- После создания сервера, подключаемся к нему:
ssh root@ip

- Создаем нового юзера (от рута может возникать много проблем с безопасностью)
adduser -username-

- Даём права суперпользователя только что созданному пользователю
usermod -aG sudo -username-

- Логинимся под вновь созданным пользователем
su -username-

- Переходим в корневую папку
cd ~

- Обновляем список пакетов (тут ничего не скачивается)
sudo apt update

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

psql -U postgres

создаем пользователя
create user testuser with password '123456';

создаем базу данных
create database db_name with owner testuser;

Мы зададим кодировку по умолчанию `UTF-8`, чего и ожидает Django. Также мы зададим схему изоляции транзакций по умолчанию «read committed», которая будет блокировать чтение со стороны неподтвержденных транзакций. В заключение мы зададим часовой пояс. По умолчанию наши проекты Django настроены на использование времени по Гринвичу (`UTC`). Все эти рекомендации взяты из проекта Django:

- *ALTER ROLE myprojectuser SET default_transaction_isolation TO 'read committed';*
- *ALTER ROLE myprojectuser SET timezone TO 'UTC';*
- *ALTER ROLE myprojectuser SET client_encoding TO 'utf8';*

- Настройка БД завершена

- Обновляем pip
sudo pip install -U pip

- Устанавливаем модули для виртуального окружения
sudo apt install python3.8-venv

- Формируем проект

mkdir project/

- Копируем проект с GitHub
git clone -ссылка на проект-

- Переходим в каталог проекта
cd -название проекта-

- Создаем виртуальное окружение
python3 -m venv .venv

- Активируем виртуальное окружение
source .venv/bin/activate или . .venv/bin/activate

pip install gunicorn

!!! Python должен работь из виртуального окружения, для того, чтобы не было конфликтов между разными проектами из-за различных версий ПО.
deactivate - выйти из виртуального окружения !!!

- Устанавливаем pip
sudo apt install pip

- Из файла requirements.txt устанавливаем все необходимые пакеты
sudo pip install -r requirements.txt

- Вносим изменения в settings.py
nano settings.py
Прописываем USER, PASSWORD, HOST
ТАМ ЖЕ ДОБАВЛЯЕМ STATIC_ROOT
STATIC_ROOT = os.path.join(BASE_DIR, 'static/')

-Дальше применяем миграции

-Не забываем создать суперпользователя (админа)
python3 manage.py createsuperuser

- Собираем статичный контент
python3 manage.py collectstatic

- Чтобы протестировать сервер разработки, необходимо разрешить доступ к порту
sudo ufw allow 8000
python3 manage.py runserver 0.0.0.0:8000

- Перед выходом из виртуальной среды нужно протестировать способность Gunicorn обслуживать приложение
gunicorn --bind 0.0.0.0:8000 northbasketball.wsgi

- Завершили настройку нашего приложения Django. Теперь мы можем выйти из виртуальной среды.

deactivate

- Теперь нам нужно реализовать более надежный способ запуска и остановки сервера приложений. Для этого мы создадим служебные файлы и файлы сокета systemd. Сокет Gunicorn создается при загрузке и прослушивает подключения. При подключении systemd автоматически запускает процесс Gunicorn для обработки подключения.

deactivate

- Создаём и открываем файл сокета systemd для Gunicorn с привилегиями `sudo`:

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

- В этом файле мы создадим раздел [Unit] для описания сокета, раздел [Socket] для определения расположения сокета и раздел [Install], чтобы обеспечить установку сокета в нужное время:

```
[Unit]
Description=gunicorn socket

[Socket]
ListenStream=/run/gunicorn.sock

[Install]
WantedBy=sockets.target
```

- Создаём и открываем служебный файл для Gunicorn с привилегиями `sudo`:

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

```
[Unit]
Description=gunicorn daemon
Requires=gunicorn.socket
```

```
After=network.target

[Service]
User=sammy
Group=www-data
WorkingDirectory=/home/sammy/myprojectdir
ExecStart=/home/sammy/myprojectdir/myprojectenv/bin/gunicorn \
        --access-logfile - \
        --workers 3 \
        --bind unix:/run/gunicorn.sock \
        myproject.wsgi:application

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

- Файл сокета /run/gunicorn.sock будет создан сейчас и будет создаваться при загрузке. При подключении к этому сокету systemd автоматически запустит gunicorn.service для его обработки. Успешность операции можно подтвердить, проверив файл сокета.

```
sudo systemctl start gunicorn.socket
```

```
sudo systemctl enable gunicorn.socket
```

- Проверьте состояние процесса, чтобы узнать, удалось ли его запустить:

```
sudo systemctl status gunicorn.socket
```

- Затем проверьте наличие файла gunicorn.sock в каталоге /run:

```
file /run/gunicorn.sock
```

- Если вы запустили только gunicorn.socket, служба gunicorn.service не будет активна в связи с отсутствием подключений к совету. Для проверки можно ввести следующую команду:

sudo systemctl status gunicorn

- Чтобы протестировать механизм активации сокета, установим соединение с сокетом через curl с помощью следующей команды:

curl --unix-socket /run/gunicorn.sock localhost

curl --unix-socket /run/gunicorn.sock localhost

- Мы настроили Gunicorn, и теперь нам нужно настроить Nginx для передачи трафика в процесс.

Для начала нужно создать и открыть новый серверный блок в каталоге Nginx sites-available:

sudo nano /etc/nginx/sites-available/myproject

Внутри прописываем

```
server {
        listen 80;
        server_name server_domain_or_IP;

        location = /favicon.ico { access_log off; log_not_found off; }
        location /static/ {
                root /home/sammy/myprojectdir;
        }

        location / {
                include proxy_params;
                proxy_pass http://unix:/run/gunicorn.sock;
        }
}
```

Теперь мы можем активировать файл, привязав его к каталогу sites-enabled:

sudo ln -s /etc/nginx/sites-available/myproject /etc/nginx/sites-enabled

Протестируйте конфигурацию Nginx на ошибки синтаксиса:
sudo nginx -t

sudo systemctl restart nginx если нет ошибок

Открываем порты
sudo ufw delete allow 8000

sudo ufw allow 'Nginx Full'

- Открываем файл для редактирования настроек postgres (для предоставления прав на управление БД для пользователя postgres)
Ищем в редакторе файлов peer и меняем на trust
sudo nano /etc/postgresql/12/main/pg_hba.conf

- Перезагружаем postgres
sudo service postgresql restart **(Старый вариант)**
sudo systemctl restart postgresql **(новый вариант, с убунту 16 начался)**