Инструкция по разворачиванию проекта на Ubuntu

# Настройка сервера и локального компьютера

Убедитесь в том, чтобы ваш сервер работал и не был приостановлен

В проекте необходимо скопировать .env файл и назвать его .env.dummy и прописать туда шаблонные данные. Этот шаблон нам в дальнейшем понадобится

Для начала необходимо подключиться к серверу по такому сетевому протоколу, который называется SSH.

*SSH (англ. Secure Shell — «безопасная оболочка») — сетевой протокол прикладного уровня, позволяющий производить удалённое управление операционной системой и туннелирование TCP-соединений (например, для передачи файлов). Схож по функциональности с протоколами Telnet и rlogin, но, в отличие от них, шифрует весь трафик, включая и передаваемые пароли. SSH допускает выбор различных алгоритмов шифрования. SSH-клиенты и SSH-серверы доступны для большинства сетевых операционных систем.*

Далее в зависимости от того на какой операционной системе вы работаете, вам необходимо скачать следующее приложение

Putty

1. Откройте PuTTY и перейдите в Connection > SSH > Auth.
2. Нажмите кнопку Обзор рядом с полем Private key file for authentication
3. Выберите созданный файл. ppk и нажмите «Открыть»
4. Перейдите в категорию «Сеанс» и введите IP-адрес вашего экземпляра в поле «Имя хоста». В поле «Сохраненные сеансы» дайте этому соединению имя.
5. Нажмите кнопку «Сохранить» справа, чтобы сохранить этот сеанс.
6. Щелкните имя сеанса, которое вы только что сохранили, и нажмите кнопку «Открыть»
7. PuTTY подключается к вашему экземпляру и запрашивает ваше имя пользователя.
8. Введите своего пользователя и нажмите Enter. Теперь вы вошли в экземпляр.

Для обновления всех пакетов запустите команды

apt-get update — обновить доступные обновления

apt-get upgrade — обновить все пакеты

Теперь можно начинать настраивать проект!

# Установка необходимых программ

Для удобства использования терминала установим пакет mc

sudo apt-get install mc

Для начала проверьте какая версия Python у вас установлен

$ python3 --version # или 'python3 -V'

Чтобы установить Django с pip, нам нужно установить пакет python3-pip в систему Ubuntu 18.04.

sudo apt-get install python3-pip

Проекты советую устанавливать в репозиторий home

Для этого перейдем в home и запустим команду

$ mkdir django && cd django

Это создаст нам папку Django и перейдет в него

Теперь, когда у нас установлен pip, это означает, что у нас есть возможность быстро установить другие необходимые пакеты для среды Python.

Но перед установкой Django нам также нужно установить virtualenv. Установите «virtualenv», выполнив команду

$ sudo apt-get install virtualenv

Теперь создайте виртуальную среду с любым именем (например, ‘venv’) с помощью установленного пакета virtualenv, а также активируйте среду, выполнив команду

$ virtualenv -p python3 venv && source venv/bin/activate

При работе в виртуальной среде python автоматически ссылается на правильную версию, поэтому мы можем использовать python вместо python3.

Теперь нам нужно установить различные библиотеки разработки, необходимые для сборки приложения. Они включают python3-dev для компиляции любых расширений Python и поддерживающих их стандартных библиотек, а также пакет libmysqlclient-dev , необходимый для сборки клиента PostreSQL

Это можно сделать, выполнив следующие команды в терминале

(venv) $ sudo apt-get install python3-dev

(venv) $ sudo apt-get install libpq-dev

Теперь нам необходимо загрузить наш проект на сервер

Мы будем это делать через Git. Для этого на сервере запускаем

git clone https://your\_repository\_name.git

Если это уже существующий проект, вам может потребоваться подготовить список требований, необходимых для запуска вашего проекта. Обычно вы можете просто создать файл requirements.txt на основе вашей среды Python.

Теперь у нас есть все важные зависимости. Итак, мы наконец-то можем установить Django и все другие инструменты, необходимые для разработки веб-приложения django.

(venv)$ pip install –r requirements.txt

# Установка и настройка PostgreSQL

Теперь давайте установим PostgreSQL. Можем пока деактивировать виртуальную среду

pip install psycopg2-binary

(venv) $ deactivate

$ sudo apt-get install postgresql postgresql-contrib

Эта команда установит PostgreSQL, но при этом вам не будет предложено задать пароль или внести какие-либо правки в конфигурацию. Поскольку это делает установку небезопасной, рассмотрим далее, как настроить установленную PostgreSQL.

Войдите в интерактивный сеанс Postgres, набрав:

sudo -u postgres psql

Сначала вы создадите базу данных для проекта Django. Каждый проект должен иметь свою изолированную базу данных из соображений безопасности. В этом руководстве мы будем вызывать базу данных myproject, но всегда лучше выбрать что-то более описательное:

CREATE DATABASE myproject;

примечание

Не забудьте заканчивать все команды SQL точкой с запятой.

Далее вы создадите пользователя базы данных, которого будете использовать для подключения и взаимодействия с базой данных. Установите надежный и безопасный пароль:

CREATE USER myprojectuser WITH PASSWORD 'password';

После этого вы измените несколько параметров подключения для только что созданного пользователя. Это ускорит операции с базой данных, так что правильные значения не нужно запрашивать и устанавливать каждый раз при установлении соединения.

ALTER ROLE myprojectuser SET client\_encoding TO 'utf8';

ALTER ROLE myprojectuser SET default\_transaction\_isolation TO 'read committed';

ALTER ROLE myprojectuser SET timezone TO 'UTC';

Вы устанавливаете кодировку по умолчанию на UTF-8, которую ожидает Django. Вы также устанавливаете схему изоляции транзакций по умолчанию на «чтение зафиксированных», которая блокирует чтение незафиксированных транзакций. Наконец, вы устанавливаете часовой пояс. По умолчанию ваши проекты Django будут настроены на использование UTC. Это все рекомендации самого проекта Django .

Теперь все, что вам нужно сделать, это предоставить пользователю вашей базы данных права доступа к базе данных, которую вы создали:

GRANT ALL PRIVILEGES ON DATABASE myproject TO myprojectuser;

Выйдите из приглашения SQL, чтобы вернуться к сеансу оболочки пользователя postgres :

\q

# Связываем базу данных с Django

Удостоверьтесь что в вашем проекте установлен django-environ

Создайте файл .env\_dev и в нем пропишите

ALLOWED\_HOSTS={{ip адрес машины}},localhost  
DEBUG=0  
SECRET\_KEY=django-insecure-lv(n=@j2obd23-\*!d6zyi3nkv#u2sc3y\_sjo\_ho9fno87gsf1u  
DB\_NAME=myproject  
DB\_USER=myprojectuser  
DB\_PASSWORD=password

Удостоверьтесь в том, чтобы в настройках вашего проекта были следующие поля

**import** environ

AWS\_SERVER = **True  
  
  
if** AWS\_SERVER:  
 env = environ.Env()  
 environ.Env.read\_env(os.path.join(BASE\_DIR, **'.env\_dev'**))  
 SECRET\_KEY = env(**'SECRET\_KEY'**)  
 DEBUG = env.bool(**'DEBUG'**, default=**False**)  
 ALLOWED\_HOSTS = env.list(**'ALLOWED\_HOSTS'**)  
 DATABASES = {  
 **'default'**: {  
 **'ENGINE'**: **'django.db.backends.postgresql'**,  
 **'NAME'**: env(**"DB\_NAME"**),  
 **'USER'**: env(**"DB\_USER"**),  
 **'PASSWORD'**: env(**"DB\_PASSWORD"**),  
 **'HOST'**: **'localhost'**,  
 }  
 }  
**else**:  
 SECRET\_KEY = **'SECRET\_KEY'** DEBUG = **True** ALLOWED\_HOSTS = [**'127.0.0.1'**, **'localhost'**, ]  
 DATABASES = {  
 **'default'**: {  
 **'ENGINE'**: **'django.db.backends.sqlite3'**,  
 **'NAME'**: BASE\_DIR / **'db.sqlite3'**,  
 }  
 }

STATIC\_URL = '/static/'

MEDIA\_URL = ' /media/'

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

MEDIA\_ROOT = os.path.join(BASE\_DIR, 'media')

Далее выполняем миграции

(venv) $ python manage.py makemigrations

(venv) $ python manage.py migrate

Теперь нам нужно создать администратора/суперпользователя.

(venv) $ python manage.py createsuperuser

Суперпользователь Django успешно создан, теперь мы можем снова выполнить команду runserver

(venv) $ python manage.py runserver 0.0.0.0:8000

Удостоверяемся что 8000 порт у нас открыт

Если порт закрыт удостоверяемся что порт не блокируется

sudo ufw status numbered

Если нужного нам порта нет в списке, включите его с помощью

sudo ufw allow 8000

Если все работает нормально чтобы собрать статические файлы в настроенном ранее месте, мы можем использовать команду collectstatic

(venv)python manage.py collectstatic

Все статические файлы будут собраны в каталоге, указанном выше. В нашем случае он static.

# Установка и настройка Gunicorn

После успешной настройки вашего проекта Django теперь мы можем установить в проект наш Gunicorn.

pip install gunicorn

Поскольку наше приложение работает нормально с runserver, теперь пришло время запустить его с помощью gunicorn. Gunicorn — это Python WSGI (интерфейс шлюза веб-сервера). Он действует как шлюз для отправки запросов в наше приложение django. Поскольку gunicorn используется здесь как модуль pip, мы должны сначала активировать нашу виртуальную среду для запуска gunicorn.

source venv/bin/activate

(venv)$ gunicorn -b 0.0.0.0:8000 **project\_name**.wsgi:application

Если все работает нормально, то выходим из виртуальной среды

deactivate

переходим по следующему пути и создаем там файл

sudo touch /etc/systemd/system/gunicorn.socket

откроем данные файл через mc нажатием f4

и пропишем следующие конфигурационные данные

[Unit]

Description=gunicorn socket

[Socket]

ListenStream=/run/gunicorn.sock

[Install]

WantedBy=sockets.target

[Unit] — для описания сокета

[Socket] — для определения местоположения сокета

[Установить] — убедиться, что сокет создан в нужное время.

Далее там же создаем файл gunicorn.service

touch gunicorn.service

и добавим туда следующие данные

[Unit]

Description=gunicorn daemon

Requires=gunicorn.socket

After=network.target

[Service]

User=root

Group=www-data

WorkingDirectory=your\_directory\_path/project\_name

ExecStart=your\_directory\_path/project\_name/venv/bin/gunicorn \

--access-logfile - \

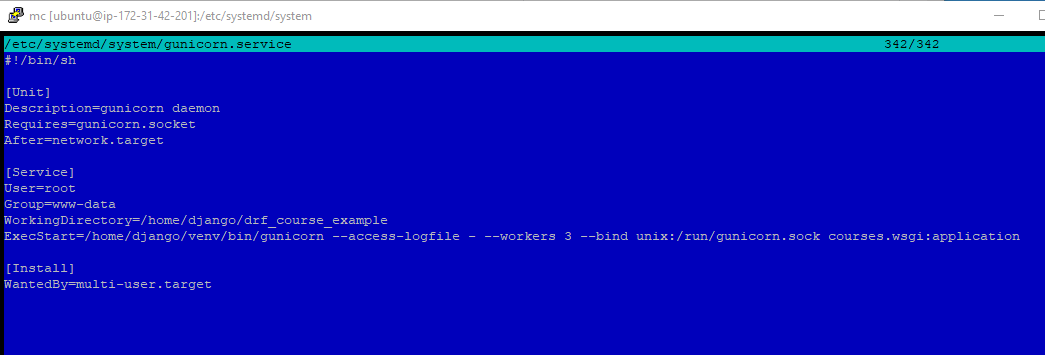
--workers 3 \

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

project\_name.wsgi:application

[Install]

WantedBy=multi-user.target



[Unit] — используется для указания метаданных и зависимостей.

[Service] — указываем пользователя и группу, под которой мы хотим запустить процесс

[Install] — это сообщит системе, с чем связать эту службу, если мы включим ее запуск при загрузке.

Теперь запустите и включите сокет gunicorn

sudo systemctl enable gunicorn.socket

sudo systemctl start gunicorn.socket

Теперь запустите и включите сервис gunicorn

sudo systemctl enable gunicorn.service

sudo systemctl start gunicorn.service

если вы хотите проверить состояние сокета и сервисов. Вы можете использовать эти команды.

sudo systemctl status gunicorn.socket

sudo systemctl status gunicorn.service

# Настройка NGINX

Для начала надо установить nginx

sudo apt-get install nginx

Прямо сейчас нам нужно настроить наш последний компонент — веб-сервер, который обрабатывает наши сетевые запросы. Мы уже установили пакет на первом этапе настройки среды. Теперь давайте создадим файл конфигурации.

sudo touch /etc/nginx/sites-available/your\_project\_name #тут должно быть имя вашего проекта

И добавьте туда эту запись

server{

listen 80;

server\_name your\_domain\_name\_or\_ip\_address;

access\_log /var/log/nginx/access.log;

error\_log /var/log/nginx/error\_log;

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

location /static/ {

root your\_directory\_path/your\_project\_name;

}

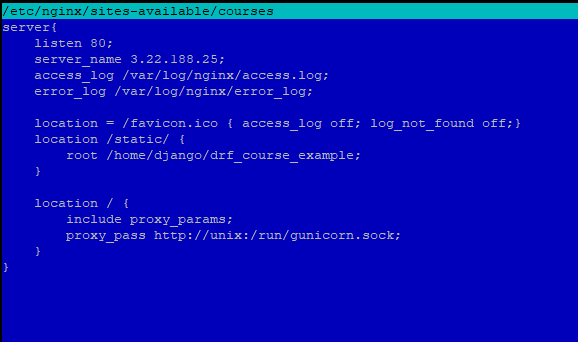
location / {

include proxy\_params;

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

}

}



Сохраните и активируйте файл, связав его с каталогом с поддержкой сайтов. Это необходимо для того, чтобы любые изменения в файле конфигурации были актуальными.

sudo ln -s /etc/nginx/sites-available/**your\_project\_name** /etc/nginx/sites-enabled

Теперь перезапустите сервер Nginx.

sudo systemctl restart nginx

Бывают случаи, когда у нас может быть некоторая опечатка в синтаксисе в файле конфигурации, вы можете проверить это с помощью этой команды.

sudo nginx -t

Наконец, нам нужно настроить параметры безопасности на сервере, чтобы получать запросы.

sudo ufw allow 'Nginx Full'

Готово!