

# Развертывание проекта на сервере

После того, как мы разработали локально большую часть функционала нашего магазина, пришло время развернуть его на внешнем сервере для демонстрации и работы в реальном режиме.

## Подготовка проекта локально

### Файл окружения requirements.txt

Для развертывания проекта на сервере, нам необходимо знать, какие модули необходимо установить для корректной работы всего проекта. Если проект уже работал с использованием виртуального окружения, то можно просто взять из него список установленных модулей с помощью команды:

```
pip3 freeze > requirements.txt
```

При работе в \*nix системах используется pip3, в Windows, скорее всего будет использован просто pip, так как это будет псевдоним для pip3.

Также, необходимо добавить модуль **psycopg2-binary**, для работы с СУБД PostgreSQL.

В случае, если разработка велась без виртуального окружения, файл requirements.txt можно заполнить вручную следующими модулями, которые мы использовали ранее:

```
psycopg2-binary
pillow
django
social_auth_app_django
requests
```

## Смена СУБД

Первым делом необходимо внести некоторые корректировки в наш код, чтобы можно было его перенести на сервер. Для начала изменим конфигурацию. Так как мы будем на сервере использовать СУБД PostgreSQL, а не файловую, как мы делали до этого, то первым делом нам необходимо изменить настройки проекта и прописать подключение к новой базе данных:

```
...
DATABASES = {
    # 'default': {
    #     'ENGINE': 'django.db.backends.sqlite3',
    #     'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    # },

    'default': {
        'NAME': 'geekshop',
        'ENGINE': 'django.db.backends.postgresql',
```

```
'USER': 'postgres',  
}  
}...
```

Для подключения к базе данных мы будем использовать стандартного пользователя, который создается по-умолчанию при установке PostgreSQL, строки со старыми настройками лучше не удалять, а закомментировать, чтобы была возможность быстро вернуться к старой базе данных для доработок локально.

## Отключаем режим отладки

При отключении режима отладки:

```
...  
DEBUG = False  
...
```

Проекту необходимо указать доступные для входа адреса, в данном случае мы разрешаем вход со всех адресов, так как пока нам неизвестно, какой у нас будет адрес у сервера, если только, для проекта не зарезервирован отдельный домен.

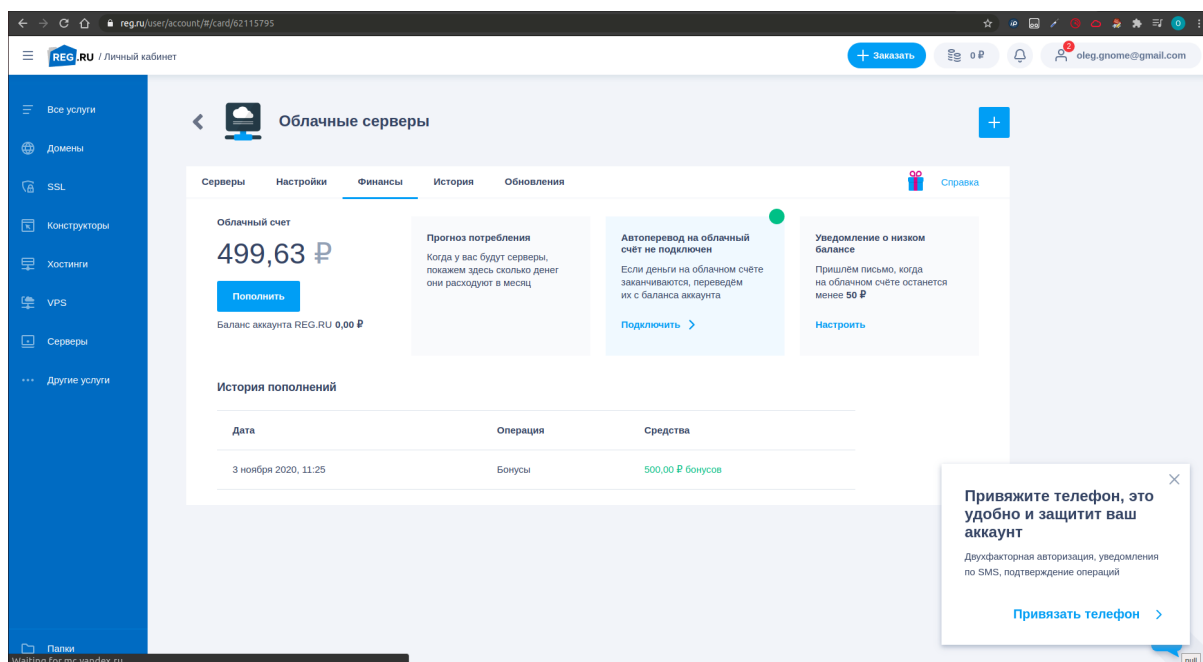
```
...  
ALLOWED_HOSTS = ['*']  
...
```

# Подготовка сервера Reg.ru

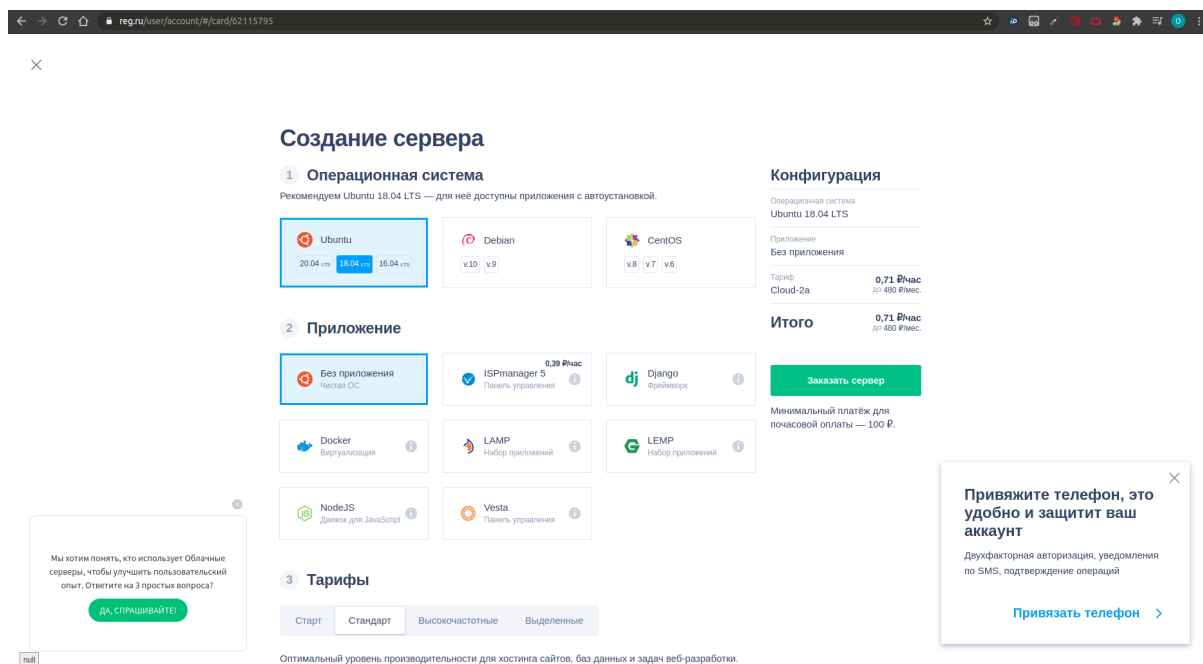
## Регистрация и аренда сервера на Reg.ru

Первым делом, необходимо зарегистрироваться на сайте [Reg.ru](https://reg.ru).

В материалах к уроку вы сможете найти ссылку на получение бонусных рублей для аренды VPS. При переходе по ней, вам будет начислен бонус, который вы сможете найти в личном кабинете перейдя в пункт меню VPS, на вкладке Финансы:



Вернувшись на вкладку Сервисы, вы сможете создать сервер, рекомендуемая конфигурация это Ubuntu 20.04, без приложений, с тарифом Start-1:



3 Тарифы

СтартСтандартВысокочастотныеВыделенные

VPS начального уровня для тестирования, разработки и хостинга сайтов. Производительность до 20% ниже, чем на тарифе Стандарт.

Скорость	Название	Диск	Память	Процессор	Стоимость
	Start-0	5 ГБ	512 МБ	1 ядро	0,32 <small>₽/час</small> <small>₽/ 215 <small>₽/мес.</small></small>
	Start-1	10 ГБ	1 ГБ	1 ядро	0,37 <small>₽/час</small> <small>₽/ 248 <small>₽/мес.</small></small>
	Start-2	15 ГБ	2 ГБ	2 ядра	0,88 <small>₽/час</small> <small>₽/ 590 <small>₽/мес.</small></small>
	Start-3	25 ГБ	4 ГБ	2 ядра	1,47 <small>₽/час</small> <small>₽/ 990 <small>₽/мес.</small></small>

4 Дополнительно

☐ Бэкапы

Создание бэкапа раз в неделю — всего 4 снимка, каждый будет храниться 4 недели. Первый бэкап будет создан ближайшей ночью.

0,07 ₽/час  
₽/ 50 ₽/мес.

5 Настройки

SSH-ключ

Добавляется при создании сервера

Новый SSH-ключ

Конфигурация

Операционная система

Ubuntu 18.04 LTS

Приложение

Без приложения

Тариф

Start-1

0,37 ₽/час  
₽/ 248 ₽/мес.

Итого

0,37 ₽/час  
₽/ 248 ₽/мес.

Заказать сервер

Минимальный платёж для почасовой оплаты — 100 ₽.

Мы хотим понять, кто использует Облачные серверы, чтобы улучшить пользовательский опыт. Ответьте на 3 простых вопроса?

ДА, СПРАШИВАЙТЕ!

Привяжите телефон, это удобно и защитит ваш аккаунт

Двухфакторная авторизация, уведомления по SMS, подтверждение операций

Привязать телефон >

После заказа, через некоторое время(в среднем от 2 до 5 минут) страница сама обновится, и появится информация по доступам к серверу, а также вся информация придет на почту, на которую производилась регистрация.

REG.RU / Личный кабинет

Все услугиДоменыSSLКонструкторыХостингиVPSСерверыДругие услуги

Облачные серверы

СерверыНастройкиФинансыИсторияОбновления

Быстрый доступ к Saffron Selenium отправлен на почту oleg.gnome@gmail.com

Подключение SSH

151.248.117.226

Пароль SSH

Eecooageiht7a

Создать сервер

Привяжите телефон, это удобно и защитит ваш аккаунт

Двухфакторная авторизация, уведомления по SMS, подтверждение операций

Привязать телефон >

## Доступ к северу

Для пользователей Windows есть возможность использовать \*nix-like ssh клиент через PowerShell, который представляет из себя обычную командную строку и имеет набор команд для работы с ssh, либо можно воспользоваться любым другим ssh клиентом, например довольно популярным Putty.

Для пользователей \*nix систем, клиент ssh встроен в обычный терминал, поэтому никаких дополнительных действий совершать не нужно и ничего устанавливать дополнительно не требуется.

Из прошлого пункта нам известны ip-адрес и пароль, с помощью которых можно попасть на сервер для работы на нем. Пользователь в данном случае будет по-умолчанию **root**.

Строка подключения из терминала выглядит следующим образом(на примере из прошлого пункта, ip-адрес необходимо будет заменить на тот, который будет указан после создания сервера):

```
ssh root@151.248.117.226
```

Если только не был использован сторонний клиент для подключения по ssh, в таком случае, необходимо следовать инструкциям данного клиента, чтобы подключиться корректно.

При использовании терминала и встроенного в него ssh клиента, доступен доступ без пароля, его можно задать с помощью команды:

```
ssh-copy-id root@151.248.117.226
```

После ввода команды, необходимо будет ввести пароль и проследовать указаниям, после этого, предыдущая команда не будет запрашивать пароль для доступа к серверу.

На сервере также необходим получить ключ, который нам понадобится в дальнейшем для копирования исходного кода из репозитория, находясь на сервере необходимо ввести команду:

```
ssh-keygen
```

Для генерации ключа, который можно получить с помощью команды:

```
cat /root/.ssh/id_rsa.pub
```

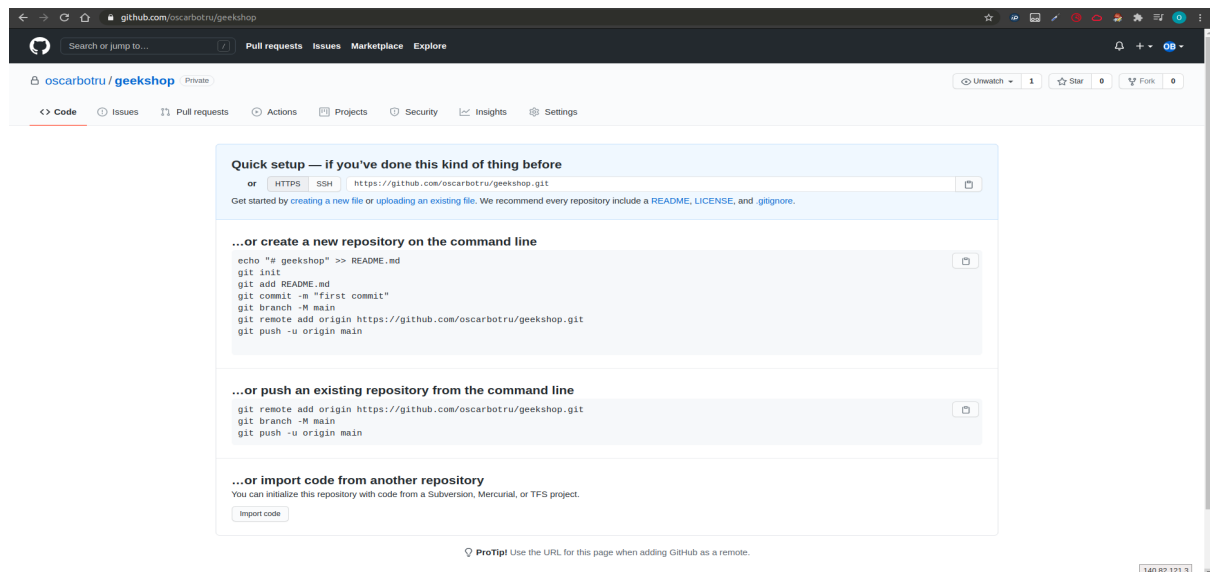
Полученный ключ необходимо сохранить, он нам пригодится на следующем шаге.

## Подготовка репозитория

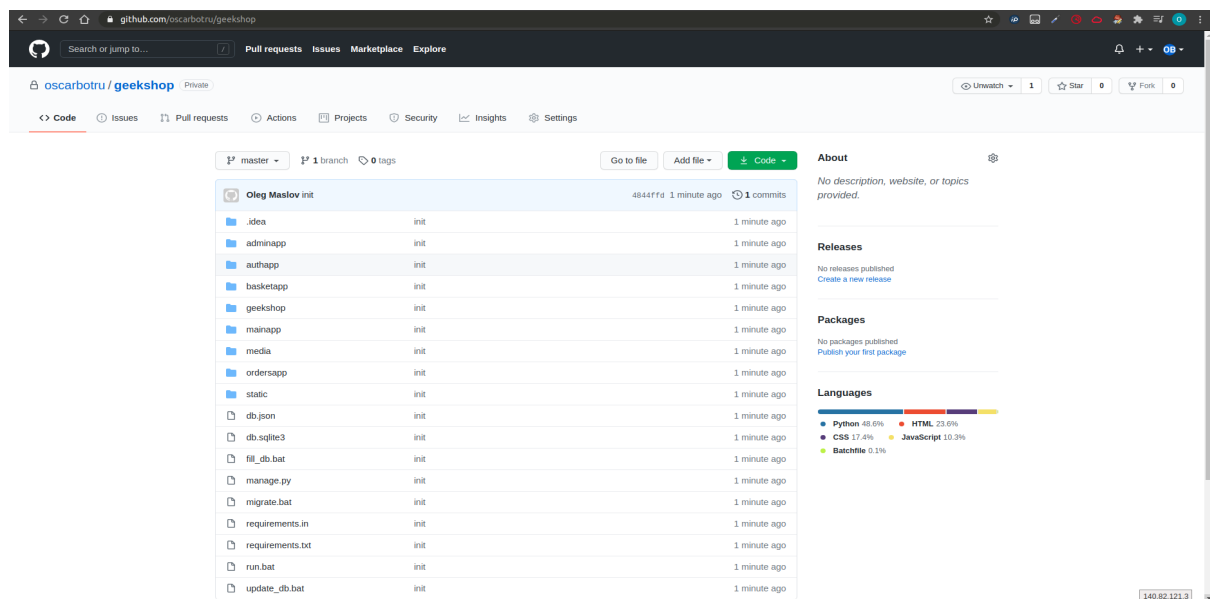
Копировать исходный код можно через репозиторий на github.com. Для этого необходимо загрузить исходный код в репозиторий. Если домашние задания сдаются через гит, то данный шаг уже готов, если через архивы и у вас установлен PyCharm, то на сайте jetbrains.com есть [инструкция](#), в которой описаны самые частые случаи, которые необходимы для работы с git репозиторием из данной IDE.

В случае, если используется другой редактор кода, и исходный код хранится не в репозитории, то необходимо создать новый репозиторий на сайте github.com (или по желанию можно

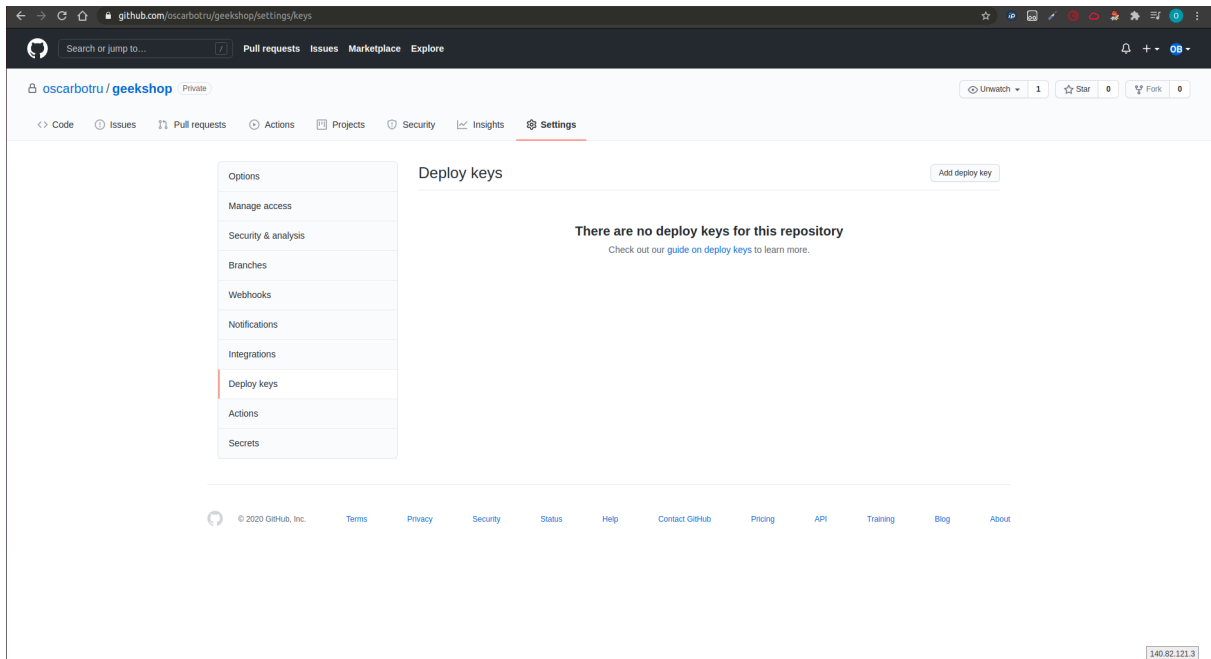
воспользоваться другими альтернативами), после того, как пустой репозиторий будет создан, отобразится инструкция по добавлению нового проекта и существующего кода(например github.com):



Необходимо воспользоваться второй инструкцией для работы с существующим кодом, и загрузить весь код в репозиторий, после загрузки, на этой же странице должен отобразиться корень проекта со всеми файлами и папками:



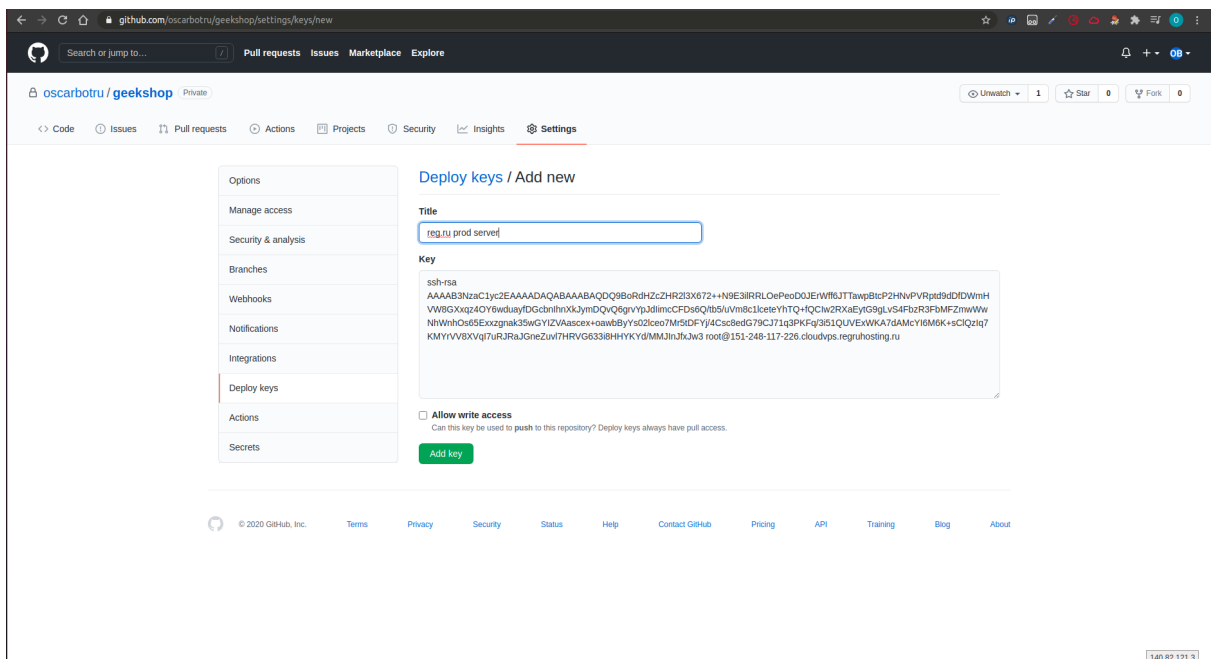
На вкладке настроек необходимо открыть пункт Deploy keys:



И в нее добавить ключ, который мы получили в прошлом пункте с помощью команды:

```
cat /root/.ssh/id_rsa.pub
```

После нажатия на кнопку добавления нового ключа, справа сверху, появится интерфейс добавления нового ключа, в большое поле необходимо вставить ключ, а в маленьком поле задать ему название:



# Развертывание проекта на сервере

## Установка необходимого ПО

Перед установкой любых пакетов, необходимо выполнить команду:

```
apt update
```

которая обновит информацию о репозиториях и появится возможность устанавливать более новые пакеты.

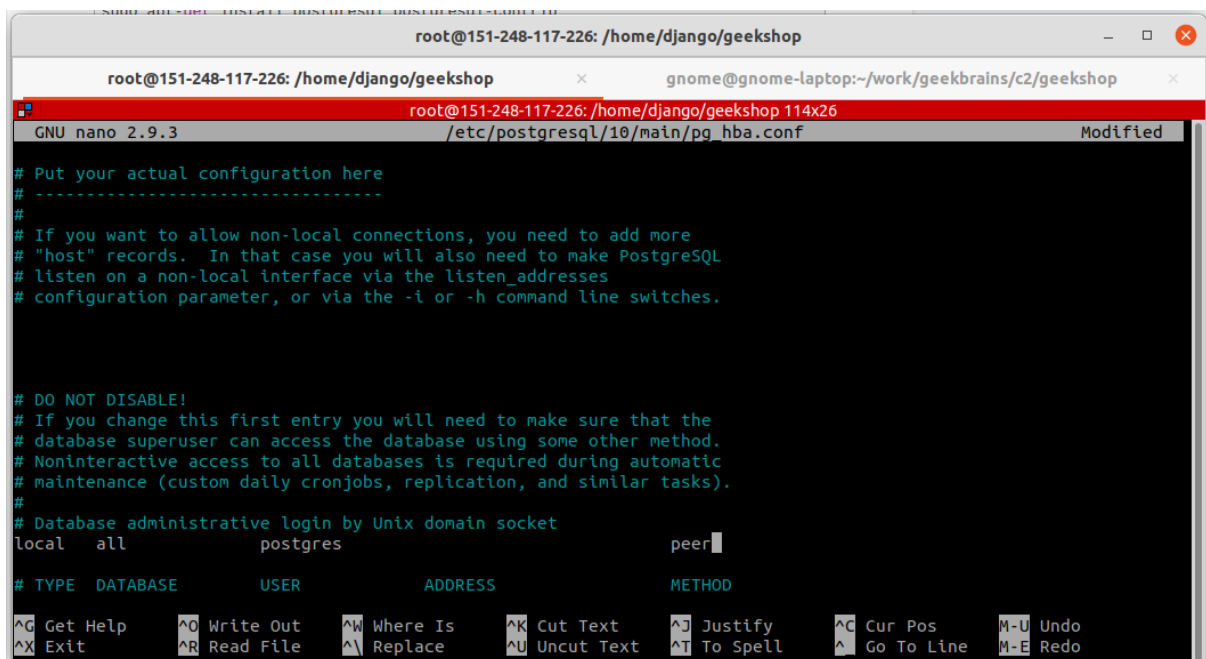
Нам понадобится веб сервер, в данном случае это будет nginx, СУБД PostgreSQL, Git и пакет для работы с виртуальными окружениями python:

```
apt install nginx
apt install postgresql postgresql-contrib
apt install python3-venv
apt install git-core
```

Для PostgreSQL необходимо внести изменения в стандартную конфигурацию, чтобы можно было использовать стандартного пользователя. Для этого откроем файл конфигурации работы с пользователями(путь может отличаться из-за версии, в данном случае установлена 10 версия):

```
nano /etc/postgresql/10/main/pg_hba.conf
```

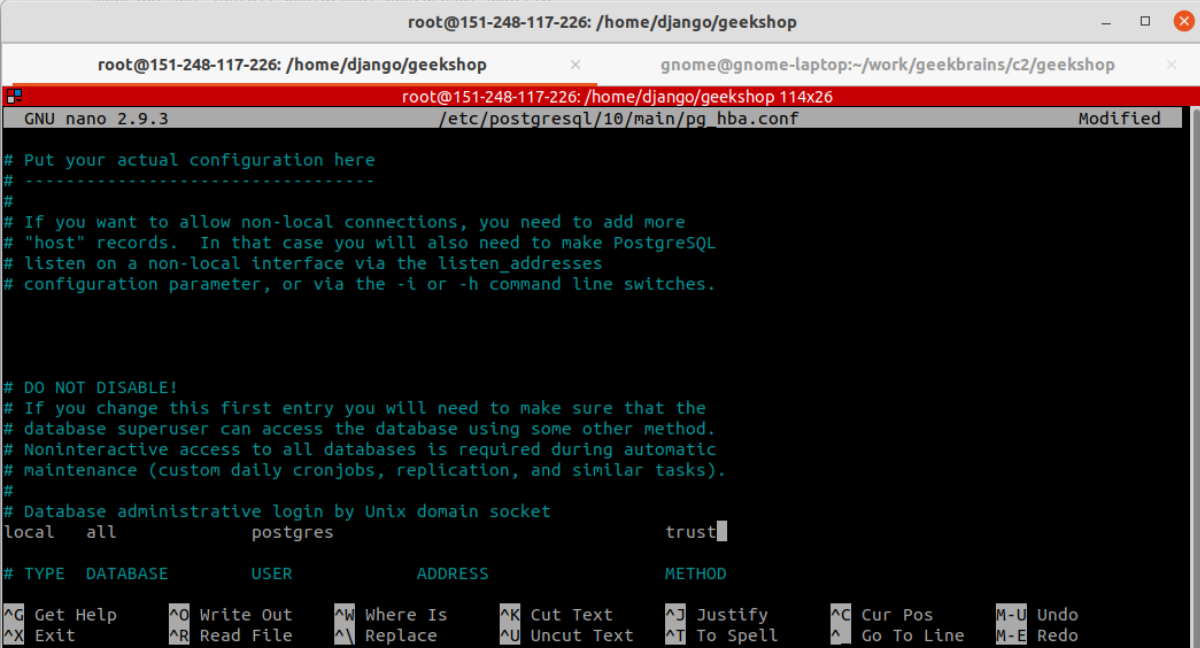
В этом файле необходимо найти строку, в которой указан пользователь postgres, хост для доступа local, как показано на ниже:



```
root@151-248-117-226: /home/django/geekshop
root@151-248-117-226: /home/django/geekshop x gnome@gnome-laptop: ~/work/geekbrains/c2/geekshop x
GNU nano 2.9.3 /etc/postgresql/10/main/pg_hba.conf Modified
# Put your actual configuration here
# -----
#
# If you want to allow non-local connections, you need to add more
# "host" records. In that case you will also need to make PostgreSQL
# listen on a non-local interface via the listen_addresses
# configuration parameter, or via the -i or -h command line switches.
#
# DO NOT DISABLE!
# If you change this first entry you will need to make sure that the
# database superuser can access the database using some other method.
# Noninteractive access to all databases is required during automatic
# maintenance (custom daily cronjobs, replication, and similar tasks).
#
# Database administrative login by Unix domain socket
local all postgres peer
# TYPE DATABASE USER ADDRESS METHOD
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos M-U Undo
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line M-E Redo
```



И заменить peer в последнем столбце на trust



```
root@151-248-117-226: /home/django/geekshop
root@151-248-117-226: /home/django/geekshop  gnome@gnome-laptop: ~/work/geekbrains/c2/geekshop
GNU nano 2.9.3 /etc/postgresql/10/main/pg_hba.conf Modified
# Put your actual configuration here
# -----
#
# If you want to allow non-local connections, you need to add more
# "host" records.  In that case you will also need to make PostgreSQL
# listen on a non-local interface via the listen_addresses
# configuration parameter, or via the -i or -h command line switches.
#
# DO NOT DISABLE!
# If you change this first entry you will need to make sure that the
# database superuser can access the database using some other method.
# Noninteractive access to all databases is required during automatic
# maintenance (custom daily cronjobs, replication, and similar tasks).
#
# Database administrative login by Unix domain socket
local all postgres trust
# TYPE DATABASE USER ADDRESS METHOD
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos M-U Undo
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line M-E Redo
```

Что даст возможность работать с этим пользователем без пароля, только с локального хоста, т.е. только внутри нашего сервера.

Необходимо сохранить изменения, выйти из редактора и перезапустить PostgreSQL:

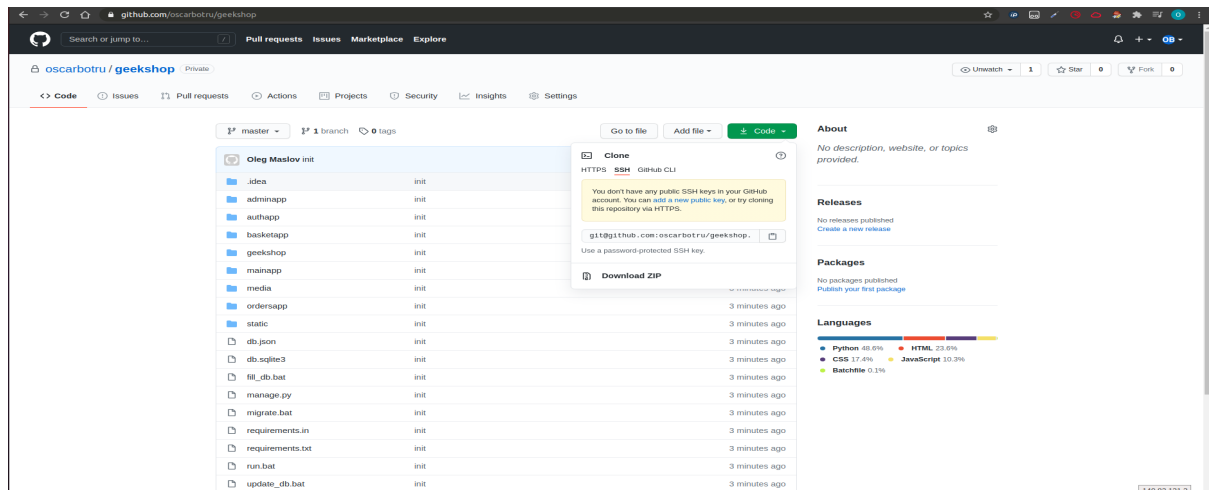
```
systemctl restart postgresql
```

## Копирование исходного кода на сервер

Обычно, проекты на чистом html, либо на php, размещают в папке **/var/www/html**, но это не безопасно в случае с Django проектом, потому что если настроить веб сервер некорректно, то файлы проекта могут оказаться доступными для скачивания, поэтому, чтобы избежать такого, создаем пользователя в группе **www-data**, которая по-умолчанию создана для работы с веб сервером:

```
useradd -g www-data -m django
cd /home/django/
```

Ссылку на исходный код мы можем получить в интерфейсе репозитория, необходимо скопировать ссылку для доступа по ssh, именно с этой настройкой и установленным Deploy key у нас не будет каждый раз запрашиваться пароль для копирования изменений из репозитория



Полученную ссылку мы вставляем в терминале, не забывая дописать команду клонирования, и переходим в папке проекта для дальнейших настроек:

```
git clone git@github.com:oscarbotru/geekshop.git
cd geekshop/
```

## Настройка окружения

Первым делом, находясь в папке с проектом, необходимо создать окружение с помощью команды:

```
python3 -m venv env
```

После этого в корне проекта появится папка **env/**, с помощью которой мы сможем активировать виртуальное окружение и установить все необходимые пакеты:

```
source env/bin/activate
pip3 install -r requirements.txt
```

После завершения установки всех пакетов, необходимо дополнительно установить еще один пакет, который позволит нам запустить наш проект в связке с nginx:

```
pip3 install gunicorn
```

Теперь можно убедиться, что наш проект работает. Первым делом необходимо применить все миграции, которые у нас есть в проекте, и после этого заполнить базу данных данными (ранее мы разрабатывали скрипт **fill\_db.py**), и проверить, что сервер разработки запускается без ошибок:

```
python3 manage.py migrate
python3 manage.py fill_db
```

```
python3 manage.py runserver
```

В случае, если все работает, то будет показано обычное сообщение о том, что сервер запустился, но пока он будет доступен извне, для этого нам необходимо настроить nginx и gunicorn.

Чтобы сервер корректно работал, на файлы проекта необходимо назначить следующие права для доступа:

```
chown -R django /home/django/  
chmod -R 755 /home/django/geekshop/
```

## Настройка веб-сервера

Первым делом необходимо настроить gunicorn, потому что nginx будет обращаться к сокету, который будет создан после старта сервиса. Для создания службы, необходимо создать файл и заполнить его:

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

```
[Unit]  
Description=gunicorn daemon  
After=network.target  
  
[Service]  
User=django  
Group=www-data  
WorkingDirectory=/home/django/geekshop  
ExecStart=/home/django/geekshop/env/bin/gunicorn --access-logfile - --workers 3 --bind  
unix:/home/django/geekshop/geekshop.sock geekshop.wsgi  
  
[Install]  
WantedBy=multi-user.target
```

Тут стоит обратить внимание на пути, необходимо следить за тем, чтобы пути и названия проекта и внутренних папок полностью совпадали.

Файл необходимо сохранить и выйти из редактора. Теперь у нас есть конфигурация для сервиса, его необходимо активировать и запустить:

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

Последняя команда покажет статус сервиса, если сервис не запустился, то необходимо вернуться на прошлый шаг и проверить все пути очень внимательно.

После того, как сервис успешно запустился, можно настроить параметры для nginx, для этого необходимо создать новый файл и внести в него конфигурацию:

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

```
server {
    listen 80;
    server_name 151.248.117.226;

    location = /favicon.ico { access_log off; log_not_found off; }
    location /static/ {
        root /home/django/geekshop;
    }

    location /media/ {
        root /home/django/geekshop;
    }

    location / {
        include proxy_params;
        proxy_pass http://unix:/home/django/geekshop/geekshop.sock;
    }
}
```

В **server\_name** необходимо написать ip-адрес сервера, на которой проводятся все работы. После сохранения и выхода из редактора, можно активировать сайт с помощью команды:

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

Проверяем настройки «nginx»:

```
sudo nginx -t
```

Перезапускаем службу «nginx» и добавляем разрешения в сетевой экран:

```
sudo systemctl restart nginx
```

После этого в браузере можно ввести ip-адрес сервера и откроется проект.