一、部署方案一：使用 Django 自带服务器部署

在 mysite 目录下是用 python 执行命令：

python manage.py runserver 0.0.0.0:8000 即可

二、部署方案二：使用 nginx +uwsgi 部署：

linux 上配置环境

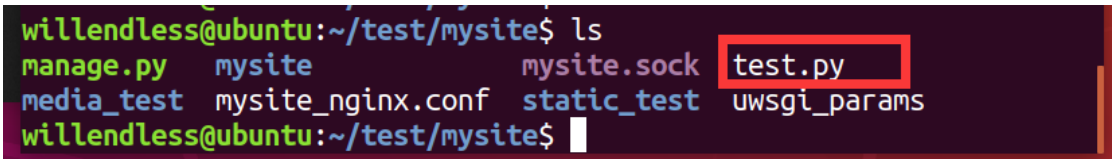
1. 安装 Django

   pip install Django

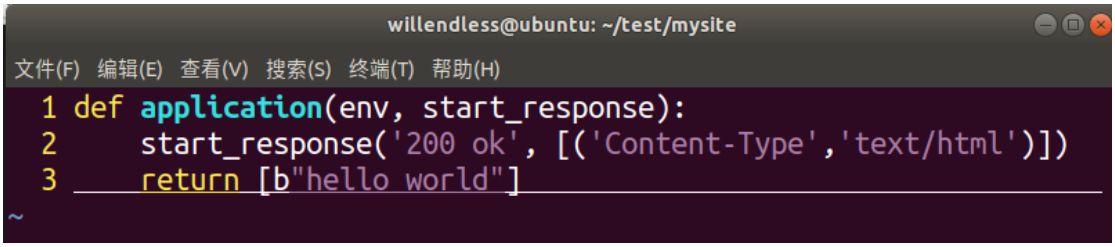
   django-admin.py startproject mysite（创建项目）

2. 安装 uwsgi

   创建测试文件

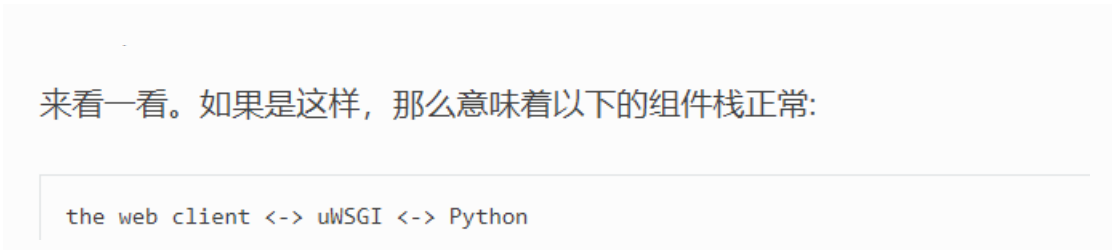


   内容：

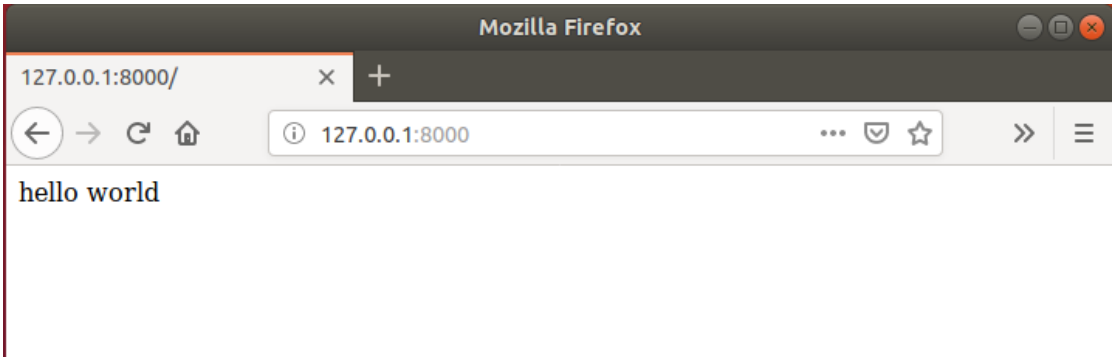


3. uWSGI 联通性测试

   a) Uwsgi –http :8000 --wsgi-file test.py



   访问 127.0.0.1:8000



B）使用 uwsgi 测试 Django 项目

测试你的Django项目

现在，我们想让uWSGI做同样的事，但是返回一个Django站点而不是 `test.py` 模块。

如果你还没有这样做，那么请确保你的 `mysite` 项目实际上正常工作:

```
python manage.py runserver 0.0.0.0:8000
```

而如果正常，则使用uWSGI来运行它:

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

- `module mysite.wsgi`: 加载指定的wsgi模块

将你的浏览器指向该服务器；如果站点出现，那么意味着uWSGI可以为你虚拟环境中的Django应用服务，而这个栈工作正常:

```
the web client <-> uWSGI <-> Django
```

4. 安装 nginx

sudo apt-get install nginx

nginx 的启动和安装:

1. Service nginx start
2. Service nginx stop

-检查 nginx 是否正常：通过端口 80 访问，应有 Welcome to nginx 页面


```
the web client <-> the web server
```

5. 配置 nginx

为你的站点配置nginx

你会需要 `uwsgi_params` 文件，可用在uWSGI发行版本的 `nginx` 目录下，或者从 https://github.com/nginx/nginx/blob/master/conf/uwsgi_params 找到。

将其拷贝到你的项目目录中。一会儿，我们将告诉nginx引用它。

现在，创建一个名为mysite_nginx.conf的文件，然后将这个写入到它里面:

```
# 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      8000;
    # the domain name it will serve for
    server_name .example.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 /path/to/your/mysite/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     /path/to/your/mysite/uwsgi_params; # the uwsgi_params file you installed
    }
}
```

这个配置文件告诉nginx提供来自文件系统的媒体和静态文件，以及处理那些需要Django干预的请求。对于一个大型部署，让一台服务器处理静态/媒体文件，让另一台处理Django应用，被认为是一种很好的做法，但是现在，这样就好了。

将这个文件链接到/etc/nginx/sites-enabled，这样nginx就可以看到它了:

```
sudo ln -s ~/path/to/your/mysite/mysite_nginx.conf /etc/nginx/sites-enabled/
```

-配置静态文件：例如图像文件，创建 static 文件夹并将相关文件放入其中

部署静态文件

在运行nginx之前，你必须收集所有的Django静态文件到静态文件夹里。首先，必须编辑mysite/settings.py，添加:

```
STATIC_ROOT = os.path.join(BASE_DIR, "static/")
```

然后运行

```
python manage.py collectstatic
```

6、配置 nginx 和 uwsgi 通信

nginx和uWSGI以及test.py

让nginx对 `test.py` 应用说句"hello world"吧。

```
uwsgi --socket :8001 --wsgi-file test.py
```

这几乎与之前相同，除了这次有一个选项不同：

- `socket :8001`：使用uwsgi协议，端口为8001

同时，已经配置了nginx在那个端口与uWSGI通信，而对外使用8000端口。访问:

http://example.com:8000/

来检查。而这是我们的栈:

```
the web client <-> the web server <-> the socket <-> uWSGI <-> Python
```

同时，你可以试着看看在http://example.com:8001的uswgi输出 - 但很有可能它不会正常工作，因为你的浏览器使用http，而不是uWSGI，但你应该能够在终端上看到来自uWSGI的输出。