

# Установка Django с PostgreSQL, Nginx и Gunicorn на Ubuntu 18.04

В инструкции описано, как установить и настроить некоторые компоненты в Ubuntu 18.04 для поддержки и обслуживания приложений Django, как настроить базу данных PostgreSQL вместо использования базы данных SQLite по умолчанию, как настроить сервер приложений Gunicorn для взаимодействия с приложениями. Затем в инструкции описана конфигурация Nginx для обращения прокси к Gunicorn.

## Что это такое?

Django - это мощный фреймворк, который позволяет запустить приложение или веб-сайт на Python. Django включает в себя упрощенный сервер разработки для локального тестирования кода.

## Загрузка и установка пакетов

Для начала обновите локальную базу пакетов:

```
sudo apt update  
sudo apt upgrade
```

*Примечание: по умолчанию на серверах Ubuntu 18.04 используется Python 3.*

Для установки необходимого набора пакетов и зависимостей выполните следующую команду:

```
sudo apt install python3-pip python3-dev libpq-dev postgresql  
postgresql-contrib nginx curl
```

В результате у вас будут установлены средства разработки Python, необходимые для сборки Gunicorn, pip, система баз данных Postgres и библиотеки, необходимые для взаимодействия с ней, а также веб-сервер Nginx.

## Создание базы данных и пользователя PostgreSQL

Во время установки Postgres был создан пользователь с именем postgres - администратор СУБД PostgreSQL. Более подробно про работу с PostgreSQL можно ознакомиться в нашей инструкции (</help/linux/ustanovka-postgresql-na-ubuntu-18-04>).

Подключитесь к СУБД с помощью следующей команды:

```
sudo -u postgres psql
```

Создайте базу данных для вашего проекта:

```
CREATE DATABASE cloudproject;
```

Далее создайте пользователя для созданной БД, указав безопасный пароль:

```
CREATE USER clouduser WITH PASSWORD 'password';
```

Для корректной работы Django кодировку необходимо установить в стандарт UTF-8:

```
ALTER ROLE clouduser SET client_encoding TO 'utf8';
```

Далее установите схему изоляции транзакции по умолчанию в «зафиксированное чтение», при котором блокируется чтение из незафиксированных транзакций  
<<https://postgrespro.ru/docs/postgresql/9.6/transaction-iso#mvcc-isolevel-table>> :

```
ALTER ROLE clouduser SET default_transaction_isolation TO 'read committed';
```

Установите рекомендуемый стандарт времени UTC:

```
ALTER ROLE clouduser SET timezone TO 'UTC';
```

В конце предоставьте новому пользователю доступ для управления созданной базой данных:

```
GRANT ALL PRIVILEGES ON DATABASE cloudproject TO clouduser;
```

Выйдите из командной строки PostgreSQL, набрав:

```
\q
```

## Создание виртуальной среды Python для проекта

Сначала необходимо настроить доступ к команде `virtualenv`, которую можно установить с помощью `pip`:

```
sudo -H pip3 install --upgrade pip
sudo -H pip3 install virtualenv
```

Флаг `-H` гарантирует, что политика безопасности устанавливает переменные окружения в домашний каталог целевого пользователя.

Создайте и перейдите в каталог, где вы собираетесь хранить файлы вашего проекта. В нашем примере используется название каталога `1cloud`:

```
mkdir ~/cloudproject
cd ~/cloudproject
```

В каталоге проекта создайте виртуальную среду Python. Выберете название среды, которое имеет отношение к вашему проекту, в нашем примере мы используем название `cloudenv`:

```
virtualenv cloudenv
```

В результате этих действий будет создано откружение в вашем каталоге, внутри будет установлена локальная версия Python и локальная версия `pip`. Это позволяет настроить изолированную среду Python для Jupyter.

Перед установкой Jupyter нам нужно активировать виртуальную среду, указав вместо `1cloud` название вашей среды:

```
source cloudenv/bin/activate
```

Когда виртуальная среда активирована, установите Django, Gunicorn и адаптер `psycopg2` PostgreSQL с помощью локального экземпляра `pip`:

```
pip install django gunicorn psycopg2-binary
```

## Создание и настройка проекта Django

### Создание проекта

Создадим проект в настроенной виртуальной среде, явно задав название проекта и путь:

```
django-admin.py startproject cloudproject ~/cloudproject
```

## Настройка параметров проекта

Для запуска созданного проекта необходимо настроить его параметры. Откройте файл настроек в текстовом редакторе, например nano:

```
nano ~/cloudproject/cloudproject/settings.py
```

Найдите директиву `ALLOWED_HOSTS` и в качестве ее значения укажите IP-адрес или доменное имя сервера:

```
ALLOWED_HOSTS = ['your_server_domain_or_IP']
```

Например:

```
ALLOWED_HOSTS = ['111.111.111.111']
```

Затем найдите раздел, который настраивает доступ к базе данных. Он начинается с ключевого слова `DATABASES`. Укажите собственные значения переменных `NAME`, `USER`, `PASSWORD` ранее созданных в PostgreSQL:

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.postgresql_psycopg2',  
        'NAME': 'cloudproject',  
        'USER': 'clouduser',  
        'PASSWORD': 'password',  
        'HOST': 'localhost',  
        'PORT': '',  
    }  
}
```

Затем переместитесь в конец файла и добавьте параметр, указывающий, где должны быть размещены статические файлы. Это необходимо для того, чтобы Nginx мог правильно обрабатывать запросы к этим элементам:

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

### Завершение начальной настройки проекта

Перенесите исходную схему базы данных в базу данных PostgreSQL, используя скрипт управления:

```
~/cloudproject/manage.py makemigrations  
~/cloudproject/manage.py migrate
```

Создайте административного пользователя для проекта, набрав:

```
~/cloudproject/manage.py createsuperuser
```

На этом шаге будет необходимо выбрать имя пользователя проекта, указать адрес электронной почты, а также выбрать и подтвердить пароль.

Соберите весь статический контент в каталог, который был настроен:

```
~/cloudproject/manage.py collectstatic
```

С помощью `ufw` откройте порт 8000 на котором по умолчанию будет запущен Django:

```
sudo ufw allow 8000
```

Наконец, вы можете протестировать проект, запустив сервер Django с помощью этой команды:

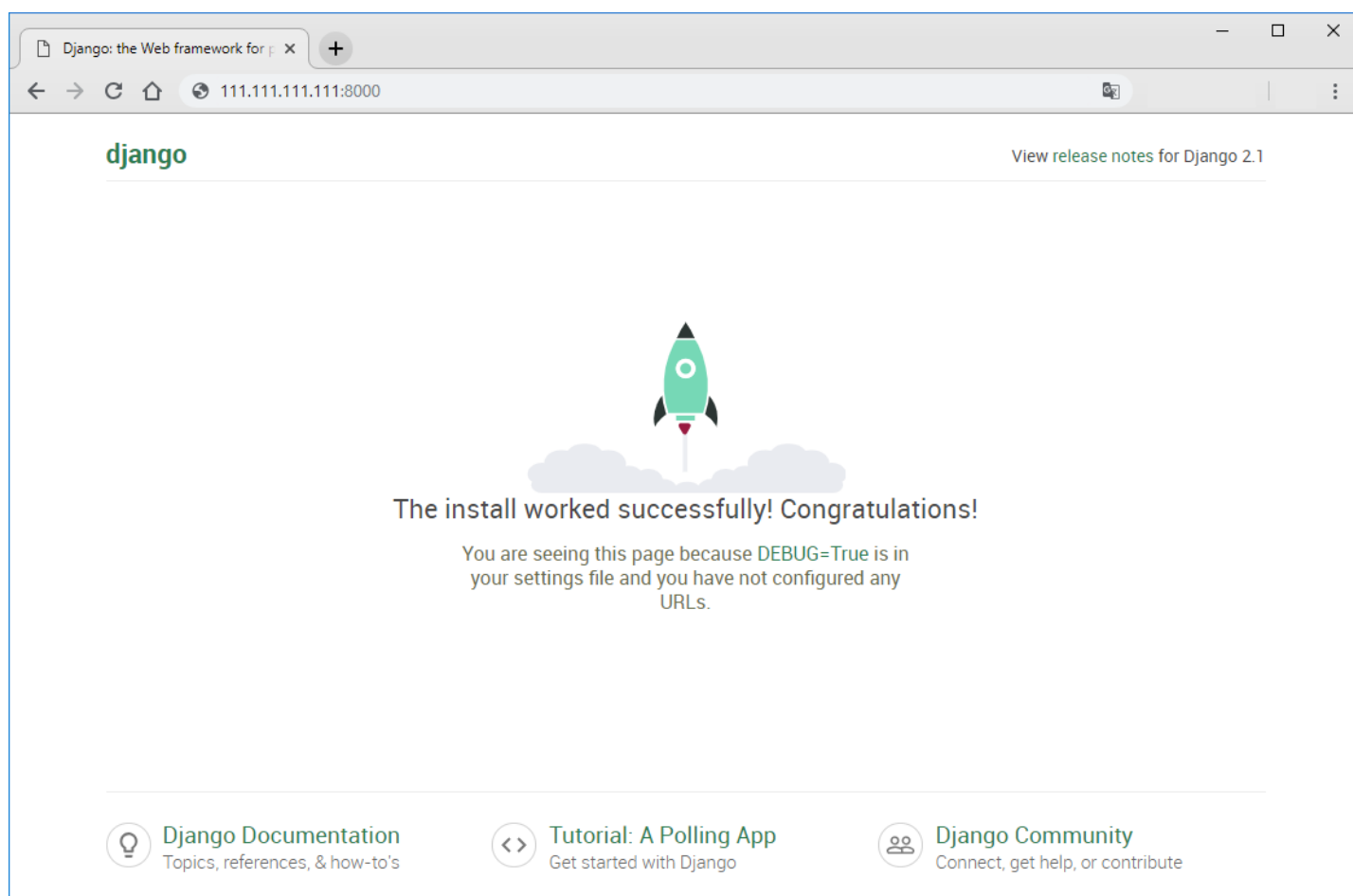
```
~/cloudproject/manage.py runserver 0.0.0.0:8000
```

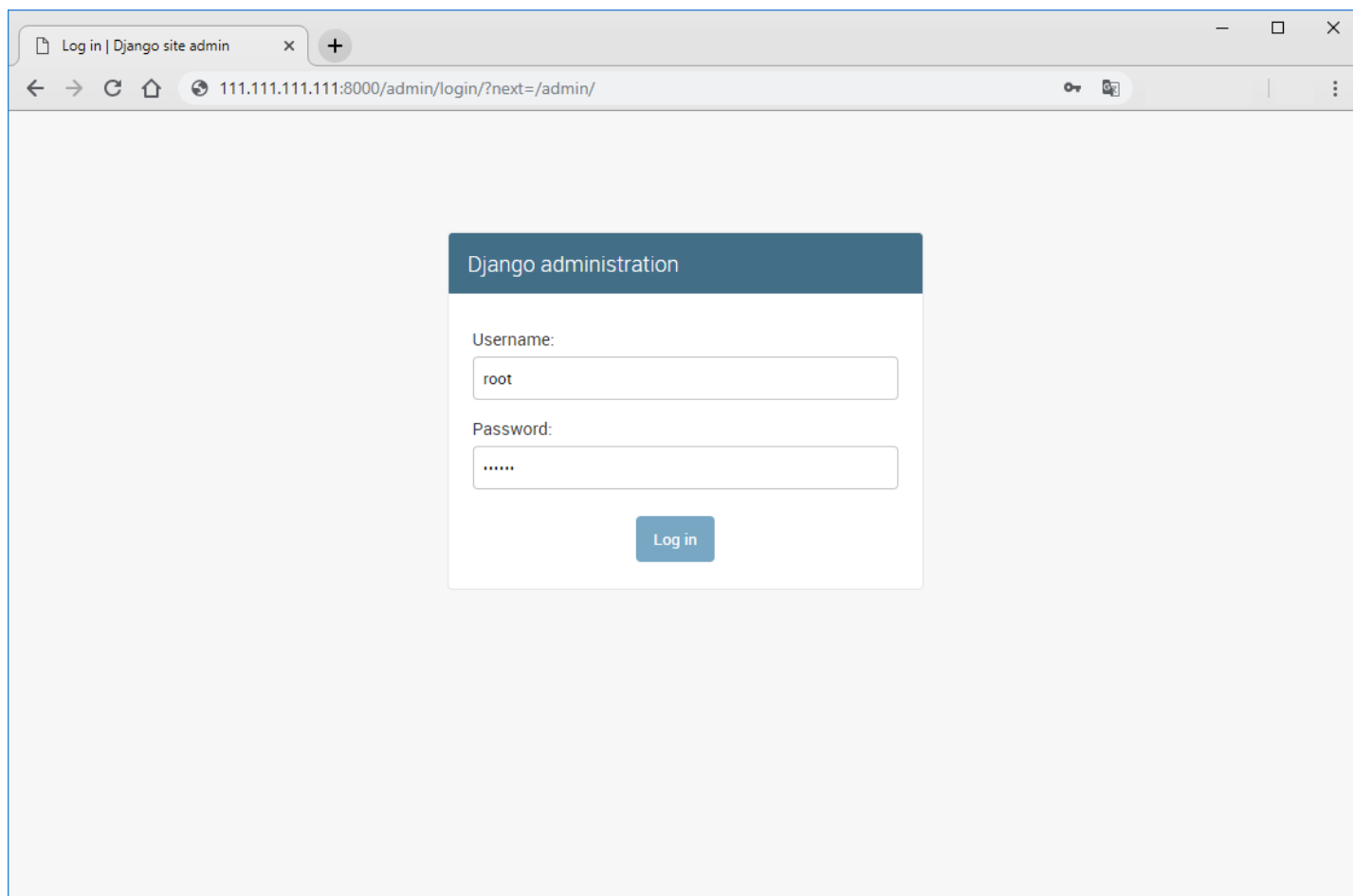
В любом удобном веб-браузере перейдите на ваше доменное имя или IP-адрес с указанием порта 8000:

*[http://server\\_domain\\_or\\_IP:8000](http://server_domain_or_IP:8000)*

Например:

*<http://111.111.111.111:8000>*





The screenshot shows a web browser window with the title "Log in | Django site admin". The address bar displays the URL "111.111.111.111:8000/admin/login/?next=/admin/". The main content area features a "Django administration" login form. The form has a dark blue header with the text "Django administration". Below the header, there are two input fields: "Username:" with the value "root" and "Password:" with masked characters "\*\*\*\*\*". A blue "Log in" button is positioned below the password field.

Log in | Django site admin

111.111.111.111:8000/admin/login/?next=/admin/

Django administration

Username:

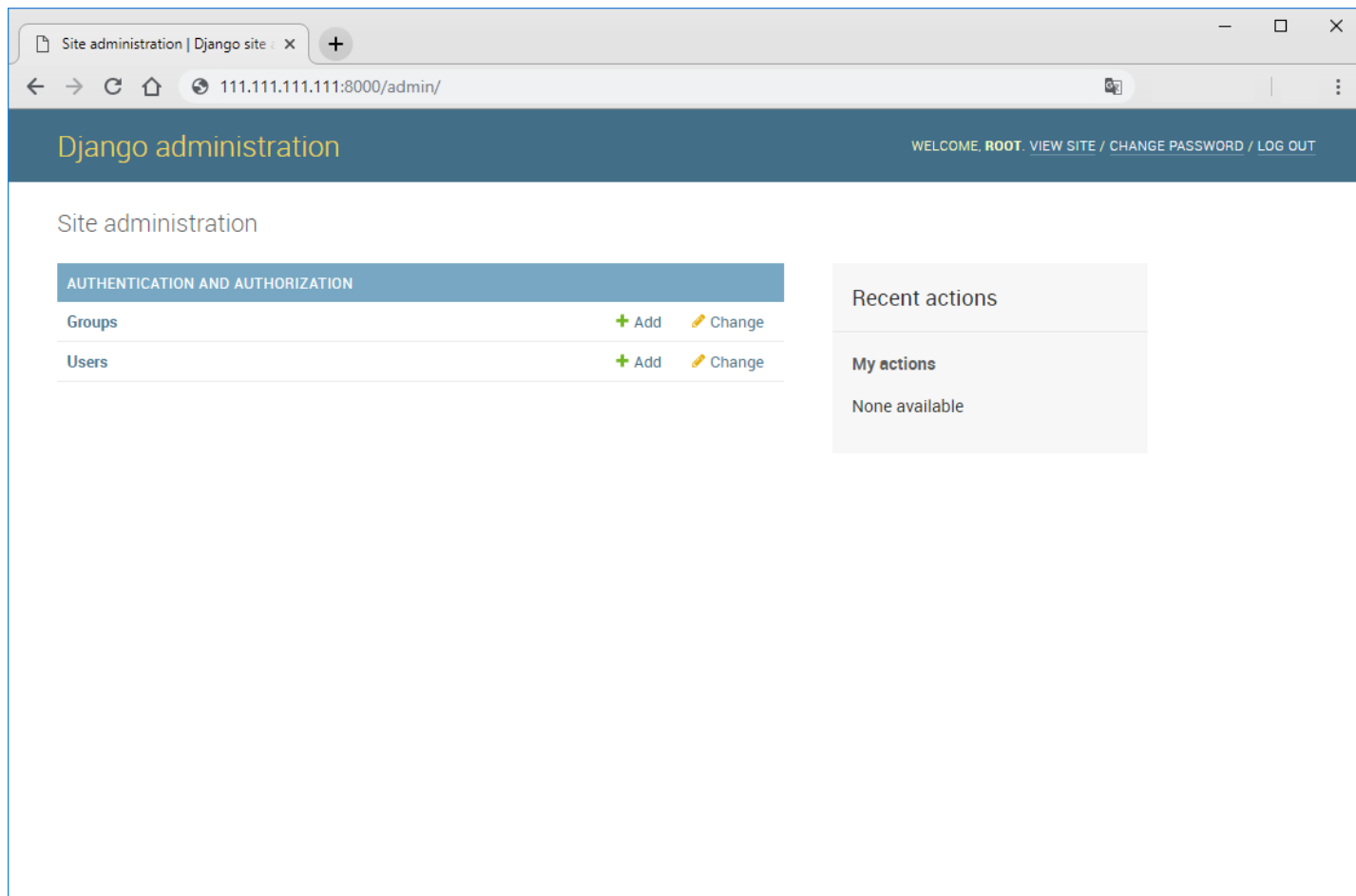
root

Password:

\*\*\*\*\*

Log in





## Тестирование работы Gunicorn

Перед тем, как покинуть виртуальную среду, необходимо протестировать Gunicorn, чтобы убедиться, что программа может обслуживать приложение. Это можно сделать, перейдя в каталог проекта и используя gunicorn для загрузки модуля WSGI:

```
cd ~/cloudproject
gunicorn --bind 0.0.0.0:8000 cloudproject.wsgi
```

В результате Gunicorn будет запущен на том же интерфейсе, на котором работал сервер разработки Django. Вы можете вернуться в браузер и снова протестировать приложение.

Теперь настройка приложения Django завершена. Необходимо выйти из виртуальной среды, набрав:

```
deactivate
```

## Создание systemd Socket и Service Files для Gunicorn

Проверка показала, что Gunicorn может взаимодействовать с приложением Django, но необходимо реализовать более надежный способ запуска и остановки сервера приложений. Для этого нужно использовать файл службы systemd.

Создайте и откройте служебный файл systemd для Gunicorn в текстовом редакторе, например nano:

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

Вставьте следующие строки:

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

[Service]
User=root
Group=www-data
WorkingDirectory=/root/cloudproject
ExecStart=/root/cloudproject/cloudenv/bin/gunicorn --access-logfile - --
workers 3 --bind unix:/root/myproject/myproject.sock
myproject.wsgi:application

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

Раздел [Unit] используется для указания метаданных и зависимостей. Здесь находится описание сервиса и информация для системы инициализации. В разделе [Service] определяется пользователь и группа, от имени которых необходимо выполнять работу. Группа www-data используется не случайно, чтобы Nginx мог

легко общаться с Gunicorn. Затем в WorkingDirectory указывается рабочий каталог и в ExecStart команда для запуска сервиса. Наконец, раздел [Install] указывает, чтобы служба стартовала, когда обычная многопользовательская система запущена:

Теперь можно запустить созданную службу Gunicorn и включить ее так, чтобы она запускалась при загрузке:

```
sudo systemctl start gunicorn  
sudo systemctl enable gunicorn
```

## Проверка Gunicorn Socket File

Для начала необходимо проверить состояние процесса, чтобы узнать, удалось ли его запустить:

```
sudo systemctl status gunicorn
```

Затем проверьте наличие файла cloudproject.sock в каталоге вашего проекта:

```
ls /root/cloudproject
```

```
Output: manage.py  cloudproject  cloudenv  cloudproject.sock  static
```

Если вы не нашли файл cloudproject.sock в каталоге, это означает, что Gunicorn не смог правильно запуститься. Проверьте журналы процесса Gunicorn, набрав:

```
sudo journalctl -u gunicorn
```

## Настройка Nginx

Теперь, когда Gunicorn настроен, нужно настроить Nginx для передачи трафика процессу. Создайте и откройте новый файл в каталоге сайтов Nginx:

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

Внутри создайте блок сервера, вставив следующие строки:

```
server {  
    listen 80;  
    server_name ;  
  
    location = /favicon.ico { access_log off; log_not_found off; }  
    location /static/ {  
        root /root/cloudproject;  
    }  
  
    location / {  
        include proxy_params;  
        proxy_pass http://unix:/root/cloudproject/cloudproject.sock;  
    }  
}
```

В блоке указано, что nginx должен прослушивать порт 80 и отвечать на доменное имя или IP-адрес сервера. Далее указано игнорировать любые проблемы с поиском значка и путь до статических ресурсов. Последний блок `location / {}` соответствует всем другим запросам.

Сохраните и закройте файл после внесения изменений. Теперь мы можем включить файл, связав его с каталогом сайтов:

```
sudo ln -s /etc/nginx/sites-available/cloudproject /etc/nginx/sites-enabled
```

Далее необходимо проверить конфигурацию Nginx на наличие синтаксических ошибок, набрав:

```
sudo nginx -t
```

Если ошибки не найдены, перезапустите Nginx, набрав:

```
sudo systemctl restart nginx
```

Наконец, нужно открыть брандмауэр для нормального трафика через порт 80. Поскольку доступ к серверу разработки больше не нужен, нужно удалить правило, которое открывает порт 8000:

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

В этом руководстве создается проект Django в собственной виртуальной среде. Мы настроили Gunicorn для перевода клиентских запросов, чтобы Django мог их обработать. После этого мы настроили Nginx в качестве обратного прокси-сервера для обработки клиентских подключений и предоставления правильного проекта в зависимости от запроса клиента.