# 第十七章 Django-博客开发-网站发布

本节所讲内容：

17.1 python环境部署

17.2 python uwsgi

17.3 django+nginx+uwsgi

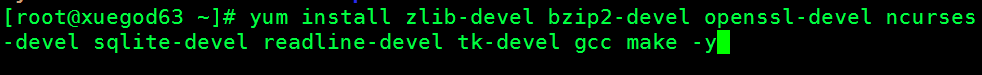
## 17.1 Python环境部署

我们今天学习的内容是如何将Django项目部署到linux服务器上，我们部署的linux系统是centos7首先，我们先在linux上搭建我们的Python3环境：

在这里首先强调一下，Centos7系统自带的Python2我们不要删除，我们要做的是在Python2和python3并存。

1. 安装Python3的依赖包
2. 命令：

[root@localhost Desktop]# yum install zlib-devel bzip2-devel openssl-devel ncurses-devel sqlite-devel readline-devel tk-devel gcc make -y



Python3 环境安装

[root@xuegod63 ~]#mkdir python37 #创建python37文件夹

[root@xuegod63 ~]#cd python37/ #切换到python37文件夹

#下载官方网站的tgz入门

[root@xuegod63 ~]yum install wget #安装wget

[root@xuegod63python37]#wget <https://www.python.org/ftp/python/3.7.2/Python-3.7.2.tar.xz> # 安装python3.7.2

[root@xuegod63 python37]# tar -xvJf  Python-3.7.2.tar.xz #解包

[root@xuegod63 python37]编译安装   ./configure prefix=/usr/local/python3   make && make install

注意：如果出现提示需要模块的Error，关于在centos下安装python3.7.0以上版本时报错ModuleNotFoundError: No module named '\_ctypes'的解决办法

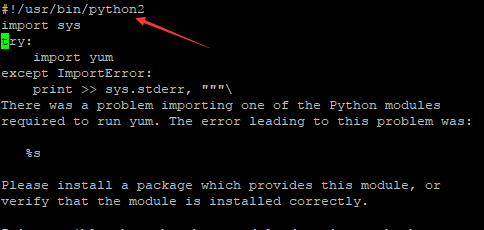
[root@xuegod63 python37]#yum install libffi-devel –y 然后再安装即可make install

然后设置python软连接，就像配置下windows下的path一个道理  
[root@xuegod63 ~]# ln -s /usr/local/python3/bin/python3 /usr/bin/python

因为执行yum需要python2版本，所以我们还要修改yum的配置，执行：

[root@xuegod63 ~]#vim /usr/bin/yum

把#! /usr/bin/python修改为#！/usr/bin/python2



同理 vi /usr/libexec/urlgrabber-ext-down 文件里面的#! /usr/bin/python 也要修改为#! /usr/bin/python2

在第一阶段中我们已经安装过python3.7.2所以我们直接拿来用就ok了

1. python-devel

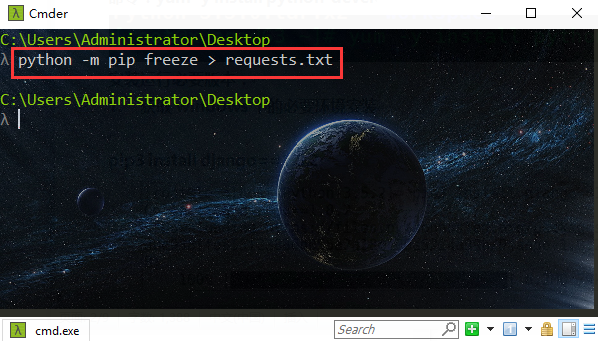
命令：yum -y install python-devel



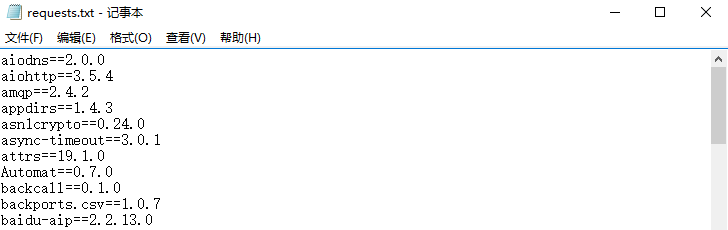
安装运行必要版本：

1. 获取windows下的必要环境安装

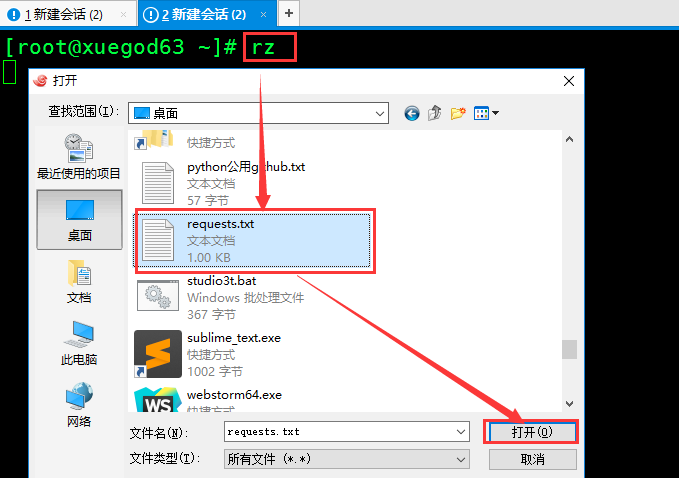
命令：python -m pip freeze > requests.txt

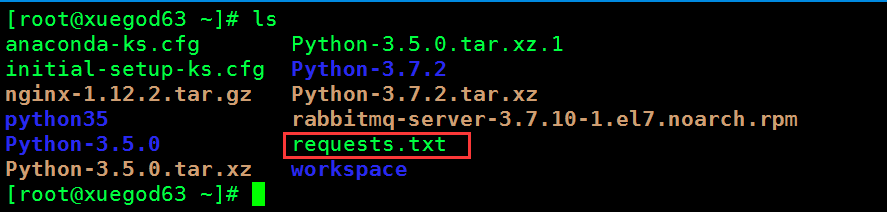


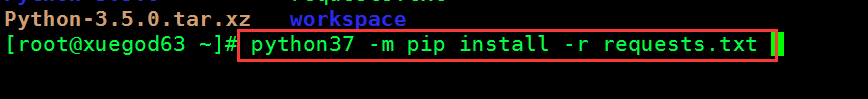
1. 打开保存到桌面的requests.txt文档 修改没必要的安装包



1. 修改没必要的第三方模块并传递给linux中







安装好所有依赖：python37 -m pip install -r requests.txt

17.1.2 运行项目

1.修改项目settings配置

在配置 uwsgi 时需要更改项目 settings.py 中的设置，更改项目如下

DEBUG = True  
ALLOWED\_HOSTS = ['\*']

但是到这里我们有个明显的问题,我们的mysql数据库指定的ip为127.0.0.1，数据存储在我的windows中（工作中常把数据存储到云服务中），所以说我们要修改settings.py中的数据库ip

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.mysql',

'NAME': 'dbname',

'HOST':'192.168.1.28',#这是windows ip地址

'PORT':'3306',

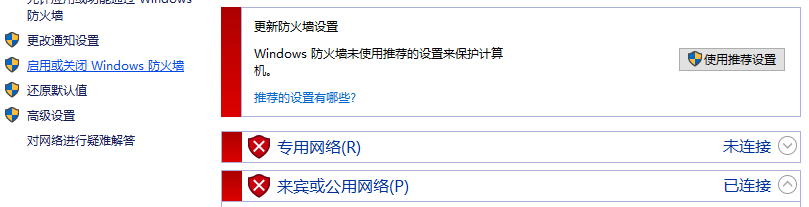
'USER':'root',

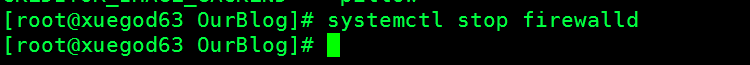
"PASSWORD":'123456'

}

}

既然有网络交互的行为，我们把windows和linux下的防火墙全部关闭

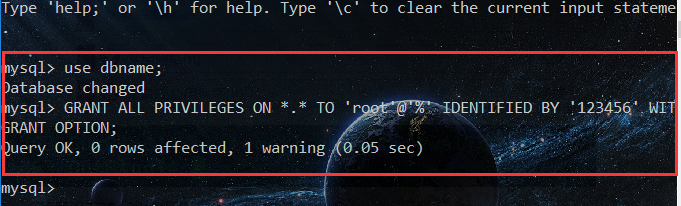




但是，明显我们linux还是没有访问权限所以说给当前IP访问权限

mysql> use dbname;

mysql> GRANT ALL PRIVILEGES ON \*.\* TO 'root'@'%' IDENTIFIED BY '123456' WITH GRANT OPTION;

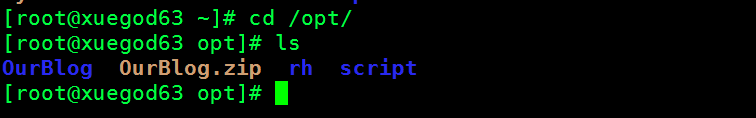


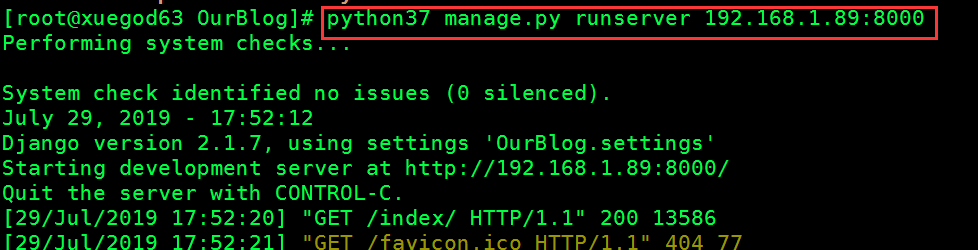
[root@xuegod63 ~]# cd /opt/

[root@xuegod63 ~]# rz

[root@xuegod63 ~]# unzip OurBlog.zip

结果如下：





运行结果如下：



注意：如果出现这个错误：(关于这个静态文件的路径我们后面会统一修改的)

ERRORS:

?: (staticfiles.E002) The STATICFILES\_DIRS setting should not contain the STATIC\_ROOT setting.

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

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

## 17.2 python uwsgi

上面我们已经完成了python+Django环境的部署，接下来我们需要了解python 的uwsgi框架。接下来我们需要了解python 的uwsgi模块。

Python manager.py runserver, 这是一款适合开发阶段使用的服务器，只支持最高200的并发，显然不适合在生产中使用。我们在生产环境当中，通常会用到apache或者nginx服务器作为Django的生产服务器。今天我们以nginx为例子。但是在学习部署之前，我们首先要掌握在Django和nginx之间的一个程序，就是wsgi，web服务器网关接口，英文叫， Web Server Gateway Interface 缩写叫WSGI，是Python应用程序或框架和web服务器之间的一种接口。

我们今天部署用的是uwsgi模块，首先安装

pip3 install uwsgi



然后配置环境变量

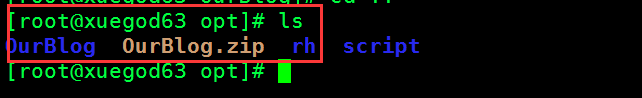
命令：ln /usr/local/python3.7/bin/uwsgi /usr/bin/uwsgi

注意标红的地方是你的python路径



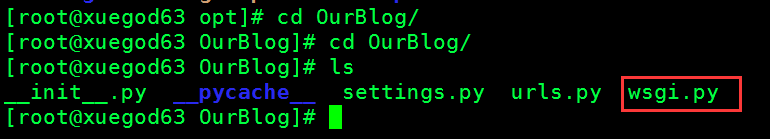
Django和WSGI的结合

我们把我们准备好的Django项目存放到/opt/下



然后尝试使用uwsgi启动Django项目

在这里我们要关注到之前大家一直忽视的一个文件了



紧接着修改OurBlog中的settings.py文件

DEBUG = Fasle  
ALLOWED\_HOSTS = ['\*']

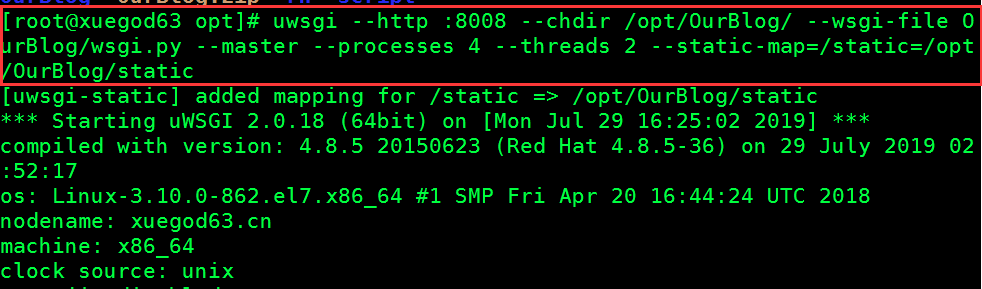
**静态文件重定位 STATIC\_ROOT = os.path.join(BASE\_DIR,‘/var/www’)**

再进行静态文件的转移 在项目目录下进行： Python manage.py collectstatic 这一步的顺序很重要，否则路径会有问题。

我们把静态文件迁移到/var/www中

这个就是我们项目的uwsgi文件

uwsgi --http :8080 --chdir /opt/OurBlog/ --wsgi-file OurBlog/wsgi.py --master --processes 4 --threads 2 --static-map=/static=/var/www



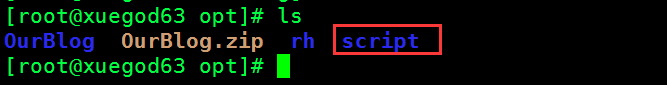
然后发起访问



## 17.3 django+nginx+uwsgi

上面我们用命令启动了uwsgi，但是体验不太好，所以我们在这里使用ini文件启动uwsgi服务器

在项目同等级目录下创建script目录



然后在当中编写uwsgi.ini文件，内容如下

[uwsgi]

socket = 127.0.0.1:9090 #在这个地方我们就写死

chdir=/opt/OurBlog

http= :8080

module=OurBlog.wsgi

master = true

processes=2

threads=2

max-requests=2000

chmod-socket=664

static-map = /static = /var/www

static-map = /media = /opt/OurBlog/media #配置静态文件

vacuum=true

daemonize = /opt/script/uwsgi.log

注意:也可以把上面代码修改如下

[uwsgi]

chdir=/opt/OurBlog #项目目录

module=OurBlog.wsgi:application #指定项目的application

socket=/opt/script/uwsgi.sock #指定sock的文件路径

workers=5 #进程个数

pidfile=/opt/script/uwsgi.pid

http=192.168.1.89:8000 #指定IP端口

static-map=/static=/opt/OurBlog/static #指定静态文件

uid=root #用户

gid=root #组

master=true #启用主进程

vacuum=true #自动移除unix Socket和pid文件当服务停止的时候

enable-threads=true #启用线程

thunder-lock=true #序列化接受的内容，如果可能的话

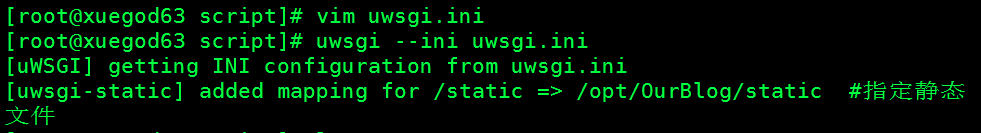
harakiri=30 #设置自中断时间

post-buffering=4096 #设置缓冲

daemonize=/opt/script/uwsgi.log #设置日志目录

然后启动uwsgi服务

uwsgi --ini uwsgi.ini(脚本名称)



然后可以正常访问



警告：操作到这一步的时候可能你会获取不到media和static以及文章，我们需要修改/opt/OuBlog/OurBlog/urls.py如下：

from django.contrib import admin

from django.urls import path, include,re\_path

from django.conf.urls import url

from django.views.static import serve

from OurBlog.settings import MEDIA\_ROOT

import xadmin

from django.conf.urls.static import static

from OurBlog import settings

from Article.views import \*

urlpatterns = [

path('xadmin/', xadmin.site.urls),

url(r'cheditor/',include('ckeditor\_uploader.urls')),

url(r'^$',index),

url('index/',index),

url('myArticle/',myArticle),

url('myPicture/',myPicture),

url('aboutMe/',aboutMe),

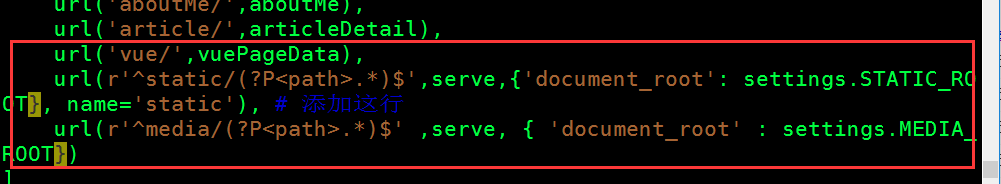
url('article/',articleDetail),

url('vue/',vuePageData),#这一行提上来

url(r'^static/(?P<path>.\*)$',serve,{'document\_root': settings.STATIC\_ROOT}, name='static'), # 添加这行

url(r'^media/(?P<path>.\*)$' ,serve, { 'document\_root' : settings.MEDIA\_ROOT})

]



如果还是出不来样式，请重启uwsgi

[root@xuegod63 script]# killall -9 uwsgi

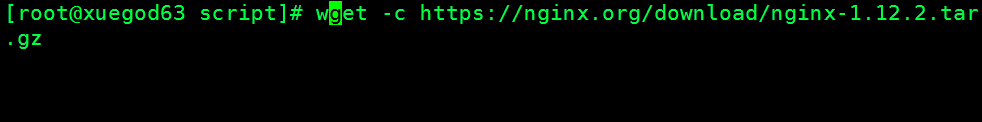
[root@xuegod63 script]# uwsgi --ini uwsgi.ini

再次刷新就OK了

17.3.2安装nginx服务器

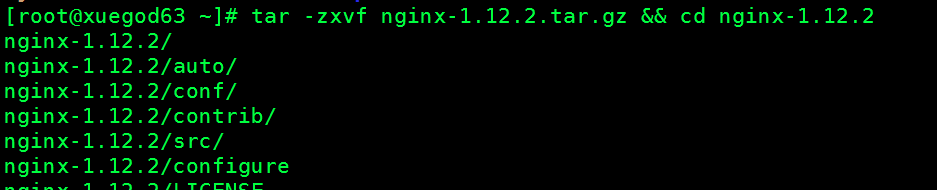
我们直接安装nginx是没有的，所以，需要我们wget

命令：wget -c https://nginx.org/download/nginx-1.12.2.tar.gz



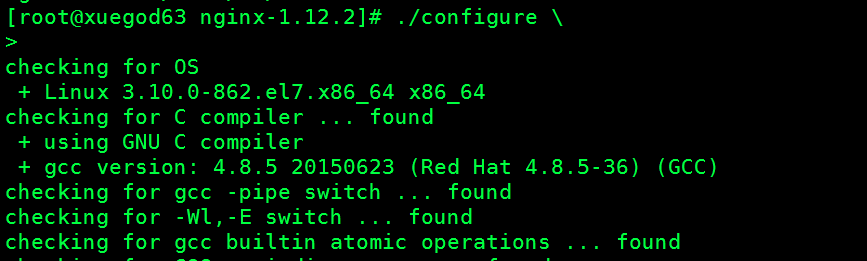
解压

命令：tar -zxvf nginx-1.12.2.tar.gz && cd nginx-1.12.2



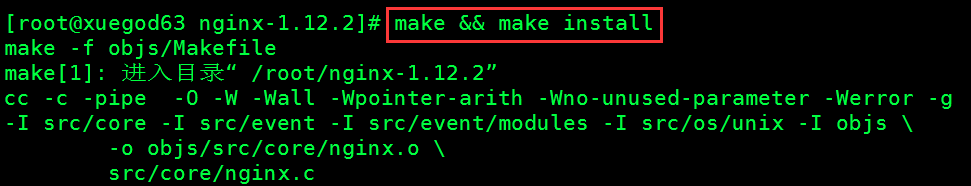
自定义配置

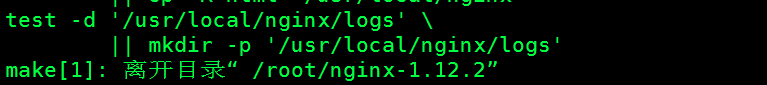
命令：./configure \



编译安装

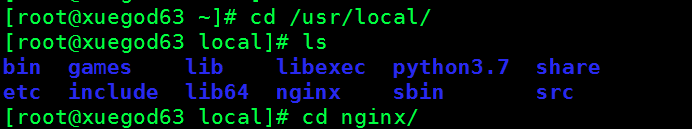
命令：make && make install





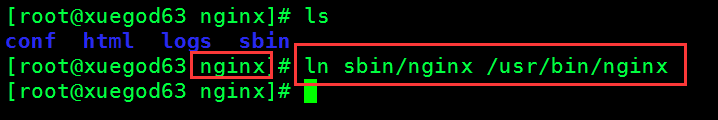
然后配置环境变量

命令：cd /usr/local/nginx

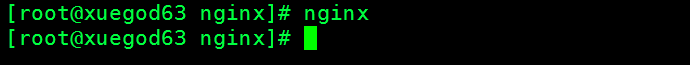


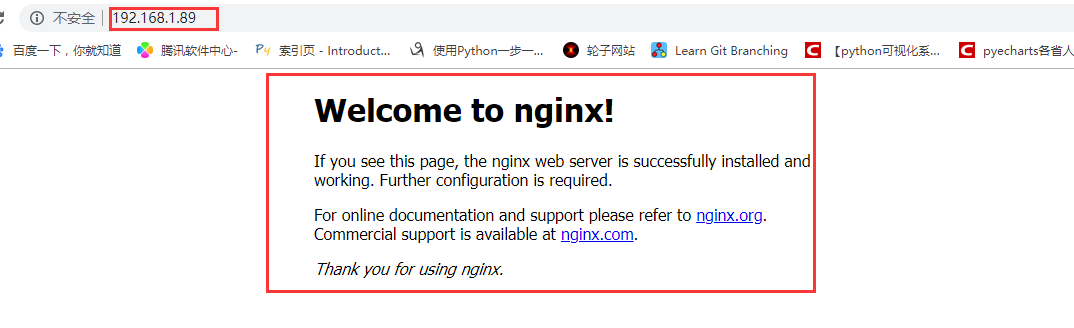
配置软连接

命令：ln sbin/nginx /usr/bin/nginx



启动nginx查看效果

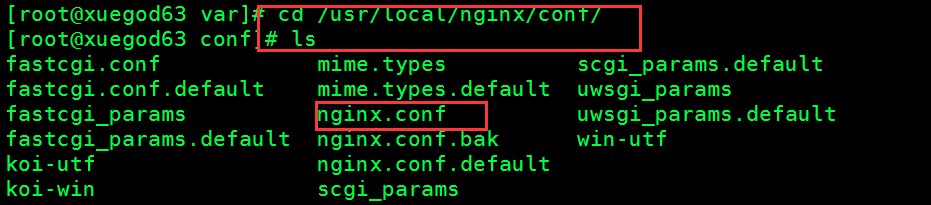




进行nginx配置，（配置nginx.conf，配置之前进行备份）

[root@xuegod63 var]# cd /usr/local/nginx/conf/

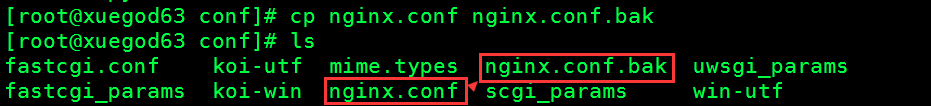
[root@xuegod63 conf]# ls



备份

[root@xuegod63 conf]# cp nginx.conf nginx.conf.bak

[root@xuegod63 conf]# ls



然后开始配置

只在server中配置

server {

listen 80;

server\_name localhost;

charset utf-8;

access\_log off;

location / {

include uwsgi\_params; #nginx加载uwsgi模块

uwsgi\_connect\_timeout 30; #连键超时时间

uwsgi\_pass 127.0.0.1:9090; #nginx对应的uwsgi socket文件

}

location /static {

alias /var/www; #静态文件路径

#index index.html index.htm; #首页

}

location /media {

alias /opt/OurBlog/media/;

}

运行代码：

#去除uwsgi进程

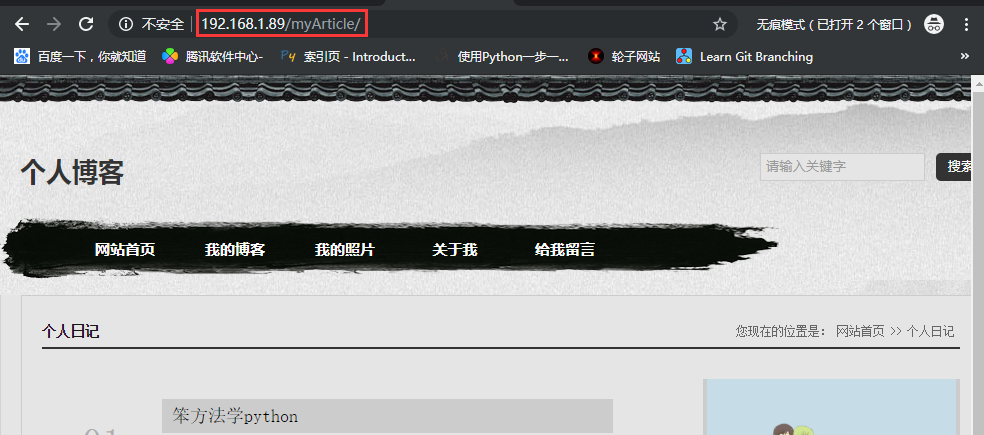
[root@xuegod63 conf]# killall -9 uwsgi

#重新加载uwsgi服务

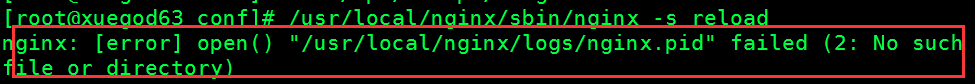
[root@xuegod63 conf]# uwsgi --ini /opt/script/uwsgi.ini

nginx重新加载

[root@xuegod63 conf]# /usr/local/nginx/sbin/nginx -s reload



如果在重启nginx中出现错误如下:



执行：

[root@xuegod63conf]#/usr/local/nginx/sbin/nginx -c /usr/local/nginx/conf/nginx.conf

再次执行就OK了

总结：

17.1 python环境部署

17.2 python uwsgi

17.3 django+nginx+uwsgi