Installeren en configuren Django op de Raspberry PI server

Je kunt het gemaakte project gaan installeren op de Raspberry Pi of bij een VPS webhosting. Voordat je kan beginnen, moet je eerst inloggen op via een Putty of een terminal-venster via het protocol SSH.

```
$ ssh ubuntu@192.168.xxx.xxx  gebruik hierbij je eigen interne ip adress.
```

Je werkt nu verder in de terminal verbinding.

Om de server geschikt maken voor het hosten van een Python Django-project gaan we de volgende onderdelen installeren en configuren. De database omgeving PostgresSQL en de opensource webserver Nginx met Gunicorn.

Gunicorn is een Python WSGI http server. WSGI staat voor Webserver Gateway Interface en zorgt dat we de Python Django webapplicatie zelfstandig kunnen draaien in de Nginx webserver-omgeving.

Database PostgreSQL: database en gebruiker

Via de terminal kun je de database-omgeving van Postgres opstarten. Let op: je komt dan gelijk in de terminal omgeving van Postgres

```
$ sudo -u postgres psql
```

Je bent nu ingelog als gebruiker ***postgres*** in de *psql* applicatie

Voer de volgende sql commando's uit. Let op: deze zijn HOOFDLETTER gevoelig.

Bij het 'password' mag je een eigen password invoeren, deze heb je wel later nodig voor de setting.py configuratie.

```
postgres=# CREATE DATABASE portfoliodb;
postgres=# CREATE USER portfoliouser WITH PASSWORD 'password';
postgres=# ALTER ROLE portfoliouser SET client_encoding TO 'utf8';
postgres=# ALTER ROLE portfoliouser SET default_transaction_isolation TO
'read committed';
postgres=# ALTER ROLE portfoliouser SET timezone TO 'UTC';
postgres=# GRANT ALL PRIVILEGES ON DATABASE portfoliodb TO portfoliouser;
postgres=# \q
```

Commando's Postgres terminal omgeving

```
\l          : list alle databases
\c <naam>   : connection naar de database
ALTER USER portfoliouser WITH PASSWORD '######";
```

Virtualenv: virtuele omgeving aanmaken voor het portfolio-project

Je gaat nu een virtuelenv installeren op de server. Je hebt dit geoefend op je eigen localhost laptop.

Via de terminal verbinding met de Raspberry PI of server ga je de volgende commando's uitvoeren.

```
$ pwd
Het commando pwd geeft weer waar je nu bent op de server, dus in beeld
komt: /home/gebruiker (gebruikersnaam waarmee je inlogt via ssh)

$ sudo -H pip3 install virtualenv

$ mkdir portfolio
$ cd portfolio
$ virtualenv portfolioenv

$ source ./portfolioenv/bin/activate
```

Nu de virtuele omgeving is geactiveerd, kun je de gewenste packages voor Python Django installeren.

Je kunt dit doen met behulp van de requirements.txt van de localhost installatie van je laptop.

Het volgende commando zorgt er voor dat de gemaakt requirements.txt wordt gebruikt voor het installeren van de gewenste Python packages:

```
(portfolioenv)…$ pip install -r requirements.txt
```

Of installeer minimaal de volgende packages:

```
(portfolioenv)…$ pip install django gunicorn psycopg2-binary pillow
```

Maak vervolgens een Django project met de naam portfolio

```
(portfolioenv)… $ django-admin.py startproject portfolio
```

Nu is de mapstructuur portfolio/portfolio

Voor de leesbaarheid hernoemen we daarom de map portfolio naar portfolio-project. Voer daarvoor het volgende commando uit

```
(portfolioenv)… $ mv portfolio portfolio-project
```

Maak in de firewall de port 8000 beschikbaar. Deze is tijdelijk nodig en wordt later uitgeschakeld met disallow.

```
(portfolioenv)… $ sudo ufw allow 8000
```

Gunicorn: Webserver Gateway Interface

Testen van de Gunicorn's mogelijkheden om portfolio te hosten

```
(portfolioenv) $ cd portfolio-project

(portfolioenv) $ gunicorn --bind 0.0.0.0:8000 portfolio.wsgi
```

Ga naar de terug naar het terminal-venster en doe het volgende:

```
CTRL-C (toetscombinatie)
(portfolioenv) $ deactivate
```

Aanmaken van een systemd socket en service files voor Gunicorn

Gunicorn draait in de achtergrond als een systemd service. Op de zelfde technische manier als Nginx webserver. Je kunt een service starten, stoppen en de status er van opvragen. Om Gunicorn te laten werken moet je een zgn. socket-bestand aanmaken. Dit doe op de volgende manier:

```
$ sudo nano /etc/systemd/system/gunicorn.socket
```

In het socket bestand verwerk je de volgende regels met code:

```
[Unit]
Description=gunicorn socket

[Socket]
ListenStream=/run/gunicorn.sock

[Install]
WantedBy=sockets.target
```

Gunicorn: service bestand

Nu ga je het gunicorn.service bestand maken. Dit gaat op de zelfde manier. Voer de volgende code uit in de terminal:

```
$ sudo nano /etc/systemd/system/gunicorn.service
```

In het service bestand verwerk je de volgende regels met code:

```
[Unit]
Description=gunicorn daemon
Requires=gunicorn.socket
After=network.target

[Service]
User=gebruikersnaam
Group=www-data

WorkingDirectory=/home/gebruikersnaam/portfolio/portfolio-project

ExecStart=/home/gebruikersnaam/portfolio/portfolioenv/bin/gunicorn \
          --access-logfile - \
          --workers 3 \
          --bind unix:/run/gunicorn.sock \
          portfolio.wsgi:application --preload

[Install]
WantedBy=multi-user.target
```

Na het afsluiten van het gemaakte bestand in Nano, moeten de gunicorn socket worden gestart en beschikbaar gemaakt na een herstart. Een service starten doe met de volgende code in terminal.

```
$ sudo systemctl start gunicorn.socket
$ sudo systemctl enable gunicorn.socket
```

Controleren van de gunicorn socket file

Nu alle bestanden voor gunicorn socket file zijn gemaakt, moet je deze nog testen op de juiste werking. Dit doe je door de volgende commando's te gebruiken:

```
$ sudo systemctl status gunicorn.socket

Op het scherm zie in elk geval dit:
Active: active (running) ……..
```

```
$ file /run/gunicorn.sock
Output /run/gunicorn.sock: socket
```

Problemen of het werkt niet? Controleer dan goed het .service bestand en het socket bestand. Een type foutje is zo gemaakt. Doorloop het proces nog een keer en check goed op type fouten.

Voor onze server moeten we nog wat lokale aanpassingen maken.
Maak een bestand aan met de naam **local_settings.py** in de map /home/**gebruikersnaam**/portfolio/portfolio-project/portfolio. Hierin moet je nog het lokale interne IP-adres van de server, het externe IP-adres van de server en eventueel een eigen domeinnaam 'toestaan' bij ALLOWED_HOSTS.

Ga naar de map

```
$ cd /home/gebruikersnaam/portfolio/portfolio-project/portfolio
```

Maak het bestand local_settings.py aan en open het:

```
$ sudo nano local_settings.py
```

Voeg daarin de volgende code. Vervang 'IP-server intern', 'IP-server extern' en eventueel 'Domeinnaam.ddns.net' voor je eigen gegevens:

```
ALLOWED_HOSTS = ['IP-server intern', 'IP-server extern',
'Domeinnaam.ddns.net']
```

Het bestand local_settings.py wordt niet vanzelf geladen in de applicatie. Daarvoor moet helemaal **onderaan** in het bestand **settings.py** hetvolgende toegevoegd worden:

```
# Laadt het bestand local_settings.py indien aanwezig.
# Daarmee kun je lokaal settings overrulen.
# Dit blok moet helemaal onderaan in settings.py staan.
try:
    from .local_settings import *
except ImportError:
    pass
```

Herstart de gunicorn server op de volgende manier.

```
$ sudo systemctl daemon-reload
$ sudo systemctl restart gunicorn
```

Nginx webserver: configureren

De Nginx gebruiken we om het http verkeer te regelen op de Raspberry PI of server. Al het verkeer dat binnenkomt op port 80 moet door Nginx geleidt worden naar Gunicorn en worden verwerkt door de Python Django omgeving. De Nginx omgeving moet geconfigureerd worden voor extern IP (internet protocol) verkeer.

Extern IP

Het externe ip adres van de router moet worden verwerkt in het default profiel van Nginx. Dit doe dit als volgt:

```
$ sudo nano /etc/nginx/sites-available/default
```

In het profiel Nginx bestand verwerk je de volgende regels met code. Let op vervang *IP adress ZIGGO/KPN* voor het externe adres van de router.

```
server {
    listen 80;
    server_name IP adress ZIGGO/KPN;

    location = /favicon.ico { access_log off; log_not_found off; }
    location /static/ {
        root /home/gebruikersnaam/portfolio/portfolio-project;
    }

    location / {
        include proxy_params;
        proxy_pass http://unix:/run/gunicorn.sock;
    }
}
```

Intern IP

Het interne ip adres van de Raspberry PI of server moet worden verwerkt in het portfolio profiel van Nginx. Dit doe dit als volgt:

```
$ sudo nano /etc/nginx/sites-available/portfolio
```

In het profiel Nginx bestand verwerk je de volgende regels met code. Let op vervang *IP adress* voor het interne adres van de Raspberry PI of gebruikte server.

```
server {
    listen 80;
    server_name 192.168.178.XXX;

    location = /favicon.ico { access_log off; log_not_found off; }
    location /static/ {
        root /home/gebruikersnaam/portfolio/portfolio-project;
    }

    location / {
        include proxy_params;
        proxy_pass http://unix:/run/gunicorn.sock;
    }
}
```

Maak een link naar de map sites-enabled om het profiel portfolio beschikbaar te maken voor de Nginx omgeving. Dit doe je met het volgende commando:

```
$ sudo ln -s /etc/nginx/sites-available/portfolio /etc/nginx/sites-enabled
```

Test of de Nginx omgeving goed is geconfigureerd met het volgende commando. Je krijgt hiermee de output: syntax ok en test is succesvol.

```
$ sudo nginx -t
```

Pas de firewall aan met de volgende commando's:

```
$ sudo ufw delete allow 8000
$ sudo ufw allow 'Nginx Full'
```

Herstart als laatste de Nginx server op de volgende manier.

```
$ sudo systemctl restart nginx
```

Controleer het resultaat in de browser van je laptop. Dit doe je door naar het interne en externe adres te gaan met de browser.

Je webserver + Django omgeving is nu klaar. Je kunt project compleet maken met de laatste versie van de ontwikkelde website door deze via GIT te Pushen en Pullen op de Raspberry PI. Kijk voor meer informatie naar de GIT tutorial.