Управление проектом на удалённом сервере, 1 спринт | Я.Шпора

Подключение к удалённому серверу по SSH-ключу

```
# Ключ -i позволяет указать путь до SSH-ключа, отличный от пути по
ssh -i путь_до_файла_с_SSH_ключом/название_файла_с_SSH_ключом_без_

# Например:
# ssh -i D:/Dev/vm_access/yc-yakovlevstepan yc-user@153.0.2.10
```

Клонирование проекта с GitHub на сервер

Находясь на сервере, выполните команду:

```
# Вместо <ваш_аккаунт> подставьте свой логин, который используете
# Вместо <название_репозитория> подставьте название репозитория,
# который хотите клонировать.
git clone git@github.com:ваш_аккаунт/название_репозитория.git
# Например:
# git clone git@github.com:yandex-practicum/project.git
```

Настройка бэкенд-приложения

1. Установите зависимости из файла *requirements.txt:*

```
# Перейдите в директорию бэкенд-приложения проекта.
# Вместо <название_проекта> укажите название проекта, с которым ра
cd название_проекта/backend/
# Создайте виртуальное окружение.
python3 -m venv venv
# Активируйте виртуальное окружение.
source venv/bin/activate
```

```
# Установите зависимости.
pip install -r requirements.txt
```

2. Выполните миграции и создайте суперюзера из директории, в которой находится файл *manage.py:*

```
# Примените миграции.
python3 manage.py migrate
# Создайте суперпользователя.
python3 manage.py createsuperuser
```

Когда вы вызываете интерпретатор Python на устройстве или сервере под управлением ОС семейства Linux, **находясь в виртуальном окружении**, версию интерпретатора указывать не обязательно. Можно использовать команду `python`.

3. Добавьте в список `ALLOWED_HOSTS` внешний IP-адрес вашего сервера, адрес `127.0.0.1`, `localhost` и домен:

```
# Вместо xxx.xxx.xxx.xxx укажите IP вашего сервера,
# а вместо ваш_домен - ваше доменное имя для этого проекта.
ALLOWED_HOSTS = ['xxx.xxx.xxx.xxx', '127.0.0.1', 'localhost', 'ваш
```

4. Отключите режим дебага для бэкенд-приложения. В файле *settings.py* установите для константы `DEBUG` значение `False`, сохраните и закройте файл настроек:

```
from pathlib import Path

...

# Вот тут нужно поменять значение с True на False.
DEBUG = False
...
```

5. Подготовьте бэкенд-приложение для сбора статики.

В файле *settings.py* укажите директорию, куда эту статику нужно сложить.

Через редактор *Nano* откройте файл *settings.py*, укажите новое значение для константы `STATIC_URL`, добавьте константу `STATIC_ROOT` и сохраните изменения в файле:

```
# Замените стандартное значение 'static' на 'static_backend',
# чтобы не было конфликта запросов к статике фронтенда и бэкенда.
STATIC_URL = 'static_backend'
# Укажите директорию, куда бэкенд-приложение должно сложить стати
STATIC_ROOT = BASE_DIR / 'static_backend'
```

6. Соберите статику бэкенд-приложения:

```
python3 manage.py collectstatic
```

7. В директории с бэкенд-приложением будет создана директория *static_backend/* со всей статикой бэкенд-приложения. Скопируйте директорию *static_backend/* в директорию */var/www/название_проекта/*:

```
sudo cp -r путь_к_директории_с_бэкенд-приложением/static_backend /
```

Настройка фронтенд-приложения

1. Установите зависимости для фронтенд-приложения. Перейдите в директорию с фронтенд-приложением и выполните команду:

```
npm i
```

2. Из директории с фронтенд-приложением выполните команду:

```
npm run build
```

3. Скопируйте статику фронта в директорию по умолчанию — */var/www/название_проекта/*:

```
# Синтаксис команды.
# Точка после build важна — будет скопировано содержимое директори
sudo cp -r путь_к_директории_с_фронтенд-приложением/build/. /var/
```

Установка и настройка WSGI-сервера Gunicorn

1. Подключитесь к удалённому серверу, активируйте виртуальное окружение Django-приложения и установите пакет `gunicorn`:

```
pip install gunicorn==20.1.0
```

2. Перейдите в директорию бэкенд-приложения, в которой лежит файл *manage.py*, и запустите Gunicorn:

```
gunicorn --bind 0.0.0.0:8000 backend.wsgi
```

3. Создайте файл конфигурации юнита *systemd* для Gunicorn в директории */etc/systemd/system/.* Назовите его по шаблону `gunicorn_название_проекта.service`.

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

> Для каждого отдельного проекта на одном сервере должен быть свой файл конфигурации Gunicorn со своим названием.

4. Подставьте в код из листинга свои данные, добавьте этот код **без комментариев** в файл конфигурации Gunicorn и сохраните изменения:

```
[Unit]
# Это текстовое описание юнита, пояснение для разработчика.
Description=gunicorn daemon

# Условие: при старте операционной системы запускать процесс толь
# как операционная система загрузится и настроит подключение к се
# Ссылка на документацию с возможными вариантами значений
```

```
# https://systemd.io/NETWORK_ONLINE/
After=network.target

[Service]
# От чьего имени будет происходить запуск:
# укажите имя, под которым вы подключались к серверу.
User=имя_пользователя_в_системе

# Путь к директории проекта.
WorkingDirectory=/home/имя_пользователя_в_системе/директория_с_про

# Команду, которую вы запускали руками, теперь будет запускать sys
ExecStart=/home/имя_пользователя_в_системе/директория_с_проектом/v

[Install]
# В этом параметре указывается вариант запуска процесса.
# Значение <multi-user.target> указывает, чтобы systemd запустил п
# доступный всем пользователям и без графического интерфейса.
WantedBy=multi-user.target
```

> Чтобы точно узнать путь до Gunicorn, активируйте виртуальное окружение и воспользуйтесь командой `which gunicorn`.

Команда `sudo systemctl` с параметрами `start`, `stop` или `restart` запустит, остановит или перезапустит Gunicorn.

Например, команда запуска юнита с именем `gunicorn_название_проекта` выглядит так:

```
sudo systemctl start gunicorn_название_проекта
```

Чтобы *systemd* следил за работой демона Gunicorn, запускал его при старте системы и при необходимости перезапускал, используйте команду:

```
sudo systemctl enable gunicorn_название_проекта
```

Установка и настройка веб- и прокси-сервера Nginx

Если на вашем сервере уже установлен Nginx, сразу переходите ко второму пункту. Если нет — начните с первого пункта.

1. Установите Nginx на удалённый сервер:

```
sudo apt install nginx -y
```

2. Запустите Nginx командой:

```
sudo systemctl start nginx
```

3. Обновите настройки Nginx. Для этого откройте файл конфигурации веб-сервера…

```
 sudo nano /etc/nginx/sites-enabled/default
```

… и допишите в конец файла новый блок *server*:

```
# ...
# Тут останется конфигурация для работы проекта Taski
# А дальше - ваш новый блок server
server {

    listen 80;
    server_name ваш_домен;

    location /api/ {
        proxy_pass http://127.0.0.1:8000;
    }

    location /admin/ {
        proxy_pass http://127.0.0.1:8000;
    }

    location / {
        root   /var/www/имя_проекта;
        index  index.html index.htm;
```

```
        try_files $uri /index.html;
    }

}
```

Сохраните изменения в файле конфигурации веб-сервера, закройте файл, проверьте его на корректность:

```
sudo nginx -t
```

Перезагрузите конфигурацию Nginx:

```
sudo systemctl reload nginx
```

Настройка файрвола ufw

Файрвол установит правило, по которому будут закрыты все порты, кроме тех, которые вы явно укажете.

1. Активируйте разрешение принимать запросы только на порты 80, 443 и 22:

```
# 80, 443
sudo ufw allow 'Nginx Full'
# 22
sudo ufw allow OpenSSH
```

С портами 80 и 443 будут работать пользователи, делая запросы к приложению, запущенному на удалённом сервере. Порт 22 нужен, чтобы вы могли подключаться к серверу по SSH.

2. Включите файрвол:

```
sudo ufw enable
```

В терминале выведется запрос на подтверждение операции. Введите `y` и нажмите **Enter.**

3. Проверьте работу файрвола:

```
sudo ufw status
```

Получение и настройка SSL-сертификата

> В этом разделе описаны шаги для получения SSL-сертификата от бесплатного центра сертификации Let's Encrypt.

Если вы уже устанавливали пакет *certbot* на удалённый сервер, переходите ко второму пункту.

1. Установите *certbot*. Зайдите на сервер и последовательно выполните команды:

```
# Установка пакетного менеджера snap.
# У этого пакетного менеджера есть нужный вам пакет — certbot.
sudo apt install snapd
# Установка и обновление зависимостей для пакетного менеджера snap
sudo snap install core; sudo snap refresh core
# При успешной установке зависимостей в терминале выведется:
# core 16-2.58.2 from Canonical✓ installed

# Установка пакета certbot.
sudo snap install --classic certbot
# При успешной установке пакета в терминале выведется:
# certbot 2.3.0 from Certbot Project (certbot-eff✓) installed

# Создание ссылки на certbot в системной директории,
# чтобы у пользователя с правами администратора был доступ к этому
sudo ln -s /snap/bin/certbot /usr/bin/certbot
```

2. Запустите *certbot* и получите SSL-сертификат. Для этого выполните команду:

```
sudo certbot --nginx
```

Далее система попросит вас указать электронную почту и ответить на несколько вопросов. Сделайте это.

Следующим шагом укажите имена, для которых вы хотели бы активировать HTTPS:

```
Account registered.

Which names would you like to activate HTTPS for?
We recommend selecting either all domains, or all domains in a Vi

1: <доменное_имя_вашего_проекта_1>
2: <доменное_имя_вашего_проекта_2>
3: <доменное_имя_вашего_проекта_3>
...

Select the appropriate numbers separated by commas and/or spaces,
blank to select all options shown (Enter 'c' to cancel):
```

Введите номер нужного доменного имени и нажмите **Enter**.

После этого *certbot* отправит ваши данные на сервер *Let's Encrypt* и там будет выпущен сертификат, который автоматически сохранится на вашем сервере в системной директории */etc/ssl/*. Также будет автоматически изменена конфигурация Nginx: в файл */etc/nginx/sites-enbled/default* добавятся новые настройки и будут прописаны пути к сертификату.

5. Проверьте конфигурацию Nginx, и если всё в порядке, перезагрузите её.