**How to run a Django server without a port?**

To get python web app output we need to run our Django lightweight server.so we need a port number.

But in aws with linux instances we are able to run sour server without a port number

Step1:

Start your ec2 linux instances and connect through putty.

Install the Packages from the Ubuntu Repositories

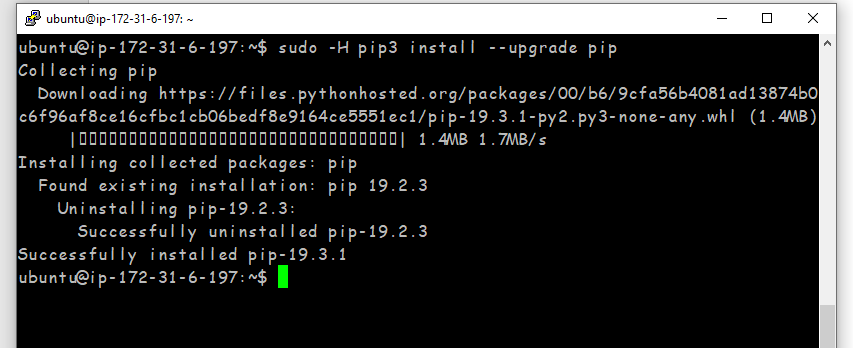
# sudo apt-get update

# sudo apt-get install python3-pip python3-dev libpq-dev nginx

Step2;

 upgrade pip and install the package by typing

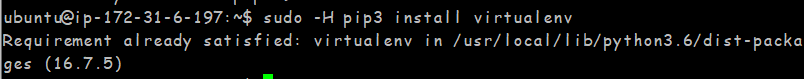
# sudo -H pip3 install --upgrade pip



Install virtual environment packages for python 3

# sudo -H pip3 install virtualenv

If u have already installed it shows like this



Step 3:

Make a directory using mkdir command. I have already this directory as src.(this directory is the whole project folder like it contain manage.py file etc)



Step4:

change the directory to src.

# cd src

Create a virtual env inside src

# Virtualenv artemus\_env

Activate environment

# source artemus\_env/bin/activate



Step-5:

Inside environment only python is applicable not python3

Create and configure a new Django project / exciting project

Install Django framework and gunicorn server

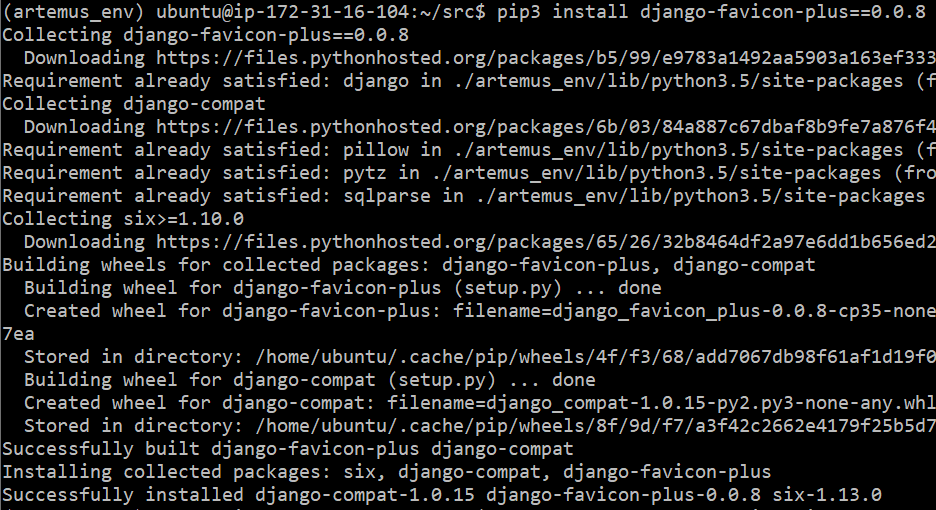
# pip3 install django gunicorn

Install pillow for image field

# pip3 install pillow

Install django-favicon-plus

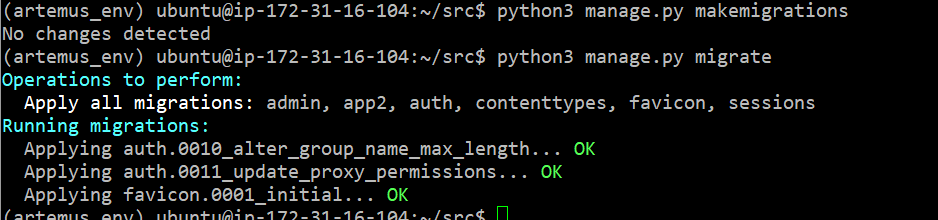
# pip3 install django-favicon-plus==0.0.8



Migrate the initial database schema to our sqllite3 database using below script

# python3 manage.py makemigrations

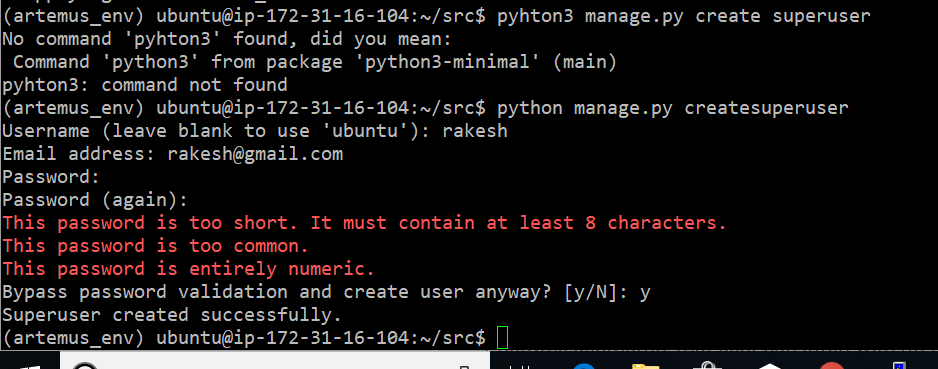
# python3 manage.py migrate



Step6:

Create an admistrative user for the project

# pyhton manage.py create superuser

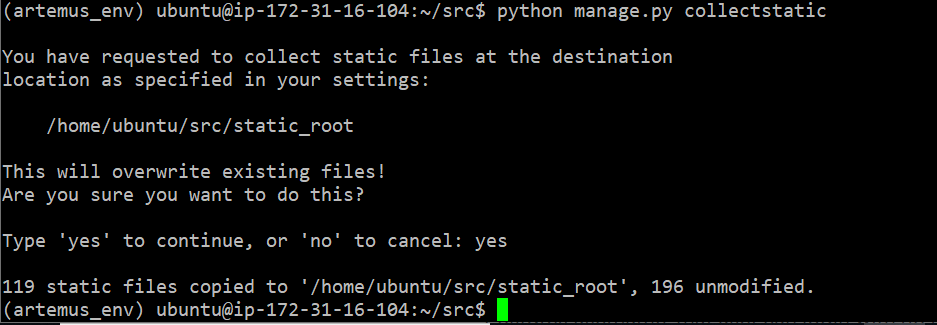


Username=rakesh

Password=12345

We can collect all of the static content into the directory location

# python manage.py collectstatic

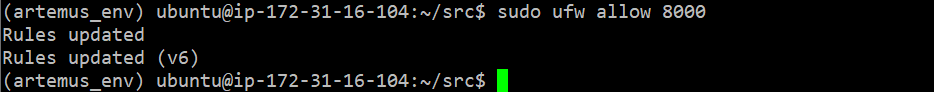


Step7:

You should have a UFW firewall protecting your server. In order to test the development server, we’ll have to allow access to the port we’ll be using.

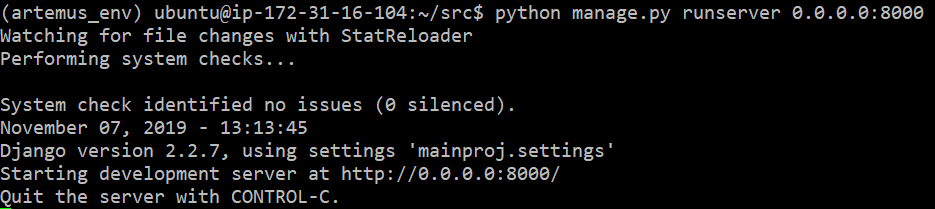
Create an exception for port 8000 by typing:

# sudo ufw allow 8000



You can test your project by starting up the Django development server with this command:

# python manage.py runserver 0.0.0.0:8000

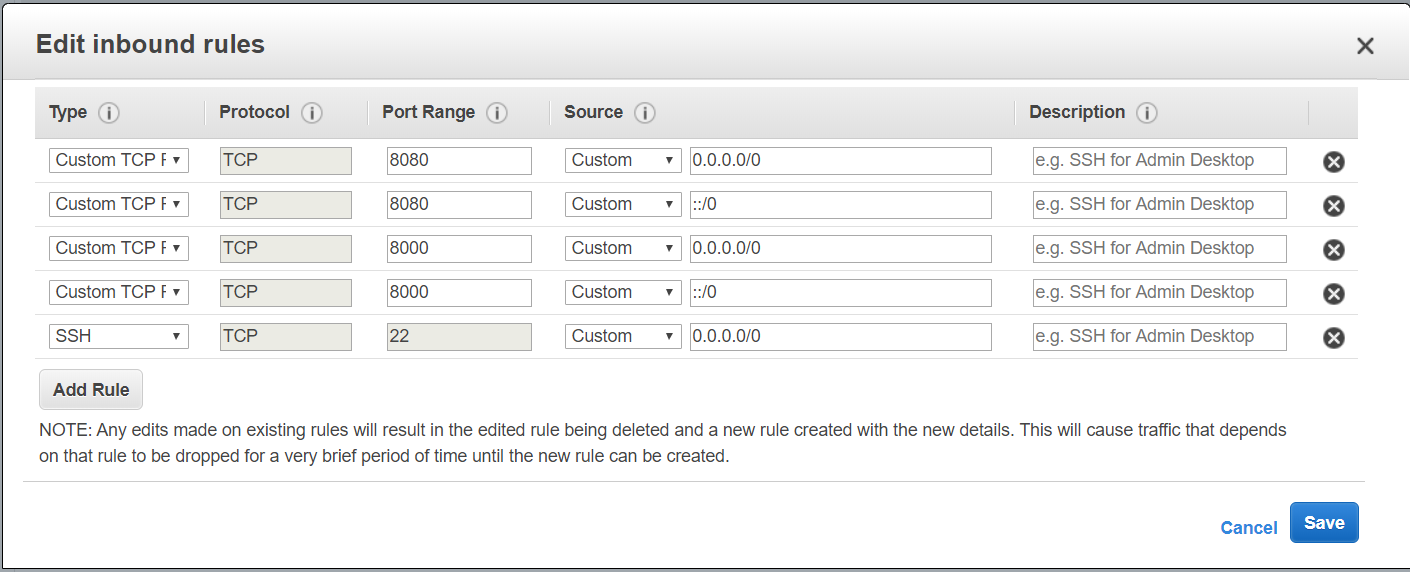


Step8:

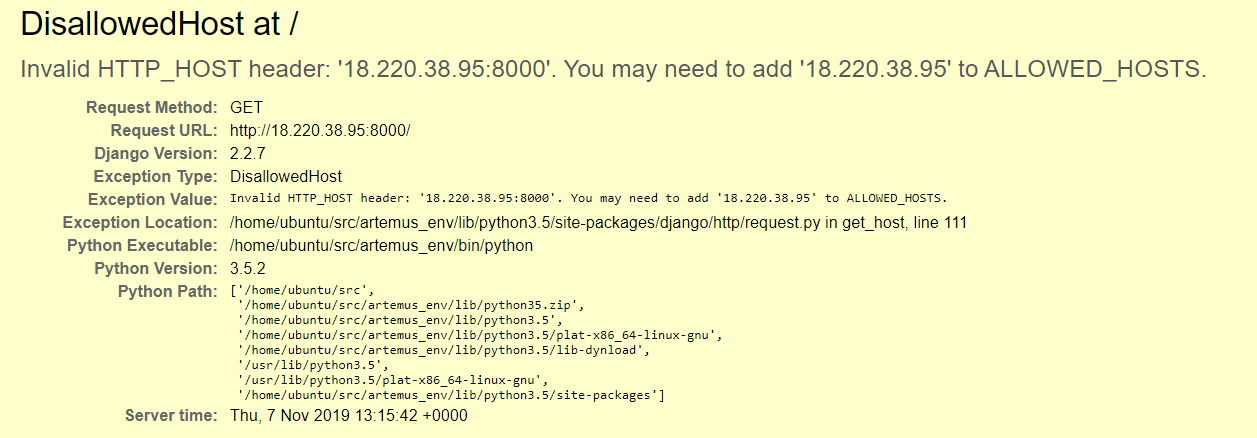
In your web browser, visit your server’s domain name or IP address followed by: 8000:

Search ec2-18-220-38-95.us-east-2.compute.amazonaws.com:8000 in browser

Make sure in your inbound of ec2 instance have assign port no 8000 if not then give that port no as shown in below figure



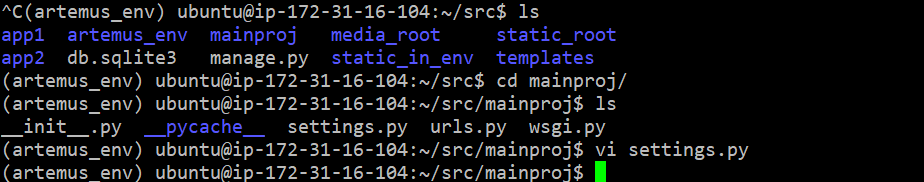
After search u get below figure bcz of not allocating your ip inside your project allowed host inside settings.py



Settings.py is available in src/mainproj

Then open the settings.py using vi command

# vi settings.py

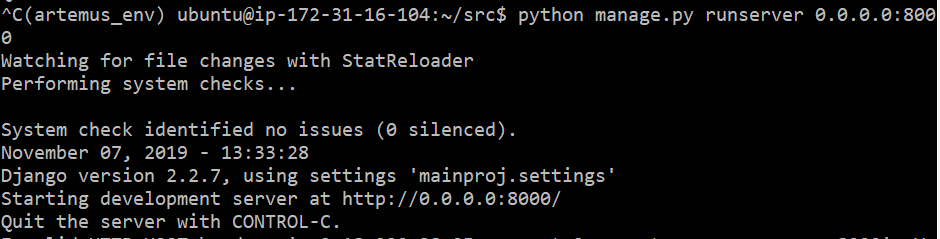


Assign your ip in ALLOWED\_HOSTS inside setting.py of project



Save with file :wq

Then go to src and runserver



Then check with your browser using “public dns:port”

# <http://ec2-18-220-38-95.us-east-2.compute.amazonaws.com:8000/>

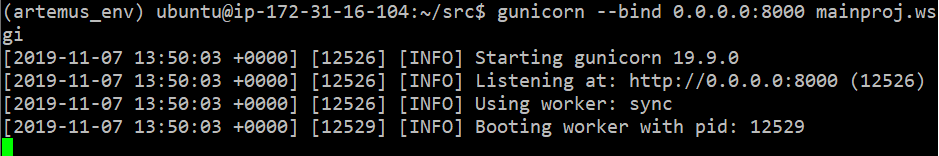
Step9:

Congratulations your project works perfectly but using port number

We are using gunicorn and nginix to run server without port number

Bind the port no with gunicorn

# gunicorn --bind 0.0.0.0:8000 mainproj.wsgi



We passed Gunicorn a module by specifying the relative directory path to Django’s wsgi.py file, which is the entry point to our application, using Python’s module syntax. Inside of this file, a function called application is defined, which is used to communicate with the application

Deactivate your virtual environment by type below code

# deactivate



Create a gunicorn systemd service file

# sudo vim /etc/systemd/system/gunicorn.service

Write below code in this file save and close it by “esc:wq”

[Unit]

Description=gunicorn daemon

After=network.target

[Service]

User=ubuntu

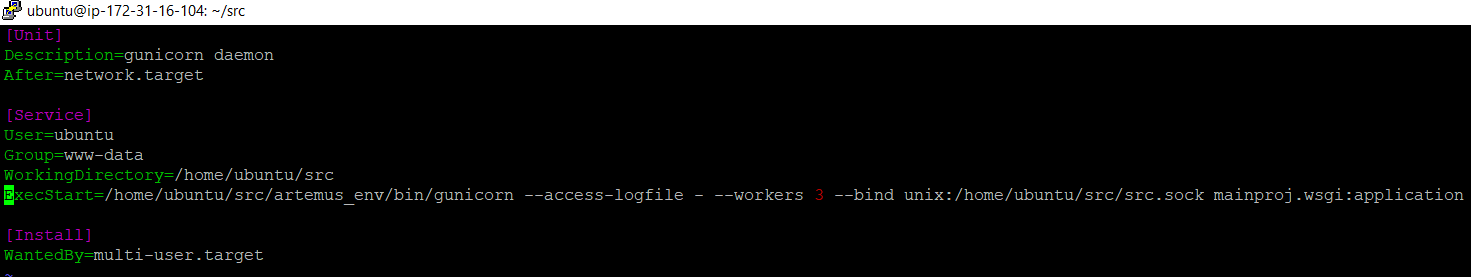
Group=www-data

WorkingDirectory=/home/ubuntu/src

ExecStart=/home/ubuntu/src/artemus\_env/bin/gunicorn --access-logfile - --workers 3 --bind unix:/home/ubuntu/src/src.sock mainproj.wsgi:application

[Install]

WantedBy=multi-user.target



Step10:

We can now start the Gunicorn service we created and enable it so that it starts at boot:

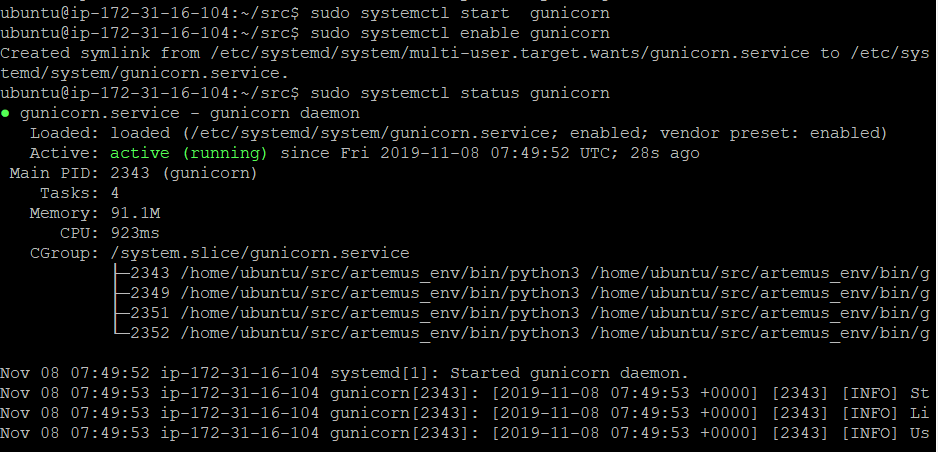
# sudo systemctl start gunicorn

# sudo systemctl enable gunicorn

## Check for the Gunicorn Socket File

Check the status of the process to find out whether it was able to start:

# sudo systemctl status gunicorn



Next, check for the existence of the src.sock file within your project directory:

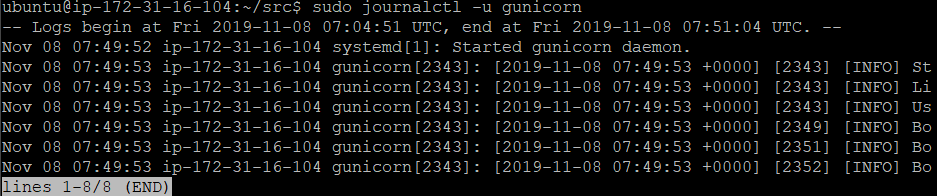
# ls



Step11:

Check the Gunicorn process logs by typing:

# sudo journalctl -u gunicorn



If you make changes to the /etc/systemd/system/gunicorn.service file, reload the daemon to reread the service definition and restart the Gunicorn process by typing:

# sudo systemctl daemon-reload

# sudo systemctl restart gunicorn

Make sure you troubleshoot any of the above issues before continuing.



## Configure Nginx to Proxy Pass to Gunicorn

Now that Gunicorn is set up, we need to configure Nginx to pass traffic to the process.

Start by creating and opening a new server block in Nginx’s sites-available directory:

Inside, open up a new server block. We will start by specifying that this block should listen on the normal port 80 and that it should respond to our server’s domain name or IP address:

# sudo vim /etc/nginx/sites-available/src

Write below code in this file

server {

listen 80;

server\_name 3.133.93.187;

location = /favicon.ico { access\_log off;log\_not\_found off; }

location /static\_in\_env/ {

root /home/ubuntu/src;

}

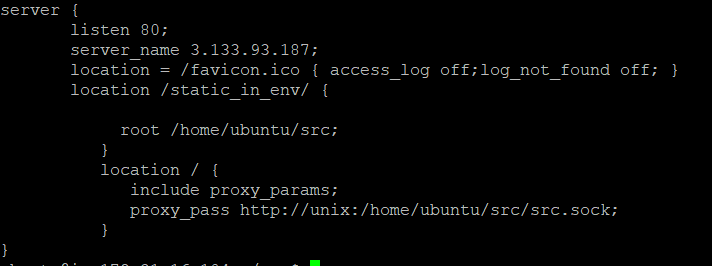
location / {

include proxy\_params;

proxy\_pass http://unix:/home/ubuntu/src/src.sock;

}

}



Step12:

Save and close the file when you are finished. Now, we can enable the file by linking it to the sites-enabled directory:

# sudo ln -s /etc/nginx/sites-available/src /etc/nginx/sites-enabled

Test your Nginx configuration for syntax errors by typing:

# sudo nginx -t

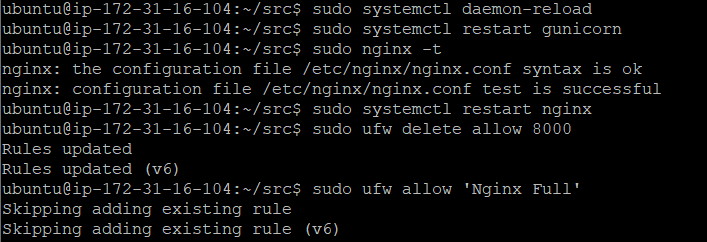
If no errors are reported, go ahead and restart Nginx by typing:

# sudo systemctl restart nginx

Finally, we need to open up our firewall to normal traffic on port 80. Since we no longer need access to the development server, we can remove the rule to open port 8000 as well:

# sudo ufw delete allow 8000

# sudo ufw allow 'Nginx Full'



You should now be able to go to your server’s domain or IP address or aws dns to view your application.

If u have any doubt follow the link in description.

<https://www.digitalocean.com/community/tutorials/how-to-set-up-django-with-postgres-nginx-and-gunicorn-on-ubuntu-16-04>