FLASK APP DEV AND DEPLOYMENT

CMPE 285 – SOFTWARE ENGINEERING PROCESSES

PROFESSOR – RICHARD SINN

ISA – PUSHYA CHANDRA LEBURU

AGENDA

- What is Flask?
- Develop a small Flask Application(s)
- Deploy Flask on AWS EC2

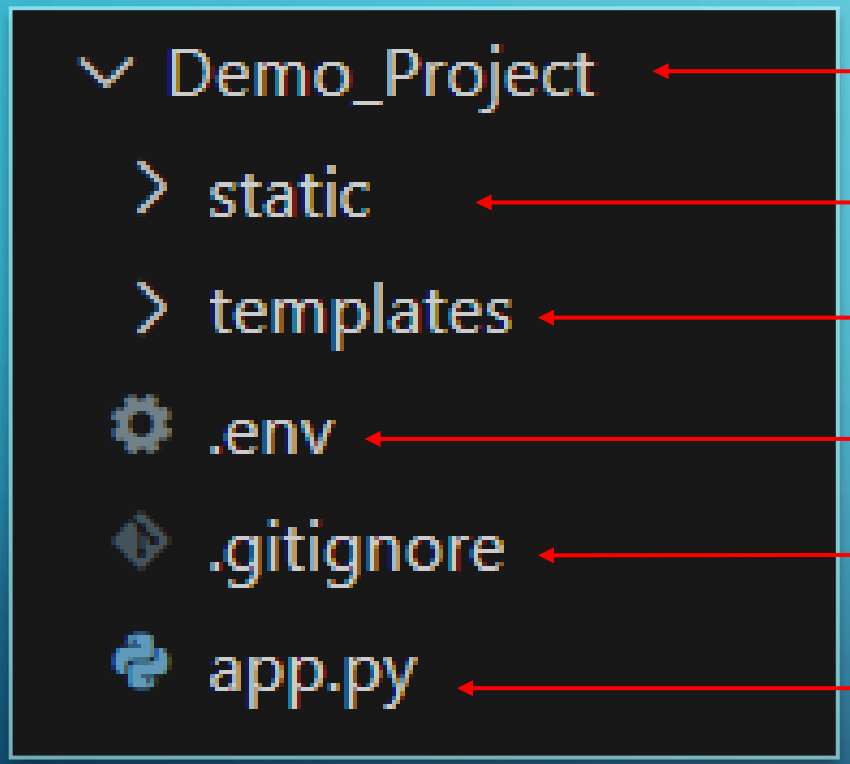
WHAT IS FLASK?

- Flask is a web application framework written in Python. It was developed by Armin Ronacher, who led a team of international Python enthusiasts called Poocco. Flask is based on the Werkzeg WSGI toolkit and the Jinja2 template engine.Both are Pocco projects.

- Flask vs Django – (micro framework vs full-featured)

DEVELOPMENT

FOLDER STRUCTURE



Project Name

This folder contains css stylesheets, images, etc

Contains html files

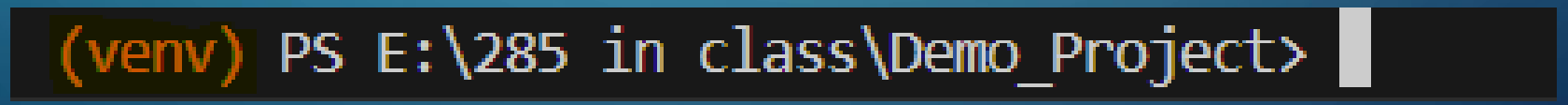
Contains sensitive information like API Keys

Git ignore to ignore file uploaded to your git

Entry Point to the application

INSTALL AND START VIRTUAL ENVIRONMENT

- Install Virtual Environment:
  - Command: pip install virtualenv
- Create Virtual Environment in your Directory
  - Command: virtualenv venv
  - This would create a folder venv in your directory
- Start the Virtual Environment in your directory
  - Command: .\venv\Scripts\activate
  - After starting the virtual environment your command line would typically have the following (venv) in it:



INSTALL FLASK

- You can install Flask using pip and it would install all the necessary packages.
  - Command: pip install Flask

```
(venv) PS E:\285 in class\Demo_Project> pip install Flask
Collecting Flask
  Using cached flask-3.0.2-py3-none-any.whl.metadata (3.6 kB)
Collecting Werkzeug>=3.0.0 (from Flask)
  Using cached werkzeug-3.0.1-py3-none-any.whl.metadata (4.1 kB)
Collecting Jinja2>=3.1.2 (from Flask)
  Using cached Jinja2-3.1.3-py3-none-any.whl.metadata (3.3 kB)
Collecting itsdangerous>=2.1.2 (from Flask)
  Using cached itsdangerous-2.1.2-py3-none-any.whl.metadata (2.9 kB)
Collecting click>=8.1.3 (from Flask)
  Using cached click-8.1.7-py3-none-any.whl.metadata (3.0 kB)
Collecting blinker>=1.6.2 (from Flask)
  Using cached blinker-1.7.0-py3-none-any.whl.metadata (1.9 kB)
Collecting colorama (from click>=8.1.3->Flask)
  Using cached colorama-0.4.6-py2.py3-none-any.whl.metadata (17 kB)
Collecting MarkupSafe>=2.0 (from Jinja2>=3.1.2->Flask)
  Using cached MarkupSafe-2.1.5-cp311-cp311-win_amd64.whl.metadata (3.1 kB)
Using cached flask-3.0.2-py3-none-any.whl (101 kB)
Using cached blinker-1.7.0-py3-none-any.whl (13 kB)
Using cached click-8.1.7-py3-none-any.whl (97 kB)
Using cached itsdangerous-2.1.2-py3-none-any.whl (15 kB)
Using cached Jinja2-3.1.3-py3-none-any.whl (133 kB)
Using cached werkzeug-3.0.1-py3-none-any.whl (226 kB)
Using cached MarkupSafe-2.1.5-cp311-cp311-win_amd64.whl (17 kB)
Using cached colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Installing collected packages: MarkupSafe, itsdangerous, colorama, blinker, Werkzeug, Jinja2, click, Flask
Successfully installed Flask-3.0.2 Jinja2-3.1.3 MarkupSafe-2.1.5 Werkzeug-3.0.1 blinker-1.7.0 click-8.1.7 colorama-0.4.6 itsdangerous-2.1.2
(venv) PS E:\285 in class\Demo_Project>
```

SIMPLE APPLICATION – HELLO WORLD

app.py

Demo_Project > app.py

```python
from flask import Flask # Imports the Flask class form the Flask library

app = Flask('__name__') # Creates an instace of the Flask Class

@app.route('/') # Aids Flask to trigger the function for a given specific URL
def hello_world():
    return 'Hello'

if(__name__=='__main__'):
    app.run() # Runs the application

```

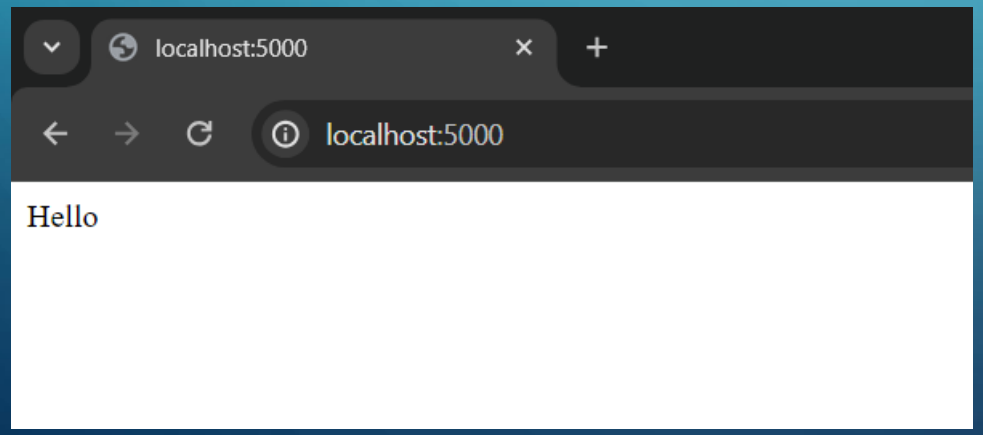
- After writing the above code in app.py, run it by using the following command from its respective directory.
  - Command: python app.py

SIMPLE APPLICATION – HELLO WORLD (CONTD.)

- Output on Command Line:

```
(venv) PS E:\285 in class\Demo_Project> python app.py
 * Tip: There are .env or .flaskenv files present. Do "pip install python-dotenv" to use them.
 * Serving Flask app '__name__'
 * 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
```

- Output on Web Browser:
  - The Default host would be 127.0.0.1 (localhost) and the default port is 5000



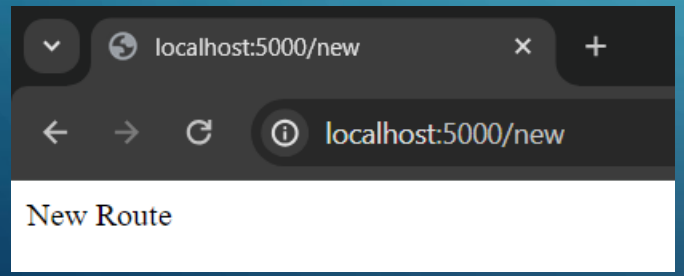
- You can change to desired host and port number in the app.run().

```
if(__name__=='__main__'):
    app.run(host="Desired host",port="Desired port number",debug=True)
```

ADDING A NEW ROUTE

- You can add a new route by simply specifying the necessary route URI in the @app.route('/<Specify required Route URI>')

- Output:



app.py

Demo_Project > app.py

```python
from flask import Flask

app = Flask('__name__')

@app.route('/')
def hello_world():
    return 'Hello'

@app.route('/new') # The New Route URI is /new
def new_route():
    return 'New Route'


if(__name__=='__main__'):
    app.run()

```

RENDER TEMPLATE

app.py

```python
from flask import Flask,render_template # Importing Render Template

app = Flask('__name__')

@app.route('/')
def home():
    return render_template('index.html') # Renders the html file and displays it when on a specific route

if(__name__=='__main__'):
    app.run()
```

index.html

```html
<!DOCTYPE html>
<html>
    <head>
        <title>Main Page</title>
        <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='css/style.css') }}">
    </head>
    <body>
        <h1>Testing!!!</h1>
    </body>
</html>
```

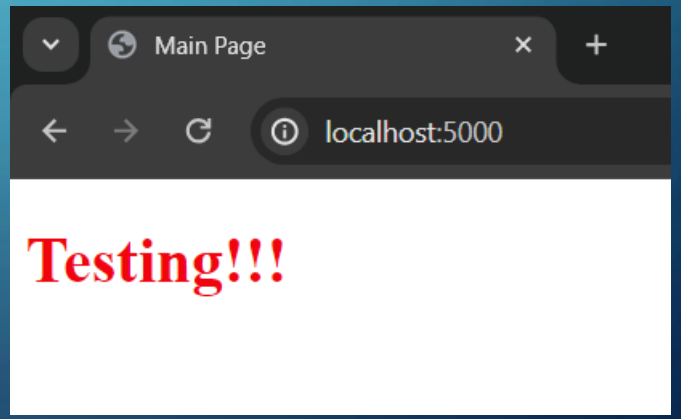
Links the stylesheet to the required css file, the css file style.css is placed in static/css/style.css

Sets the color of text in h1 tag to red

style.css

```css
h1{
    color: red;
}
```

Output:



VARIABLE PASSING TO TEMPLATE

app.py

Demo_Project > app.py

```python
from flask import Flask,render_template

app = Flask('__name__')

@app.route('/hello/<name>')
def hello(name):
    return render_template('hello.html',name=name)

if(__name__=='__main__'):
    app.run()

```

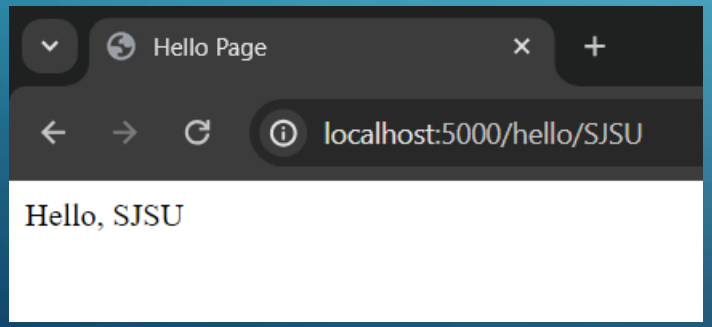
hello.html

Demo_Project > templates > hello.html

```html
<!DOCTYPE html>
<html>
    <head>
        <title>Hello Page</title>
    </head>
    <body>
        Hello, {{name}}
    </body>
</html>

```

Output:



CALCULATOR EXAMPLE – REQUEST OBJECT

- You have to import ‘request’
- The calculator function receives a POST request. It takes form data and extracts floating-point integers and an operation. It then computes the result (add, subtract, multiply, or divide) based on the operation and send the result to the html file rendering it

app.py

Demo_Project > app.py

```python
from flask import Flask,render_template,request

app = Flask('__name__')

@app.route('/calculator', methods=['GET', 'POST'])
def calculator():
    result = None
    if request.method == 'POST':
        num1 = request.form.get('num1', type=float)
        num2 = request.form.get('num2', type=float)
        operation = request.form.get('operation')

        if operation == 'add':
            result = num1 + num2
        elif operation == 'subtract':
            result = num1 - num2
        elif operation == 'multiply':
            result = num1 * num2
        elif operation == 'divide':
            result = num1 / num2 if num2 != 0 else 'Infinity'
    return render_template('calculator.html', result=result)

if(__name__=='__main__'):
    app.run()
```

CALCULATOR EXAMPLE – REQUEST OBJECT (CONTD.)

- Following is the html code for calculator.html, this file must be placed in the templates folder.
- Here you can input 2 numbers and select the operation that you want to perform.
- The it would generate the result and display it on the screen

calculator.html

Demo_Project > templates > calculator.html

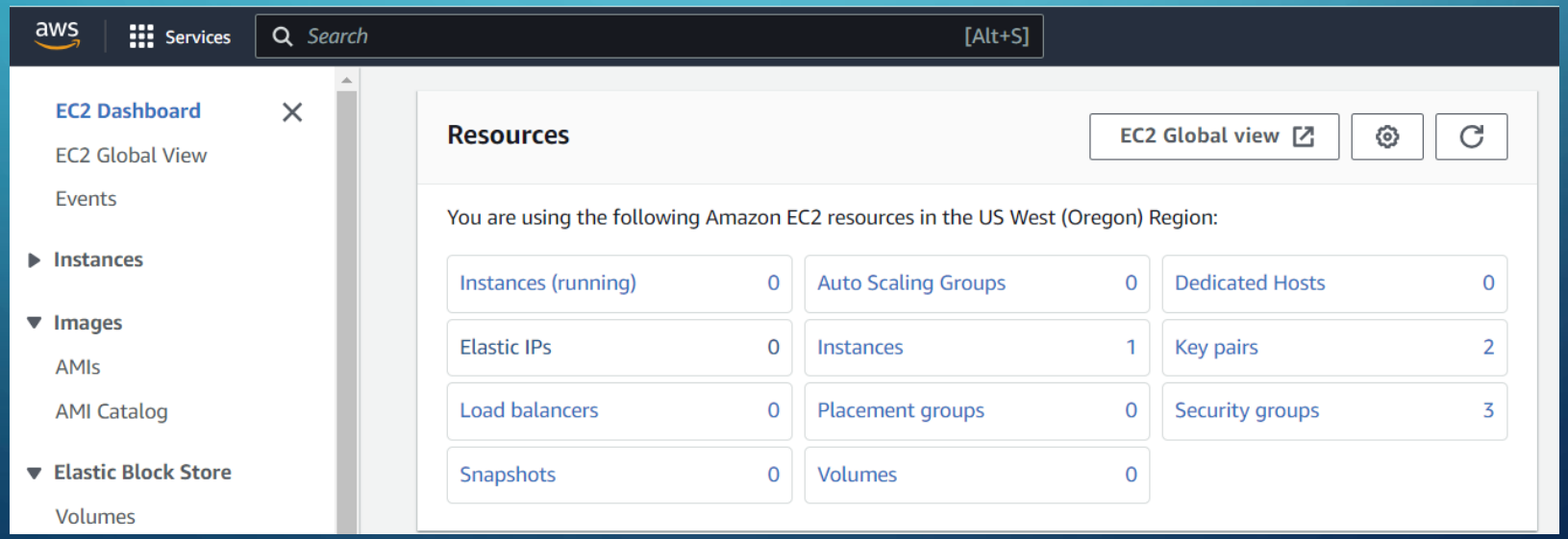
```
<!DOCTYPE html>
<html>
<head>
    <title>Calculator</title>
    <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='css/style.css') }}">
</head>
<body>
    <h1>Calculator</h1>
    <form method="post">
        <input type="number" name="num1" placeholder="Number 1" required>
        <input type="number" name="num2" placeholder="Number 2" required>
        <select name="operation">
            <option value="add">Add</option>
            <option value="subtract">Subtract</option>
            <option value="multiply">Multiply</option>
            <option value="divide">Divide</option>
        </select>
        <button type="submit">Calculate</button>
    </form>
    {% if result is not none %}
        <div class="result">Result: {{ result }}</div>
    {% endif %}
</body>
</html>

```

DEPLOYMENT

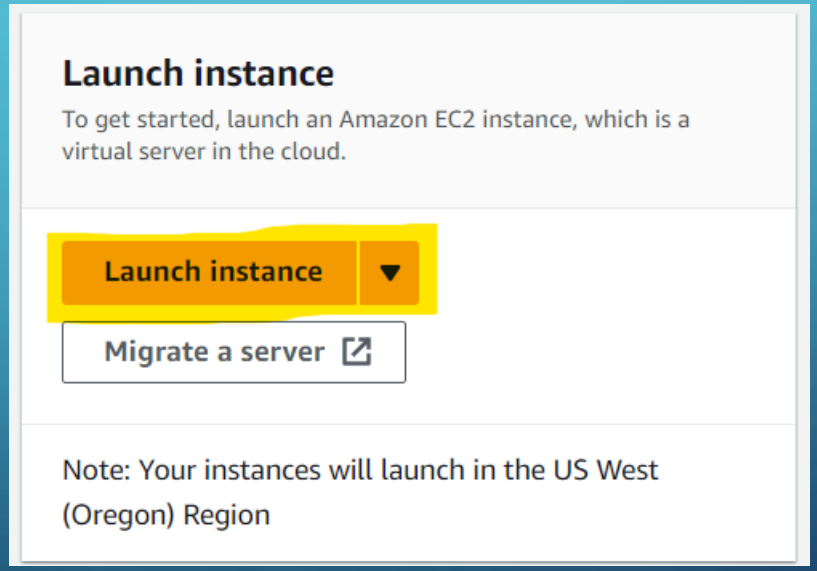
STEPS TO DEPLOY ON AWS EC2

- Login to AWS console on the following link: https://aws.amazon.com/console/
- Look for EC2 on the search bar and select it, you will be redirected to EC2 Dashborad



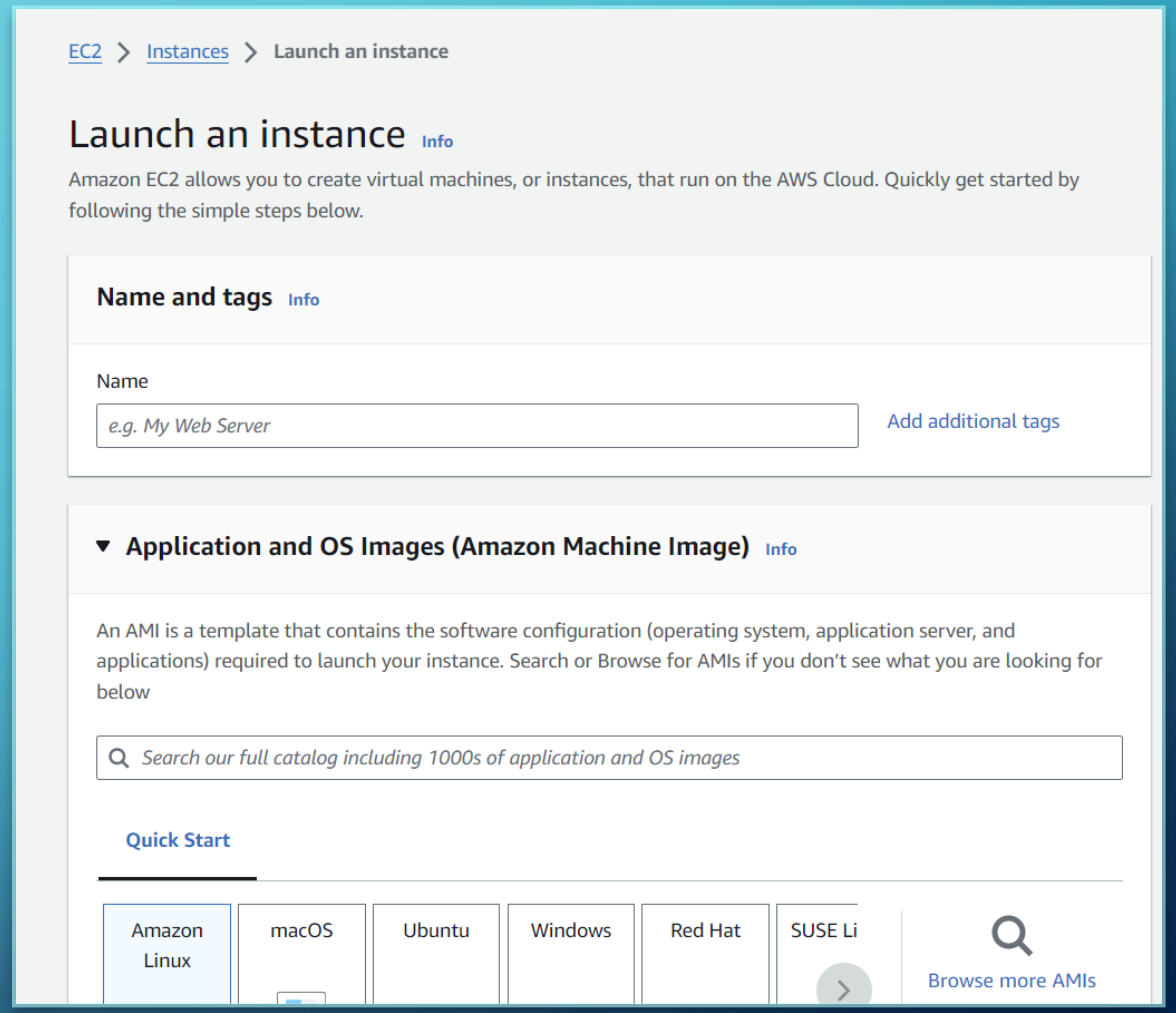
STEPS TO DEPLOY ON AWS EC2 (CONTD.)

- On the EC2 Dashboard click on Launch Instance to create and launch a new instance.



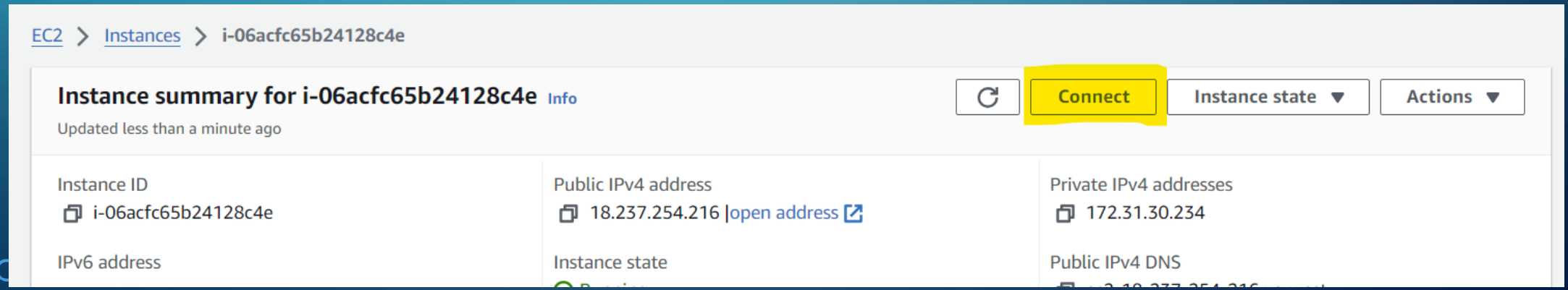
STEPS TO DEPLOY ON AWS EC2 (CONTD.)

- In the following page, name your instance
- Select the Operating system, we will be using ‘Ubuntu’
- Create a key pair(if needed)
- And Launch the instance



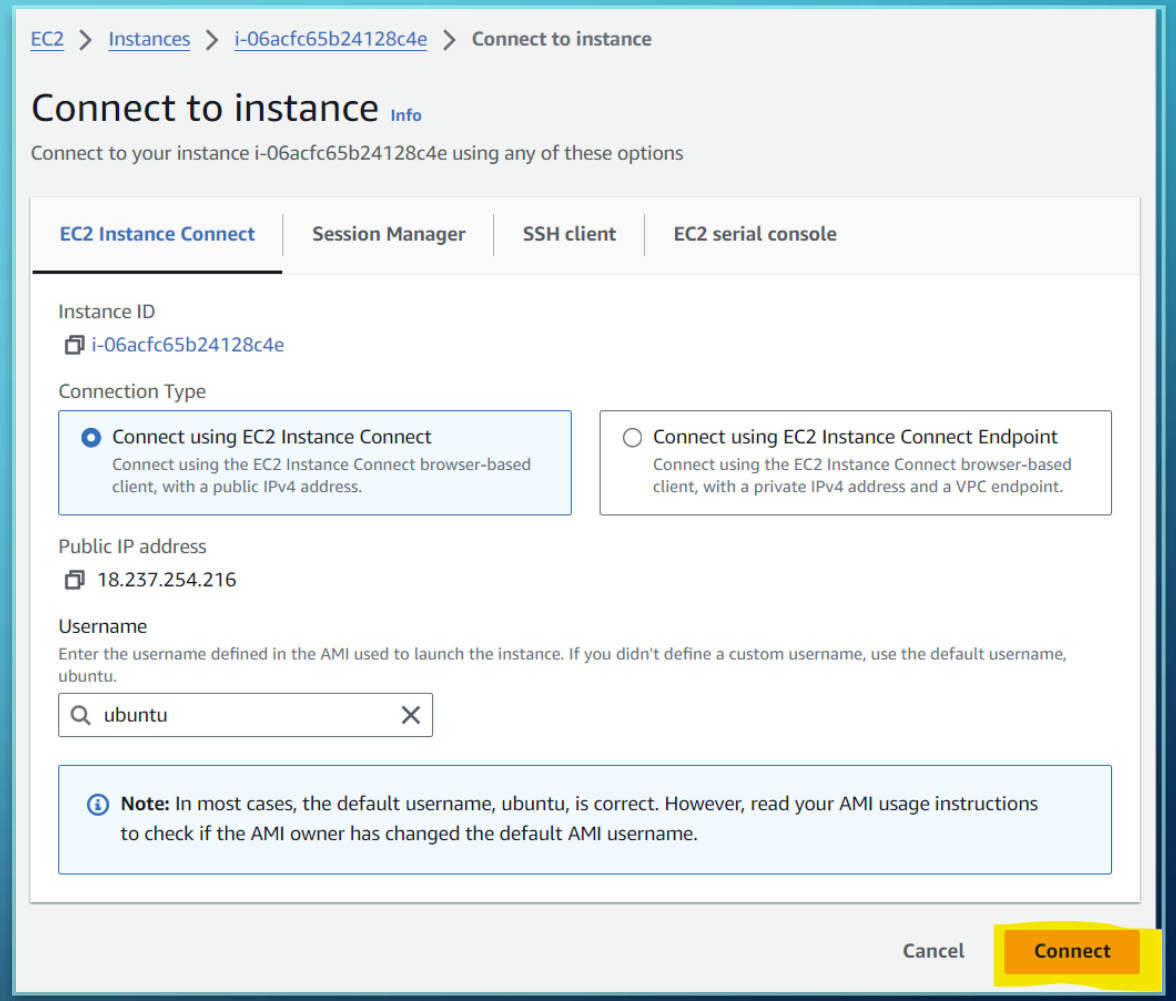
STEPS TO DEPLOY ON AWS EC2 (CONTD.)

- Once the instance is created you can find it under EC2 > Instances (From the EC2 Dashboard)
- Run it open it
- Now Click on Connect to connect to the instance



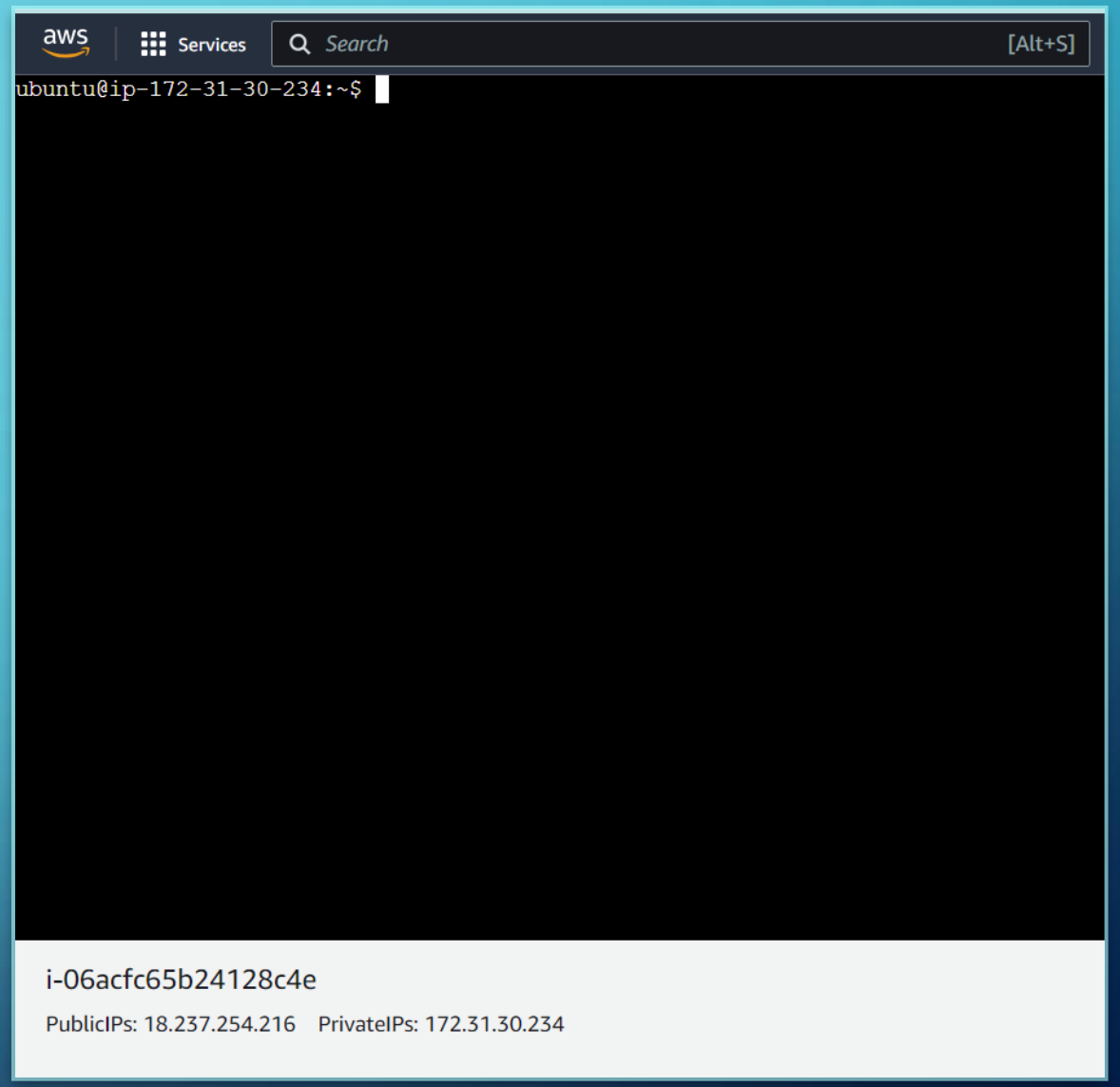
STEPS TO DEPLOY ON AWS EC2 (CONTD.)

- It will prompt to a new page, Here click on connect to connect to the instance.



STEPS TO DEPLOY ON AWS EC2 (CONTD.)

- The following screen will open and now you can access the instance.



STEPS TO DEPLOY ON AWS EC2 (CONTD.)

- First, we will update the package manager:
  - Command: sudo apt-get update
- Now Check and install python and pip if not installed.
- Install git, we will be cloning the code directly from the github.
  - Command: sudo apt-get install git
- Clone your Git Repository to the instance.
  - Command: git clone <your-repository>

STEPS TO DEPLOY ON AWS EC2 (CONTD.)

- Navigate to the location where you have app.py

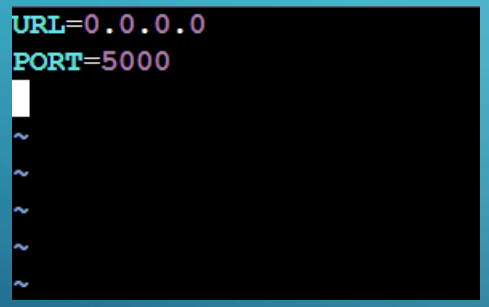
```
ubuntu@ip-172-31-30-234:~/Flask_Dev_Deployment/Demo_Project$ ls
app.py  static  templates
ubuntu@ip-172-31-30-234:~/Flask_Dev_Deployment/Demo_Project$
```

- Install Flask
  - Command: pip install Flask

```
ubuntu@ip-172-31-30-234:~/Flask_Dev_Deployment/Demo_Project$ pip install Flask
Defaulting to user installation because normal site-packages is not writeable
Collecting Flask
  Downloading flask-3.0.2-py3-none-any.whl (101 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 101.3/101.3 KB 3.0 MB/s eta 0:00:00
Collecting Werkzeug>=3.0.0
  Downloading werkzeug-3.0.1-py3-none-any.whl (226 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 226.7/226.7 KB 7.3 MB/s eta 0:00:00
Collecting click>=8.1.3
  Downloading click-8.1.7-py3-none-any.whl (97 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 97.9/97.9 KB 9.8 MB/s eta 0:00:00
Collecting itsdangerous>=2.1.2
  Downloading itsdangerous-2.1.2-py3-none-any.whl (15 kB)
Collecting blinker>=1.6.2
  Downloading blinker-1.7.0-py3-none-any.whl (13 kB)
Collecting Jinja2>=3.1.2
  Downloading Jinja2-3.1.3-py3-none-any.whl (133 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 133.2/133.2 KB 10.1 MB/s eta 0:00:00
Requirement already satisfied: MarkupSafe>=2.0 in /usr/lib/python3/dist-packages (from Jinja2>=3.1.2->Flask) (2.0.1)
Collecting MarkupSafe>=2.0
  Downloading MarkupSafe-2.1.5-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (25 kB)
Installing collected packages: MarkupSafe, itsdangerous, click, blinker, Werkzeug, Jinja2, Flask
  WARNING: The script flask is installed in '/home/ubuntu/.local/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed Flask-3.0.2 Jinja2-3.1.3 MarkupSafe-2.1.5 Werkzeug-3.0.1 blinker-1.7.0 click-8.1.7 itsdangerous-2.1.2
ubuntu@ip-172-31-30-234:~/Flask_Dev_Deployment/Demo_Project$
```

STEPS TO DEPLOY ON AWS EC2 (CONTD.)

- Create a .env file here to have the URL and PORT set on it
  - Command: vi .env
  - Then press ‘i’ to inset into the file and insert the below text.



  - Save and exit by pressing ‘esc’ key and typing ‘:wq’ and ‘enter’ key

STEPS TO DEPLOY ON AWS EC2 (CONTD.)

Note:

- The app.py file has been changed to accommodate reading from .env file
- The following changes are made in app.py. (imports section and app.run() function parameters)

```
import os

from dotenv import load_dotenv
load_dotenv()

from flask import Flask, request, render_template
```

```
if __name__ == '__main__':
    app.run(
        host=os.getenv('URL'),
        port=os.getenv('PORT'),
        debug=True
        )
```

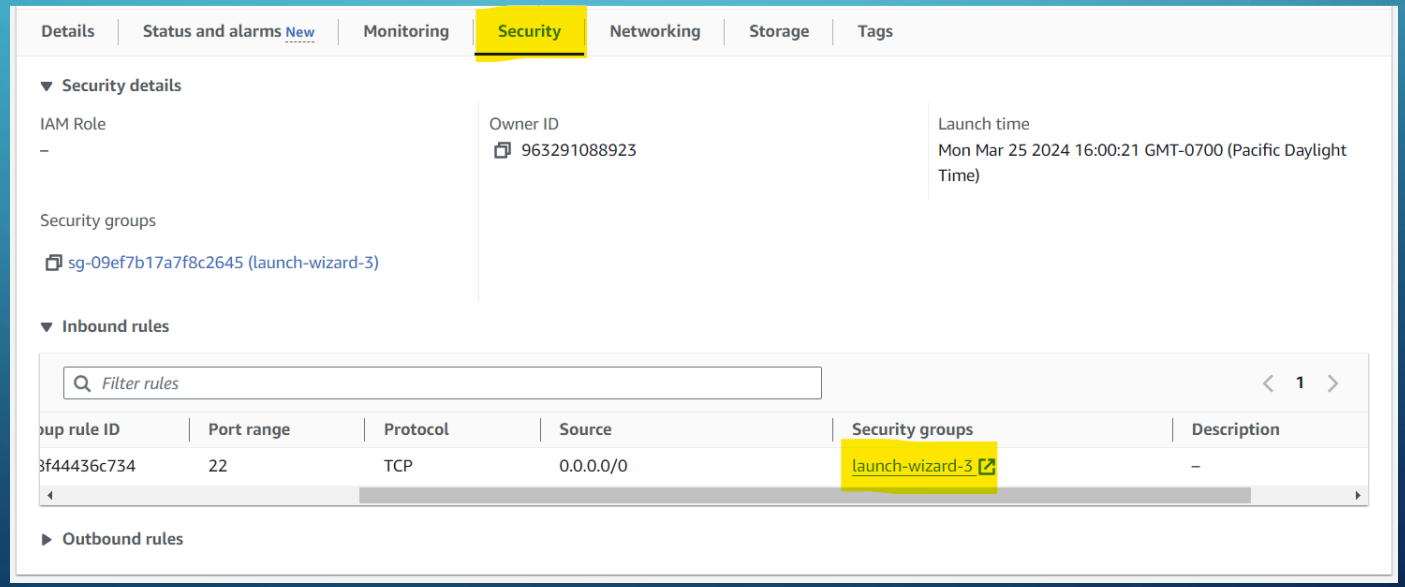
STEPS TO DEPLOY ON AWS EC2 (CONTD.)

- Install python-dotenv to enable python code to read from .env file:
  - Command: pip install python-dotenv
- Start the python code, by running the following command
  - Command: python3 app.py

```
ubuntu@ip-172-31-30-234:~/Flask_Dev_Deployment/Demo_Project$ python3 app.py
 * Serving Flask app '__name__'
 * Debug mode: on
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://172.31.30.234:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 120-727-364
```

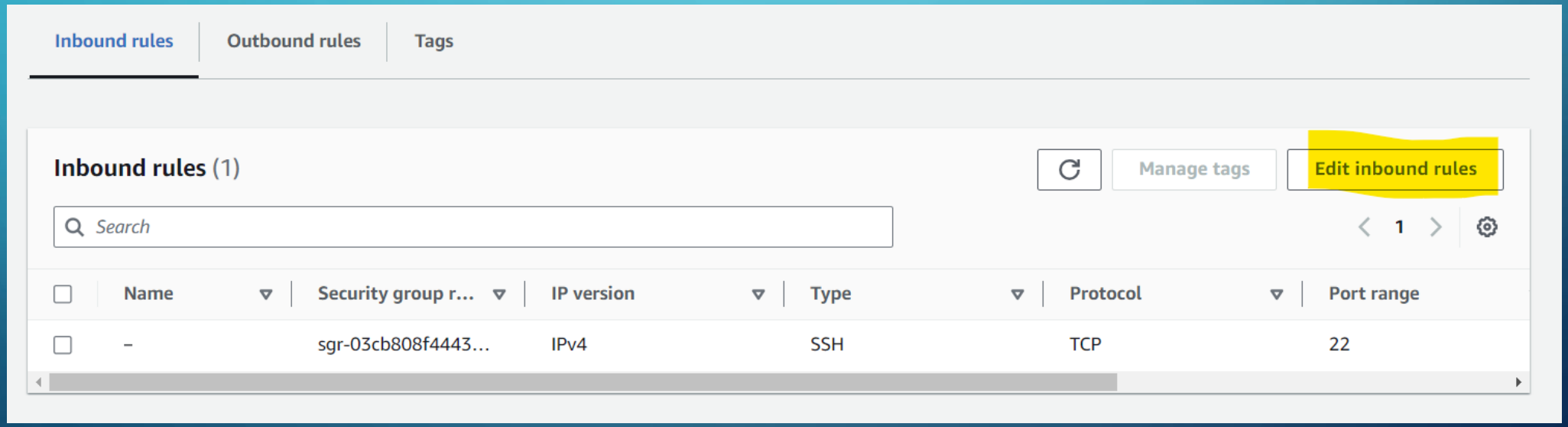
STEPS TO DEPLOY ON AWS EC2 (CONTD.)

- You have to edit the Inbound rules for this instance to make it accessible from the browser.
- Go to the Security section on your instance summary page and select the Security tab, next select the Inbound Rule Section and launch the security group.



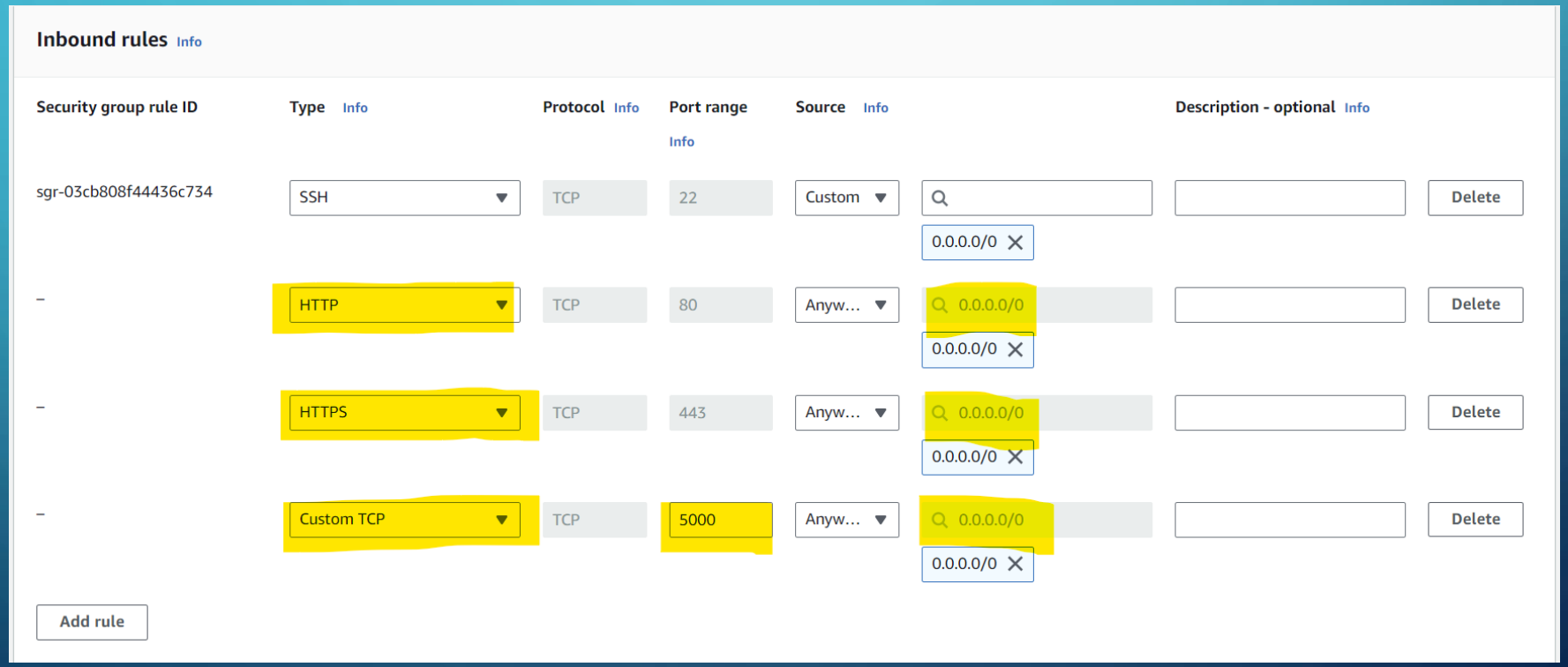
STEPS TO DEPLOY ON AWS EC2 (CONTD.)

- Once it has been launched, please select the Edit Inbound Rule button to edit the rules



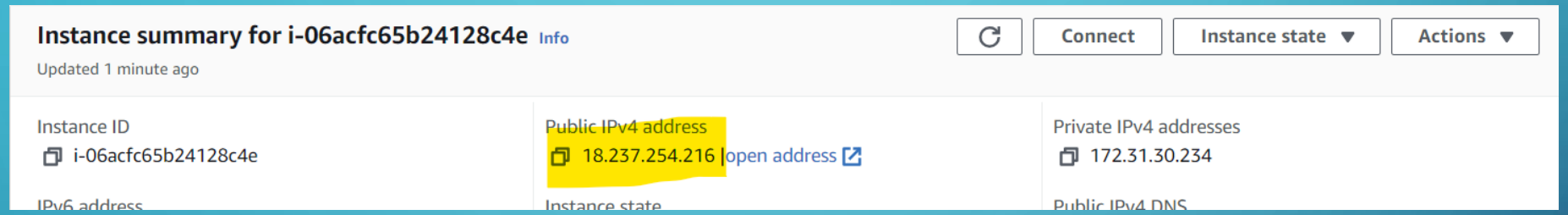
STEPS TO DEPLOY ON AWS EC2 (CONTD.)

- In the Edit mode add the following rules and Save the Ru

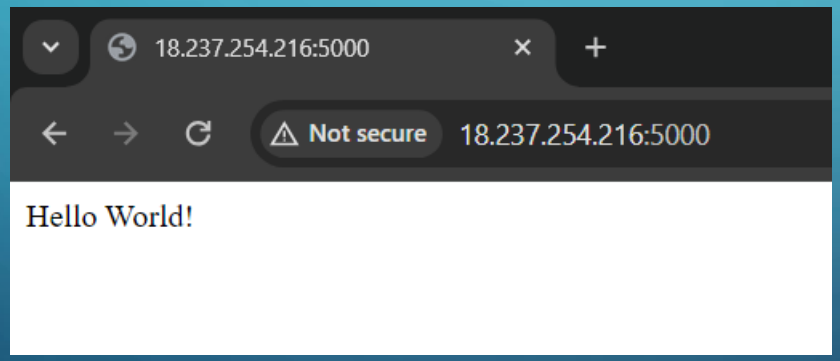


STEPS TO DEPLOY ON AWS EC2 (CONTD.)

- Find the public IP Address of your instance form the Instance summary page.



- Now you would be able to access the Deployed application on the browser



STEPS TO DEPLOY ON AWS EC2 (CONTD.)

- The code has been deployed in developer mode. Although it works, we must deploy the code in production with a WSGI server like Gunicorn.
- Install Gunicorn
  - Command: pip install gunicorn
- Start the Flask application by running the following commad:
  - Command: gunicorn --bind 0.0.0.0:5000 app:app