Django – NGINX: deploy your Django project on a production server

**1. Install required packages using apt**

sudo apt install nginx uwsgi uwsgi-plugin-python3

Why do you need uWSGI? In very simple terms NGINX on its own cannot run a Python process to host your application, for this you’ll need a so called *application server* that will host a Python process running your Django project. NGINX and uWSGI will “talk” each other using the uwsgi protocol.

**2. Create directories for your static and media files**

Static files are “not-python” files needed by your Django project, for example Javascript, CSS and images. Media files will be the files uploaded by the users of your application. Not every application will let users upload files, but it’s a very common scenario. Django will not serve static and media files by itself. We’ll leverage NGINX to serve them.

First of all you have to create the directories. Here I assume that you are currently using the user **ubuntu** with the default home directory **/home/ubuntu**:

mkdir -p /home/ubuntu/static /home/ubuntu/media

sudo chown www-data.www-data /home/ubuntu/media

The second command will make the user named **www-data** the owner of the **/home/ubuntu/media** directory. **www-data** will be the user running your Python process in uWSGI, and that user should be able to write in the media directory to correctly save user uploaded files.

**3. Setup your Django project and install requirements**

This step really depends on your particular Django application, for the purpose of this tutorial I will assume that your Django project is installed in the directory **/home/ubuntu/django\_project/** with the following structure:

/home/ubuntu/django\_project/

├── app1

│   ├── admin.py

│   ├── \_\_init\_\_.py

│   ├── migrations

│   │   └── \_\_init\_\_.py

│   ├── models.py

│   ├── tests.py

│   ├── views.py

├── manage.py

└── project

   ├── \_\_init\_\_.py

   ├── settings

  │   └── \_\_init\_\_.py

  │   └── base.py

  │   └── production.py

   ├── urls.py

   ├── wsgi.py

Also I will assume that you installed all your Python requirements, for example using apt or pip.

I always follow a best practice when starting a new Django project, by splitting the monolithic **settings.py** file in different files, one for each deploy environment (local, test, production, …). You can read a more in depth explanation of this approach if you aren’t used to it.

In our case Django will use the module **project/settings/production.py** for his settings. Here we set the **STATIC\_ROOT** and **MEDIA\_ROOT** variables to the directories we created at step 2:

from .base import \*

ALLOWED\_HOSTS = [ 'www.example.com' ] # customize with your domain name

DATABASES = {

'default': { ... } # here the configuration for your database

}

STATIC\_ROOT = '/home/ubuntu/static'

MEDIA\_ROOT = '/home/ubuntu/media'

**4. Collect static files**

Run the following command to collect all static files for your Django project:

./manage.py collectstatic

This command will copy all static files (Javascript, CSS, images) for all your Django apps in the **STATIC\_ROOT** directory configured in **production.py**. For instance **/home/ubuntu/static**.

**5. Configure uWSGI to host your Django project**

Create a file named **django.ini** in the **/etc/uwsgi/apps-enabled/** directory. The content of the file should be something like this:

[uwsgi]

chdir = /home/ubuntu/django\_project # customize with your django installation directory

env = DJANGO\_SETTINGS\_MODULE=project.settings.production # customize with your settings module

wsgi-file = project/wsgi.py # customize with the relative path to your wsgi.py file

workers = 1

Restart uWSGI with:

service uwsgi restart

You should find the uWSGI logs in **/var/log/uwsgi/apps/django.log**. Therefore you can check them to see if the Python process started correctly or there are issues.

**6. Configure NGINX to serve your application**

Create a file named **django** in the **/etc/nginx/sites-enabled/** directory. The content of the file should be something like this:

server {

listen 80;

server\_name www.example.com; # customize with your domain name

location / {

# django running in uWSGI

uwsgi\_pass unix:///run/uwsgi/app/django/socket;

include uwsgi\_params;

uwsgi\_read\_timeout 300s;

client\_max\_body\_size 32m;

}

location /static/ {

# static files

alias /home/ubuntu/static/; # ending slash is required

}

location /media/ {

# media files, uploaded by users

alias /home/ubuntu/media/; # ending slash is required

}

}

Restart NGINX with:

service nginx restart

**7. Enjoy your Django application**

Point the browser to your domain, and you should see your Django application in all of its glory!