# 使用Nginx+uWSGI+Django方法部署Django程序

Django的部署可以有很多方式，采用nginx+uwsgi的方式是其中比较常见的一种方式。在这种方式中，我们的通常做法是，将nginx作为服务器最前端，它将接收WEB的所有请求，统一管理请求。nginx把所有静态请求自己来处理（这是Nginx的强项）。然后，Nginx将所有非静态请求通过uWSGI传递给Django，由Django来进行处理，从而完成一次WEB请求。可见，uWSGI的作用就类似一个桥接器。起到桥梁的作用。

NOTE：不使用Nginx，只使用uWSGI+django也是可以实现WEB服务的。uWSGI也可以直接处理WEB请求。

为了完成上述的方式部署，我将分成部分来分别进行阐述。

第一步先解决uWSGI与django的桥接。解决在没有Nginx的情况下，如何使用uWSGI+Django来实现一个简单的WEB服务器。

第二步解决uwsgi与Nginx的桥接。通过nginx与uwsgi的桥接，打通nginx与django的连通，从而比较完美的实现django的部署。

## 环境介绍

1. Mac OS
2. django 1.4.11
3. uWSGI 2.0.15
4. Nginx 1.13-2.5

## 使用uWSGI+Django服务器

### 安装uWSGI

安装pip

安装uWSGI

$ pip install uwsgi

or

$ brew install uwsgi

### 测试uWSGI

在你的机器上写一个test.py

# test.py

def application(env, start\_response):

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

return "Hello World"

然后执行shell命令：

uwsgi --http-socket :8000 --plugin python --wsgi-file test.py

<http://127.0.0.1:8001/>看在网页上是否有Hello World

### 运行项目

首先需要确保在开发环境下项目可以正常运行

$python manage.py runserver

确保项目能够正常运行

运行以下命令

$uwsgi --http-socket :8092 --plugin python --home=/Users/xiawei/Documents/codes/PycharmProjects/virtualenv\_django1.11.4 --module DjangoDemo.wsgi

**--http-socket**指定端口号，指定使用http方式，这样不通过Nginx等web服务器，也可以访问这个程序。如果使用web服务器，必须使用--uwsgi-socket方式。

-**-plugin python** 这个必须要，否则后面的--home 等option会报错。

**--home** 指定python环境，这里指定了一个虚拟环境

**--module**指定。DjangoDemo.wsgi就是指wsgi.py文件,该文件由start project时生成，在DjangoDemo／Django Demo文件夹下。

该命令运行在manage.py同级目录之下。

## Uwsgi+Nginx

安装Nginx

在manage.py同级目录下新建配置文件djangodemo\_nginx.conf。该文件用于配置nginx服务，内容如下：

# the upstream component nginx needs to connect to

# 反向代理服务器列表，这里必须包含最后location里面uwsgi\_pass的服务  
upstream django {  
 server 127.0.0.1:8100;   
 #server unix:///tmp/mysite.sock; # for a file socket, more effective  
}  
# configuration of the server  
server {  
 # the port your site will be served on  
 listen 8080;  
 server\_name localhost;  
 charset utf-8;  
 # max upload size  
 client\_max\_body\_size 75M; # adjust to taste  
 # Django media  
 #location /media {  
 # alias /path/to/your/media; # your Django project's media files - amend as required  
 # alias /Users/xiawei/Documents/codes/PycharmProjects/DjangoDemo/polls/static/polls/images;  
 #}  
 location /static {  
 alias /Users/xiawei/Documents/codes/PycharmProjects/DjangoDemo/polls/static; # your Django project's static files - amend as required  
 }  
 # Finally, send all non-media requests to the Django server.  
 location / {  
 #uwsgi\_pass localhost:8001;  
 proxy\_pass http://django;  
 uwsgi\_pass 127.0.0.1:8100; #该服务地址与上节启动的uwsgi服务地址对应  
 include uwsgi\_params; # the uwsgi\_params file you installed  
 }  
}

ln -s /Users/xiawei/Documents/codes/PycharmProjects/DjangoDemo/mysite\_nginx.conf /usr/local/etc/nginx/servers

建立软连接，每次nginx启动都会加载server目录的配置文件，这样就可以加载上一步骤的配置文件了。

同级目录下建立wsgi\_params文件，内容如下：

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;

### 部署静态文件

运行Nginx之前，你需要把所有的静态文件同步到静态文件夹（static folder）中，首先你需要编辑DjangoDemo/DjangoDemo/settings.py。加入下面这行配置：

STATIC\_ROOT = os.path.join(BASE\_DIR, "static/")

然后运行命令

$python manage.py collectstatic

运行nginx

运行1.2.3节项目

访问地址localhost:8080

## 常见问题

### 端口被占用

probably another instance of uWSGI is running on the same address (:8002).

bind(): Address already in use [core/socket.c line 764]

按照端口号进行查询

$lsof –I :8002

|  |
| --- |
| COMMAND  PID USER   FD   TYPE             DEVICE SIZE/OFF NODE NAME  uwsgi   2208   tu    4u  IPv4 0x53492abadb5c9659      0t0  TCP \*:teradataordbms (LISTEN)  uwsgi   2209   tu    4u  IPv4 0x53492abadb5c9659      0t0  TCP \*:teradataordbms (LISTEN) |

然后根据PID用以下命令kill相关进程

sudo kill -9 2208 2209

按照程序名称查询

ps aux | grep uwsgi

## uGWSGI为什么需要nginx

uGWSGI能够配合Django处理所有的动态请求，但是其对静态资源的处理并不是好，一事性能的问题，二是各种HTTP请求缓冲头，处理也不如Nginx完善。

一种情况，本地有多个 web 服务，有 Python、php、java 编写的，都想监听 80 端口，这个时候就必须有一个负责转发的服务了。

如果本机确定只跑这一个服务，但是 uwsgi 和 gevent 对于静态资源处理的并不是很好，一是性能问题，二是各种 HTTP 请求缓存头，处理的也没有 Nginx 完善。

然后还有一些安全问题，Nginx 作为专业服务器，暴露在公网相对比较安全（虽然有著名的心血漏洞），uwsgi 和 gevent 的话，漏洞恐怕只比 Nginx 多而不是少。

再来就是支持的协议，uwsgi 和 gunicon 早期是不支持 https 的，只能提供 http 给浏览器访问。虽然现在这两者都支持了，但是以后的 spdy 和http2，恐怕也是 nginx 跟进更快一些。

还有一些运维优势，比如服务器被人 CC，这是一种非常常见的情况，nginx 可以比较方便的把一些 IP 加入黑名单，直接改配置文件就好了。要是 uwsgi 或者 gunicorn，恐怕还要修改自己应用的代码，把 IP 过滤写进去。

题主说只考虑单台机器的情况，但是如果不考虑的话，那一个 nginx 做负载均衡那就几乎是必须了。

Nginx更安全

* Nginx能更好地处理静态资源（通过一些http request header）
* Nginx也可以缓存一些动态内容
* Nginx可以更好地配合CDN
* Nginx可以进行多台机器的负载均衡
* 不需要在wsgi server那边处理keep alive
* 让Nginx来处理slow client
* 还有一个更隐蔽的区别是，像uWSGI支持的是wsgi协议，Nginx支持的是http协议，它们之间是有区别的。  
  一句话来说，如果访问量不大，其实也没必要加个Nginx，uWSGI或者Gunicorn足以胜任。

## 参考文档

1. uWSGI官方文档

（<http://uwsgi.readthedocs.io/en/latest/tutorials/Django_and_nginx.html?highlight=django>）

1. Deploy Django Project in nginx with uWSGI on Mac OS X （<https://www.sean-lan.com/2016/09/15/django-uwsgi-nginx/>）
2. django官方部署uwsgi

（<https://docs.djangoproject.com/en/1.11/howto/deployment/wsgi/uwsgi/>）

1. 知乎uWSGI

（<https://www.zhihu.com/topic/19607701/hot>）