

28.11.2019 в 04:52 [LegGnom](#)

Django - NGINX: запускаем наш проект на сервере

[Django](#) - [NGINX](#) - это популярная и хорошо протестированная комбинация, используемая для развертывания веб-приложений в продакшене. В этом посте я объясню шаги, необходимые для развертывания вашего проекта [Django](#) на сервере с использованием Ubuntu 18.04.

Чтобы развернуть Django - NGINX на вашем сервере, выполните следующие простые шаги.

1. Установите необходимые пакеты, используя apt

```
sudo apt install nginx uwsgi uwsgi-plugin-python3
```

Зачем вам нужен [UWSGI](#) ? Проще говоря, NGINX сам по себе не может запустить процесс Python для размещения вашего приложения, для этого вам понадобится так называемый *сервер приложений*, на котором будет размещен процесс Python, выполняющий ваш проект Django. NGINX и uWSGI будут «общаться» друг с другом по протоколу uwsgi.

2. Создайте каталоги для ваших статических и медиа файлов

Статические файлы - это файлы «не-python», необходимые для вашего проекта Django, например Javascript, CSS и изображения. Медиа-файлы - это файлы, загруженные пользователями вашего приложения. Не каждое приложение позволяет пользователям загружать файлы, но это очень распространенный сценарий. Django не будет обслуживать статические и медиа файлы самостоятельно. Мы будем использовать NGINX, чтобы работать с ними.

Прежде всего, вы должны создать каталоги. Здесь я предполагаю, что в настоящее время используете пользователя **Ubuntu** с домашним каталогом `~`.



умолчанию **/home/ubuntu:**

```
mkdir -p /home/ubuntu/static /home/ubuntu/media
sudo chown www-data.www-data /home/ubuntu/media
```

Вторая команда сделает пользователя с именем **www-data** владельцем каталога **/home/ubuntu/media**. **www-data** будет пользователем, выполняющим ваш процесс Python в uWSGI, и этот пользователь должен иметь возможность писать в каталог мультимедиа, чтобы правильно сохранять загруженные пользователем файлы.

3. Настройте свой проект Django и установите требования

Этот шаг действительно зависит от вашего конкретного приложения Django, для целей данного руководства я буду предполагать, что ваш проект Django установлен в каталоге **/home/ubuntu/django_project/** со следующей структурой:

```
/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
```

Также я предполагаю, что вы установили все свои зависимости Python, например, используя apt или pip.

Я всегда следую рекомендациям при запуске нового проекта Django, разбивая монолитный файл **settings.py** на разные файлы, по одному для каждой среды развертывания (local, test, production и т.д.).

В нашем случае Django будет использовать модуль **project/settings/production.py** для своих настроек. Здесь мы устанавливаем переменные **STATIC_ROOT** и **MEDIA_ROOT** для каталогов, которые мы создали на шаге 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. Соберите статические файлы

Запустите следующую команду, чтобы собрать все статические файлы для вашего проекта Django:

```
./manage.py collectstatic
```

Эта команда скопирует все статические файлы (javascript, CSS, изображения) для всех ваших приложений Django в каталог **STATIC_ROOT**, настроенный в **production.py**. Например, **/home/ubuntu/static**.

5. Настройте uWSGI для размещения вашего проекта Django.

Создайте файл с именем **django.ini** в каталоге **/etc/uwsgi/apps-enabled/**.

Содержимое файла должно быть примерно таким:

```
[uwsgi]
chdir = /home/ubuntu/django_project # customize with your django i
env = DJANGO_SETTINGS_MODULE=project.settings.production # customi
```



```
wsgi-file = project/wsgi.py # customize with the relative path to your wsgi.py file
workers = 1
```

Перезапустите uWSGI с помощью:

```
service uwsgi restart
```

Вы должны найти логи uWSGI в **/var/log/uwsgi/apps/django.log**. Поэтому вы можете проверить их, чтобы увидеть, правильно ли запущен процесс Python или есть проблемы.

6. Настройте NGINX для обслуживания вашего приложения

Создайте файл с именем **django** в каталоге **/etc/nginx/sites-enabled/**. Содержимое файла должно быть примерно таким:

```
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
    }
}
```

Перезапустите NGINX с помощью:

```
service nginx restart
```

7. Наслаждайтесь своим приложением Django

Направьте браузер на свой домен, и вы увидите приложение Django во всей его красе!

[#Python](#) [#Django](#) [#Nginx](#) [#uWSGI](#)

Поделится публикацией



11607



0



0



Комментарии

Чтобы оставить комментарий, необходимо [авторизоваться](#)

Присоединяйся в тусовку

Возможно вам понравится

[Переменные времени сборки в Go](#)

[Проблема с зазубренными краями на градиентах.](#)

[Изучаем Go - создание загрузчика \(часть 3\)](#)



В подарок 100\$ на счет при регистрации

Получить

[Реклама на DevGang](#)

Генератор ЧПУ

Библиотека переводов

DevGang

Служба поддержки
support@dev-gang.ru

☰ Меню