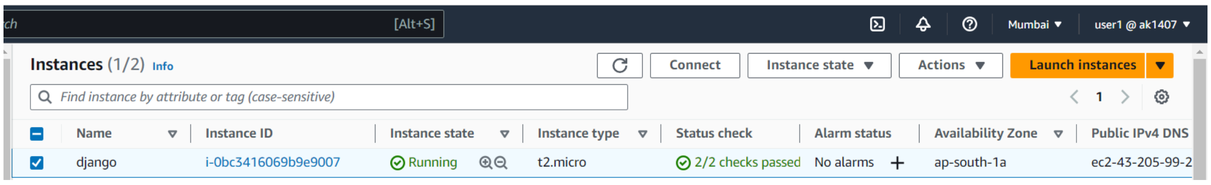
Deploying a Django application on EC2 instance

Step1 : Launch an EC2 instance. Name the server, select AMI as Ubuntu, select key-pair, network settings be default. Launch the instance.

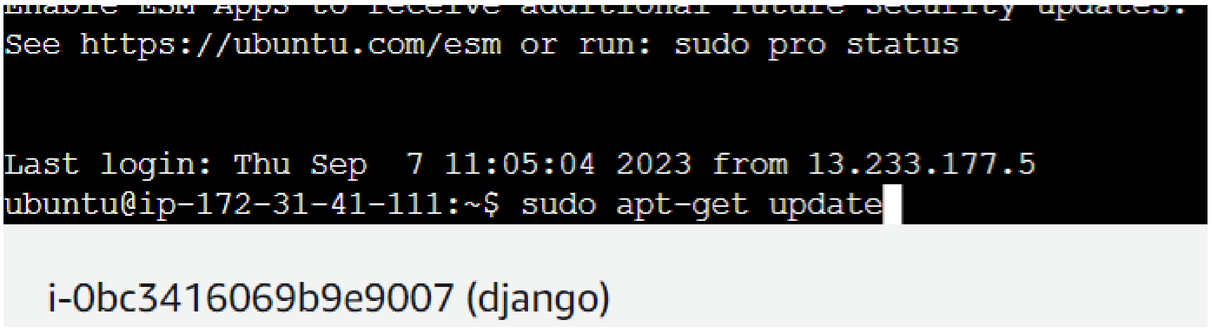


	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
✓	django	i-0bc3416069b9e9007	Running	t2.micro	2/2 checks passed	No alarms	ap-south-1a	ec2-43-205-99-2

Step 2 : Connect to shell.

Use Commands :

Sudo apt-get update



```
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


Last login: Thu Sep  7 11:05:04 2023 from 13.233.177.5
ubuntu@ip-172-31-41-111:~$ sudo apt-get update
```

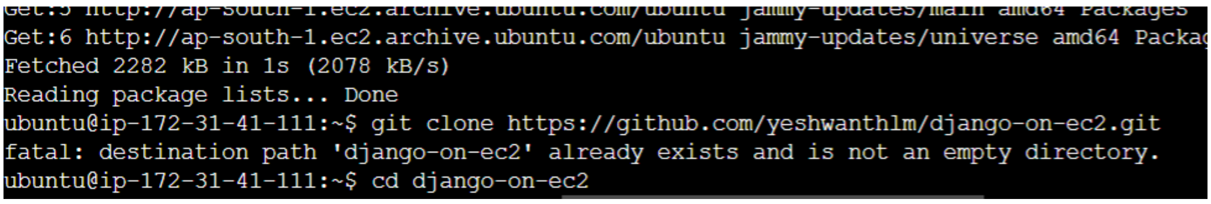
i-0bc3416069b9e9007 (django)

Create directory

git clone https://github.com/yeshwanthlm/django-on-ec2.git

change directory to Django

cd django-on-ec2



```
Get:5 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages
Get:6 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packag
Fetched 2282 kB in 1s (2078 kB/s)
Reading package lists... Done
ubuntu@ip-172-31-41-111:~$ git clone https://github.com/yeshwanthlm/django-on-ec2.git
fatal: destination path 'django-on-ec2' already exists and is not an empty directory.
ubuntu@ip-172-31-41-111:~$ cd django-on-ec2
```

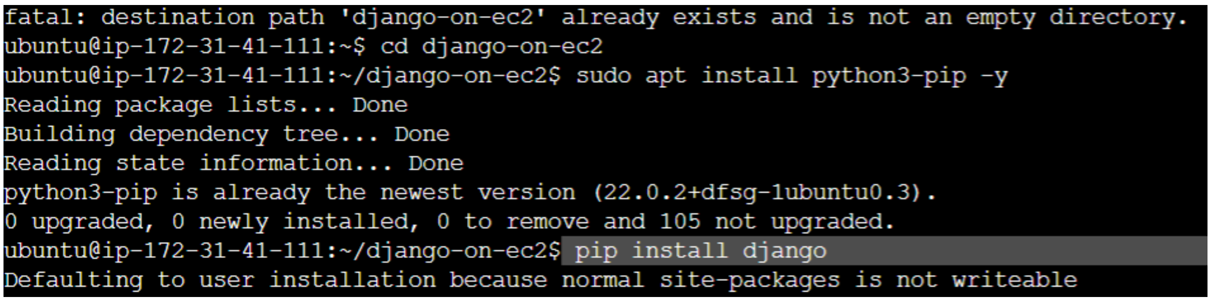
Step 3 :Installing Django on your computer

Pip install Django==4.2.5

Step 4 : Download Django using pip

Sudo app install python3-pip -y

Pip install Django

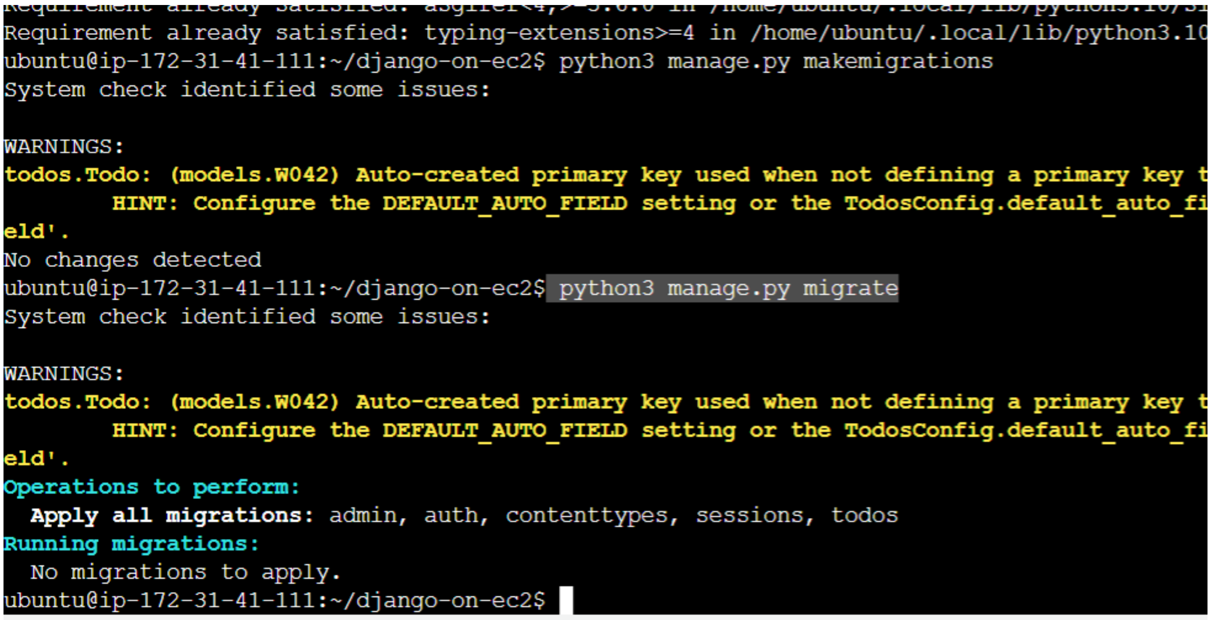

```
fatal: destination path 'django-on-ec2' already exists and is not an empty directory.
ubuntu@ip-172-31-41-111:~$ cd django-on-ec2
ubuntu@ip-172-31-41-111:~/django-on-ec2$ sudo apt install python3-pip -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
python3-pip is already the newest version (22.0.2+dfsg-1ubuntu0.3).
0 upgraded, 0 newly installed, 0 to remove and 105 not upgraded.
ubuntu@ip-172-31-41-111:~/django-on-ec2$ pip install django
Defaulting to user installation because normal site-packages is not writeable
```

Step 5 : Create the database migrations to run this APP

Python3 namage.py makemigrations

To apply this migrations use following command

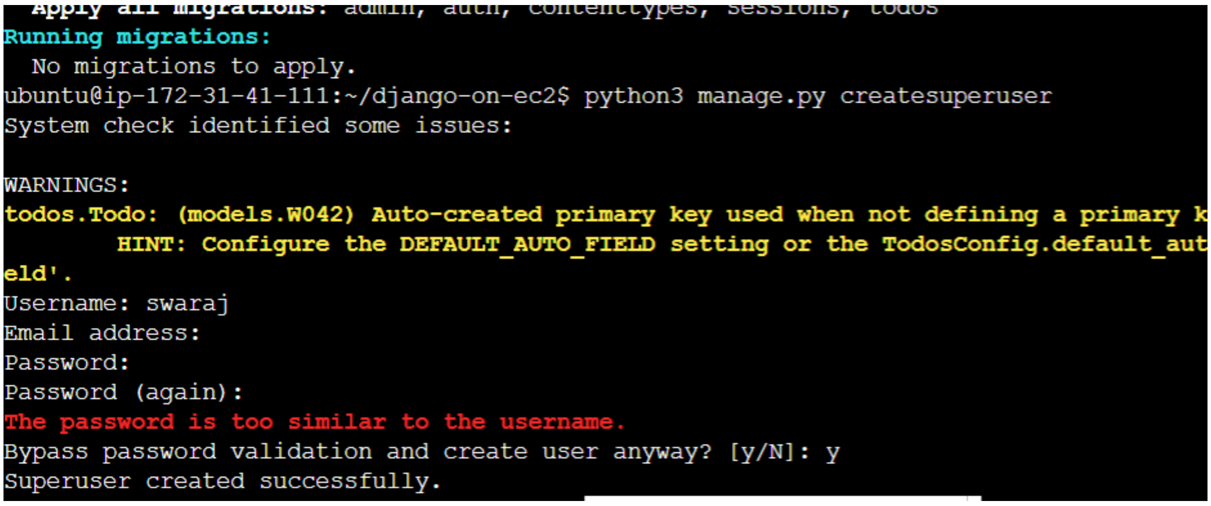
Python3 manage.py migrate


```
Requirement already satisfied: typing-extensions>=4 in /home/ubuntu/.local/lib/python3.10
ubuntu@ip-172-31-41-111:~/django-on-ec2$ python3 manage.py makemigrations
System check identified some issues:

WARNINGS:
todos.Todo: (models.W042) Auto-created primary key used when not defining a primary key t
        HINT: Configure the DEFAULT_AUTO_FIELD setting or the TodosConfig.default_auto_fi
eld'.
No changes detected
ubuntu@ip-172-31-41-111:~/django-on-ec2$ python3 manage.py migrate
System check identified some issues:

WARNINGS:
todos.Todo: (models.W042) Auto-created primary key used when not defining a primary key t
        HINT: Configure the DEFAULT_AUTO_FIELD setting or the TodosConfig.default_auto_fi
eld'.
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions, todos
Running migrations:
  No migrations to apply.
ubuntu@ip-172-31-41-111:~/django-on-ec2$
```

Step 6 : Create an admin user

Python3 manage.py createsuperuser

After this command enter a username, email address and password

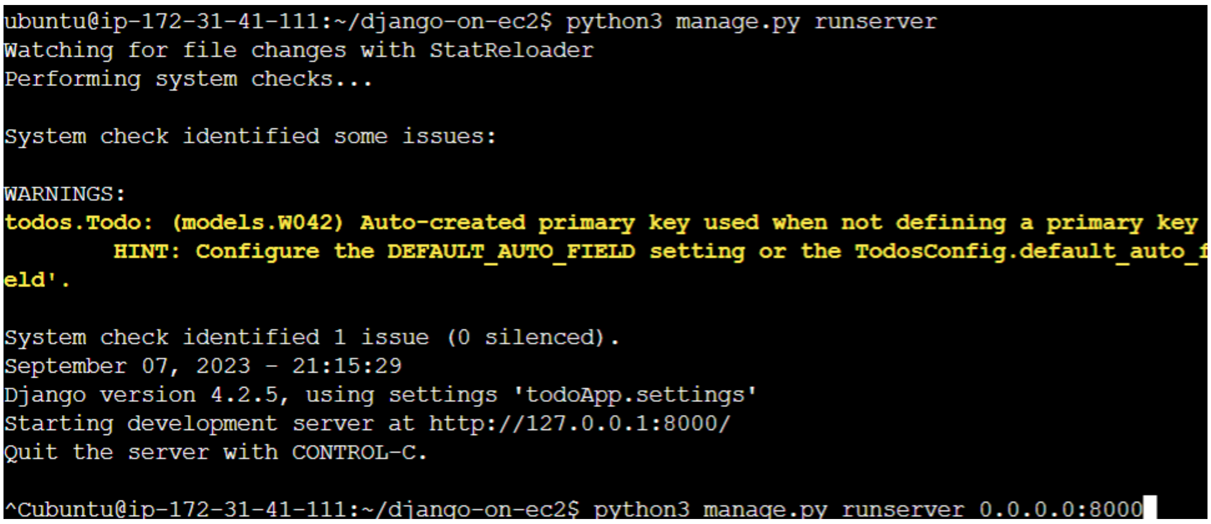


```
  Apply all migrations: admin, auth, contenttypes, sessions, todos
Running migrations:
  No migrations to apply.
ubuntu@ip-172-31-41-111:~/django-on-ec2$ python3 manage.py createsuperuser
System check identified some issues:

WARNINGS:
todos.Todo: (models.W042) Auto-created primary key used when not defining a primary k
        HINT: Configure the DEFAULT_AUTO_FIELD setting or the TodosConfig.default_aut
eld'.
Username: swaraj
Email address:
Password:
Password (again):
The password is too similar to the username.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
```

Step 7 : Starting the APP

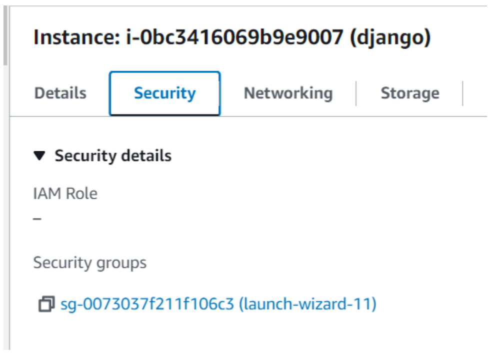
Python3 manage.py runserver

Python3 manag.py runserver 0.0.0.0:8000



```
ubuntu@ip-172-31-41-111:~/django-on-ec2$ python3 manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified some issues:

WARNINGS:
todos.Todo: (models.W042) Auto-created primary key used when not defining a primary key
        HINT: Configure the DEFAULT_AUTO_FIELD setting or the TodosConfig.default_auto_f
eld'.

System check identified 1 issue (0 silenced).
September 07, 2023 - 21:15:29
Django version 4.2.5, using settings 'todoApp.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.

^Cubuntu@ip-172-31-41-111:~/django-on-ec2$ python3 manage.py runserver 0.0.0.0:8000
```

Step 8 : Go to instance security, Click on security rules, Click on Edit Inbound rules,
Add type : Custom TCP, Port range to 8000, Select IPv4 Anywhere.



Instance: i-0bc3416069b9e9007 (django)

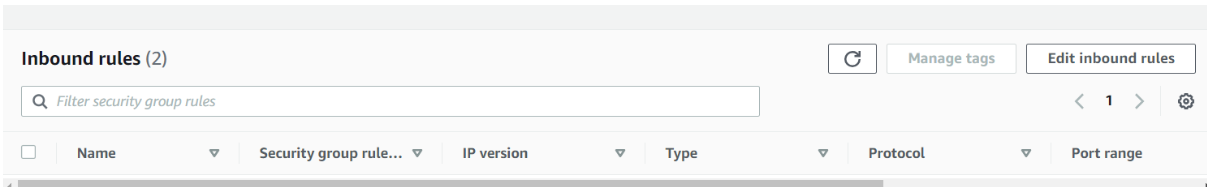
▼ Security details

IAM Role

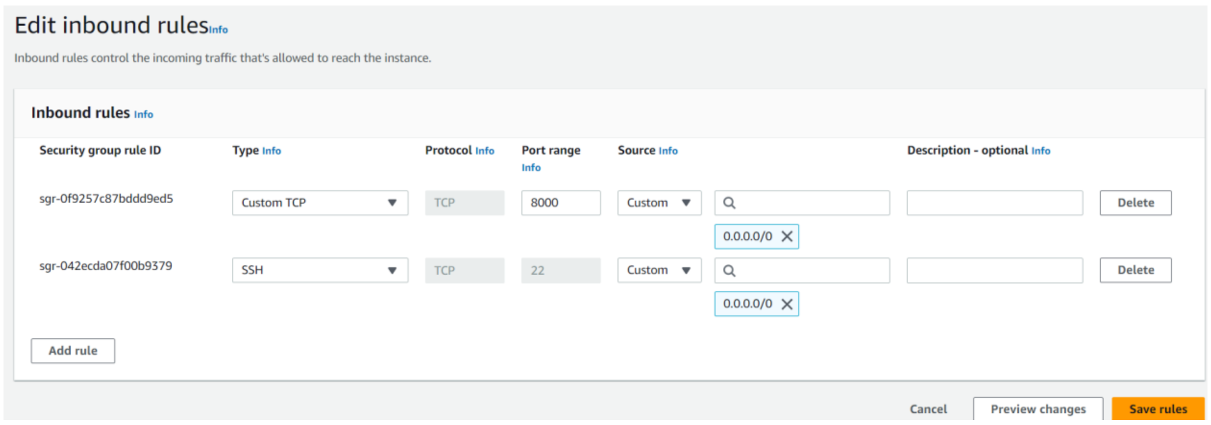
–

Security groups

sg-0073037f211f106c3 (launch-wizard-11)



	Name	Security group rule...	IP version	Type	Protocol	Port range

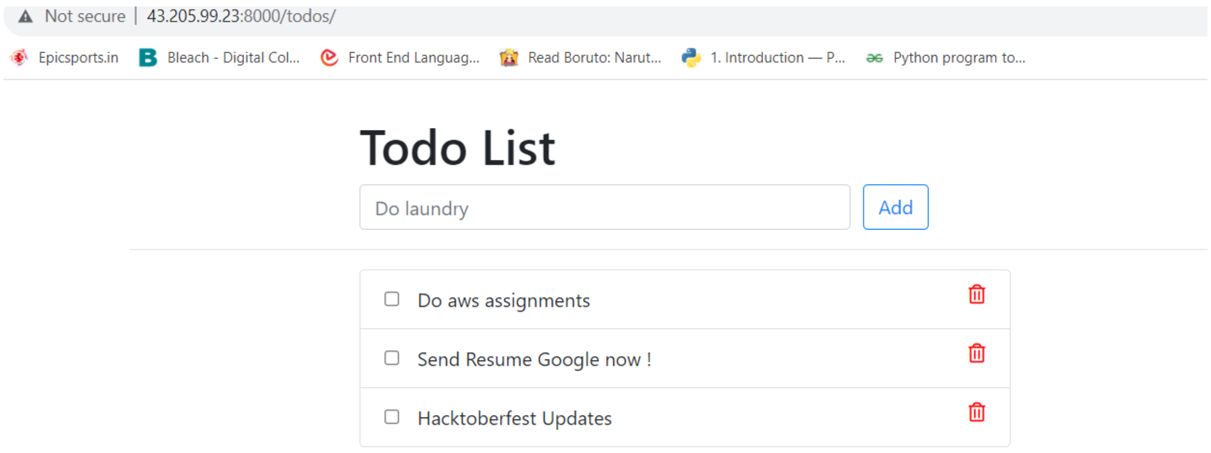

Edit inbound rules Info

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

Inbound rules Info

Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info		Description - optional Info
sgr-0f9257c87bddd9ed5	Custom TCP	TCP	8000	Custom	0.0.0.0/0	
sgr-042ecda07f00b9379	SSH	TCP	22	Custom	0.0.0.0/0	

Step 9 : Copy the IP address of the instance and Paste in any browser
<public ip : 8000>


Not secure | 43.205.99.23:8000/todos/

Todo List

Do laundry

- ☐ Do aws assignments
- ☐ Send Resume Google now !
- ☐ Hacktoberfest Updates

Deploying a Flask application on EC2 instance

Step1 : Launch an EC2 instance. Name the server, select AMI as Ubuntu, select key-pair, In network settings set http, http, ssh. Launch the instance.

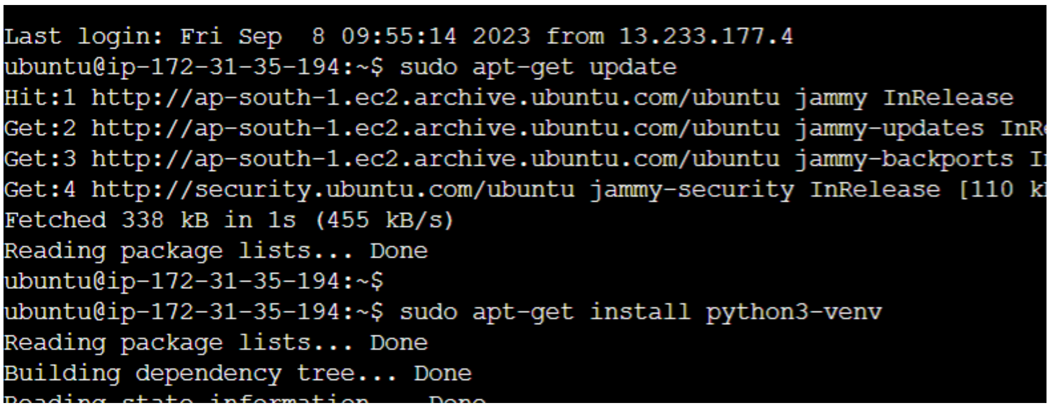
Step 2 : Connect the instance

Use these commands

Sudo apt-get update

Install Python Virtual environment

Sudo apt-get install python3-venv



```
Last login: Fri Sep  8 09:55:14 2023 from 13.233.177.4
ubuntu@ip-172-31-35-194:~$ sudo apt-get update
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRe
Get:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports I
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 k
Fetched 338 kB in 1s (455 kB/s)
Reading package lists... Done
ubuntu@ip-172-31-35-194:~$
ubuntu@ip-172-31-35-194:~$ sudo apt-get install python3-venv
Reading package lists... Done
Building dependency tree... Done
```

Step 3 : Create a new directory

Mkdir helloworld

Cd helloworld

Step 4 : Create an virtual environment

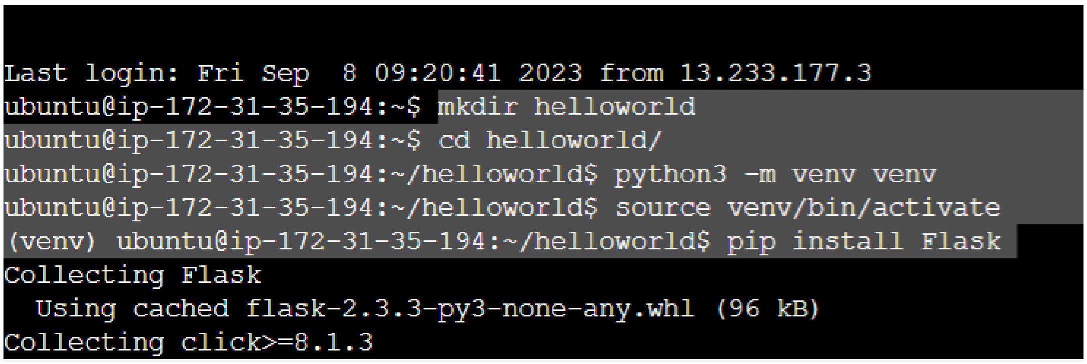
Python3 -m venv venv

Step 5 : Activate the virtual environment

Source venv/bin/activate

Step 6 : Install Flask

Pip install flask

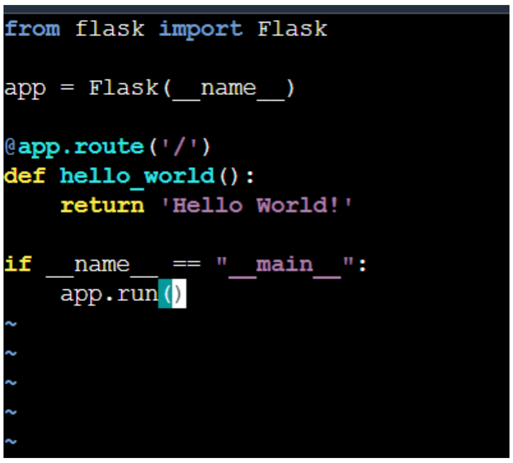


```
Last login: Fri Sep  8 09:20:41 2023 from 13.233.177.3
ubuntu@ip-172-31-35-194:~$ mkdir helloworld
ubuntu@ip-172-31-35-194:~$ cd helloworld/
ubuntu@ip-172-31-35-194:~/helloworld$ python3 -m venv venv
ubuntu@ip-172-31-35-194:~/helloworld$ source venv/bin/activate
(venv) ubuntu@ip-172-31-35-194:~/helloworld$ pip install Flask
Collecting Flask
  Using cached flask-2.3.3-py3-none-any.whl (96 kB)
Collecting click>=8.1.3
```

Step 7 : Create an Flask API

Sudo vi app.py

Enter the flask code in the file



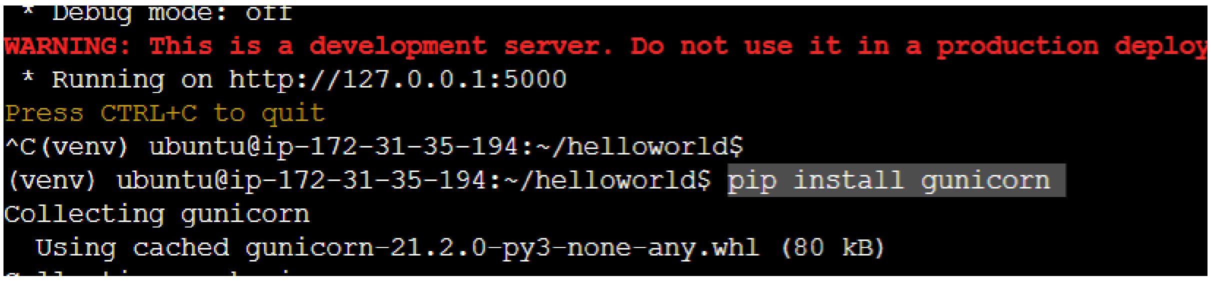
```
from flask import Flask

app = Flask(__name__)

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

if __name__ == "__main__":
    app.run()
~
~
~
~
~
```

Use following command to check whether the file is working or not

Python app.py

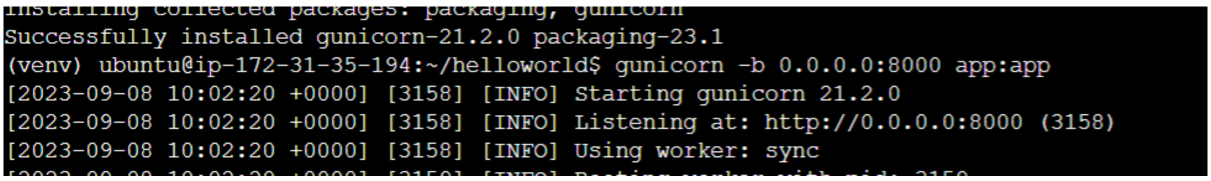
Step 8 : Installing Gunicorn

Pip install gunicorn



```
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deploy
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
^C(venv) ubuntu@ip-172-31-35-194:~/helloworld$
(venv) ubuntu@ip-172-31-35-194:~/helloworld$ pip install gunicorn
Collecting gunicorn
  Using cached gunicorn-21.2.0-py3-none-any.whl (80 kB)
```

gunicorn -b 0.0.0.0:8000 app:app

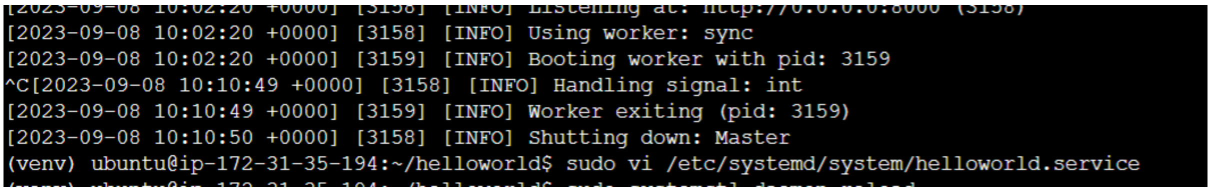


```
Installing collected packages: packaging, gunicorn
Successfully installed gunicorn-21.2.0 packaging-23.1
(venv) ubuntu@ip-172-31-35-194:~/helloworld$ gunicorn -b 0.0.0.0:8000 app:app
[2023-09-08 10:02:20 +0000] [3158] [INFO] Starting gunicorn 21.2.0
[2023-09-08 10:02:20 +0000] [3158] [INFO] Listening at: http://0.0.0.0:8000 (3158)
[2023-09-08 10:02:20 +0000] [3158] [INFO] Using worker: sync
[2023-09-08 10:02:20 +0000] [3158] [INFO] Booting worker with pid: 3159
```

Step 9 : We create a .service file in the /etc/systemd/system folder, and specify what would happen to gunicorn when the system reboots. We will be adding 3 parts to systemd Unit file Unit, Service, Install

Unit — This section is for description about the project and some dependencies Service — To specify user/group we want to run this service after. Also some information about the executables and the commands. Install — tells systemd at which moment during boot process this service should start. With that said, create an unit file in the /etc/systemd/system directory

Sudo vi etc/system/helloworld.service



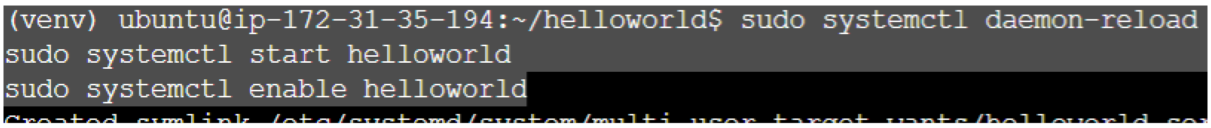
```
[2023-09-08 10:02:20 +0000] [3158] [INFO] Listening at: http://0.0.0.0:8000 (3158)
[2023-09-08 10:02:20 +0000] [3158] [INFO] Using worker: sync
[2023-09-08 10:02:20 +0000] [3159] [INFO] Booting worker with pid: 3159
^C[2023-09-08 10:10:49 +0000] [3158] [INFO] Handling signal: int
[2023-09-08 10:10:49 +0000] [3159] [INFO] Worker exiting (pid: 3159)
[2023-09-08 10:10:50 +0000] [3158] [INFO] Shutting down: Master
(venv) ubuntu@ip-172-31-35-194:~/helloworld$ sudo vi /etc/systemd/system/helloworld.service
(venv) ubuntu@ip-172-31-35-194:~/helloworld$ sudo systemctl daemon-reload
```

A file will be created insert the following in 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
```

Step 10 : Start the service

```
sudo systemctl daemon-reload
sudo systemctl start helloworld
sudo systemctl enable helloworld
```



```
(venv) ubuntu@ip-172-31-35-194:~/helloworld$ sudo systemctl daemon-reload
sudo systemctl start helloworld
sudo systemctl enable helloworld
Created symlink /etc/systemd/system/multi-user.target.wants/helloworld.se
```

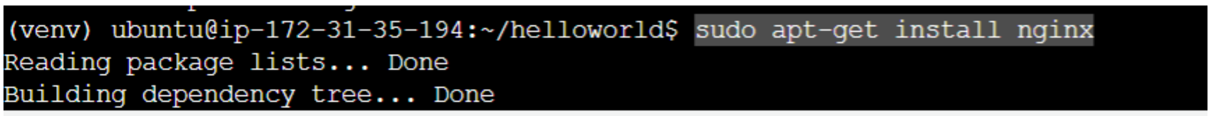
Step 11 : Check if the app is running with

```
curl localhost:8000
```

Step 12 : Install Nginx

Run Nginx Webserver to accept and route request to Gunicorn Finally, we set up Nginx as a reverse-proxy to accept the requests from the user and route it to gunicorn.

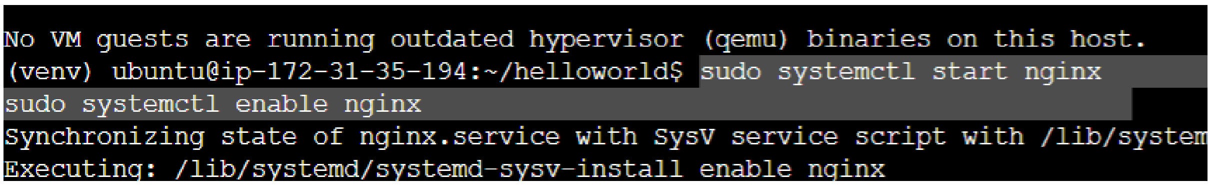
Sudo apt-get nginx



```
(venv) ubuntu@ip-172-31-35-194:~/helloworld$ sudo apt-get install nginx
Reading package lists... Done
Building dependency tree... Done
```

Sudo systemctl start nginx

Sudo systemctl enable nginx

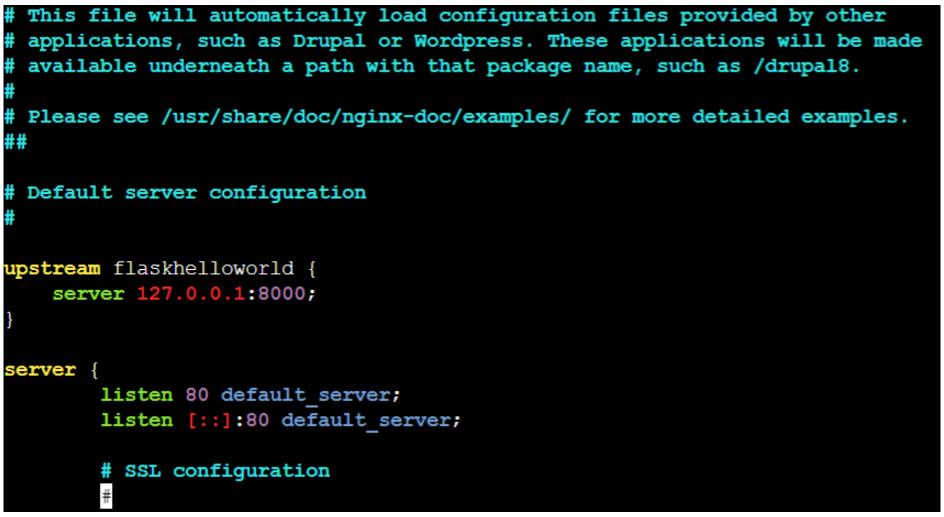


```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
(venv) ubuntu@ip-172-31-35-194:~/helloworld$ sudo systemctl start nginx
sudo systemctl enable nginx
Synchronizing state of nginx.service with SysV service script with /lib/system
Executing: /lib/systemd/systemd-sysv-install enable nginx
```

Step 13 :

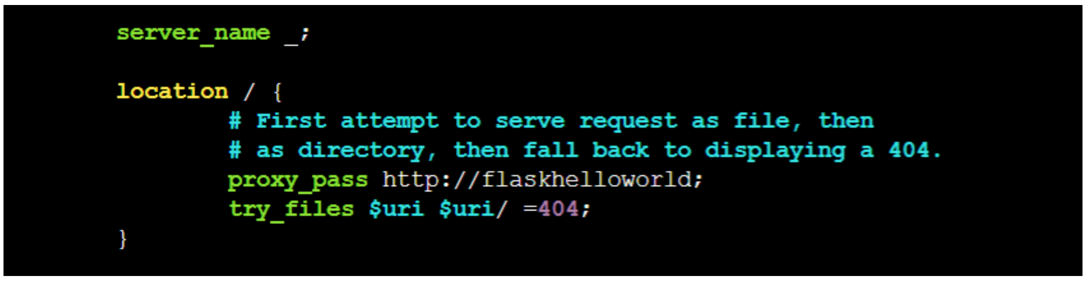
sudo vi /etc/nginx/sites-available/default

Add upstream part in the code before the server



```
# 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 flaskhelloworld {
    server 127.0.0.1:8000;
}

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

        # SSL configuration
        #
```

Add proxy_pass http://flaskhelloworld; in the location above the try_files.



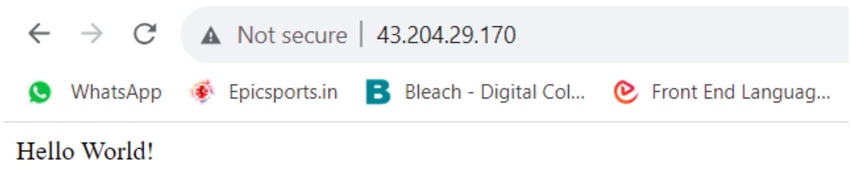
```
        server_name _;

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

Edit the file and enter the following code

Step 14 ; Restart Nginx

Sudo systemctl restart nginx

Step 15 : Enter your IP in any browser. Output will be displayed



Hello World!