# 准备工作

1. Setting.py 里设置关掉日志 DEBUG=FALSE ，当是True时，django帮我们把静态文件做处理，关了就django不会处理，导致找不到原来的静态文件
2. Setting.py ALLOWED\_HOSTS=[‘\*’]

2、Setting.py 里设置static\_root,并把该文件夹chmode 777

3、Python manage.py collectstatic把静态文件打包，注意修改存放静态文件的文件夹的权限为777

4、可能会修改nginx默认端口

# Nginx

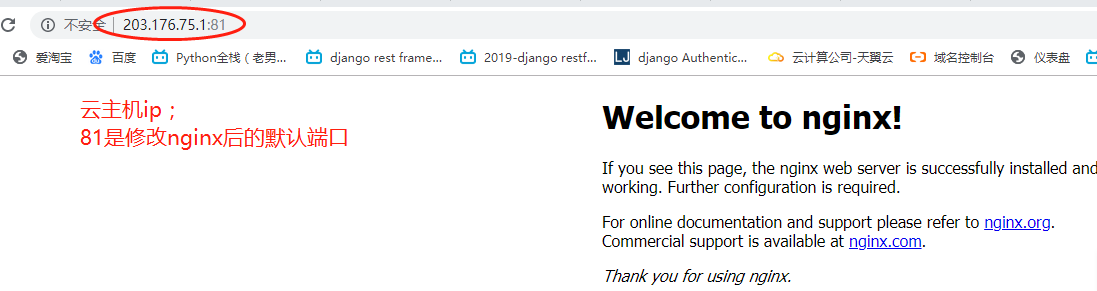
## 安装Nginx（apt-get默认安装）

sudo apt-get update

sudo apt-get install nginx

## 验证nginx

本地浏览里输入nginx所在服务器的ip，端口号，默认端口是80，如果80端口不可用，需要修改nginx的默认端口，见修改nginx默认端口章节



## Nginx配置

标准化:为了项目间的管理，在/etc/nginx/sites-available创建project\_name.conf，然后把它软链接到/etc/nginx/sites-enabled里

cd /etc/nginx 配置目录

vim nginx.conf 主配置文件

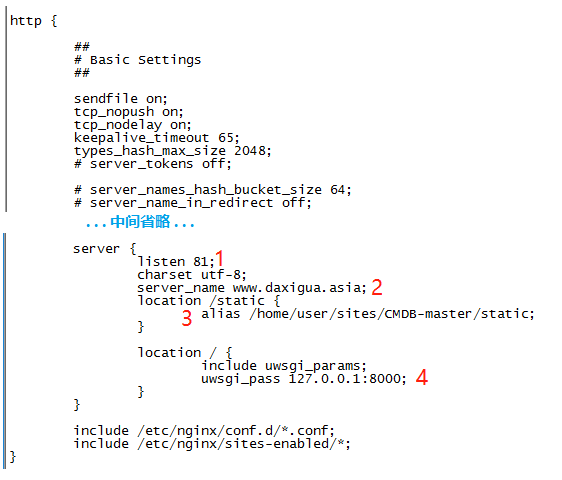
### 处理静态文件（集中与分开配置）

1.都配置在主配置文件里情况

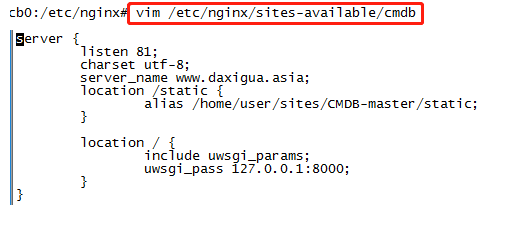
cd /etc/nginx 配置目录

vim nginx.conf 主配置文件

nginx.conf 内容如下图:



上图说明:

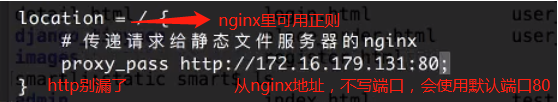
1. 可以配置ngixn默认端口，如这里我使用的81默认端口(我修改过nginx默认端口号)
2. 域名可换成ip（203.176.75.1），当为域名时，浏览器只能通过域名访问，不能通过ip 访问，当为ip时，只能通过ip访问，暂时不知道怎么配置
3. Alias处理静态文件，注意与root的区别，nginx经过处理后的静态地址，可以去uwsgi日志里去找，调试能通
4. 可换成ip（203.176.75.1），因为nginx与uwsgi在同ip上，所以此处可用127
5. Server 包含在http字典里，注意层级关系
6. 分开配置，配置与在主配置相同（例子）
7. touch /etc/nginx/sites-available/cmdb
8. 
9. 将available链接到enable去: ln -s /etc/nginx/sites-available/cmdb cmdb

### 反向代理

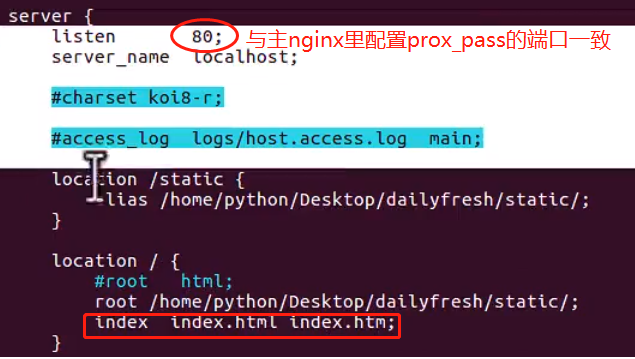
自己还没实践过，下面教程传递流程是:

浏览器请求/ --> 主nginx --> 从nginx 80端口的服务器 --> 找到从ngxin里配置的index --> 磁盘里的index页面

1. 主nginx里配置proxy\_pass，主ngxin内容如下:

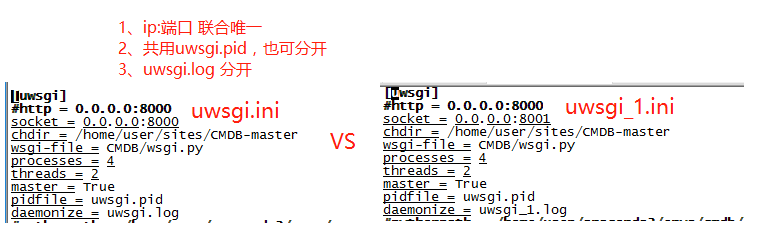


1. 从ngxin里配置好请求静态文件配置，从ngxin内容如下:



### 负载均衡

1. 项目里再创建个uwsgi.ini副本，uwsgi.ini与uwsgi.ini副本内容如下:



1. Ngxin里配置，这里是单独放文件里的，与放主配置文件里配置相同



1. 验证是否起作用
2. uwsgi --ini uwsgi.ini和uwsgi --ini uwsgi\_1.ini，启动uwsgi；
3. 启动nginx；
4. 分别查看uwsgi日志

tail -f /home/user/sites/CMDB-master/uwsgi.log

tail -f /home/user/sites/CMDB-master/uwsgi\_1.log

1. 浏览器里发送请求，看到请求是否交替出现在uwsgi.log、uwsgi\_1.log 中;

## 查看nginx日志



1. /var/log/nginx/access.log

每一个访问请求都会默认记录在这个文件中, 除非你做了其它设置.

1. tail -f /var/log/nginx/error.log

## 查看nginx占用的端口

但完全没有项目时，Nginx只会占用一个默认端口，且默认端口可改



## nginx常用命令

service nginx start

service nginx stop

service nginx restart

service nginx status 状态

sudo nginx -s reload #要检查配置文件是否错误，建议用这个替换

# Uwsgi

## 系统里安装uwsgi，conda时不用

virtualenv是基于系统的python版本，隔离出来的环境，环境里的python版本是系统已有的python版本

Pip install uwsgi 或 Pip3 install uwsgi ，conda虚拟环境里不要用这个安装方式，采用源码安装，方便各个虚拟环境里uwsgi与python版本绑定

如果报错，很可能是gcc版本不兼容导致。

gcc改变版本,遇到编译内核等问题可能会依赖于某个gcc版本，而ubuntu中自带的gcc版本过高或者过低，需要改变到合适的版本

sudo apt-get install gcc-4.7 安装4.7版本

sudo rm /usr/bin/gcc（删除已有软连接）

sudo ln -s /usr/bin/gcc-4.7 /usr/bin/gcc

gcc -v

## Conda里安装uwsgi（源码安装，方便一个虚拟环境里uwsgi对应自己的python）

1. wget <http://projects.unbit.it/downloads/uwsgi-latest.tar.gz>
2. tar zxvf uwsgi-latest.tar.gz
3. Cd uwsgi-2.0.18
4. python uwsgiconfig.py --build 注意因为是在conda虚拟环境里，如cmbd。所以此处的python就是cmdb里的python版本与uwsgi绑定
5. python setup.py install python解释同上
6. 在虚拟环境中查看uwsgi版本，可能会少文件，如找不到libpcre.so.1，用find命令查找是否有这文件，找到后做个链接

如:sudo ln -s /lib/x86\_64-linux-gnu/libpcre.so.3 /usr/lib/libpcre.so.1

/usr/lib/libpcre.so.1固定的，/lib/x86\_64-linux-gnu/libpcre.so.3 需要修改成世find后的实际路径

## 验证uwsgi

见百度

## 启动和停止uwsgi

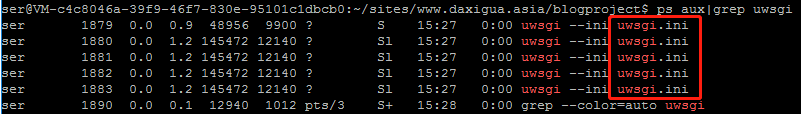
Uwsgi --ini uwsgi.ini 启动

Ps aux|grep uwsgi 查看进程

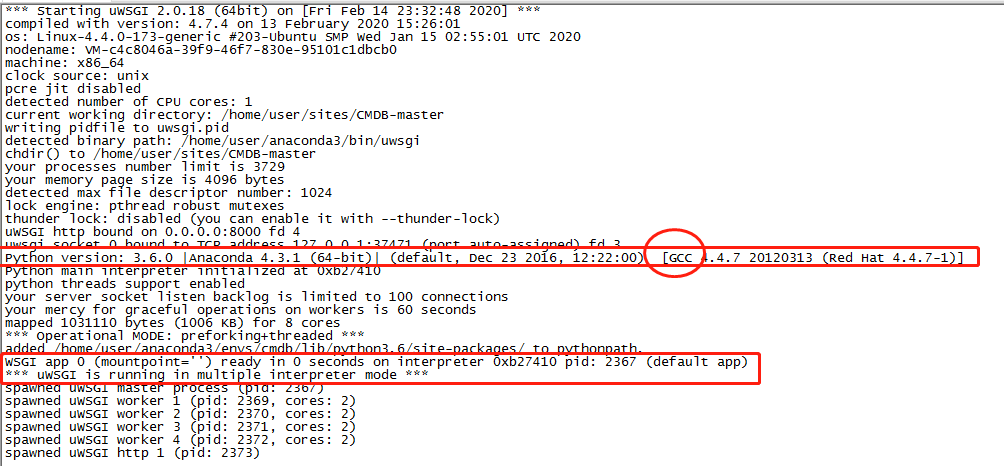
Uwsgi --stop uwsgi.pid 终止uwsgi进程

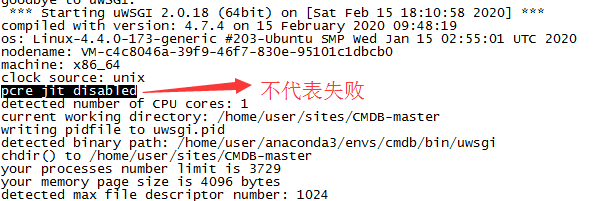
~~Uwsgi --reload uwsgi.ini 经过验证不起作用，不会重新加载配置~~

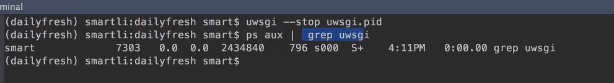




## 浏览器能成功请求的log

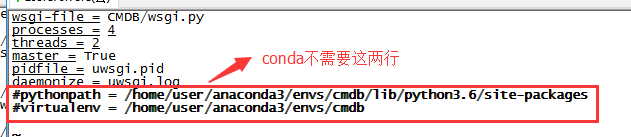






## Uwsgi.ini配置

项目根目录下新建uwsgi.ini，写入内容如下



[uwsgi]

http = 0.0.0.0:8000

chdir = /home/user/sites/CMDB-master

wsgi-file = CMDB/wsgi.py

processes = 4

threads = 2

master = True

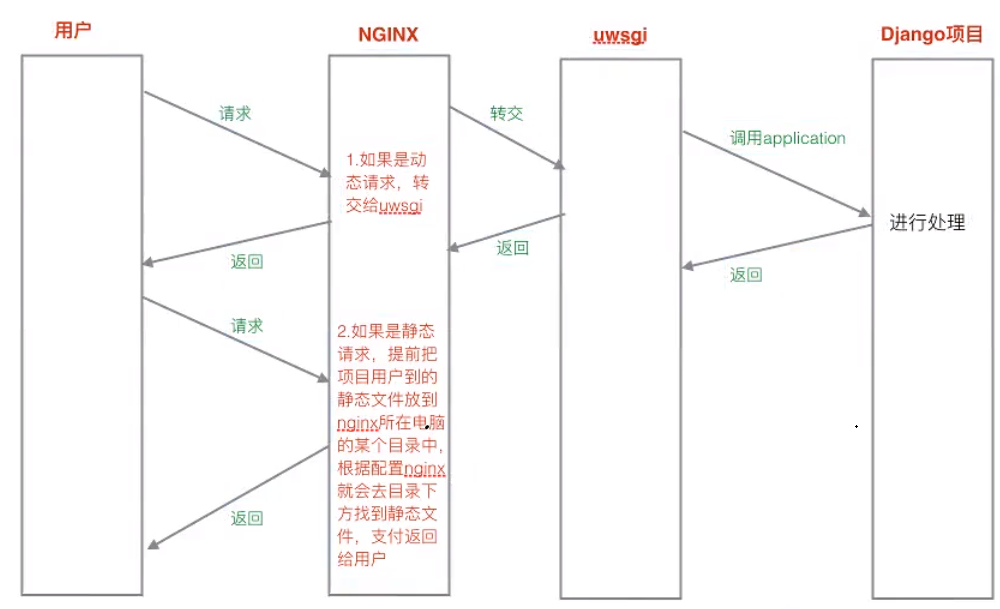
pidfile = uwsgi.pid

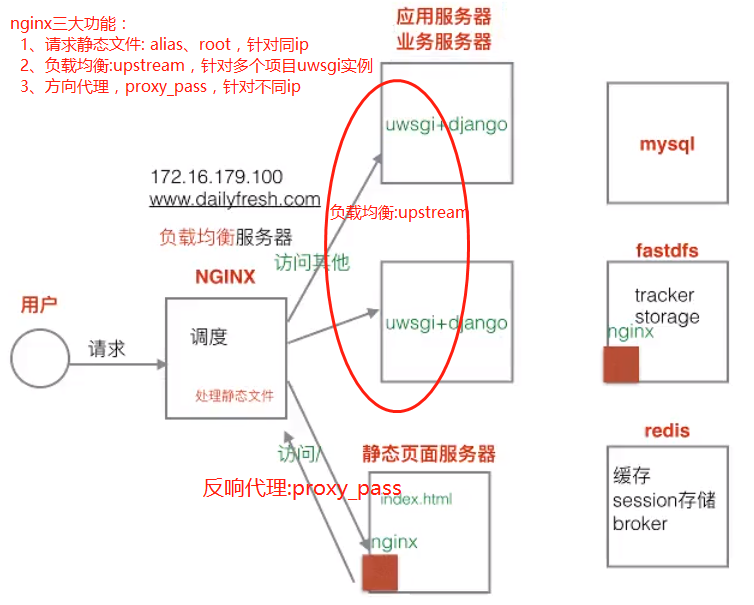
daemonize = uwsgi.log

## 查看uwsgi日志



# 部署总结





# 踩过的坑

1. 生产环境关闭setting.py的debug，开启debug ，nginx也能成功请求静态文件，建议还是关闭，因为如果网页报错，

会内部信息暴露给客户，不安全；

2、runserver用开发服务器时，setting.py的debug必须开启才能找到静态文件；

3、Nginx里Server 里配置listen，这个端口号可配置ngxin默认端口号，也可配置不同的端口号，浏览器端通过此端口号访问；

4、uwsgi与nginx通信时，uwsgi里配置socket，http用户与浏览器交互；

5、Python版本与uwsgi版本配套；

6、每个项目的nginx 的server的端口不能一样，每个项目的uwsgi的端口也不能一样。否则端口冲突，无法启动；

# Nginx里整合fdfs

具体参照: <https://www.cnblogs.com/handsomeye/p/9451568.html>

<https://www.jianshu.com/p/bad096f5123a>

## 安装fdfs

### 安装fdfs依赖包 libfastcommon

下载libfastcommon，解压，进入解压的包里，执行

./make.sh

./make.sh install

### 安装fdfs包

下载fdfs，解压，进入解压的包里，执行

./make.sh

./make.sh install

### 配置fdfs的tracker

1. 进入 /etc/fdfs
2. cp tracker.conf.sample tracker.conf
3. Vim tracker.conf
4. 修改数据项

base\_path=/home/user/fdfs\_data/tracker

port=23000 #tracker默认23000

http.server\_port=80 #http端口，需要和nginx相同

### 配置fdfs的storage

A.进入 /etc/fdfs

B. cp storage.conf.sample storage.conf

C. Vim tracker.conf

D.修改数据项

base\_path=/home/user/fdfs\_data/storage #storage存储data和log的跟路径，必须提前创建好

port=23000 #storge默认23000，同一个组的storage端口号必须一致

group\_name=group1 #默认组名，根据实际情况修改

store\_path\_count=1 #存储路径个数，需要和store\_path个数匹配

store\_path0=/home/user/fdfs\_data/storage/path0 #如果为空，则使用base\_path

tracker\_server=10.0.0.105:22122 # 云服务器时，配置内网ip

### 启动

启动tracker（支持start|stop|restart）：

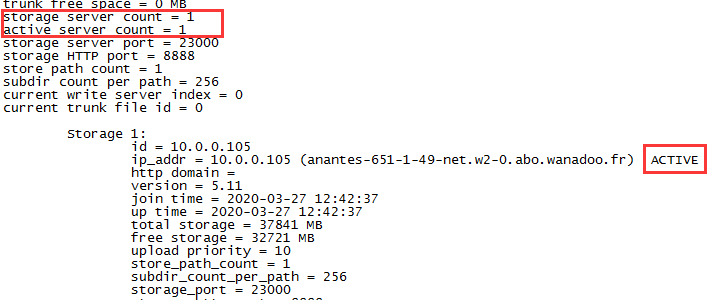
/usr/bin/fdfs\_trackerd /etc/fdfs/tracker.conf start

启动storage（支持start|stop|restart）：

/usr/bin/fdfs\_storaged /etc/fdfs/storage.conf start

### 通过monitor来查看storage是否成功绑定

/usr/bin/fdfs\_monitor /etc/fdfs/storage.conf，如果找不到fdfs\_monitor，用whereis fdfs\_monitor，找到fdfs\_monitor可执行文件路径



### 测试fdfs

A.进入 /etc/fdfs

B. cp client.conf.sample client.conf

C. Vim client.conf

D. 修改里面数据

base\_path

tracker\_server

1. fdfs\_upload\_file /etc/fdfs/client.conf /home/user/V1.0.38.tar.gz，返回文件路径，则搭建成功
2. /etc/fdfs# /usr/bin/fdfs\_upload\_file /etc/fdfs/client.conf /home/user/software/fdfs\_V5.11.tar.gz

## 安装nginx（源码安装），并整合fastdfs

### 1、准备两个包，源码包放置在/usr/local/src中

下载Nginx安装包,fastdfs-nginx-module包

wget http://nginx.org/download/nginx-1.15.2.tar.gz

wget https://github.com/happyfish100/fastdfs-nginx-module/archive/V1.20.tar.gz

解压两个包，将解压后的两个源码包复制到/usr/local/src

### 安装依赖库

apt-get update

apt-get install libpcre3 libpcre3-dev openssl libssl-dev libperl-dev

sudo apt-get install zlib1g-dev

### Nginx配置中加载fastdfs-nginx-module模块

cp -r fastdfs-nginx-module-1.20/ /usr/local/src/ 复制到/usr/local/src/目录下

./configure --prefix=/usr/local/nginx --add-module=/usr/local/src/fastdfs-nginx-module-1.20/src/

--prefix 安装路径

### 4、编译安装nginx：

make

make install

报错1:编译报错

/usr/include/fastdfs/fdfs\_define.h:15:27: fatal error: common\_define.h: No such file or directory  
 解决方案：修改fastdfs-nginx-module-1.20/src/config文件，然后重新configure,make,make install

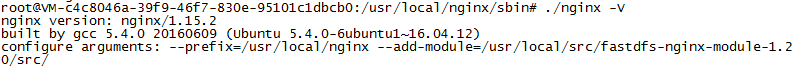
ngx\_module\_incs="/usr/include/fastdfs /usr/include/fastcommon/"

CORE\_INCS="$CORE\_INCS /usr/include/fastdfs /usr/include/fastcommon/"

### 查看nginx版本：

Cd cd /usr/local/nginx/sbin

./nginx -V



### 查看安装路径：whereis nginx

查看nginx路径，找到对于的log，可执行文件

### 7.配置Nginx和fastdfs-nginx-module模块

A.配置mod-fastdfs.conf，并拷贝到/etc/fdfs文件目录下

cd fastdfs-nginx-module-1.20/src/

cp mod\_fastdfs.conf /etc/fdfs

1. 进入/etc/fdfs修改mod-fastdfs.conf：

base\_path=/home/user/fastdfs #存log

tracker\_server=10.0.0.105:22122 #tracker的云主机内网地址

url\_have\_group\_name=true #url是否包含group名称

storage\_server\_port=23000 #需要和storage配置的相同

store\_path\_count=1 #存储路径个数，需要和store\_path个数匹配

store\_path0=/home/user/fastdfs/storage/path0 #文件存储的位置，注意该配置文件里有2个store\_path0，都配置成一样

1. 配置nginx，在http包里加入如下：

server {

listen 8888;

server\_name localhost;

location ~/group[0-9]/{

ngx\_fastdfs\_module; #匹配到后，交给这个模块处理

}

error\_page 500 502 503 504 /50x.html;

#location = /50x.html {

# root html;

#}

}

### 8.启动nginx

Cd 到 --prefix=/usr/local/nginx 的目录

./nginx

./nginx -s reload

报错1: 启动nginx时，很可能会报错

[2017-03-26 11:52:26] ERROR - file: ini\_file\_reader.c, line: 315, include file "http.conf" not exists, line: "#include http.conf"

[2017-03-26 11:52:26] ERROR - file: /opt/fastdfs-nginx-module/src/common.c, line: 155, load conf file "/etc/fdfs/mod\_fastdfs.conf" fail,

ret code: 22017/03/26 11:52:26 [alert] 12135#0: worker process 12136 exited with fatal code 2 and cannot be respawned

说明你的/etc/fdfs/下没有http.conf和mime.types这两个文件，你需要将这两个文件从解压目录中复制过去，然后从来一遍就可以了。

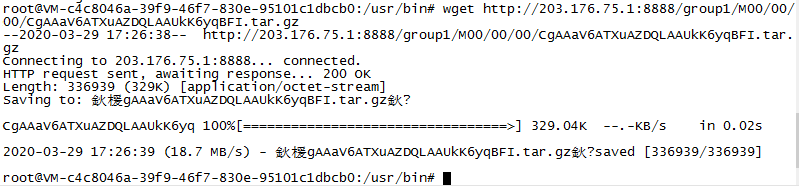
### 测试

用fdfs client上传文件

root@VM-c4c8046a-39f9-46f7-830e-95101c1dbcb0:/usr/bin# ./fdfs\_upload\_file /etc/fdfs/client.conf /home/user/software/fdfs\_V5.11.tar.gz

返回: group1/M00/00/00/CgAAaV6AXM6AeyMDAAUkK6yqBFI.tar.gz

云主机里:wget [http://203.176.75.1:8888/group1/M00/00/00/CgAAaV6ATXuAZDQLAAUkK6yqBFI.tar.gz](http://203.176.75.1/group1/M00/00/00/CgAAaV6ATXuAZDQLAAUkK6yqBFI.tar.gz)



或本地浏览器输入 [http://203.176.75.1:8888/group1/M00/00/00/CgAAaV6ATXuAZDQLAAUkK6yqBFI.tar.gz](http://203.176.75.1/group1/M00/00/00/CgAAaV6ATXuAZDQLAAUkK6yqBFI.tar.gz)

# Redis

修改配置/etc/redis/redis.conf ：

requirepass 123456

daemonize yes 守护进程

redis-server /etc/redis/redis.conf 带配置启动

Kill -9 pid 关闭redis

# Mysql

/etc/init.d/mysql stop /start/restart 脚本方式启动

service mysql stop /start/restart 服务方式启动

mysql –uroot -p 本地连接mysql

killall -TERM mysqld

service mysql status

# 附录

## 附-修改nginx默认端口

因天翼云主机没备案是不开放80端口的，为了能访问ngin，修改nginx的默认80端口。

修改教程地址:<https://blog.csdn.net/YLD10/article/details/80242487>

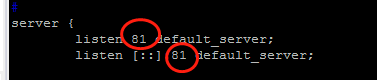
cd /etc/nginx 配置目录

vim nginx.conf 主配置文件

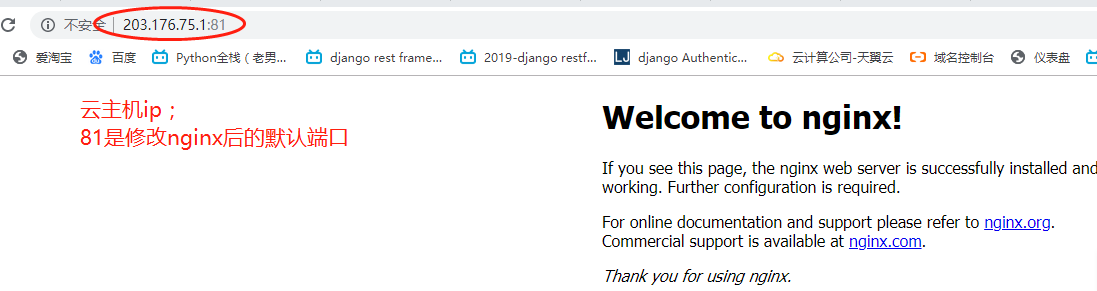
 ->



vim /etc/nginx/sites-available/default （最终修改的是这个文件里的）

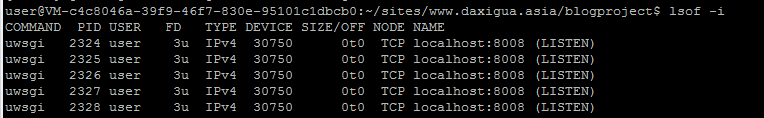
修改nginx默认端口由80改成81 

本地打开浏览器，输入IP:端口号，检查nginx能否访问

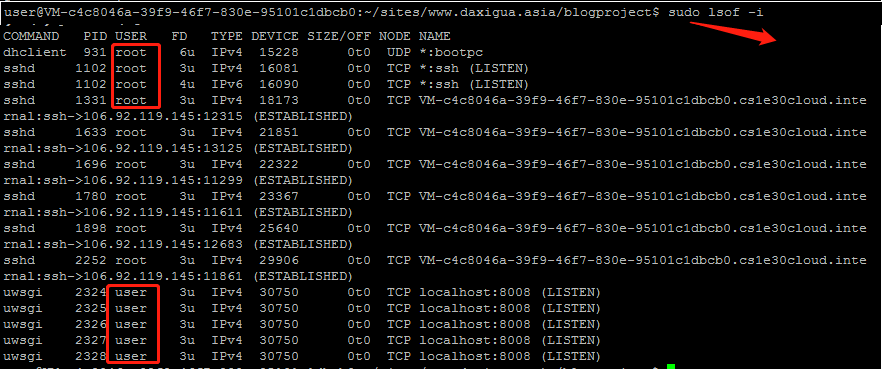


## 附-用户权限问题

不加sudo，只能看到属于自己用户的进程



加上sudo，即管理员权限，可以看到所以用户的进程

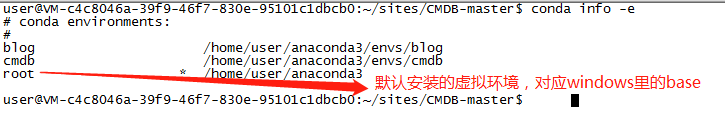


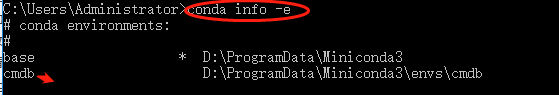
## 附-Conda虚拟环境第三方包位置

用sudo find / -name site-packages -type d 查找到

Root虚拟环境包放置位置:/home/user/anaconda3/lib/python3.6/site-packages/

Blog虚拟环境包放置位置:/home/user/anaconda3/envs/blog/lib/python3.5/site-packages





## 附-Uwsgi与python、gcc的兼容问题

### 分析

whereis uwsgi

uwsgi:

/usr/bin/uwsgi

uwsgi -> /etc/alternatives/uwsgi\*

uwsgi -> /usr/bin/uwsgi-core\*

/home/user/anaconda3/bin/uwsgi

==========================================

根据

whereis uwsgi

uwsgi:

/usr/bin/uwsgi

找到系统的，系统的gcc版本是5.4，所以以前的uwsgi是根据5.4编译的

\*\*\* Starting uWSGI 2.0.12-debian (64bit) on [Sat Feb 15 11:12:31 2020] \*\*\*

compiled with version: 5.4.0 20160609 on 28 September 2018 15:49:44

==========================================

根据

whereis uwsgi

uwsgi:

/usr/bin/uwsgi

/home/user/anaconda3/bin/uwsgi

root虚拟环境的，user用户也是用的这个，conda里的cmdb、blog虚拟环境都是用的这个，uwsgi根据4.7版本gcc编译

Starting uWSGI 2.0.18 (64bit) on [Sat Feb 15 11:15:20 2020] \*\*\*

compiled with version: 4.7.4 on 13 February 2020 15:26:01

==========================================

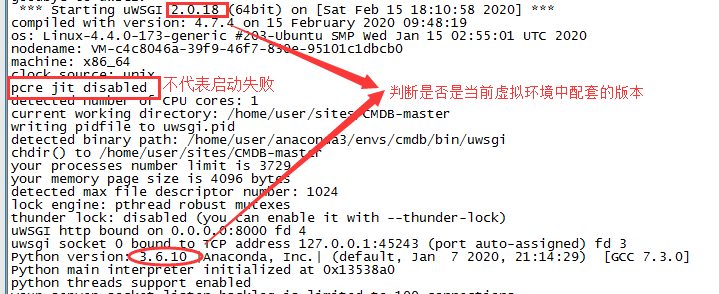
### 以上小结

1. 目前遇到的兼容问题，主要是python版本与uwsgi版本没有对应

如: whereis uwsgi ->/home/user/anaconda3/envs/cmdb/bin/uwsgi 知道cmdb虚拟环境里uwsgi可执行文件位置，执行该文件，可得到uwgi版本

Python ->可直接知道cmdb里pyton版本

当uwsgi --ini uwsgi.ini后 tail -f uwsgi.log查看日志，查看里面uwsgi调用的python版本是哪个，判断是否配套



1. conda中的虚拟化环境的gcc版本与root、user系统里gcc版本是共用的，都是4.7

3、终端里输入 wusgi，表明uwsgi版本是2.0.18，gcc版本是4.7.4，如下:

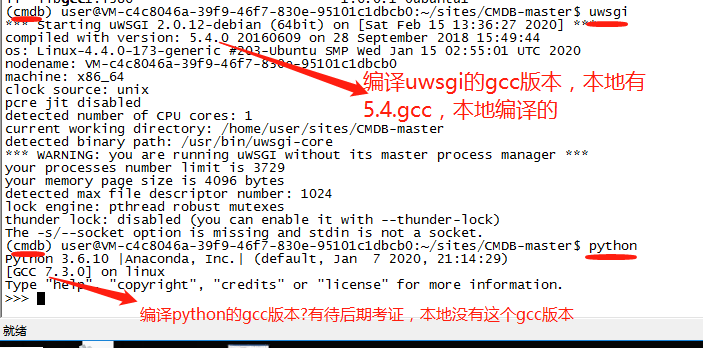
Starting uWSGI 2.0.18 (64bit) on [Sat Feb 15 11:15:20 2020] \*\*\*

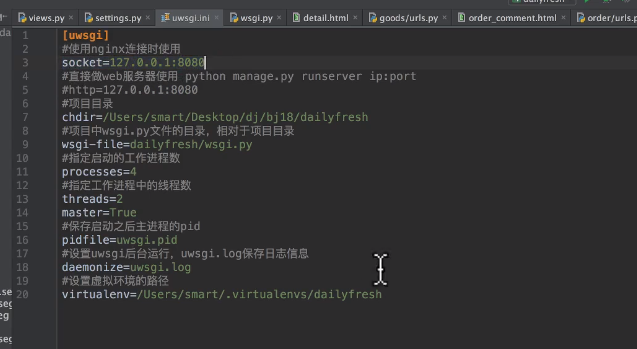
compiled with version: 4.7.4 on 13 February 2020 15:26:01

**4、linux安装软件，库文件和可执行文件分开放的，库文件放xx/lib，可执行文件放xx/bin，不像windows里一般安装在同一个文件夹里**

5、gcc在不同conda虚拟环境中，版本一样，可进入conda不同的虚拟环境中，用uwsgi命令查看版本，当conda虚拟环境中没有uwsgi，会调用系统原生的uwsgi

6、虚拟环境里uwgic与python命令之后的解释





## 附-ln命令

ln -s /etc/nginx/sites-available/cmdb cmdb

-s 软链接，不加该参数，是硬链接

最后一个路径是快捷图标所在的位置

## 附-配置代码

~/sites/CMDB-master/uwsgi.ini内容:

[uwsgi]

#http = 0.0.0.0:8000

socket = 0.0.0.0:8001

chdir = /home/user/sites/CMDB-master

wsgi-file = CMDB/wsgi.py

processes = 4

threads = 2

master = True

pidfile = uwsgi.pid

daemonize = uwsgi.log

----------------------------------------------------------------------------------

/etc/nginx/sites-available/cmdb内容:

upstream cmdb{

server 127.0.0.1:8000;

server 127.0.0.1:8001;

}

server {

listen 81;

charset utf-8;

server\_name www.daxigua.asia;

location /static {

alias /home/user/sites/CMDB-master/static;

}

location / {

include uwsgi\_params;

uwsgi\_pass cmdb;

}

}

## 附-参考资料

视频资料 <https://www.bilibili.com/video/av38692008?p=3>