Deploying a Flask Application on AWS EC2

This guide covers the steps required to deploy a Flask application on an AWS EC2 instance, including configuration of the virtual environment, setting up Gunicorn as the application server, and ensuring the application is accessible via the public IP address.

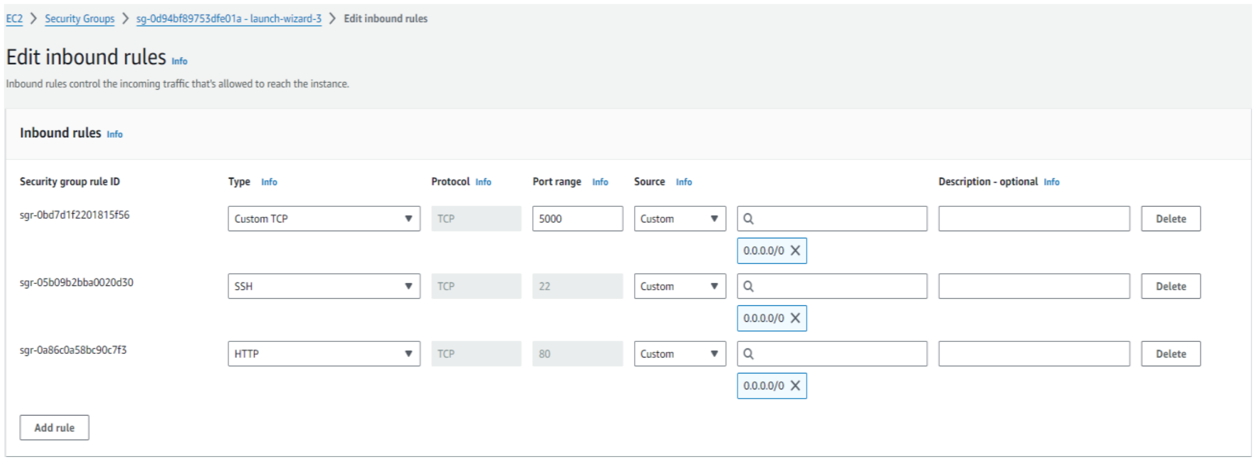
Prerequisites

- AWS account with access to the EC2 service
- Basic knowledge of the Flask framework
- An existing Flask application

Steps

1. Launch an EC2 Instance

1. Log in to the AWS Management Console.
2. Navigate to the EC2 Dashboard and click on "Launch Instance".
3. Select an Amazon Machine Image (AMI). The Amazon Linux 2 AMI is a good choice for most applications.
4. Choose an instance type. For development and testing, the `t2.micro` instance is sufficient.
5. Configure the instance details as needed. The default settings are usually fine.
6. Add storage if necessary. The default settings should suffice.
7. Configure the security group:
   - Add a rule to allow inbound SSH traffic (port 22) from your IP address.
   - Add a rule to allow inbound HTTP traffic (port 80) from anywhere.
   - Add a rule to allow inbound traffic on the port your Flask application will use (e.g., port 5000).
8. Launch the instance and download the private key file (`.pem`) for SSH access.


EC2 > Security Groups > sg-0d94bf89753dfe01a - launch-wizard-3 > Edit inbound rules

Edit inbound rules Info

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

Inbound rules Info

Security group rule ID	Type	Protocol	Port range	Source		Description - optional
sgr-0bd7d1f2201815f56	Custom TCP	TCP	5000	Custom	0.0.0.0/0	
sgr-05b09b2bba0020d30	SSH	TCP	22	Custom	0.0.0.0/0	
sgr-0a86c0a58bc90c7f3	HTTP	TCP	80	Custom	0.0.0.0/0	

2. Connect to Your EC2 Instance

1. Open a terminal on your local machine.
2. Change the permissions of your private key file:

```
chmod 400 path/to/your-key-file.pem
```

3. Connect to the EC2 instance using SSH:

```
ssh -i path/to/your-key-file.pem ec2-user@your-ec2-public-ip
```

4. Set Up the Environment on the EC2 Instance

1. Update the package list and install necessary packages:

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

2. Install `virtualenv`:

```
pip3 install virtualenv
```

3. Create a directory for your Flask application and navigate into it:

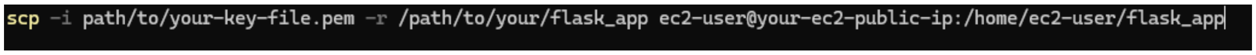
```
mkdir flask_app
cd flask_app
```

4. Set up a virtual environment:

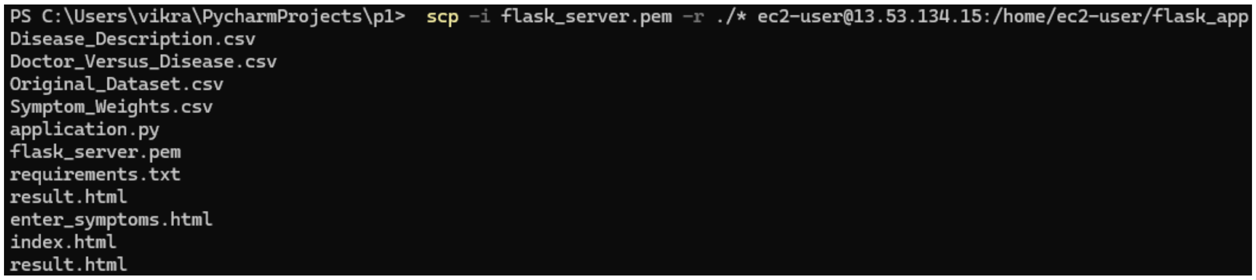
```
virtualenv venv
source venv/bin/activate
```

4. Transfer Your Flask Application to the EC2 Instance

1. On your local machine, use `scp` to transfer your Flask application files and `requirements.txt` to the EC2 instance

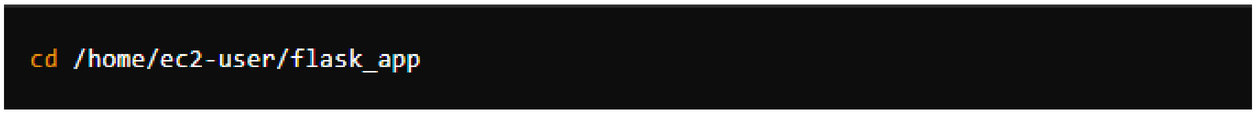


```
scp -i path/to/your-key-file.pem -r /path/to/your/flask_app ec2-user@your-ec2-public-ip:/home/ec2-user/flask_app
```



```
PS C:\Users\vikra\PycharmProjects\p1>  scp -i flask_server.pem -r ./* ec2-user@13.53.134.15:/home/ec2-user/flask_app
Disease_Description.csv
Doctor_Versus_Disease.csv
Original_Dataset.csv
Symptom_Weights.csv
application.py
flask_server.pem
requirements.txt
result.html
enter_symptoms.html
index.html
result.html
```

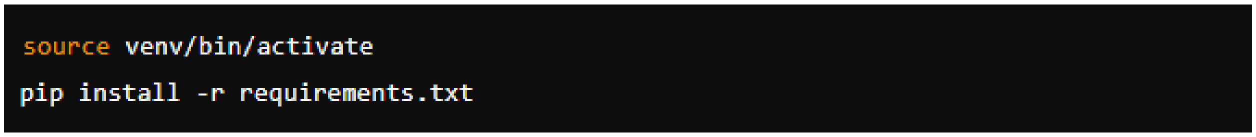
2. On the EC2 instance, navigate to the Flask application directory:



```
cd /home/ec2-user/flask_app
```

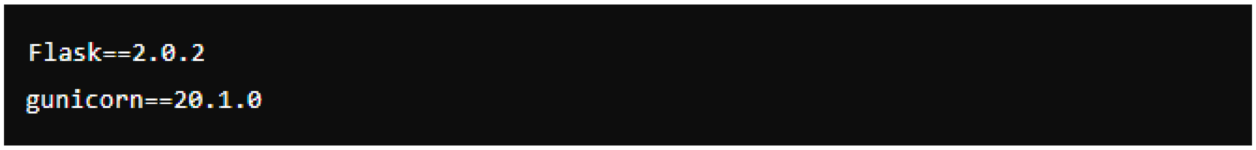
5. Install Dependencies

1. With the virtual environment activated, install the required packages using `requirements.txt`:



```
source venv/bin/activate
pip install -r requirements.txt
```

2. Ensure `requirements.txt` includes Flask and Gunicorn. It should look something like this:



```
Flask==2.0.2
gunicorn==20.1.0
```

6. Configure Gunicorn

1. Ensure your Flask application is correctly defined in `application.py` (or your main Flask application file). For example:

```
from flask import Flask

application = Flask(__name__)

@application.route('/')
def hello():
    return "Hello, World!"

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

2. Start Gunicorn, binding it to all network interfaces (0.0.0.0) and the desired port (e.g., 5000):

```
gunicorn --bind 0.0.0.0:5000 application:application
```

7. Configure Security Group

Ensure your EC2 instance's security group allows inbound traffic on port 5000. If not already configured, update the security group to allow this traffic.

8. Access Your Application

1. Open a web browser and navigate to:

```
http://your-ec2-public-ip:5000
```

9. Optional: Set Up Nginx as a Reverse Proxy

To make your Flask application accessible via the standard HTTP port (80), you can set up Nginx as a reverse proxy:

1. Install Nginx:

```
sudo amazon-linux-extras install nginx1 -y
sudo systemctl start nginx
sudo systemctl enable nginx
```

2. Configure Nginx to forward requests to Gunicorn. Edit the Nginx configuration file:

```
sudo nano /etc/nginx/nginx.conf
```

3. Save the file and restart Nginx:

```
sudo systemctl restart nginx
```

4. Now, you should be able to access your Flask application by navigating to your EC2 instance's public IP without specifying the port:

```
http://your-ec2-public-ip
```