**1. Покупка VPS, домена и их связывание**

Покупаем VPS и домен. Привязываем домен к IP.  
<https://www.youtube.com/watch?v=vTPE5UoJioU> – привязка домена к IP.

**2. Первоначальная подготовка сервера**

При покупке VPS на почту приходит сообщение с логином и паролем для подключения по ssh. Логин, как правило, всегда root.

Вводим в терминале своего компьютера команду, затем пароль.

ssh root@tuna.com.ru

Возможны случаи, когда ключ к машине уже прописан, но он не совпадает. Тогда нужно выполнить   
 rm .ssh/known\_hosts

Еще можно подключаться, например root@192.168.1.1, (но зачем?)

Сразу меняем пароль

passwd root

Делаем обновление пакетов

apt-get update && apt-get upgrade

Посмотрим на состояние локали

perl -v

Наиболее вероятна ситуация, когда будет несколько предупреждений вида  
perl: warning: Please check that your locale settings:  
 LANGUAGE = (unset),  
 LC\_ALL = (unset),  
 ...

Генерируем локаль

locale-gen en\_US en\_US.UTF-8 ru\_RU.UTF-8

Затем реконфигурируем локаль машины.

dpkg-reconfigure locales

Вновь проверяем на наличие ошибок

perl -v

По идее, их быть не должно.

**3. Удаление Apache2**

Проверим, установлен ли Apache

service apache2 status

Если нет, то переходим к установке nginx.

Останавливаем apache и начинаем его удалять

service apache2 stop

---

Вот инструкция, как его удалить  
<http://razvlek-info.net/article/udalyaem-apache-polnostyu-debian-ili-ubuntu.html>

После нее выполняем

apt-get remove --purge apache\*

Теперь, по идее, apache удален. Но по адресу tuna.com.ru по прежнему apache? Это объяснимо. Браузер для ip машины берет статику с /var/www/html. Там есть файл index.html (или другой), в котором гипертекстовый шаблон этого самого apache. Можно просто переименовать index.html в .index.html. Все! Больше по адресу сайта не должно быть стартовой страницы apache.

Можно смело устанавливать nginx.

---

Если ссылка рабочая, то выполняем все инструкции по ней и переходим к установке nginx.

Продублирую ее тут, на случай, если ссылка уже недоступна.

Сначала ищем в dpkg все, что касается Apache.

dpkg -l | grep -i apache

Удаляем то, что нашлось

apt-get remove --purge apache2 apache2-doc apache2-utils … и прочие (одной командой)

Далее

apt-get autoremove

Потом

whereis apache2

Может быть, появится список директорий. Их нужно вручную удалить.

rm -rf /etc/apache2

rm -rf /usr/lib/apache2

Теперь возвращаемся назад! Там есть несколько инструкций после ссылки.

**4. Установка Nginx**

Устанавливаем Nginx.

apt-get install nginx

После установки запускаем nginx.

service nginx start

Смотрим статус:

service nginx status

Откроем страницу tuna.com.ru. Должна появиться служебная страница nginx.

**5. Установка инструментов python разработчика**

Во-первых, выполнять операции из под рута --- плохая практика. После того, как разобрались с Apache и Nginx, все остальное можно делать из под обычного пользователя. Создадим его.

adduser aleksey

adduser aleksey sudo

login aleksey

Теперь можно подключаться по ssh через ssh aleksey@tuna.com.ru

Установим инструменты python3

sudo apt-get install python3-dev python3-setuptools

А также pip3

sudo apt-get install python3-pip

Теперь установим virtualenv с помощью pip3

pip3 install virtualenv

Конечно же, нужен текстовый редактор

sudo apt-get install emacs

Сразу после установки надо добавить свой файл .emacs

Также сразу добавляем alias-ы в .bashrc

Например alias emc="emacs -nw"

Установим git, чтобы скачать себе репозиторий с моим сайтом.

sudo apt-get install git

**6. Подготовка виртуального окружения**

Во-первых клонируем репозиторий.

git clone <https://github.com/ferc255/socialnetwork>

Где подразумевается расположение виртуального окружение? Какое у него должно быть название? Обычно это можно узнать (конкретно я так делаю) в файле set\_paths.sh внутри проекта.

Узнаем это и создаем его в нужной папке

virtualenv env

Интересная фича. Можно подключиться к файлам машины через GUI интерфейс. То есть можно получить доступ к файлам сервера прямо со своего ноутбука хоть из консоли, хоть с помощью Nautilus (или другой файловый менеджер).

Для этого надо установить  
 sudo apt-get install sshfs

А затем указать, где будет храниться ссылка на файлы этой виртуальной машины  
 sshfs root@tuna.com.ru:/ /home/user/Develop/sites/socialnetwork/

Активируем виртуальное окружение

source env/bin/activate

Устанавливаем Django в наше виртуальное окружение:

pip3 install Django

...и uwsgi

pip3 install uwsgi

Надо проверить, нормально ли работает uwsgi. Создадим файл test.py с таким содержимым

def application(env, start\_response):

start\_response('200 OK', [('Content-Type','text/html')])

return [b"Hello World"]

И запускаем

uwsgi --http :8000 --wsgi-file test.py

**7. Настройка базы данных**

Скачиваем postgresql

apt-get install libpq-dev postgresql

Заходим в управление postgres

sudo –u postgres –i

Создаем базу данных сайта

createdb my\_site

Создаем пользователя, который администрирует эту базу

createuser -P alexsql

Заходим в режим исполнения postgres

psql

Передаем все права пользователю alexsql

postgres=# GRANT ALL PRIVILEGES ON DATABASE my\_site TO django;

Иногда, особенно при переносе баз, ругается при “migrate”. Для этого надо прописать:

psql database -c "GRANT ALL ON ALL TABLES IN SCHEMA public to user;"

где database – имя рабочей БД, user – имя пользователя БД в settings.py.

Устанавливает утилиту, которая нужна django, что контактировать с postgresql

pip3 install psycopg2

В файле settings.py теперь указываем такую базу данных

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.postgresql\_psycopg2',

'NAME': 'my\_site',

'USER': 'alexsql',

'PASSWORD': '1234567890',

'HOST': 'localhost',

'PORT': '',

}

}

Нельзя убирать ни одно из полей. Работать не будет, проверено. Пароль можно спрятать где-то за пределами репозитория и сюда сделать import, например.

**8. Первый запуск**

Во-первых, конечно, нужно установить requirements

pip3 install –r requirements.txt

Все делаем в виртуальном окружении.  
Если что-то не установилось, устанавливаем вручную pip3 install X==2.3.0

Надо внести несколько изменений в settings.py, касаемо продакшн.

MEDIA\_ROOT = "/home/aleksey/my\_site/media/"

STATIC\_ROOT = "/home/aleksey/my\_site/static/"

STATICFILES\_DIRS для продакшн надо комментировать. Вся статика и так будет лежать в корне проекта.

Добавляем в ALLOWED\_HOSTS домены

ALLOWED\_HOSTS = [

‘tuna.com.ru’,

‘www.tuna.com.ru’,

‘37.140.197.29’,

‘37.140.197.29:80’

]

Делаем миграцию, если необходимо

python3 manage.py migrate

Если запустить правильно настроенный set\_paths.sh, то вполне можно пользоваться django-admin вместо py3 manage.py

Создаем суперпользователя

python3 manage.py createsuperuser

Запускаем сервер

python3 manage.py runserver 0.0.0.0:8080

Ничего страшного, что статика не загрузилась, это нормально. Так должно быть! Позже самому nginx-у надо будет указать, где она лежит.

Если проект разрабатывался так, что статика не в корне, а в каждом приложении своя директория static, то надо выполнить

python3 manage.py collectstatic

Теперь можно запустить my\_site через uwsgi, и если это не сработало, значит где-то совершена ошибка. Вполне вероятно, о ней сообщат логи uwsgi.

uwsgi --http :8000 --module my\_site.wsgi

Вместо my\_site папку, в которой settings.py обычно лежит, то есть имя проекта.

**9. Начальная настройка nginx**

Создаем папку deployment в корне проекта.

В ней создаем файл uwsgi\_params с таким содержанием

uwsgi\_param QUERY\_STRING $query\_string;

uwsgi\_param REQUEST\_METHOD $request\_method;

uwsgi\_param CONTENT\_TYPE $content\_type;

uwsgi\_param CONTENT\_LENGTH $content\_length;

uwsgi\_param REQUEST\_URI $request\_uri;

uwsgi\_param PATH\_INFO $document\_uri;

uwsgi\_param DOCUMENT\_ROOT $document\_root;

uwsgi\_param SERVER\_PROTOCOL $server\_protocol;

uwsgi\_param REQUEST\_SCHEME $scheme;

uwsgi\_param HTTPS $https if\_not\_empty;

uwsgi\_param REMOTE\_ADDR $remote\_addr;

uwsgi\_param REMOTE\_PORT $remote\_port;

uwsgi\_param SERVER\_PORT $server\_port;

uwsgi\_param SERVER\_NAME $server\_name;

В той же папке создаем файл my\_site\_nginx.conf с таким содержимым

# the upstream component nginx needs to connect to

upstream django {

server unix:///home/aleksey/socialnetwork/uwsgi\_nginx.sock; # for a file socket

# server 127.0.0.1:8001; # for a web port socket (we'll use this first)

}

# configuration of the server

server {

# the port your site will be served on

listen 8000;

# the domain name it will serve for

server\_name tuna.com.ru; # substitute your machine's IP address or FQDN

charset utf-8;

# max upload size

client\_max\_body\_size 75M; # adjust to taste

# Django media

location /media {

alias /home/aleksey/socialnetwork/media; # your Django project's media files - amend as required

}

location /static {

alias /home/aleksey/socialnetwork/static; # your Django project's static files - amend as required

}

# Finally, send all non-media requests to the Django server.

location / {

uwsgi\_pass django; - именно django, не менять!

include /home/aleksey/socialnetwork/deployment/uwsgi\_params; # the uwsgi\_params file you installed

}

}

В папке /etc/nginx/sites-enabled создаем ссылку на файл mysite\_nginx.conf, чтобы nginx увидел его.

sudo ln -s /home/aleksey/socialnetwork/deployment/socn\_nginx.conf /etc/nginx/sites-enabled

Обязательно указывать полный адрес каждого файла/директории.

Перезапускаем nginx:

sudo service nginx restart

Помещаем файл с именем, например, media.png в папку /home/aleksey/socialnetwork/media.

В браузере переходим по адресу yourserver.com:8000/media/media.png и, если видим наш файл, значит мы все сделали правильно.

**10. Настройка uwsgi**

Пробуем запустить приложение через сокет

uwsgi --socket uwsgi\_nginx.sock --module my\_site.wsgi --chmod-socket=666

Опять же, my\_site — имя главного приложения проекта

На 8000 порту должно все работать.

Очень удобно все опции, с которыми мы запускаем uWSGI, указать в ini файле, а при запуске передавать только путь к этому файлу. Создаем файл my\_site\_uwsgi.ini в нашей папке deployment. Указываем в нем следующее содержимое:

[uwsgi]

# Настройки, связанные с Django

# Корневая папка проекта (полный путь)

chdir = /home/aleksey/socialnetwork

# Django wsgi файл

module = my\_site.wsgi

# полный путь к виртуальному окружению

home = /home/aleksey/env

# общие настройки

# master

master = true

# максимальное количество процессов

processes = 10

# полный путь к файлу сокета

socket = /home/aleksey/socialnetwork/uwsgi\_nginx.sock

# права доступа к файлу сокета

chmod-socket = 666

# очищать окружение от служебных файлов uwsgi по завершению

vacuum = true

Запускаем этот файл:

uwsgi --ini my\_site\_uwsgi.ini

На 8000 порту должно все работать.

Хорошо, работает. Теперь надо, чтобы это работало как сервис. То есть, чтобы не было необходимости постоянно держать открытым терминал с запущенной командой.

Для этого будет использоваться supervisor. Но он работает из-под рута. Поэтому надо теперь установить uwsgi из-под рута.

deactivate  
su - root  
 pip3 install uwsgi

Пробуем запустить глобально

uwsgi --ini my\_site\_uwsgi.ini

Не работает на 8000 порту? Это вполне нормально. Надо установить дополнительную утилиту.

apt-get install uwsgi-plugin-python3

И еще добавить в файл my\_site\_uwsgi.conf следующую строку

plugins = python3

Теперь должно работать.

Также должно быть все в порядке при запуске в режиме императора.

uwsgi --emperor "/home/aleksey/socialnetwork/deployment/my\_site\_uwsgi.ini"

**11. Настройка supervisor**

Работаем все также под рутом. Устанавливаем supervisor.

apt-get install supervisor

Для него надо написать конфиг. Этот конфиг должен находиться в папке /etc/supervisor/conf.d/.

Туда доступа у обычного пользователя нет. И это правильно. Последнее, что нам осталось сделать под рутом, это разрешить запись в файл конфигурации supervisor-а.

Заходим в /etc/supervisor/conf.d и создаем файл my\_site.conf

touch my\_site.conf  
 chmod o+w my\_site.conf

Все, выходим из рута. Все остальное можно делать из-под обычного пользователя.

login aleksey

Запишем в my\_site.conf следующий конфиг

[program:my\_site]

command=uwsgi --emperor "/home/django/my\_site/deployment/my\_site\_uwsgi.ini"

stdout\_logfile=/home/django/my\_site/deployment/uwsgi.log

stderr\_logfile=/home/django/my\_site/deployment/uwsgi\_err.log

autostart=true

autorestart=true

Перезапускаем supervisor

sudo service supervisor restart

Возможно, надо выполнить еще 2 команды. Но обычно обходится и без них.  
supervisorctl reread  
supervisorctl update

Теперь на 8000 порту должно все работать.

**12. Окончательные мелкие штрихи**

Изменим в deployment/my\_site\_nginx.conf порт, на котором должно работать приложение. Поставим 80 и сделаем

sudo service nginx restart

В файле settings.py DEBUG надо поставить False.

**\*13. Extra**

Ссылка на обучающее видео, благодаря которому эта шпаргалка вообще существует  
<https://www.youtube.com/watch?v=lhO0gcwT0rA>

......

При любых изменениях в проекте

Чтобы изменения вступили в силу и отображались в браузере надо выполнить 3 команды

service supervisor stop

touch my\_site\_uwsgi.ini

service supervisor start

......

Если хочется разграничить settings...

В my\_site\_uwsgi.conf добавляем строку

env = DJANGO\_SETTINGS\_MODULE=socn.settings.abs\_settings

......

Перенос БД

Чтобы сделать выгрузку и перенести текущую БД, превращаем ее в json

python3 manage.py dumpdata --natural-foreign --exclude auth.permission

--exclude contenttypes --indent 4 > datadump.json

Чтобы полученный файл datadump.json загрузить в проект, вводим

python3 manage.py loaddata datadump.json

......

..............

Войти в psql как пользователь alex и работать с базой данных my\_test

psql alex –h localhost –d my\_test

..............

Postgres работает с папкой var/lib/postgresql

..............

Для переноса БД можно воспользоваться pg\_dump (плохой способ):

pg\_dump the\_db\_name > the\_backup.sql

Then copy the backup to your development server, restore with:

psql the\_new\_dev\_db < the\_backup.sql

https://stackoverflow.com/questions/1237725/copying-postgresql-database-to-another-server

..............