x86_64.whl (20 kB)
Installing collected packages: zipp, MarkupSafe, Werkzeug, Jinja2, itsdangerous, importlib-metadata, click, blinker, Flask
Successfully installed Flask-3.1.0 Jinja2-3.1.5 MarkupSafe-3.0.2 Werkzeug-3.1.3 blinker-1.9.0 click-8.1.8 importlib-metadata-8.6.1 itsdangerous-2.2.0 zipp-3.21.0
[ec2-user@ip-172-30-5-173 ~]$
```

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

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

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

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

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

PS C:\Users\HP> scp -i "C:\Users\HP\Downloads\key.pem" "C:\Users\HP\OneDrive\Desktop\app.py" ec2-user@3.238.247.34:/home/ec2-user/app.py
The authenticity of host '3.238.247.34 (3.238.247.34)' can't be established.
ED25519 key fingerprint is SHA256:cNg44Xau19f4Bv6gUVSM6b5FGV2Jq7hNwb7Lveh82M.
This host key is known by the following other names/addresses:
  C:\Users\HP/.ssh/known_hosts:1: ec2-3-238-247-34.compute-1.amazonaws.com
Are you sure you want to continue connecting (yes/no/[fingerprint])?
Please type 'yes', 'no' or the fingerprint:
Warning: Permanently added '3.238.247.34' (ED25519) to the list of known hosts.
app.py
PS C:\Users\HP> | 100% 186 0.5KB/s 00:00
```

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

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

**python3 app.py**

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

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

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

Edit inbound rules [Info](#)

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

Security group rule ID	Type <a href="#">Info</a>	Protocol <a href="#">Info</a>	Port range <a href="#">Info</a>	Source <a href="#">Info</a>	Description - optional <a href="#">Info</a>	
sgr-0a38e766a15ab6c6a	Custom TCP	TCP	5000	Custom	0.0.0.0/0	Delete

- Save the rules.

## STEP 6: Access the Application:

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

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



## Expected Outcome

By completing this POC, you will:

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