**Django+Uwsgi+Nginx项目部署文档**

**WSGI**

在生产环境中使用WSGI作为python web的服务器

WSGI：全拼为Python Web服务器网关接口，Python Web服务器网关接口，是项目默认会生成一个wsgi.py文件，确定了设置模块

uWSGI实现了WSGI的所有接口，是一个快速，自我修复，开发人员和系统管理员友好的服务器，C语言编写，效率高

**Nginx**

使用nginx的的作用主要包括负载均衡，反向代理

项目通过Django+Uwsgi+Nginx进行线上服务器部署

**1、文件打包传服务器，通过xshell：**

文件 > 传输 > ZMODEM > 用ZMODEM发送 > 文件或压缩包

Linux下压缩包解压命令:

zip格式 : unzip 压缩包路径

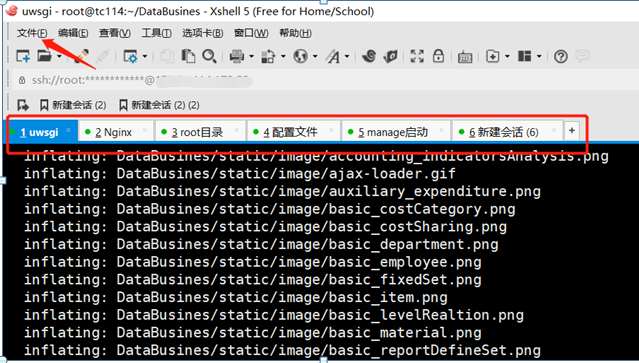
tar.gz格式 ： tar zxvf 压缩包路径

(rar格式压缩包解压较为复杂，尽量别传rar格式)

**2、Xshell使用技巧：**

文件 > 新建，开启多终端

建议开多终端，这样对uwsgi、Nginx、项目代码进行调试修改时，可以避免在一个终端下来回切换目录，提高工作效率，具体开终端的个数根据实际需求来定，并且右击tab终端名重命名，更加方便知道哪个终端对应做哪些事情



**3、修改配置文件问题**

不管修改uwsgi的配置文件uwsgi.ini还是修改nginx配置文件nginx.conf，修改完都必须重启服务才能生效，并且启动服务要在指定的目录下面重启

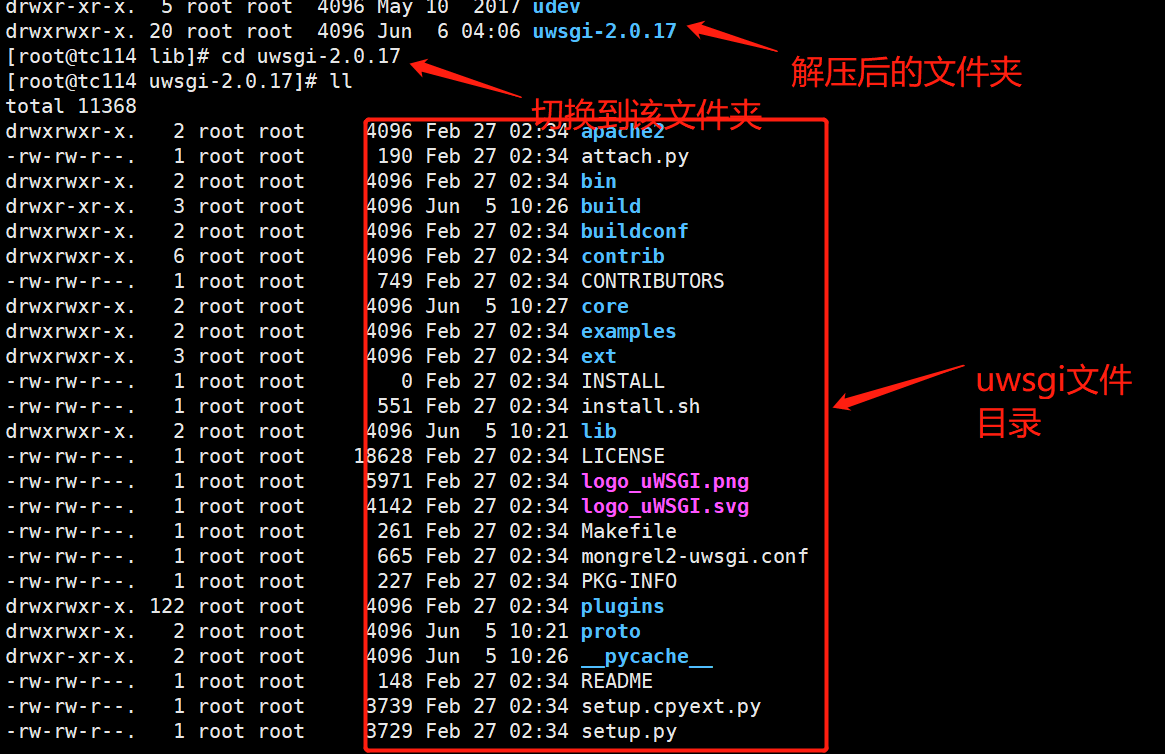
**4、Uwsgi的安装**

方法1、pip install uwsgi（有网的情况下）

方法2、没网情况下去官网下载uwsgi压缩包，为tar.gz格式，传到服务器，进行解压，解压路径/lib/目录下面，然后切换到uwsgi文件目录，执行以下两个命令，即可完成安装，示意图如下（解压路径可以自定义，记下来，方便以后进行维护）

python setup.py build

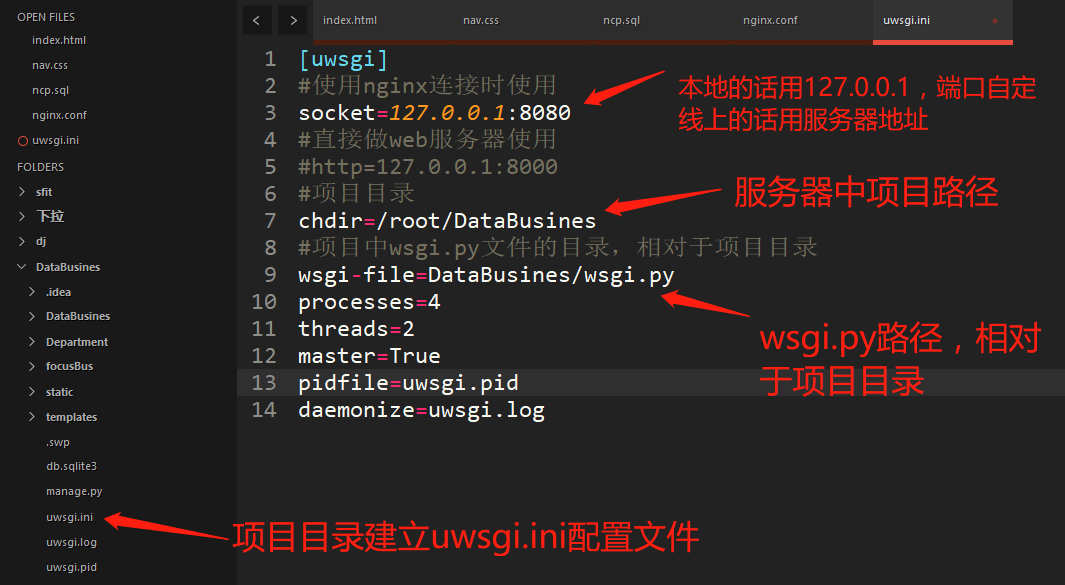
python setup.py install



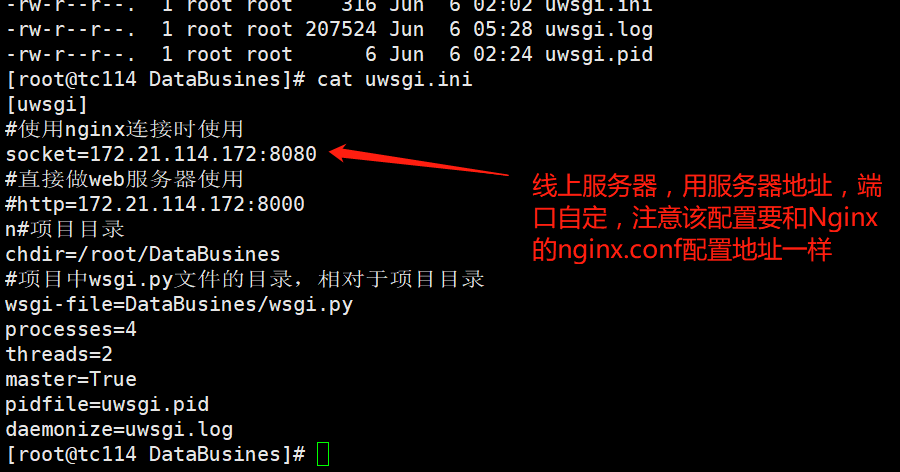
**5、Django项目中配置uwsgi**

1、项目目录（例如本例中DataBusines）下创建uwsgi.ini文件，配置如下

本地测试一般用：127.0.0.1即可，端口可以自定



线上的话用线上服务器IP，端口自定，该配置访问地址需要和nginx.conf中的配置一样



例如：这是后面的nginx.conf配置文件，两者地址和端口需要一致



**6、Uwsgi的使用（启动、查看进程、关闭）**

启动uwsgi.ini,需要切换到项目目录

启动uwsgi： uwsgi --ini uwsgi.ini

查看uwsgi进程：ps ajx|grep uwsgi

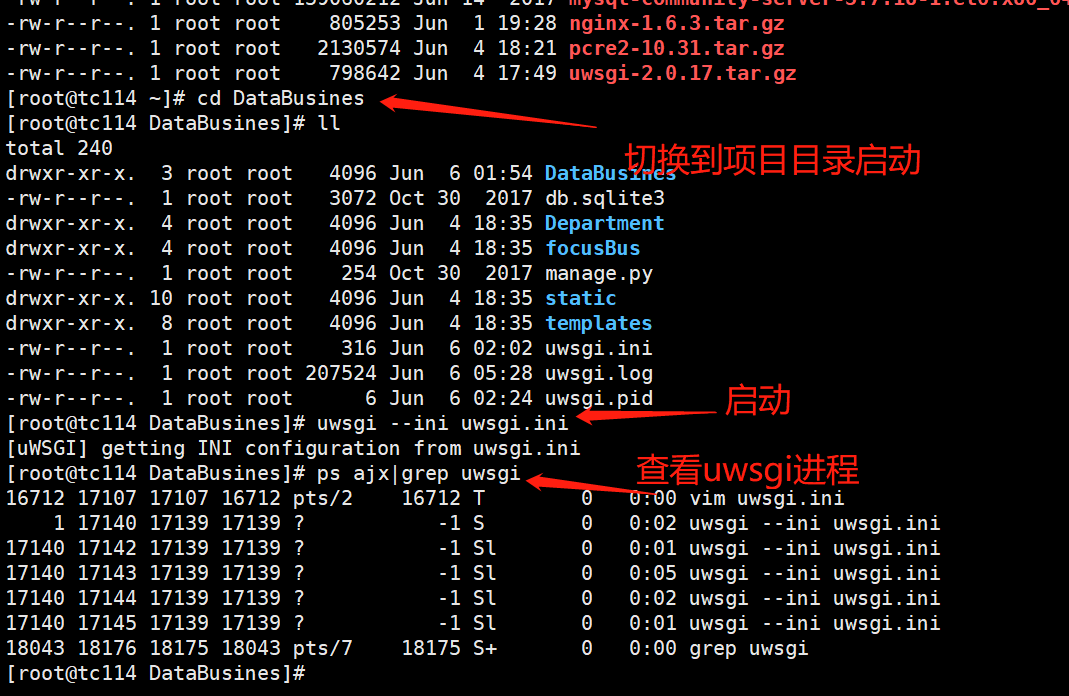
关闭uwsgi：

查阅相关资料文档，提到多种命令关闭方式，关闭命令的意义在于修改配置文件后，一般需要重启uwsgi才会生效

1、uwsgi --stop uwsgi.pid（不好用，经常报pid找不到）

2、sudo pkill -f uwsgi -9（不好用，有可能报错，无效的-9）

3、killall -9 uwsgi（该命令最好用）



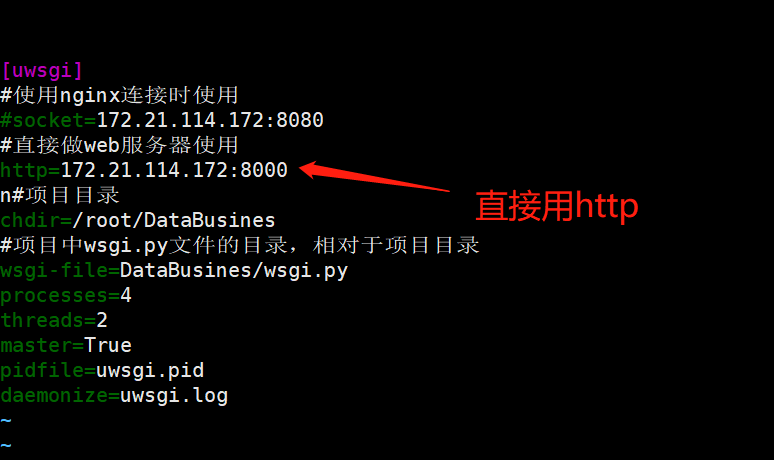
**7、通过uwsgi网页访问**

因为uwsgi本身就是web服务器，我们可以通过更改配置直接通过uwsgi进行访问网页

如下图：我们在服务器通过vi更改配置文件为http请求模式，更改后保存并重启uwsgi服务器，在我们自己的浏览器访问设置的IP和端口，成功显示页面，证明uwsgi配置成功

http模式: 直接用uwsgi时使用

socket模式: 使用Nginx时使用





**8、Nginx的安装**

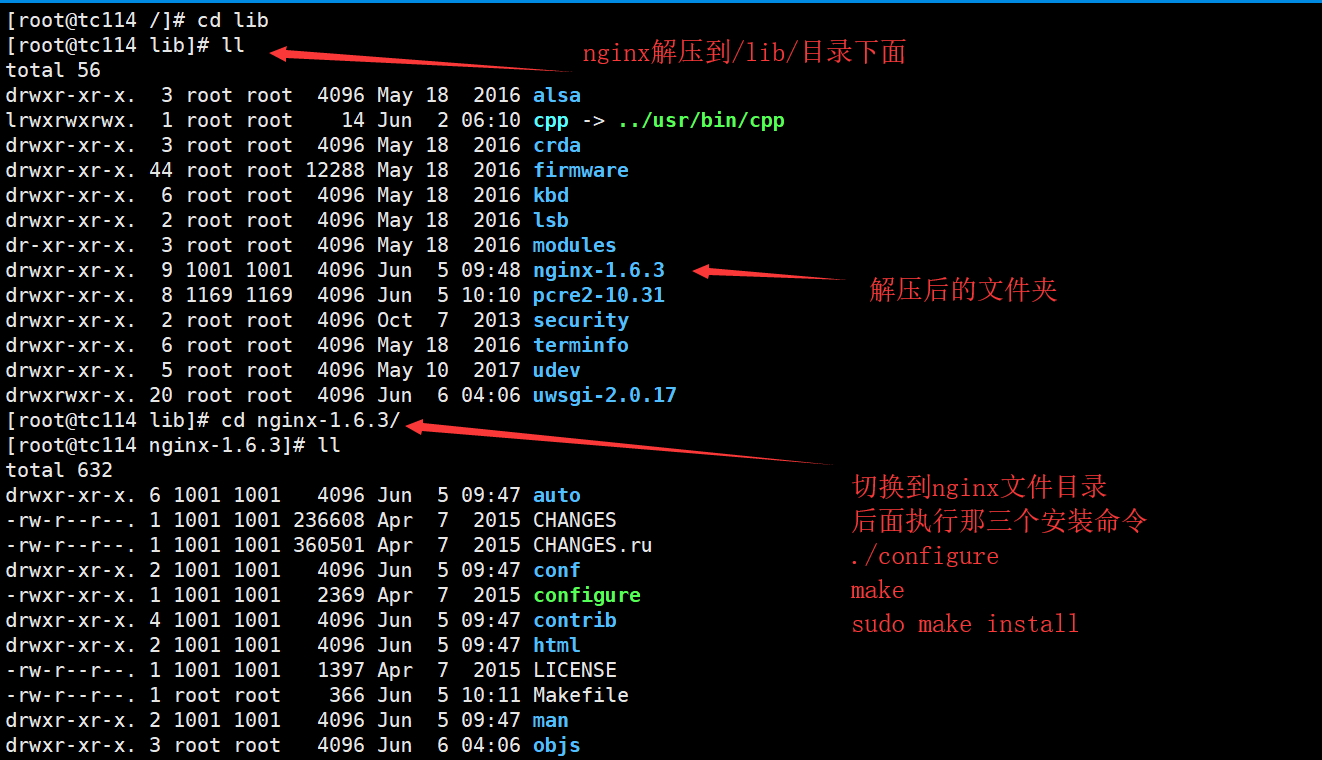
方法1、pip install nginx（官方提供有该方法，但是之前在本地测试遇到坑，没有配好，建议通过方法二中压缩包方式安装）

方法2、没网情况下去官网下载nginx压缩包，为tar.gz格式，传到服务器，进行解压，解压路径/lib/目录下面，然后切换到nginx文件目录，执行以下三个命令，进行安装

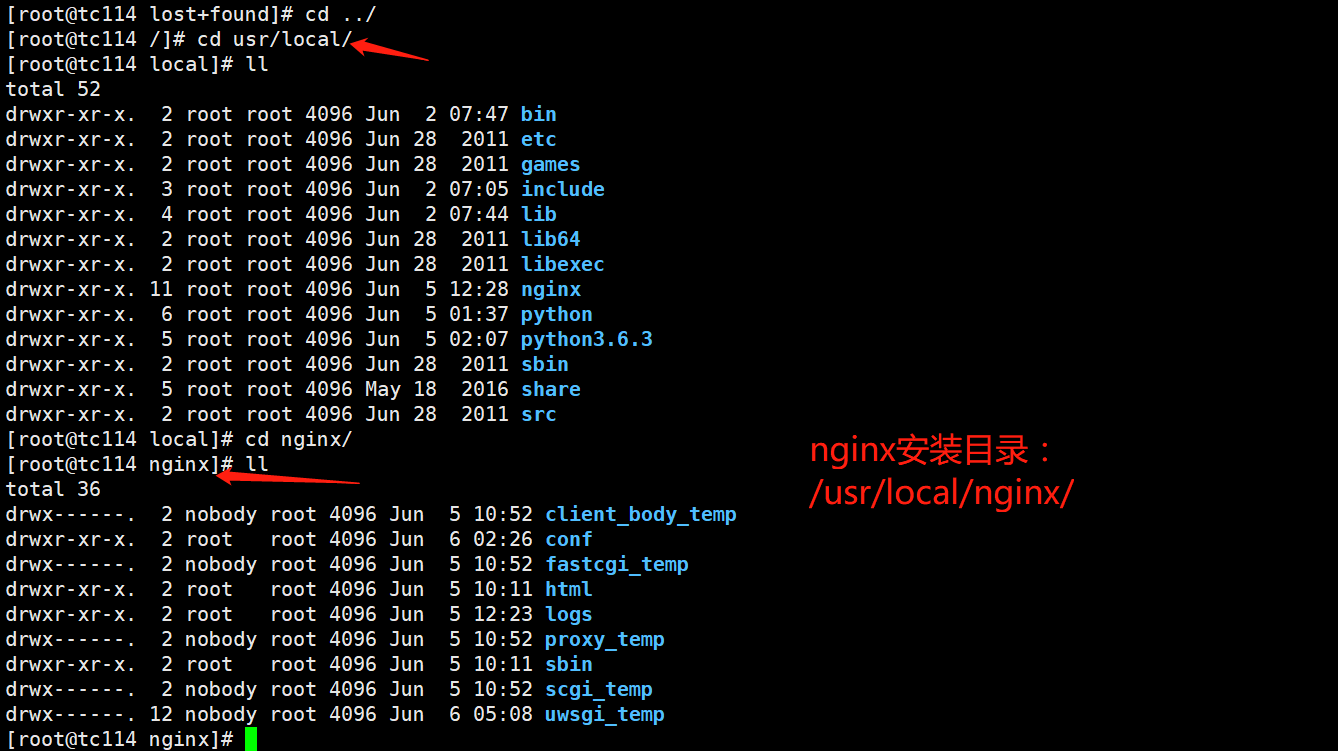
./configure

make

sudo make install



执行完以上命令后，nginx被安装在了/usr/local/nginx/，安装成功



**9、nginx的使用（启动、查看进程、关闭）**

进入nginx安装目录：cd /usr/local/nginx/

启动nginx： sudo sbin/nginx

查看nginx进程： ps ajx|grep nginx

关闭uwsgi：

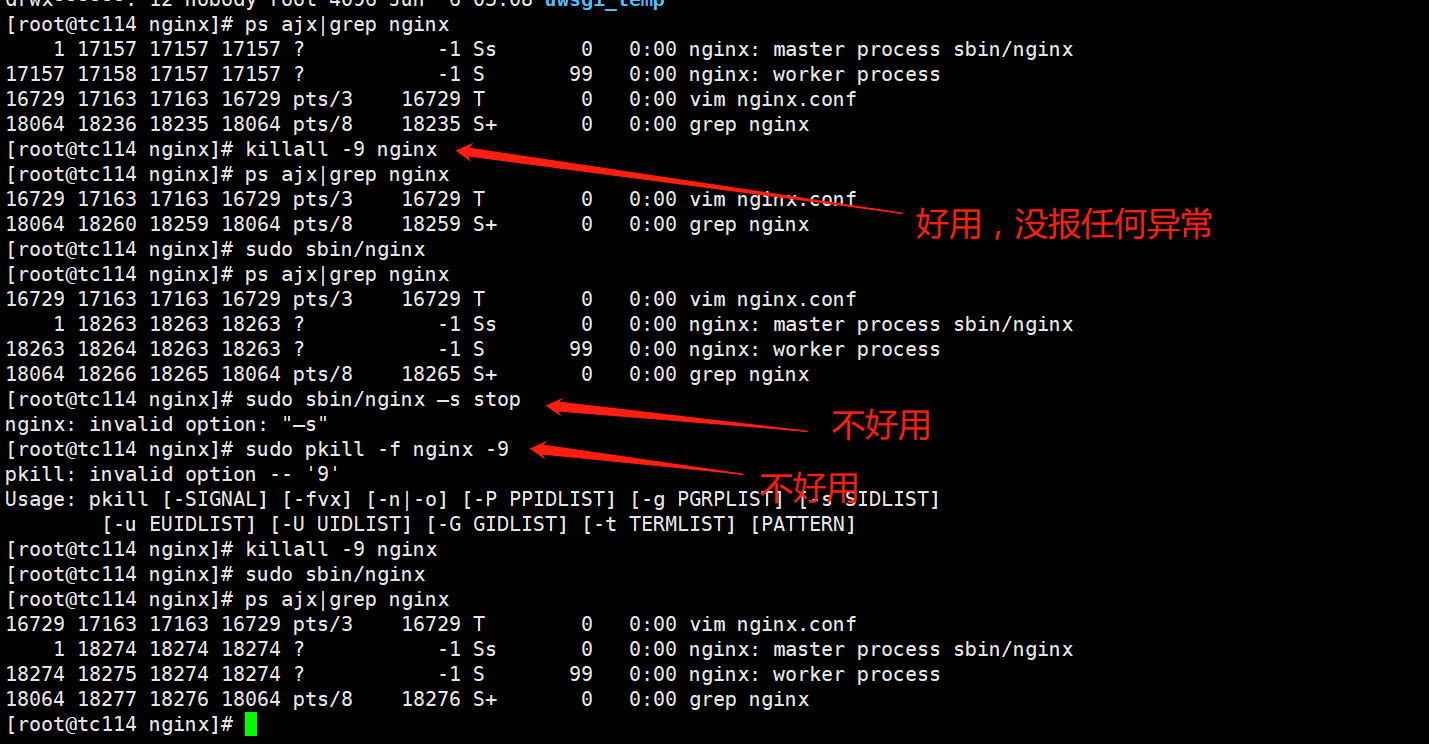
查阅相关资料文档，提到多种命令关闭方式，关闭命令的意义在于修改配置文件后，一般需要重启nginx才会生效

1、sudo sbin/nginx –s stop（不好用,报异常无效的-s）

2、sudo pkill -f uwsgi -9（不好用，报错，无效的-9）

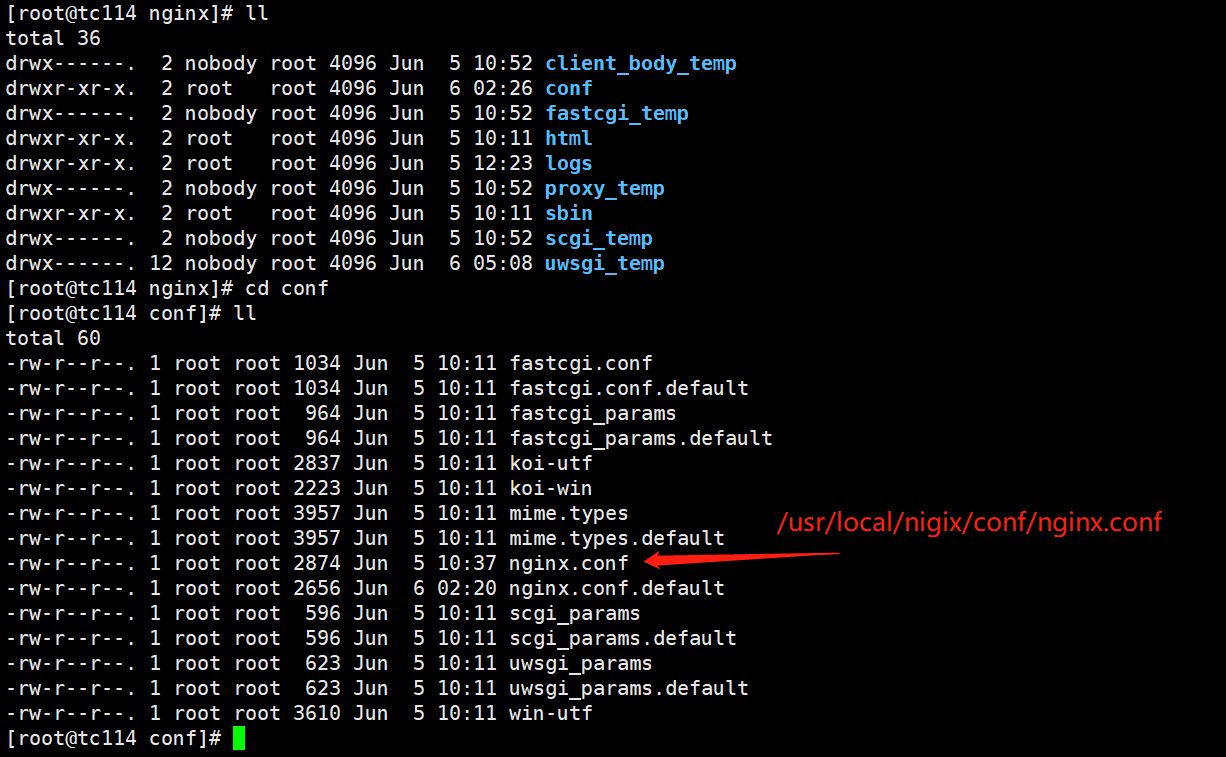
3、killall -9 nginx（该命令最好用）

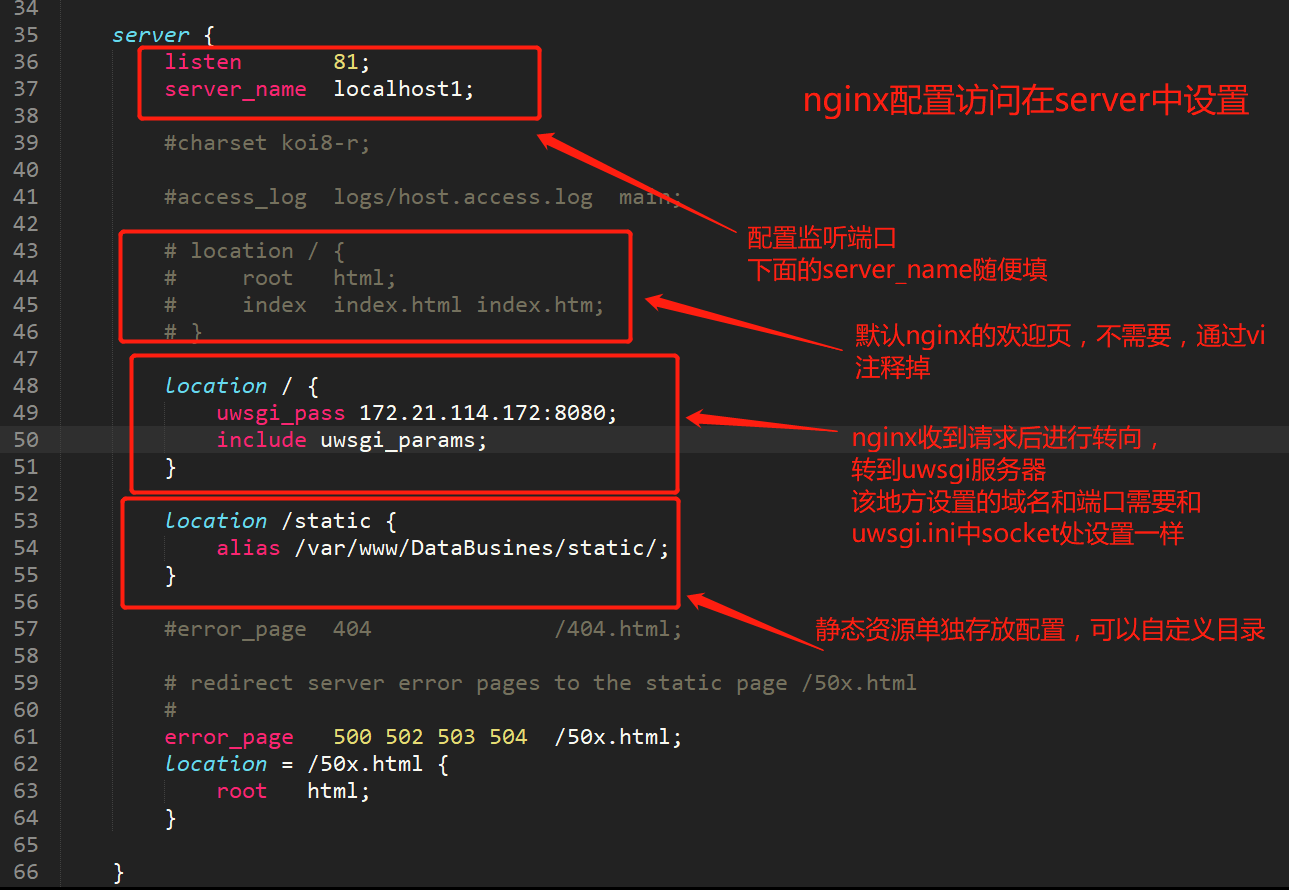
命令报错示意



**10、nginx的配置文件nginx.conf配置**

具体如图中所示，配置文件目录/usr/local/nginx/conf/nginx.conf,配置文件的修改需要通过vi进行修改

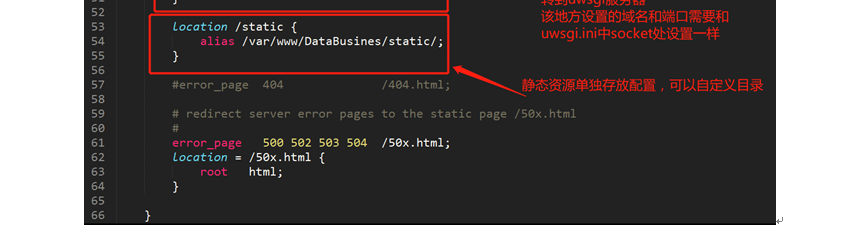




**11、静态资源配置**

配置静态资源目录是因为让静态资源通过nginx可以直接返回，不需要通过uwsgi，也就是让uwsgi只处理后端逻辑，不处理静态资源，优化性能

1、静态资源在nginx.conf中的配置，路径可以自定义



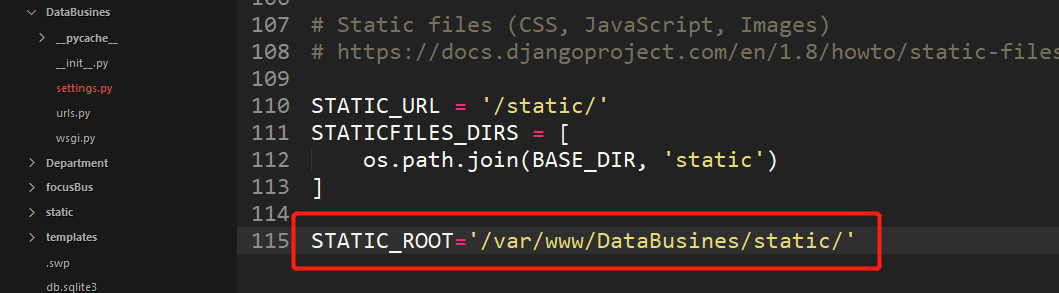
2、在服务器上创建如下目录

sudo mkdir –vp /var/www/DataBusines/static/

3、修改目录权限

sudo chmod 777 /var/www/DataBusiness/static/

4、项目代码中配置settings，加入该目录（本地演示在IDE中，线上可以用vi）



5、收集所有静态文件到static\_root指定目录

服务器上切换到项目目录（DataBusines），执行如下命令收集

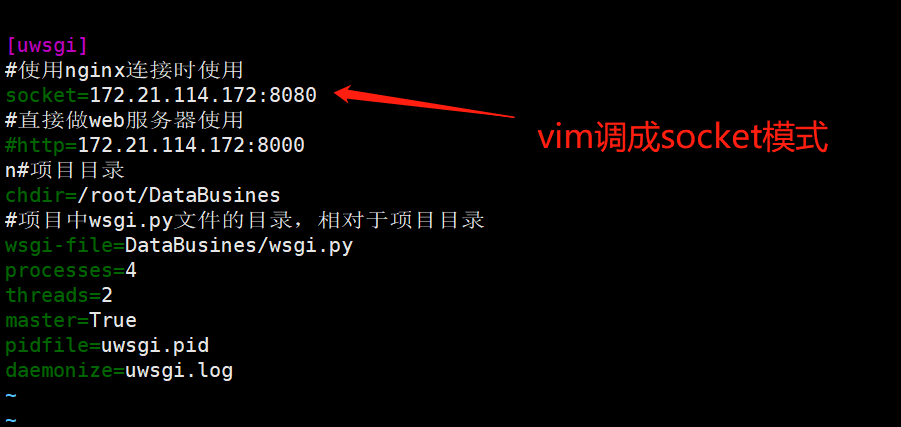
python manage.py collectstatic

6、查看静态资源目录

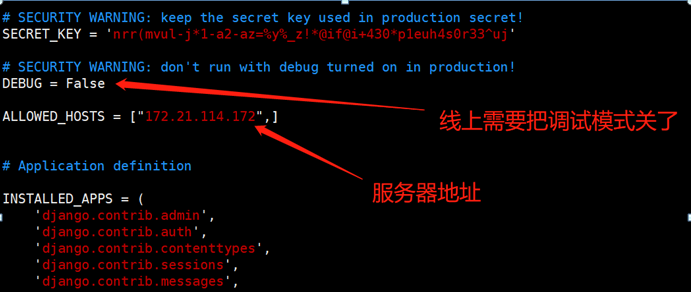


**12、更改uwsgi.ini配置**

刚才在做uwsgi时候，用了http配置，现在nginx正式搭建起来，需要改成socket配置，修改完毕要重启uwsgi



Settings.py需要debug设置为不调试，允许访问的地址设置为服务器地址



以上步骤完成后，访问服务器主机地址和端口，如果nginx.conf中配置的为80端口，则地址栏不需要输入端口，因为浏览器请求端口也是默认为80端口，非80端口的需要自己在ip后面添加

