Tutorial Django Digital Ocean

Artigo original em:

<https://www.digitalocean.com/community/tutorials/how-to-set-up-django-with-postgres-nginx-and-gunicorn-on-ubuntu-20-04>

0 - Pre-requisitos:

<https://www.digitalocean.com/community/tutorials/initial-server-setup-with-ubuntu-20-04>

ubuntu server com privilégios sudo

1 – Instalando Pacotes:

Python 3

* sudo apt update
* sudo apt install python3-pip python3-dev libpq-dev postgresql postgresql-contrib nginx curl

2 – Criando banco de dados e usuário PostgreSQL

sudo -u postgres psql

2.1 - Primeiro, crie o banco de dados para seu projeto:

* postgres=# CREATE DATABASE myproject;

2.2 – Crie o usuario

* postgres=# CREATE USER myprojectuser WITH PASSWORD 'password';

adicionando mais algumas configurações no BD

* postgres=# ALTER ROLE myprojectuser SET client\_encoding TO 'utf8';
* postgres=# ALTER ROLE myprojectuser SET default\_transaction\_isolation TO 'read committed';
* postgres=# ALTER ROLE myprojectuser SET timezone TO 'UTC';
* postgres=# GRANT ALL PRIVILEGES ON DATABASE myproject TO myprojectuser;

2.3 - Para sair:

* postgres=# \q

3 – Criando o ambiente virtual:

* sudo -H pip3 install --upgrade pip
* sudo -H pip3 install virtualenv
* mkdir ~/myprojectdir
* cd ~/myprojectdir
* virtualenv myprojectenv
* source myprojectenv/bin/activate

Instalando Django, Gunicorn, and the psycopg2 PostgreSQL no ambiente virtual

pip install django gunicorn psycopg2-binary

4 – Criando e configurando o projeto Django

* (myprojectenv) $ django-admin.py startproject myproject ~/myprojectdir

Ajuste o arquivo settings.py:

~/myprojectdir/myproject/settings.py

4.1 Hosts permitidos

. . .

# The simplest case: just add the domain name(s) and IP addresses of your Django server

# ALLOWED\_HOSTS = [ 'example.com', '203.0.113.5']

# To respond to 'example.com' and any subdomains, start the domain with a dot

# ALLOWED\_HOSTS = ['.example.com', '203.0.113.5']

ALLOWED\_HOSTS = ['your\_server\_domain\_or\_IP', 'second\_domain\_or\_IP', . . ., 'localhost']

4.2 Banco de Dados

. . .

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.postgresql\_psycopg2',

'NAME': 'myproject',

'USER': 'myprojectuser',

'PASSWORD': 'password',

'HOST': 'localhost',

'PORT': '',

}

}

. . .

4.3 Arquivos estáticos

. . .

STATIC\_URL = '/static/'

STATIC\_ROOT = os.path.join(BASE\_DIR, 'static/')

5 – Configurando um pouco mais o projeto:

No ambiente virtual:

* ~/myprojectdir/manage.py makemigrations
* ~/myprojectdir/manage.py migrate
* ~/myprojectdir/manage.py createsuperuser
* ~/myprojectdir/manage.py collectstatic

5.1 Criando excessao para porta 8000do firewall:

sudo ufw allow 8000

Teste o projeto com os comandos abaixo

~/myprojectdir/manage.py runserver 0.0.0.0:8000

http://server\_domain\_or\_IP:8000

Finalize o projeto com CTRL+C

6 – Teste o GUINICORN para o projeto

* cd ~/myprojectdir
* gunicorn --bind 0.0.0.0:8000 myproject.wsgi

É pra rodar o servidor da mesma forma que feito antes

Caso tenha erro entrar no link abaixo:

https://www.digitalocean.com/community/tutorials/how-to-set-up-uwsgi-and-nginx-to-serve-python-apps-on-ubuntu-14-04#definitions-and-concepts

Finalizamos a criação do projeto, podemos desativar o ambiente virtual

* deactivate

7 – Criando o systemd Socket e arquivos de serviços para o GUINICORN

7.1 Criando e abrindo o arquido para o systemd socket com privilegio sudo:

* sudo nano /etc/systemd/system/gunicorn.socket

com o conteúdo abaixo:

/etc/systemd/system/gunicorn.socket

[Unit]

Description=gunicorn socket

[Socket]

ListenStream=/run/gunicorn.sock

[Install]

WantedBy=sockets.target

Salve e saia do arquivo

Crie o arquivo de serviço systemd com privilégio sudo:

Com o mesmo nome, só alterando a extensão

* sudo nano /etc/systemd/system/gunicorn.service

/etc/systemd/system/gunicorn.service

[Unit]

Description=gunicorn daemon

Requires=gunicorn.socket

After=network.target

[Service]

User=sammy

Group=www-data

WorkingDirectory=/home/sammy/myprojectdir

ExecStart=/home/sammy/myprojectdir/myprojectenv/bin/gunicorn \

--access-logfile - \

--workers 3 \

--bind unix:/run/gunicorn.sock \

myproject.wsgi:application

[Install]

WantedBy=multi-user.target

Salve e saia, agora voce esta pronto para checar o **Gunicorn Socket File**

* sudo systemctl status gunicorn.socket

Output

● gunicorn.socket - gunicorn socket

Loaded: loaded (/etc/systemd/system/gunicorn.socket; enabled; vendor prese>

Active: active (listening) since Fri 2020-06-26 17:53:10 UTC; 14s ago

Triggers: ● gunicorn.service

Listen: /run/gunicorn.sock (Stream)

Tasks: 0 (limit: 1137)

Memory: 0B

CGroup: /system.slice/gunicorn.socket

Check gunicorn sock com o comando /run

$ file /run/gunicorn.sock

Output

/run/gunicorn.sock: socket

Se der algum erro cheque o log com o commando abaixo:

sudo journalctl -u gunicorn.socket

Teste a ativaçao do socket

sudo systemctl status gunicorn

Output

● gunicorn.service - gunicorn daemon

Loaded: loaded (/etc/systemd/system/gunicorn.service; disabled; vendor preset: enabled)

Active: inactive (dead)

curl --unix-socket /run/gunicorn.sock localhost

sudo systemctl status gunicorn

Output

● gunicorn.service - gunicorn daemon

Loaded: loaded (/etc/systemd/system/gunicorn.service; disabled; vendor preset: enabled)

Active: active (running) since Fri 2020-06-26 18:52:21 UTC; 2s ago

TriggeredBy: ● gunicorn.socket

Main PID: 22914 (gunicorn)

Tasks: 4 (limit: 1137)

Memory: 89.1M

CGroup: /system.slice/gunicorn.service

├─22914 /home/sammy/myprojectdir/myprojectenv/bin/python /home/sammy/myprojectdir/myprojectenv/bin/gunicorn --access-logfile - --workers 3 --bind unix:/run/gunico>

├─22927 /home/sammy/myprojectdir/myprojectenv/bin/python /home/sammy/myprojectdir/myprojectenv/bin/gunicorn --access-logfile - --workers 3 --bind unix:/run/gunico>

├─22928 /home/sammy/myprojectdir/myprojectenv/bin/python /home/sammy/myprojectdir/myprojectenv/bin/gunicorn --access-logfile - --workers 3 --bind unix:/run/gunico>

└─22929 /home/sammy/myprojectdir/myprojectenv/bin/python /home/sammy/myprojectdir/myprojectenv/bin/gunicorn --access-logfile - --workers 3 --bind unix:/run/gunico>

Jun 26 18:52:21 django-tutorial systemd[1]: Started gunicorn daemon.

Jun 26 18:52:21 django-tutorial gunicorn[22914]: [2020-06-26 18:52:21 +0000] [22914] [INFO] Starting gunicorn 20.0.4

Jun 26 18:52:21 django-tutorial gunicorn[22914]: [2020-06-26 18:52:21 +0000] [22914] [INFO] Listening at: unix:/run/gunicorn.sock (22914)

Jun 26 18:52:21 django-tutorial gunicorn[22914]: [2020-06-26 18:52:21 +0000] [22914] [INFO] Using worker: sync

Jun 26 18:52:21 django-tutorial gunicorn[22927]: [2020-06-26 18:52:21 +0000] [22927] [INFO] Booting worker with pid: 22927

Jun 26 18:52:21 django-tutorial gunicorn[22928]: [2020-06-26 18:52:21 +0000] [22928] [INFO] Booting worker with pid: 22928

Jun 26 18:52:21 django-tutorial gunicorn[22929]: [2020-06-26 18:52:21 +0000] [22929] [INFO] Booting worker with pid: 22929

Se algum problema ocorrer, cheque os logs

* sudo journalctl -u gunicorn

Check your /etc/systemd/system/gunicorn.service file for problems. 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

Garanta que concluiu o passo anterior até seguir em frente:

8 – Configure o NGINX para passar pelo proxy do Gunicorn

* sudo nano /etc/nginx/sites-available/myproject

Próximo passo vamos remover erros em procurar o favicon.ico e vamos identificar o diretório de arquivos estáticos, vamos criar também um bloco para os outros requests:

/etc/nginx/sites-available/myproject

server {

listen 80;

server\_name server\_domain\_or\_IP;

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

location /static/ {

root /home/sammy/myprojectdir;

}

location / {

include proxy\_params;

proxy\_pass http://unix:/run/gunicorn.sock;

}

}

Salve e feche o arquivo.

Habilitamos as configurações com:

sudo ln -s /etc/nginx/sites-available/myproject /etc/nginx/sites-enabled

Teste o Nginx

sudo nginx -t

Se não tiver erros reinicie com o comando abaixo:

sudo systemctl restart nginx

Finalmente precisamos abrir o firewall na porta 80.

Quando não precisamos mais acessar o servidor de desenvolvimento nós podemos remover a regra para a porta 8000 também:

* sudo ufw delete allow 8000
* sudo ufw allow 'Nginx Full'