链滴

django + uwsgi + nginx 部署小结

作者：flhuoshan
原文链接：https://ld246.com/article/1525480091859
来源网站：链滴

1. 环境准备

本文基于centos7.x + python3.x完成django2.0.x项目的部署。

- 需要准备以下环境：

1. python3.x,但不可卸载python2，因为centos的某些系统功能依赖python2。
2. pyenv,因为多个python版本共存的问题，需要在版本间切换，所以安装pyenv。
3. django2.0.x及相关依赖python库，

首先导出项目需要的依赖库：pip freeze > requirements.txt

然后安装依赖库：pip install -r requirements.txt

4. 上传工程文件，本例中上传到目录：/usr/local/deploy/demo

2. uwsgi

首先，切换到python3，

输入：pip install uwsgi

然后，检测是否安装成功，
进入到目录/usr/local/deploy/demo中，

输入：uwsgi --http :8001 --module demo.wsgi

进入浏览器该端口下是否运行正常。

输入：ctrl + c 退出

但是，这种方式依赖于工程本身的配置文件，耦合太高，我采用了配置文件的方式，新建并进入文件/usr/local/deploy/scripts。

vim uwsgi.ini

在文件中输入以下内容，并保存。

```
# uwsig使用配置文件启动
[uwsgi]
# 项目目录
chdir=/usr/local/deploy/demo
# 指定项目的application
module=demo.wsgi:application
# 指定sock的文件路径
socket=/usr/local/deploy/scripts/uwsgi.sock
socket-timeout=360
# 进程个数
```

```
workers=5
pidfile=/usr/local/deploy/scripts/uwsgi.pid
# 指定IP端口
http=:8001
# 指定静态文件
static-map=/static=/usr/local/deploy/demo/static
# 启动uwsgi的用户名和用户组
uid=root
gid=root
# 启用主进程
master=true
# 自动移除unix Socket和pid文件当服务停止的时候
vacuum=true
# 序列化接受的内容，如果可能的话
thunder-lock=true
# 启用线程
enable-threads=true
# 设置自中断时间
harakiri=360
harakiri-verbose=true

# 设置内存限制
limit-as=3096
# 设置缓冲
post-buffering=4096
# 设置日志目录
daemonize=/usr/local/deploy/scripts/uwsgi.log
```

切换到/usr/local/deploy/demo项目目录下,依次执行：

```
python manage.py makemigrations
python manage.py migrate
python manage.py collectstatic
```

启动uwsgi：

```
启动：
/root/.pyenv/shims/uwsgi --ini /usr/local/deploy/scripts/uwsgi.ini
重启：
/root/.pyenv/shims/uwsgi --reload /usr/local/deploy/scripts/uwsgi.pid
关闭：
/root/.pyenv/shims/uwsgi --stop /usr/local/deploy/scripts/uwsgi.pid
```

查看8001端口，看是否启动好了。查看日志：

```
tail -500f /usr/local/deploy/scripts/uwsgi.log
```

3. nginx

安装nginx

```
yum install -y nginx
```

测试：

```
定位：cd /usr/sbin
启动：./nginx
停止：./nginx -s stop
重启：./nginx -s reload
```

修改配置文件nginx.conf：

```
vim /etc/nginx/nginx.conf
```

编辑后内容如下（主要是删除了没必要的注释及server部分）：

```
# For more information on configuration, see:
#   * Official English Documentation: http://nginx.org/en/docs/
#   * Official Russian Documentation: http://nginx.org/ru/docs/

user nginx;
worker_processes auto;
error_log /var/log/nginx/error.log;
pid /run/nginx.pid;

# Load dynamic modules. See /usr/share/nginx/README.dynamic.
include /usr/share/nginx/modules/*.conf;

events {
    worker_connections 1024;
}

http {
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile            on;
    tcp_nopush          on;
    tcp_nodelay         on;
    keepalive_timeout   65;
    types_hash_max_size 2048;


    include             /etc/nginx/mime.types;
    default_type        application/octet-stream;

    # Load modular configuration files from the /etc/nginx/conf.d directory.
    # See http://nginx.org/en/docs/ngx_core_module.html#include
    # for more information.
    include /etc/nginx/conf.d/*.conf;


}
```

新增配置文件demo.conf:

```
touch /etc/nginx/conf.d/demo.conf
vim /etc/nginx/conf.d/demo.conf
```

编辑并保存，内容如下：

```
server { # 这个server标识我要配置了
        listen 8000; # 我要监听那个端口
        server_name xxx.xxx.xxx.xxx; # 你访问的路径前面的url名称
        access_log /var/log/nginx/access.log main; # Nginx日志配置
        charset utf-8; # Nginx编码
        gzip_types text/plain application/x-javascript text/css text/javascript application/x-httpd
php application/json text/json image/jpeg image/gif image/png application/octet-stream; #
持压缩的类型


        error_page 404 /404.html; # 错误页面
        error_page 500 502 503 504 /50x.html; # 错误页面

        # 指定项目路径uwsgi
        location / { # 这个location就和咱们Django的url(r'^admin/', admin.site.urls),
        uwsgi_send_timeout 300;        # 指定向uWSGI传送请求的超时时间，完成握手后向uWSGI传
请求的超时时间。
        uwsgi_connect_timeout 300;   # 指定连接到后端uWSGI的超时时间。
        uwsgi_read_timeout 300;        # 指定接收uWSGI应答的超时时间，完成握手后接收uWSGI应
的超时时间

        include uwsgi_params; # 导入一个Nginx模块他是用来和uWSGI进行通讯的
        uwsgi_pass unix:/usr/local/deploy/scripts/uwsgi.sock; # 指定uwsgi的sock文件所有动态请
就会直接丢给他
        }

        # 指定静态文件路径
        location /static {
        alias /usr/local/deploy/demo/static/;
        }
}
```

启动nginx：

```
cd /usr/bin
./nginx
```

查看日志：

```
运行日志：tail -500f /var/log/nginx/access.log
错误日志：tail -500f /var/log/nginx/error.log
```

访问8000端口，看是否正常。

4. 踩坑记录

4.1 uwsgi无法访问静态文件

解决办法：启动前，运行python manage.py collecstatic

4.2 uwsgi超过10s自动发送重复任务

解决办法：修改harakiri参数改大些。

4.3 uwsgi无法访问部分css文件，报404.

解决办法：排查后发现引入css时是这样的

<link rel="stylesheet" href={% static "report/css/bootstrap.css" %} />

django解析后成了：[localhost/report/css/boostrap.css/]，而不是[localhost/report/css/boostrap.css]，导致无法找到资源，改为

<link rel="stylesheet" href='{% static "report/css/bootstrap.css" %}' />

4.4 长连接超过一分钟报50x或者0错误

解决办法：这是由于请求应答时间过长造成的，需要在demo.conf中设置uwsgi_read_timeout参数，将它设置长一些

5. 参考资料

---

https://blog.csdn.net/hshl1214/article/details/46789969
https://www.cnblogs.com/chenice/p/6921727.html
https://segmentfault.com/q/1010000006068317
http://www.mynetcow.com/show_art.php?id=383【安装nginx遇到的报错】