Nginx + uWsigi + Django 运行环境部署

部署前的准备工作:

1. 安装virtualenv

   使用virtualenv为我们要部署的项目创建一个“独立”的python运行环境

   ```
   virtualenv env27
   cd env27
   source bin/active
   ```

   注: env27这个目录为部署项目的python运行环境，后文中会有涉及

2. 安装uWSGI

   ```
   pip install uwsgi
   ```

3. 整个部署过程中需要准备的几个配置文件:

- mingjia_uwsgi.ini
- uwsgi_params 下载地址
- mingjia_nginx.conf

使用uWSGI运行Django项目:

当我们安装好uWSGI模块之后可以测试一下uWSGI是否能够正常运行:

test.py

```
# test.py
def application(env, start_response):
    start_response('200 OK', [('Content-Type','text/html')])
    return [b"Hello World"] # python3
```

```
    #return ["Hello World"] # python2
```

使用下面的命令运行：

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

参数的含义：

- http :8000 -> 使用http协议，使用8000端口
- wsgi-file test.py -> 加载具体的文件 test.py

当我们打开网页在地址栏输入http:127.0.0.1:8000 ,如果能收到Hello World的相应则表明uWSGI运行正常。

开始部署项目

下面我们开始使用uWGSI来作为服务器运行我们的Django项目，当然在运行前我们需要先在Django提供环境中测试下当前的项目是否能够正常工作：

```
python manager runserver 0.0.0.0:8000
```

如果能够正常工作的话，我们在终端上输入下面的指令，来让我们的项目运行在uWSGI下：

```
uwsgi --http :8000 --module mysite.wsgi
```

参数说明：

- mysite.wsgi为你要运行的项目下的 wsgi.py文件

注意！如果命令运行后出现" ImportError: No module named xxx"错误，请检查virtualenv环境是否激活,以及依赖的包是否都已经安装。

通过浏览器访问下你的项目，如果可以正常返回数据(可以先忽视未被加载静态文

件),则说明你已经成功了一大半，这个时候要冷静不要被胜利的喜悦冲昏了头脑。

下面我们创建mingjia_uwsgi.ini配置文件：

mingjia_uwsgi.ini

```
# mysite_uwsgi.ini file
[uwsgi]

# Django-related settings
# the base directory (full path)
chdir           = /root/workspace/http/mingjia
# Django's wsgi file
module          = mingjia.wsgi
# the virtualenv (full path)
home            = /root/workspace/env27

# process-related settings
# master
master          = true
# maximum number of worker processes
processes       = 10
# the socket (use the full path to be safe
socket          = :8001
# ... with appropriate permissions - may be needed
# chmod-socket    = 664
# clear environment on exit
vacuum          = true
```

需要配置的参数:

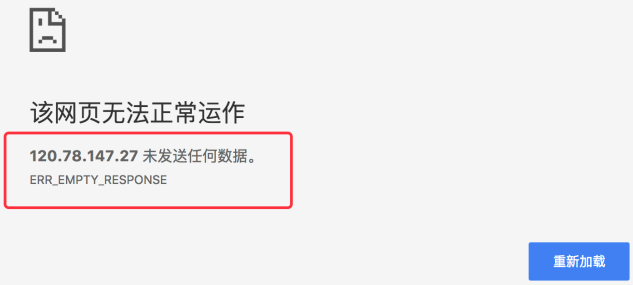
- chdir 要部署的Django项目的根路径
- module Django的wsgi文件
- home python的安装目录
- socket 指定端口，这里的端口要和下面nginx中配置的端口一致

其余的参数可以直接使用默认值即可

下面我们可以直接通过以下命令运行:

```
uwsgi --ini mingjia_uwsgi.ini
```

接下来当我们通过浏览器去访问的时候会发现浏览器并没有接收到服务器发来的数据



这里为问题主要是因为 mingjia_uwsgi.ini 中

```
socket          = :8001
```

这一行配置，把socket修改成http就可以正常访问，并获取页面数据了，当然这里不需要修改，这一行的配置是为了能够使uWSGI与Nginx互相通信。

下面我们开始配置Nginx来实现与uWSGI的相互通信

下面我们开始创建 nginx 配置文件,具体的内容如下:

mingjia_nginx.conf

```
# mysite_nginx.conf

# the upstream component nginx needs to connect to
upstream django {
    # server unix:///path/to/your/mysite/mysite.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      80;
    # the domain name it will serve for
    server_name .merpyzf.com; # 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 /path/to/your/mysite/media;  # your Django
project's media files - amend as required
    # }

    location /static {
        alias /root/workspace/http/mingjia/mingjia_admin/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     /root/workspace/http/mingjia/uwsgi_params; #
the uwsgi_params file you installed
    }
}
```

做个简要说明，这个文件要配置的地方不多：

1. 下面的这个server地址的端口号要与上面 mingjia_uwsgi.ini 文件中的端口号保持一致，这样才能连通。

```
upstream django {
    # server unix:///path/to/your/mysite/mysite.sock; # for a file
socket
    server 127.0.0.1:8001; # for a web port socket (we'll use this
first)
```

```
}
```

2. listen 80

   这个端口号为访问Nginx服务器时的端口

3. alias 的内容为项目中存放静态文件的文件夹的全路径

```
    location /static {
        alias /root/workspace/http/mingjia/mingjia_admin/static;
        # your Django project's static files - amend as required
    }
```

4. 最后一个 uwsgi_params 就是已经准备好的配置文件，直接拷贝所在路径就可以了

这个conf文件告诉nginx媒体资源和静态文件在文件系统中的具体位置，以及处理一些动态的需要Django进行干预的请求。

下面将这个文件使用软连接到 '/etc/nginx/sites-enabled/'目录下，确保Nginx可以识别

```
 sudo ln -s /root/workspace/http/mingjia/mingjia_nginx.conf
/etc/nginx/sites-enabled/
```

然后将sites-enabled下的default这个默认配置文件删除，注意！如果不删除的话Nginx读取的仍然是default中的配置信息

下面重启Nginx即可:

```
/etc/init.d/nginx restart
```

如果以上的步骤操作全都无误的话，那么正常情况下你的项目就可以Nginx服务器下正常访问了，但总会有出乎意料的问题！按照以上的步骤我已成功在树莓派上部署，但当真正部署到阿里云的Ubuntu上的时候，出现了403权限拒绝,静态文件无法加载的问题。于是开始了一个多小时的错误排查，文件路径没问题，配置没问题，于是开始怀疑人生，喝了口水，突然想起是不是文件权限的问题。不出所料，最后

的错误定位在家目录的权限上

```
drwxr-xr-x   2 root root  4096 Aug 17 15:46 bin
drwxr-xr-x   3 root root  4096 Aug 17 15:45 boot
drwxr-xr-x  20 root root  4160 Oct 17 20:12 dev
drwxr-xr-x  91 root root  4096 Oct 26 17:15 etc
drwxr-xr-x   2 root root  4096 Apr 13  2016 home
lrwxrwxrwx   1 root root    32 Aug 17 15:43 initrd.img -
drwxr-xr-x  20 root root  4096 Aug 17 15:46 lib
drwxr-xr-x   2 root root  4096 Aug 17 15:46 lib64
drwx------   2 root root 16384 Aug 17 15:42 lost+found
drwxr-xr-x   4 root root  4096 Aug 17 15:43 media
drwxr-xr-x   2 root root  4096 Feb 16  2017 mnt
drwxr-xr-x   2 root root  4096 Feb 16  2017 opt
dr-xr-xr-x 123 root root     0 Oct 17 20:12 proc
drwx------   9 root root  4096 Oct 26 17:44 root
drwxr-xr-x  21 root root   740 Oct 26 17:44 run
drwxr-xr-x   2 root root  4096 Aug 17 15:48 sbin
drwxr-xr-x   3 root root  4096 Oct 24 09:54 srv
dr-xr-xr-x  13 root root     0 Oct 18 04:12 sys
drwxrwxrwt   7 root root  4096 Oct 26 17:47 tmp
drwxr-xr-x  10 root root  4096 Aug 17 15:42 usr
drwxr-xr-x  13 root root  4096 Oct 25 20:16 var
lrwxrwxrwx   1 root root    29 Aug 17 15:43 vmlinuz -> b
```

可以看到root目录是不具备读权限的，即我们普通用户起初不具备读权限,只需要给该目录赋予读取权限即可解决403权限拒绝问题。

简单粗暴:

```
chmod 777 root/
```

这样部署上去的项目就可以正常访问了，当然要确保Nginx和uWSGI同时运行，这样才能正常访问。

到这里这篇文章就算结束了，也算是我两天时间研究Nginx,uWsigi,Django部署的一个总结，希望能给大家带来一些帮助，欢迎大家指正错误。如果觉着文章比较啰嗦，可以直接查看官网提供的搭建教程。

参考:

http://uwsgi-docs.readthedocs.io/en/latest/tutorials/Django_and_nginx.html?

highlight=nginx

http://uwsgi-docs.readthedocs.io/en/latest/Nginx.html?highlight=Nginx