Деплой djnago проекта по шагам.

У Вас куплена свежая машинка, мы первый раз к ней подключаемся.

Используем мы Ubuntu 14.04.

(В конце файла еще немного полезного материала. Посмотри сейчас!)

1. Если у вас уже прописан ключ к вашей машине, но он не совпадает, очищаете его

rm .ssh/known\_hosts

2. Подключаетесь и вводите пароль от сервера.

ssh root@5.45.120.175

3. Обновляете пароль рута, если он вас не устраивает

passwd root

4. Обновляем списки репозиториев.

apt-get update

5. Апгредим.

apt-get upgrade

6. Если присутствуют проблемы локали, исправляем их сразу.

perl -v

7. Смотрим, если ругается и есть предупреждения генерируем локаль.

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

8. Затем реконфигурируем локаль машины.

dpkg-reconfigure locales

9. Вновь проверяем на наличие ошибок

perl -v

10. Ура их нет! Двигаемся дальше. Устанавливаем Nginx.

apt-get install nginx

11. После установки запускаем nginx.

service nginx start

12. Смотрим статус:

service nginx status

13. Если все ок, и он запущен можете открыть страничку вашего сайта и вы увидите стартовую страницу nginx. Если же сервер не запустился, проверте, не запущен ли у вас apache

service apache2 status

14. Если он запущен стопайте его. Его необходимо полностью удалить.

service apache2 stop

------------------------

14.5 (моя) Сначала ищем в dpkg все, что касается Apache.

dpkg -l | grep -i apache

Удаляем то, что нашлось

apt-get remove --purge apache2 apache2-doc … 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

После этого, на всякий случай,

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! Должен появиться nginx по ip-адресу в браузере.

----------

15. Деинсталируем апач и все его конфиги такой вот не хитрой командой:

apt-get purge apache2 apache2-utils apache2.2-bin apache2-common

16. Если какие то пакеты, которые собираемся удалить отсутствуют, просто уберите их из команды. Далее:

apt-get autoremove

17. Наконец, надо проверить наличие конфигурационных файлов или мануалов, связанных с Apache2, но до сих пор не удаленных.

whereis apache2

18. Удаляете все эти директории вручную. Например, вот так:

rm -rf /etc/apache2

19. Если все удалено, аллилуя. Запускаем nginx. И смотрим статус:

service nginx start

service nginx status

20. Если все ок, заходите на вашу страничку и вы увидете стартовую страницу nginx. Идем дальше. Cоздайте пользователя для старта django-приложения. Создайте из-под него виртуальное окружение.

apt-get install python3-dev python3-setuptools

easy\_install-3.4 virtualenv

adduser django

login django

cd /home/django

virtualenv venv

21. Теперь, переносим проект в эту папку.Устанавливаем git, nano если они не установлены.

apt-get install git nano

22. Создаем ключ ssh. Все поля оставляем пустыми.

ssh-keygen -t rsa -C «your\_mail@mail.com” (исправить на двойные кавычки)

23. Переходим в директорию ключа, открываем необходимый нам публичный ключ. Советую с помощью утилиты sshfs подключиться к вашей машине через gui интерфейс. Вам будет необходимо указать адрес машины и папку, в которую поместите файлы машины. Например вот так.

sshfs root@5.45.120.175:/ /home/user/Develop/sites/my\_site/

24. Затем копируете ключ.

/root/.ssh/id\_rsa.pub

25. Копируем его в настройки пользователя git репозитория

26. После того, как вы это сделаете, вы сможете спокойно клонировать по ssh репозиторий. Заходите под рутом.

su - root

cd /home/django/

git clone git@github.com:user/my\_site.git

27. Симлинк /etc/nginx/sites-enabled/default можно удалить

28. Логинимся под юзером django.

login django

29. Активируем виртуальное окружение.

source venv/bin/activate

30. Устанавливаем Django в наше виртуальное окружение:

pip install Django

31. Переходим в корневую папку проекта.

cd my\_site

32. Один из хороших способов установить uWSGI:

pip install uwsgi

33. Проверка. Создаем файл test.py:

nano test.py

34. С таким содержимым:(Если не создается, создайте от рута)

def application(env, start\_response):

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

return [b"Hello World"]

35. Запускаем uWSGI:

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

36. Если все работает, нажимаем ctrl-c, завершая процесс и продолжаем дальше.

37. Теперь конфигурируем наш сайт. Устанавливаем postgresql. Из под рута.

apt-get install libpq-dev postgresql postgresql-contrib

su – postgres // лучше sudo –u postgres –i (my mark)

createdb my\_site

createuser -P django

psql

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

su - root

login django

source /home/django/venv/bin/activate

pip install psycopg2

my: 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.

38. Если необходимо перенести БД со старой на новую делается это так:

Шаг 1. Выполняется со старыми настройками DATABASES в settings.py

python manage.py dumpdata > datadump.json

my: У меня не работал loaddata после этого. На stackoverflow нашел такое решение:

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

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

https://stackoverflow.com/questions/42125730/how-to-manage-py-loaddata-in-django

Шаг 2. Выполняется с новыми настройками DATABASES в settings.py

python manage.py loaddata datadump.json

39. После этого устанавливаем все библиотеки что есть в requirements.txt. Для этого заходим в папку my\_site проектом и вводим следующую команду.Разумеется с помощью виртуального окружения.

pip install -r requirements.txt

40. Если не установилась какая-нибдуь библиотека X устанавливайте более позднюю версию:

pip install X==2.9.0

41. В файле …/my\_site/settings.py обязательно нужно указать настройки базы данных:

DATABASES = {

'default': {

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

'NAME': 'my\_site',

'USER': 'django',

'PASSWORD': '1234567890',

'HOST': 'localhost',

'PORT': '',

}

}

42. В файле settings.py должен быть такой путь:

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

44. Делаем миграцию, если это необходимо:

python manage.py migrate

45. Создаем супер пользователя.

python manage.py createsuperuser

45.5(моя вставочка) надо добавить в ALLOWED\_HOSTS в settings.py:

ALLOWED\_HOSTS = [

‘tuna.com.ru’,

‘www.tuna.com.ru’,

‘37.140.197.29’,

‘37.140.197.29:80’

]

(код не копировать, не тестил)

46. Запускаем сервер

python manage.py runserver 0.0.0.0:8080

47. Если ошибок не возникло, заходим и настраиваем сайт через админку(если у вас новая свежая бд без данных):

48. После того, как вы сконфигурировали проект идем дальше. Собираем всю статику(Скорее всего это сработает только из под рута):

python manage.py collectstatic

49. Теперь, вы можете запустить my\_site через uwsgi, и если это не сработало, значит вы где то совершили ошибку. Вполне вероятно, о ней сообщат логи uwsgi.

uwsgi --http :8000 --module my\_site.wsgi (вместо my\_site папку, в которой settings.py

обычно лежит)

50. Теперь нам необходимо сконфигурировать nginx. Для начала, создайте файл в корне проекта my\_site с названием 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;

51. Вообще, лучше поместите этот файл в корень проекта, в папку с названием deployment, чтобы файлы проекта различались с файлами деплоя.

52. теперь в тойже папке создайте файл my\_site\_nginx.conf

53. Наполните его таким содержимым:

# the upstream component nginx needs to connect to

upstream django {

server unix:///home/django/my\_site/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 my\_site.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/django/my\_site/media; # your Django project's media files - amend as required

}

location /static {

alias /home/django/my\_site/static; # your Django project's static files - amend as required

}

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

location / {

uwsgi\_pass django;

include /home/django/my\_site/deployment/uwsgi\_params; # the uwsgi\_params file you installed

}

}

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

sudo ln -s /home/django/rss\_news/deployment/rss\_news\_nginx.conf /etc/nginx/sites-enabled

55. Перезапускаем nginx:

/etc/init.d/nginx restart

56. Помещаем файл с именем, например, media.png в папку /home/django/my\_site/media.

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

58. Пробуем запустить через сокет:

uwsgi --socket uwsgi\_nginx.sock --wsgi-file test.py --chmod-socket=666 (не работает, и не надо, не трогаем)

59. nginx + uWSGI + Django. Запускаем: В браузере переходим на yourserver.com:8000/ и видим стартовую страницу Django.

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

(my\_site опять же имя главного приложения django)

60. Мы собрали всю цепочку, но настройка еще не закончена, идем дальше.

61. 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:

#mysite\_uwsgi.ini

[uwsgi]

# Настройки, связанные с Django

# Корневая папка проекта (полный путь)

chdir = /home/django/my\_site

# Django wsgi файл

module = my\_site.wsgi

# полный путь к виртуальному окружению

home = /home/django/venv

# общие настройки

# master

master = true

# максимальное количество процессов

processes = 10

# полный путь к файлу сокета

socket = /home/django/my\_site/uwsgi\_nginx.sock

# права доступа к файлу сокета

chmod-socket = 666

# очищать окружение от служебных файлов uwsgi по завершению

vacuum = true

env = DEBUG\_MODE=False

daemonize=/var/log/uwsgi/my\_site.log

62. Запускаем этот файл:

uwsgi --ini my\_site\_uwsgi.ini

63. Проверяем. Все работает? Дальше.До сих пор uWSGI был установлен в виртуальном окружении. Чтобы была возможность автоматически запускать uWSGI при старте операционной системы, мы установим его глобально. Деактивируем виртуальное окружение:

deactivate

64. Устанавливаем pip и pip3глобально.

apt-get install python-pip python3-pip

64. Затем, устанавливаем uwsgi глобально.

pip3 install uwsgi

65. Проверяем запуск:

uwsgi --ini my\_site\_uwsgi.ini

66. Если сервер обслуживает несколько проектов, каждый из которых использует uWSGI, то нужно исползовать режим Emperor. В этом режиме uWSGI просматривает папку с конфигурационными файлами и для каждого файла запускает отдельный процесс (вассал).

Если один из конфигурационных файлов будет изменен, uWSGI перезапустит соответствующего вассала.

Создаем папку для конфигурационных файлов:

sudo mkdir /etc/uwsgi

sudo mkdir /etc/uwsgi/vassals

67. Создаем в ней ссылку на my\_site\_uwsgi.ini:

sudo ln -s /home/django/my\_site/deployment/my\_site\_uwsgi.ini /etc/uwsgi/vassals/

68. Запускаем uWSGI в режиме Emperor, потом отключаем если все робит(Возможно вам придется перезапустить вашу БД такой командой sudo service postgresql restart):

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

69. Устанавливаем супервизор

apt-get install supervisor

70. Создаем файл конфигурации в папке etc/supervisor/conf.d/my\_site.conf:

echo\_supervisord\_conf > /etc/supervisord.conf (в папке /)

71. В Этом файле указываем следующий текст:

[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

72. Индексируем этот файл: (Если появляются какие то ошибки, перезапускаем supervisor(service supervisor restart)) Так же, смотрим логи.

supervisorctl reread

supervisorctl update

73. Все должно работать. Управлять supervisor можно как сервисом:

service supervisor start|stop|status ...

74. Если происходят какие то изменения в коде, выполняем эту команду в папке deployment:

service supervisor stop

touch my\_site\_uwsgi.ini

service supervisor start

75. Команда supervisorctl покажет запущенные приложения императором.

76. Если необходимо запустить проект в режиме дебага, меняем значение DEBUG\_MODE=False на DEBUG\_MODE=True в файле конфигурации uwsgi.

77. Вот и весь деплой)

--------------

Ссылки на видеообучение:

<https://www.youtube.com/watch?v=vTPE5UoJioU> – привязка домена к VPS

<https://www.youtube.com/watch?v=lhO0gcwT0rA&t=2s> – сам деплой, откуда и взята шпаргалка

---------------

--------------

Чтобы созданного пользователя django добавить в группу sudo, надо прописать в терминале за root-а:

adduser django sudo

---------------

Войти в postgres

sudo –u postgres –i

---------------

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:

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

----------------