Guide pour installer Django Web Framework sur Debian 12

Ibrahim ALAME

14/02/2023

Django est un framework Web Python gratuit et open source utilisé pour développer des sites Web et des applications dynamiques. Il est utilisé pour le développement d'applications Python complexes et basées sur des bases de données. Django suit l'architecture MVC (Model-View-Controller), qui permet aux développeurs d'écrire moins de code et de créer un nouveau site Web en peu de temps. Django peut être installé sur n'importe quel système d'exploitation exécutant Python, notamment Windows, macOS, Linux/Unix et Solaris.

Ce guide vous montrera comment installer le framework web Django sur un serveur Debian 12. Vous commencerez avec Django en créant votre premier projet Django en utilisant PostgreSQL comme base de données par défaut, le serveur Gunicorn WSGI et Nginx comme proxy inverse.

1 Conditions préalables

Pour commencer, assurez-vous d'avoir les éléments suivants :
- Une machine Debian 12.
- Un utilisateur non root avec les privilèges d'administrateur sudo.

2 Installation des dépendances

Dans la première étape, vous installerez les dépendances de packages pour l'installation de votre framework Web Django, cela inclut les packages suivants :
- Serveur PostgreSQL : Par défaut, Django utilise SQLite, et cet exemple vous montrera comment utiliser PostgreSQL comme base de données pour votre projet Django.
- Superviseur : C'est un gestionnaire de processus, et vous exécuterez votre application Django avec Gunicorn et Supervisor.
- Serveur Web Nginx : cet exemple vous montrera comment utiliser Nginx comme proxy inverse pour votre projet Django. Cela permet à votre projet Django d'être accessible via un nom de domaine local

Tout d'abord, exécutez la commande apt ci-dessous pour mettre à jour et actualiser l'index de votre package.

```
sudo apt update
```

Installez maintenant les dépendances de packages telles que l'environnement virtuel Python, le gestionnaire de packages pip, le pilote PostgreSQL et libpq5, Nginx et Supervisor.

```
sudo apt install build-essential python3-dev python3-pip python3-venv nginx supervisor
postgresql libpq5 libpq-dev
```

Tapez y pour confirmer et procéder à l'installation.

```
root@debian12:~#
root@debian12:~# sudo apt install build-essential python3-dev python3-pip python3-venv nginx supervisor postgresql libpq5 libpq-dev
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
build-essential is already the newest version (12.9).
build-essential set to manually installed.
nginx is already the newest version (1.22.1-9).
postgresql is already the newest version (15+248).
libpq5 is already the newest version (15.3-0+deb12u1).
libpq5 set to manually installed.
The following additional packages will be installed:
  libexpat1-dev libjs-jquery libjs-sphinxdoc libjs-underscore libpython3-dev libpython3.11 libpython3.11-dev python3-distutils python3-lib2to3
  python3-pip-whl python3-setuptools python3-setuptools-whl python3-wheel python3.11-dev python3.11-venv zlib1g-dev
Suggested packages:
  postgresql-doc-15 python-setuptools-doc supervisor-doc
The following NEW packages will be installed:
  libexpat1-dev libjs-jquery libjs-sphinxdoc libjs-underscore libpq-dev libpython3-dev libpython3.11 libpython3.11-dev python3-dev
  python3-distutils python3-lib2to3 python3-pip python3-pip-whl python3-setuptools python3-setuptools-whl python3-venv python3-wheel
  python3.11-dev python3.11-venv supervisor zlib1g-dev
0 upgraded, 21 newly installed, 0 to remove and 30 not upgraded.
Need to get 14.4 MB of archives.
After this operation, 54.6 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

Une fois les dépendances installées, exécutez les commandes suivantes pour vérifier les services PostgreSQL, Nginx et Supervisor et assurez-vous que ces services sont en cours d'exécution et activés.

Vérifiez le service PostgreSQL à l'aide de la commande ci-dessous.

```
sudo systemctl is-enabled postgresql
sudo systemctl status postgresql
```

Si PostgreSQL est en cours d'exécution et activé, vous devriez obtenir en dessous de la sortie.

```
root@debian12:~#
root@debian12:~# sudo systemctl is-enabled postgresql
enabled
root@debian12:~# sudo systemctl status postgresql
● postgresql.service - PostgreSQL RDBMS
     Loaded: loaded (/lib/systemd/system/postgresql.service; enabled; preset: enabled)
     Active: active (exited) since
   Main PID: 707 (code=exited, status=0/SUCCESS)
        CPU: 2ms
```

Vérifiez le service Nginx à l'aide de la commande ci-dessous.

```
sudo systemctl is-enabled nginx
sudo systemctl status nginx
```

Si Nginx est en cours d'exécution et activé, vous trouverez ci-dessous le résultat que vous devriez obtenir.

Enfin, vérifiez le superviseur à l'aide de la commande ci-dessous.

```
sudo systemctl is-enabled supervisor
sudo systemctl status supervisor
```

Vous devriez voir que le superviseur est en cours d'exécution et activé.

3 Installer Django via Pip

Le framework Web Django peut être installé de différentes manières, notamment manuellement via Git, via le gestionnaire de packages Pip Python, ou combiné avec un environnement isolé avec le module venv et Pip. Dans cet exemple, vous installerez Django via le gestionnaire de packages Pip sur l'environnement Python isolé.

Connectez-vous à votre utilisateur via la commande ci-dessous.

```
su - username
```

Créez maintenant un nouveau répertoire de projet /testdjango et accédez-y.

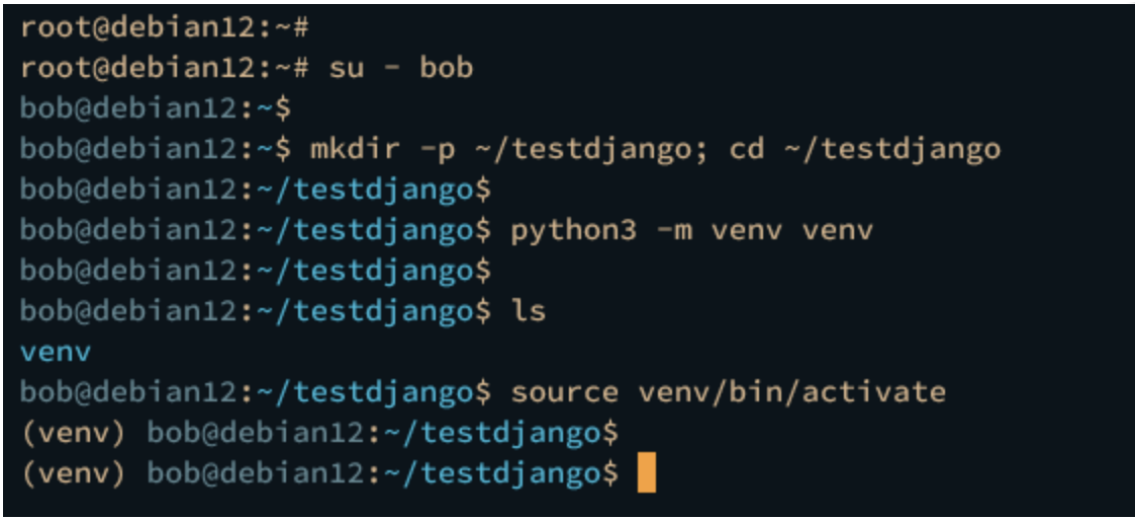
```
mkdir -p ~/testdjango; cd ~/testdjango
```

Ensuite, exécutez la commande suivante pour créer un nouvel environnement virtuel Python appelé venv dans le répertoire de votre projet actuel.

```
python3 -m venv venv
```

Activez-le ensuite à l'aide de la commande ci-dessous. Une fois l'environnement virtuel activé, votre invite actuelle deviendra comme (venv) bob@hostname : .

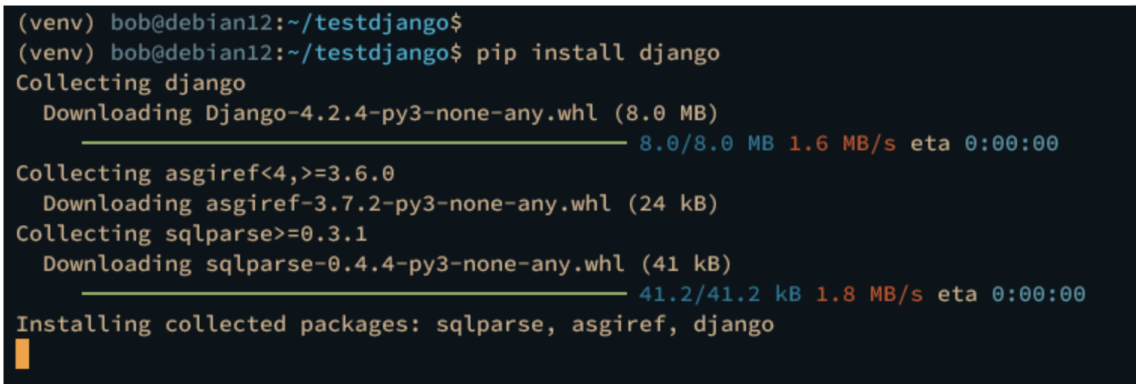
```
source venv/bin/activate
```



Dans l'environnement virtuel venv, exécutez la commande pip ci-dessous pour installer le framework Web Django. Cela installera Django sous votre environnement virtuel Python, et non à l'échelle du système.

```
pip install django

or

pip install django==4.2.4
```

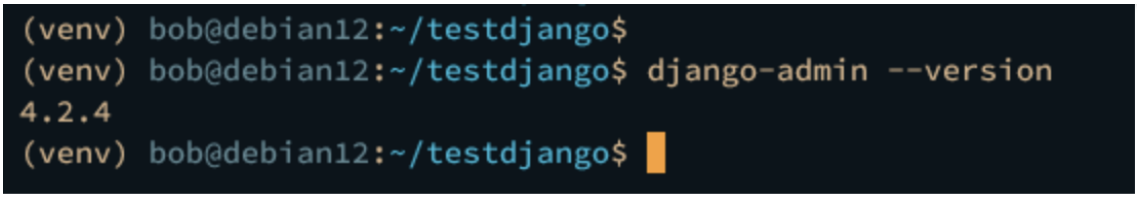
Ci-dessous, l'installation de Django est en cours d'exécution.



Vérifiez votre version de Django une fois l'installation terminée à l'aide de la commande django-admin ci-dessous.

```
django-admin --version
```

Le résultat suivant confirme que Django 4.2.4 est installé dans l'environnement virtuel venv via le gestionnaire de packages Pip Python.



4 Création du premier projet Django

Cette section vous apprendra comment créer le premier projet Django et utiliser le serveur PostgreSQL comme base de données par défaut. Pour y parvenir, procédez comme suit :

— Préparation de la base de données et de l'utilisateur.
— Création du projet Django via django-admin.
— Migration de base de données et génération de fichiers statiques.
— Création d'utilisateurs administrateurs et exécution de Django.

4.1 Préparer la base de données et l'utilisateur

Exécutez la commande pip ci-dessous pour installer le package Python psycopg2 dans votre environnement virtuel. Il s'agit du pilote Python qui sera utilisé par Django pour se connecter au serveur de base de données PostgreSQL.

```
pip install psycopg2
exit
```

```
(venv) bob@debian12:~/testdjango$
(venv) bob@debian12:~/testdjango$ pip install psycopg2
Collecting psycopg2
  Downloading psycopg2-2.9.7.tar.gz (383 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 383.5/383.5 kB 796.8 kB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Installing collected packages: psycopg2
  DEPRECATION: psycopg2 is being installed using the legacy 'setup.py install' method, bec
' package is not installed. pip 23.1 will enforce this behaviour change. A possible replac
n can be found at https://github.com/pypa/pip/issues/8559
  Running setup.py install for psycopg2 ... done
Successfully installed psycopg2-2.9.7
(venv) bob@debian12:~/testdjango$
```

Exécutez maintenant la commande suivante pour vous connecter au shell PostgreSQL.

```
sudo -u postgres psql
```

Exécutez les requêtes suivantes pour créer une nouvelle base de données et un nouvel utilisateur pour votre projet Django. L'exemple suivant créera une nouvelle base de données Djangodb, l'utilisateur Django et le mot de passe p4ssw0rd.

```
CREATE USER django WITH PASSWORD 'p4ssw0rd';
CREATE DATABASE djangodb OWNER django;
```

```
root@debian12:~#
root@debian12:~# sudo -u postgres psql
could not change directory to "/root": Permission denied
psql (15.3 (Debian 15.3-0+deb12u1))
Type "help" for help.

postgres=# CREATE USER django WITH PASSWORD 'p4ssw0rd';
CREATE ROLE
postgres=# CREATE DATABASE djangodb OWNER django;
CREATE DATABASE
postgres=#
```

Ensuite, exécutez les requêtes suivantes pour vérifier la base de données de liste et l'utilisateur sur votre serveur PostgreSQL.

```
\du
\l
```

Vous devriez voir la base de données Djangodb et l'utilisateur Django est créé.

```
postgres=#
postgres=# \du
                                   List of roles
 Role name |                         Attributes                         | Member of
-----------+------------------------------------------------------------+-----------
 django    |                                                            | {}
 postgres  | Superuser, Create role, Create DB, Replication, Bypass RLS | {}
                                                                        | {}

postgres=# \l
                                                  List of databases
   Name    |  Owner   | Encoding |   Collate   |    Ctype    | ICU Locale | Locale Provider |
-----------+----------+----------+-------------+-------------+------------+-----------------+---
 djangodb  | django   | UTF8     | en_US.UTF-8 | en_US.UTF-8 |            | libc            |
 postgres  | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |            | libc            |
 template0 | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |            | libc            | =
           |          |          |             |             |            |                 | p
 template1 | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |            | libc            | =
           |          |          |             |             |            |                 | p
                                                                                             |
(5 rows)
```

Tapez quit pour quitter le serveur PostgreSQL.

4.1.1 Création d'un projet Django

Avant de créer le projet Django, connectez-vous à votre utilisateur et activez l'environnement virtuel venv.

```
su - bob
cd testdjango; source venv/bin/activate
```

Pour créer un nouveau projet Django, exécutez la commande django-admin ci-dessous. Dans ce cas, vous créerez un nouveau projet testapp sur votre répertoire de travail actuel.

```
django-admin startproject testapp
```

Une fois le projet créé, le nouveau répertoire testapp sera créé sur votre répertoire de travail.

```
root@debian12:~#
root@debian12:~# su - bob
bob@debian12:~$
bob@debian12:~$ cd testdjango; source venv/bin/activate
(venv) bob@debian12:~/testdjango$
(venv) bob@debian12:~/testdjango$ django-admin startproject testapp .
(venv) bob@debian12:~/testdjango$
(venv) bob@debian12:~/testdjango$ ls
manage.py  testapp  venv
(venv) bob@debian12:~/testdjango$
```

Ensuite, exécutez la commande suivante pour générer un secret aléatoire pour votre projet Django. Assurez-vous de copier le résultat, car vous l'utiliserez pour sécuriser votre installation Django.

```
python3 -c 'from django.core.management.utils import get_random_secret_key; print(get_random_secret_key())'
```

Utilisez maintenant votre éditeur préféré et ouvrez le fichier testapp/settings.py.

```
nano testapp/settings.py
```

En haut de la ligne, insérez la configuration suivante.

```
import os
```

Insérez votre clé secrète dans le paramètre SECRET_KEY.

```
SECRET_KEY = 'fzahzbm*wrxoleqb0^-3%%tf^y!b6lsc5-c#2^@#s6gkyrl2ef'
```

Saisissez votre adresse IP locale et votre nom de domaine local dans le paramètre ALLOWED_HOSTS.

```
ALLOWED_HOSTS = ['127.0.0.1','192.168.10.15','first-django.dev']
```

Modifiez la configuration de la base de données par défaut avec les détails du serveur PostgreSQL comme ceci :

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'djangodb',
        'USER': 'django',
        'PASSWORD': 'p4ssw0rd',
        'HOST': '127.0.0.1',
        'PORT': '5432',
    }
}
```

Enfin, ajoutez le paramètre STATIC_ROOT pour définir un répertoire de stockage des fichiers statiques.

```
STATIC_ROOT = os.path.join(BASE_DIR, "static/")
```

Enregistrez le fichier et quittez l'éditeur lorsque vous avez terminé.

4.1.2 Migration de base de données et génération de fichiers statiques

Avant de migrer la base de données, exécutez la commande suivante pour vous assurer que vous disposez d'une configuration de base de données appropriée.

```
python3 manage.py check --database default
```

Si aucune erreur, exécutez la commande suivante pour migrer la base de données.

```
python3 manage.py makemigrations
python3 manage.py migrate
```

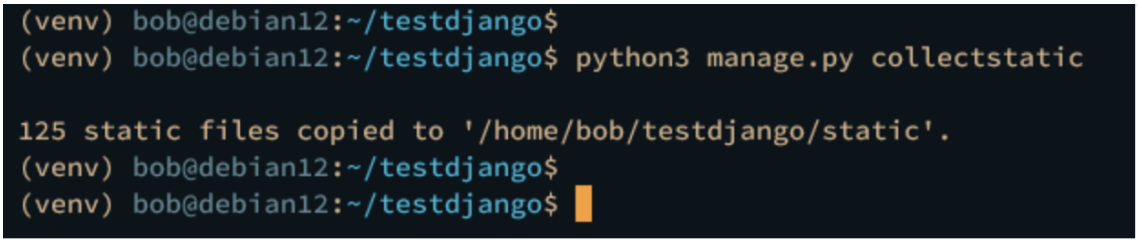
Si tout se passe bien, vous devriez voir le processus de migration de la base de données comme ceci :

```
(venv) bob@debian12:~/testdjango$
(venv) bob@debian12:~/testdjango$ python3 manage.py check --database default
System check identified no issues (0 silenced).
(venv) bob@debian12:~/testdjango$
(venv) bob@debian12:~/testdjango$ python3 manage.py makemigrations
No changes detected
(venv) bob@debian12:~/testdjango$ python3 manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
(venv) bob@debian12:~/testdjango$
```

Enfin, exécutez la commande suivante pour générer des fichiers statiques pour votre projet Django. Une fois la commande exécutée, le nouveau répertoire statique sera créé et des fichiers statiques y seront générés.

```
python3 manage.py collectstatic
```

Vous trouverez ci-dessous le résultat lors de la génération de fichiers statiques.

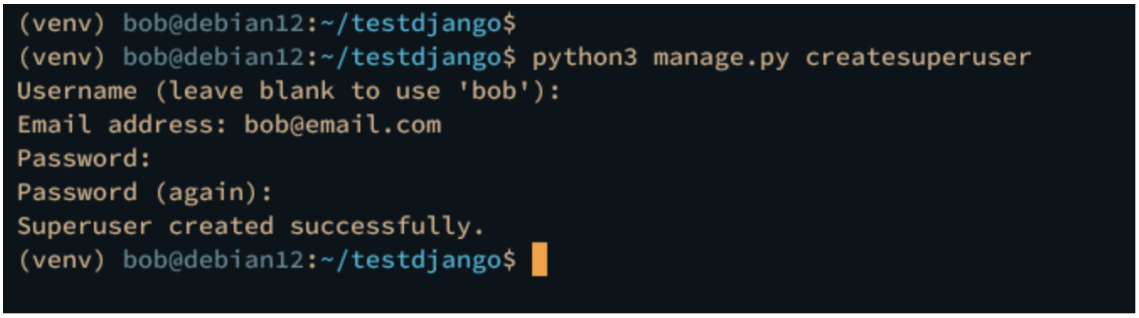


4.1.3 Création d'un utilisateur administrateur et exécution de Django

Exécutez la commande suivante pour créer l'utilisateur administrateur pour votre projet Django.

```
python3 manage.py createsuperuser
```

Saisissez votre adresse e-mail et votre mot de passe lorsque vous y êtes invité.



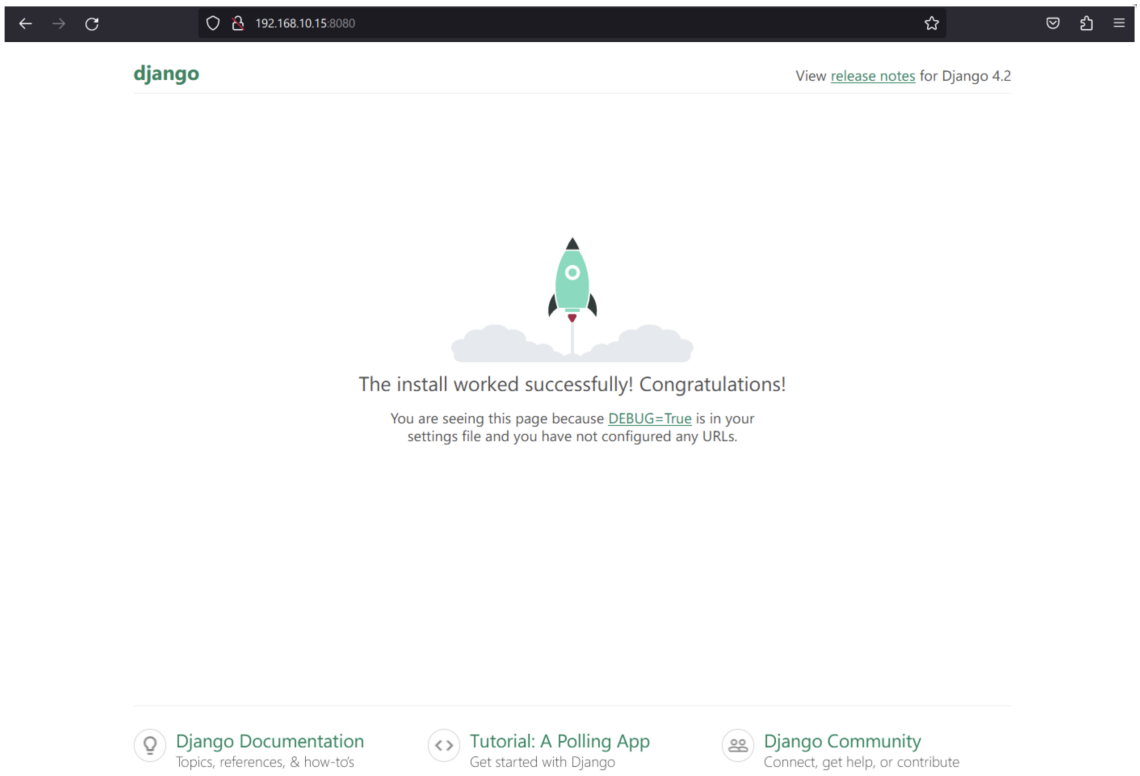
Une fois l'utilisateur administrateur créé, exécutez la commande ci-dessous pour exécuter votre projet Django.

```
python3 manage.py runserver 0.0.0.0:8080
```

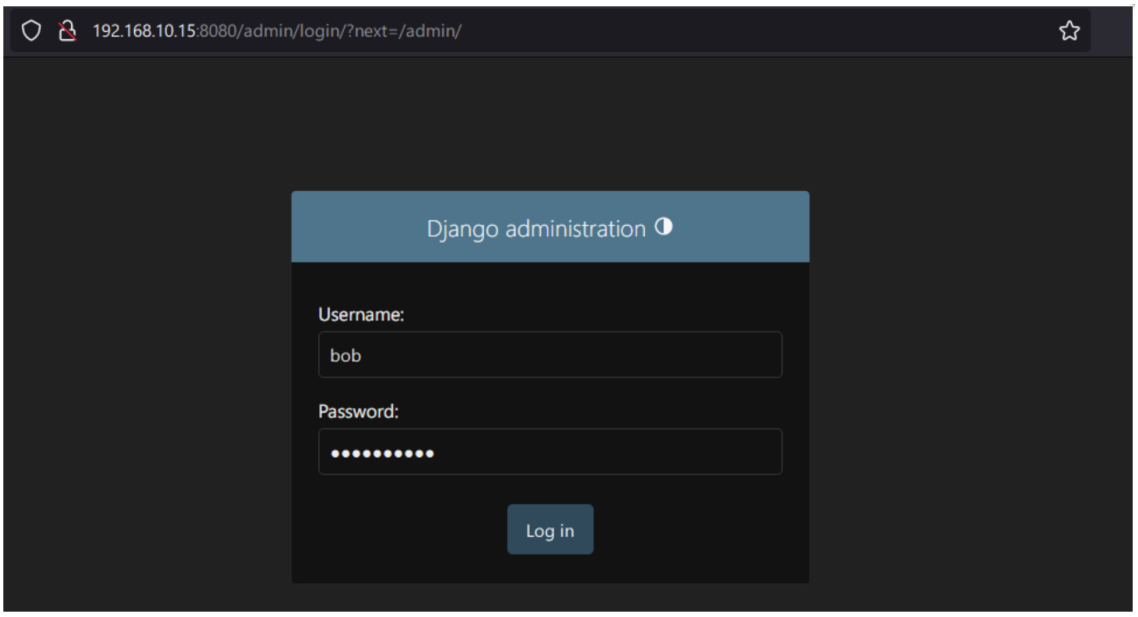
Une fois la commande exécutée, votre projet Django sera exécuté sur votre adresse IP locale sur le port 8080.

```
(venv) bob@debian12:~/testdjango$
(venv) bob@debian12:~/testdjango$ python3 manage.py runserver 0.0.0.0:8080
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

Django version 4.2.4, using settings 'testapp.settings'
Starting development server at http://0.0.0.0:8080/
Quit the server with CONTROL-C.
```

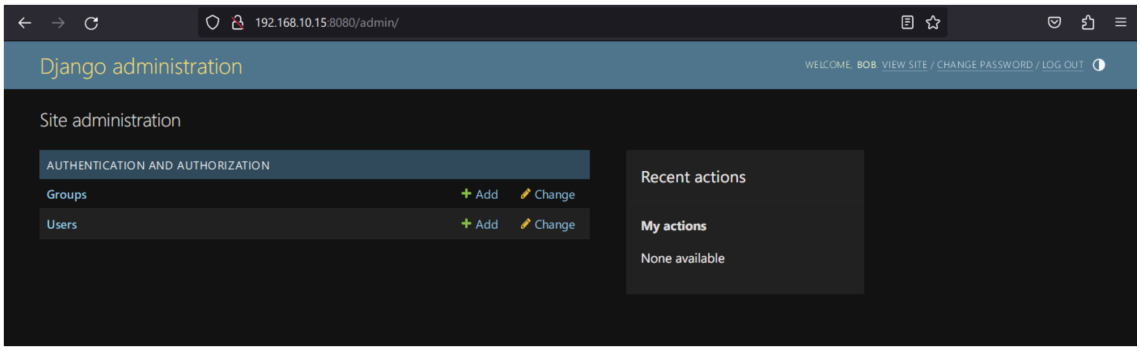
Ouvrez maintenant votre navigateur Web et visitez l'adresse IP du serveur suivie du port 8080, http ://192.168.10.15 :8080/ . Si votre installation de Django réussit, vous devriez voir la page Django index.html par défaut comme celle-ci :



Accédez maintenant à votre administration Django via le chemin URL /admin, http ://192.168.10.15 :8080/admin . Saisissez votre utilisateur et votre mot de passe administrateur Django, puis cliquez sur Se connecter .



Vous devriez voir un exemple de tableau de bord utilisateur Django comme celui-ci :



Appuyez sur Ctrl+c pour terminer le processus.

5 Exécuter Django avec Gunicorn et Supervisor

À ce stade, vous avez terminé l'installation de Django et créé votre premier projet Django. Dans l'étape suivante, vous configurerez Django pour qu'il s'exécute en arrière-plan en utilisant le serveur Gunicorn WSGI et le gestionnaire de processus Supervisor.

5.1 Installation de Gunicorn

Dans l'environnement virtuel venv, exécutez la commande pip ci-dessous pour installer gunicorn.

```
pip install gunicorn
```

```
(venv) bob@debian12:~/testdjango$
(venv) bob@debian12:~/testdjango$ pip install gunicorn
Collecting gunicorn
  Downloading gunicorn-21.2.0-py3-none-any.whl (80 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 80.2/80.2 kB 441.4 kB/s eta 0:00:00
Collecting packaging
  Downloading packaging-23.1-py3-none-any.whl (48 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 48.9/48.9 kB 1.5 MB/s eta 0:00:00
Installing collected packages: packaging, gunicorn
Successfully installed gunicorn-21.2.0 packaging-23.1
(venv) bob@debian12:~/testdjango$
```

Exécutez maintenant la commande suivante pour désactiver l'environnement virtuel venv et revenir à votre utilisateur root.

```
deactivate
exit
```

5.2 Exécuter Django avec Gunicorn et Supervisor

Créez une nouvelle configuration de superviseur /etc/supervisor/conf.d/testapp.conf à l'aide de la commande suivante de l'éditeur nano.

```
sudo nano /etc/supervisor/conf.d/testapp.conf
```

Insérez la configuration suivante et assurez-vous de modifier les informations détaillées du nom de l'application, du chemin du projet et de l'utilisateur avec vos informations. Dans cet exemple, vous exécuterez votre projet Django sous le socket UNIX /home/bob/testdjango/testapp.sock .

```
[program:testapp]
command=/bin/bash -c 'source /home/bob/testdjango/venv/bin/activate; gunicorn -t 3000 --workers 3 --bind unix:unix:/home/b
directory=/home/bob/testdjango
user=bob
group=www-data
autostart=true
autorestart=true
stdout_logfile=/home/bob/testdjango/testapp.log
stderr_logfile=/home/bob/testdjango/error.log
```

Enregistrez et quittez le fichier lorsque vous avez terminé.

Enfin, exécutez la commande suivante pour redémarrer le service superviseur et appliquer les modifications. Ensuite, vérifiez le service du superviseur pour vous assurer que le service est en cours d'exécution.

```
sudo systemctl restart supervisor
sudo systemctl status supervisor
```

```
root@debian12:~#
root@debian12:~# sudo nano /etc/supervisor/conf.d/testapp.conf
root@debian12:~#
root@debian12:~# sudo systemctl restart supervisor
root@debian12:~# sudo systemctl status supervisor
● supervisor.service - Supervisor process control system for UNIX
     Loaded: loaded (/lib/systemd/system/supervisor.service; enabled; preset: enabled)
     Active: active (running) since
       Docs: http://supervisord.org
   Main PID: 2511 (supervisord)
      Tasks: 1 (limit: 4642)
     Memory: 20.1M
        CPU: 467ms
     CGroup: /system.slice/supervisor.service
             └─2511 /usr/bin/python3 /usr/bin/supervisord -n -c /etc/supervisor/supervisord.conf
```

5.3 Vérification de Django via Supervisorctl

Exécutez la commande superviseurctl ci-dessous pour vérifier l'état de l'application en cours d'exécution dans le superviseur.

```
sudo supervisorctl status
```

Vous devriez voir que l' application testapp s'exécute sur le PID 2577 .

```
root@debian12:~#
root@debian12:~# sudo supervisorctl status
testapp                          RUNNING   pid 2577, uptime 0:01:14
root@debian12:~#
root@debian12:~#
```

Enfin, vérifiez votre application Django via la commande curl ci-dessous.

```
curl --unix-socket /home/bob/testdjango/testapp.sock 127.0.0.1
```

Si tout se passe bien, vous devriez voir le code source de la page index.html de votre projet Django.

```
root@debian12:~#
root@debian12:~# curl --unix-socket /home/bob/testdjango/testapp.sock 127.0.0.1

<!doctype html>

<html lang="en-us" dir="ltr">
    <head>
        <meta charset="utf-8">
        <title>The install worked successfully! Congratulations!</title>
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <style>
          html {
            line-height: 1.15;
          }
          a {
            color: #19865C;
          }
```

6 Configurer Nginx comme proxy inverse pour Django

Maintenant que votre projet Django s'exécute en arrière-plan, l'étape suivante consiste à configurer Nginx comme proxy inverse pour Django. L'exemple suivant utilisera un nom de domaine local.

Créez une nouvelle configuration de bloc de serveur Nginx /etc/nginx/sites-available/django à l'aide de l'éditeur nano suivant.

```
sudo nano /etc/nginx/sites-available/django
```

Insérez la configuration suivante et assurez-vous de modifier le nom de domaine dans le paramètre server_name .

```
server {
    listen 80;
    server_name first-django.dev;

    location = /favicon.ico { access_log off; log_not_found off; }
    try_files $uri @django;
    location /static {
        alias /home/bob/testdjango/static/;
    }

    location @django {
        proxy_set_header Host $http_host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_pass http://unix:/home/bob/testdjango/testapp.sock;
    }
}
```

Enregistrez et quittez le fichier lorsque vous avez terminé.

Exécutez maintenant la commande suivante pour activer le bloc serveur /etc/nginx/sites-available/django , puis vérifiez la syntaxe Nginx pour vous assurer que vous disposez de la syntaxe appropriée.

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

Vous devriez obtenir que la syntaxe de sortie est correcte - le test est réussi lorsque vous disposez de la syntaxe Nginx appropriée.

Enfin, redémarrez votre service Nginx en exécutant la commande suivante et en appliquant les modifications que vous avez apportées.

```
sudo systemctl restart nginx
```

Avec cela, votre projet Django est accessible via un nom de domaine local. Sur votre ordinateur local, modifiez le fichier /etc/hosts pour Linux ou C :
Windows
System32
drivers
etc
hosts pour Windows. Ensuite, définissez l'adresse IP et le nom de domaine de votre serveur comme suit.

```
192.168.10.15  first-django.dev
```

Enregistrez et quittez le fichier lorsque vous avez terminé. Revenez à votre navigateur Web et visitez votre nom de domaine local, vous devriez voir la page index.html par défaut de votre projet Django.

7 Conclusion

En résumé, en suivant ce guide étape par étape, vous avez installé le framework web Django avec PostgreSQL, Nginx, Gunicorn et Supervisor sur Debian 12. Vous avez appris à créer un projet Django avec PostgreSQL comme base de données par défaut. et exécutez Django avec Gunicorn et Supervisor. Vous pouvez désormais créer et développer votre application avec Django.