--

注意：

如果你要在windows上访问，需要对server.example.com进行自定义域名解析

修改文件 C:\Windows\System32\drivers\etc\hosts

进程的最大连接数受linux系统的最大打开文件数限制，在执行操作系统命令“ulimit -n 65535”后worker\_connections的设置才能生效。

worker\_connections 65535;

YUM环境准备：

yum install epel-release -y

yum clean all

yum makecache

lnmp(Linux+Nginx+Maria+PHP)

lamp(Linux+Apache+Maria+PHP)

nginx反向代理

反向代理(缓存加速)

squid varnish ats nginx等

LB(负载均衡）

nginx haproxy lvs等

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

SNAT:源地址转换

DNAT：目标地址转换

squid

nginx 反向代理

client(宿主机) 192.168.2.x

|

|

192.168.2.19 eth0

nginx（虚拟机1）

192.168.224.10 eth1

|

|

web1(虚拟机2) web2(虚拟机3）

192.168.224.11 192.168.224.12

(lnmp)

1.安装nginx(需要本地源,epel源)

# yum install nginx

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

例一:使用前端nginx代理后面一台web

client(宿主机) 192.168.2.x

|

|

| 192.168.2.19

nginx（虚拟机1）

192.168.224.10

|

|

|

web1(虚拟机1) 192.168.224.11

在Nginx服务器执行：

#yum -y install nginx

# cat /etc/nginx/nginx.conf |grep -v '#'

user nginx;

worker\_processes auto;

error\_log /var/log/nginx/error.log;

pid /run/nginx.pid;

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;

include /etc/nginx/conf.d/\*.conf;

server {

listen 80 default\_server;

listen [::]:80 default\_server;

server\_name 192.168.2.19; --改成模拟nginx的外网ip

root /usr/share/nginx/html;

index index.php index.html; -- 添加

include /etc/nginx/default.d/\*.conf;

location / {

proxy\_pass http://192.168.224.11/;

proxy\_set\_header Host $host;

proxy\_set\_header X-Forwarded-For $remote\_addr;

} --这个例子主要讨论这一段 这两句与代理的网站跳转有关

error\_page 404 /404.html;

location = /40x.html {

}

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

location = /50x.html {

}

}

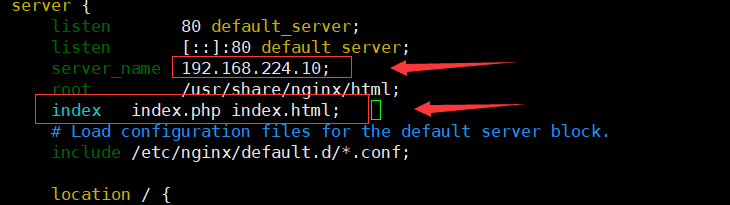
}

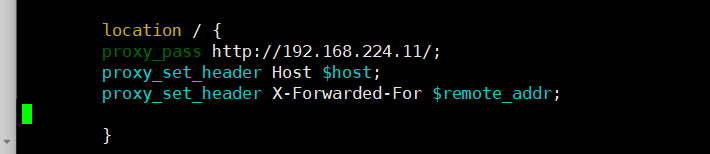
Note:

--说明:下面这两句是做外网转内网双网段架构必需要加的

proxy\_set\_header Host $host;

proxy\_set\_header X-Forwarded-For $remote\_addr; #这里是获取客户端的ip，并转发给后端。





#systemctl restart nginx

#lsof -i:80

2.在web1安装nginx，

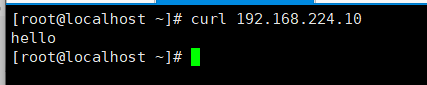
yum install -y nginx

echo hello > /usr/share/nginx/html/index.html

systemctl restart nignx

iptables -F 清空防火墙

curl 192.168.224.10



客户端使用firefox验证

访问 http://192.168.2.19/ --得到后台web1上安装的http server



验证:

把后面的web关闭，客户端也访问不了，说明nginx默认没有缓存功能

Q:为什么需要代理？ 安全DMZ，缓存

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

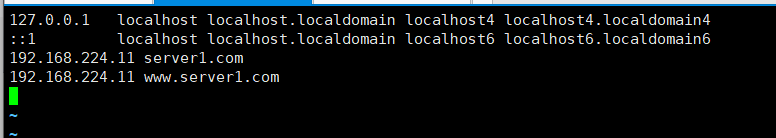
nginx 301永久重定向配置

举个例子，最近我想做一个查询域名的站点，我注册了域名server1.com。

我想主要使用server1.com，访问www.server1.com会301跳转到server1.com

首先，设置server1.com和www.server1.com解析到同一ip，

vim /etc/hosts



在10这台机添加hosts



其次，设置好域名解析后修改nginx配置文件

修改nginx.conf 的server\_name部分

server\_name server1.com www.server1.com;

if ($host ~\* www.server1.com) {

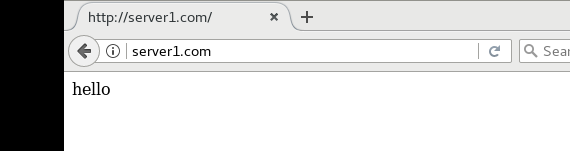
rewrite ^/(.\*)$ http://server1.com/$1 permanent;

}

然后重新加载nignx

systemctl reload nginx

最后，重新加载nginx配置，访问http://www.server1.com 就会301重定向到http://server1.com了



nginx 80端口重定向至443（http跳转https）

以域名http://server1.com跳转https://server1.com为例

server {

listen 443;

server\_name server1.com;

#开启ssl证书认证

ssl on;

access\_log /data/nginx/server1.com.access.log main;

keepalive\_timeout 60;

#证书路径，根据实际情况改写

ssl\_certificate /data/nginx/cert/server1.com.com.pem;

ssl\_certificate\_key /data/nginx/cert/server1.com.com.key;

ssl\_session\_cache shared:SSL:1m;

ssl\_session\_timeout 5m;

#禁止在header中出现服务器版本，防止黑客利用版本漏洞攻击

server\_tokens off;

}

server {

listen 80;

server\_name server1.com;

#核心代码

rewrite ^(.\*)$ https://${server\_name}$1 permanent;

}

或者：

server {

listen 80;

server\_name server1.com;

rewrite ^(.\*)$ https://${server\_name}$1 permanent;

}

如果要二个都可以访问，不用单独配置这个server,直接在443下面新增一个80就可以了

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

当网址为http://时，默认端口为80

当网址为https://时，默认端口为443

以上两种都可以省略端口号，可以直接写成http://zzuxh.top/或https://zzuxh.top/，后边跟上我们项目的接口URI。

如果是http的8080端口，或者是https的8443端口，则不能省略，要在域名之后加上端口，如 http://server1.com:8080/ 、 https://server1.com:8443/

但默认情况下我们的server1.com里边只有一个http请求对应的8080端口connector是开启的，剩下的http和https端口：80,443,8443，前两个没有，最后一个被注释了，都需要手动配置启用。

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

什么是网站数据切分?

其实也是七层调度

比如我要把新浪新闻，新浪体育给分开

方法1:

用dns的二级域名(直接dns解析成不同的ip)

新浪新闻 news.sina.com 新浪国内新闻　news.sina.com/china/ --说明没有用二级域名

新浪国际新闻 news.sina.com/world/

　新浪国内新闻　china.news.sina.com --用了四级域名

新浪国际新闻 world.news.sina.com

新浪体育　 sports.sina.com 新浪体育nba sports.sina.com/nba/

新浪体育nba nba.sports.sina.com

方法2:

前端使用代理(squid,varnish,apache,nginx,haproxy)

通过代理软件七层调度来分离

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

例二:locate网站数据切分

client(宿主机) 192.168.2.x

192.168.2.19

nginx（虚拟机1）

192.168.224.10

web1(虚拟机2) web2(虚拟机3）

192.168.224.11 192.168.224.12

把例一nginx配置文件里的那五行改成下面两段(加在server { }配置段中 ):

# vim /etc/nginx/nginx.conf

将原来的文件中'location'内容替换为如下:

location /nba/ {

proxy\_pass http://192.168.224.11/;

proxy\_set\_header Host $host;

proxy\_set\_header X-Forwarded-For $remote\_addr;

}

location /cba/ {

proxy\_pass http://192.168.224.12/;

proxy\_set\_header Host $host;

proxy\_set\_header X-Forwarded-For $remote\_addr;

}

# systemctl restart nginx

在client2操作

yum install -y httpd

echo web2 > /var/www/html/index.html

systemctl restart httpd

curl 192.168.224.12

客户端验证

http://192.168.2.19/nba/

hello

http://192.168.2.19/cba/

web2

例三:网站动静分离

client(宿主机) 192.168.2.x

192.168.2.19

nginx（虚拟机1）

192.168.224.10

web1(虚拟机2) web2(虚拟机3）

192.168.224.11 192.168.224.12

把例二的配置再改成如下(加在server { }配置段中 ):

location ~ \.(html|htm|gif|jpeg|jpg|css|js|png|swf)$ {

proxy\_pass http://192.168.224.11;

proxy\_set\_header Host $host;

proxy\_set\_header X-Forwarded-For $remote\_addr;

}

location ~ \.(php|cgi|txt)$ {

proxy\_pass http://192.168.224.12; #使用正则表达式IP后面不用加/号，会报错

proxy\_set\_header Host $host;

proxy\_set\_header X-Forwarded-For $remote\_addr;

}

# systemctl restart nginx

分别创建1.txt 和 1.html 在两台web服务器上。

[root@client1 ~]# vim /var/www/html/1.txt

web1txt

[root@client1 ~]# vim /var/www/html/1.html

web1.html

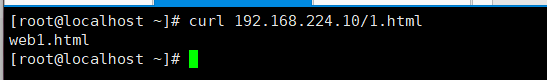
vim /var/www/html/1.txt

web2.txt

vim /var/www/html/1.html

web2.html

验证：访问 http://nigixIP/1.txt http://nigixIP/1.html 观察内容的变化



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

例四:代理后端时使用负载均衡(load balance)

client(宿主机) 192.168.2.x

192.168.2.19

nginx（虚拟机1）

192.168.224.10

web1(虚拟机2) web2(虚拟机3）

192.168.224.11 192.168.224.12

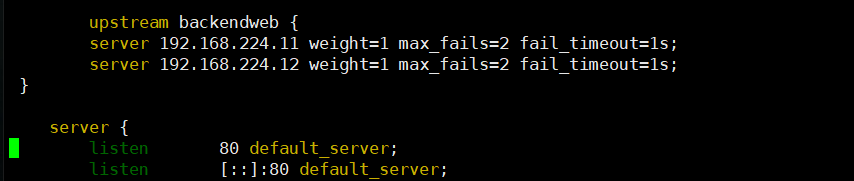
下面一段加到server之前

upstream backendweb {

server 192.168.224.11 weight=1 max\_fails=2 fail\_timeout=1s;

server 192.168.224.12 weight=1 max\_fails=2 fail\_timeout=1s;

}



--weight代表权重,max\_fails=2 fail\_timeout=1s代表健康检查(检查后台web是否ok，访问超时1秒，并两次超时，则认为不健康)

把例三的配置再改成如下(加在server { }配置段中 ):

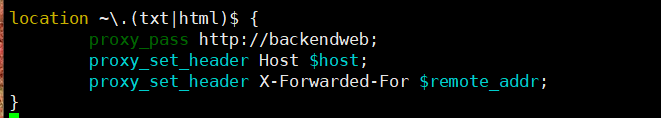
location ~ \.(txt|html)$ {

proxy\_pass http://backendweb; --backendweb是一个名称，对应上面upstream的配置

proxy\_set\_header Host $host;

proxy\_set\_header X-Forwarded-For $remote\_addr;

}



# systemctl reload nginx

客户端验证

访问http://192.168.2.19/1.txt

--验证时，会发现客户端针对同一个URL的访问也会一次web1一次web2，这再次验证说明了nginx默认并没有squid或varnish那样的缓存功能

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

负载均衡(lb load banlance)一般要注意四个方面:

1,算法 round-robin

2,健康检查

3,会话保持

4,数据一致 rsync drbd 　　 共享存储 　　分布式存储

client request

LB

web1 web2

例五:使用ip\_hash，实现同一IP客户端一旦调到一台，就一直调那一台

client(宿主机) 192.168.2.x

192.168.2.19

nginx（虚拟机1）

192.168.224.10

web1(虚拟机2) web2(虚拟机3）

192.168.224.11 192.168.224.12

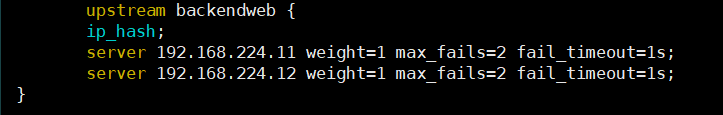
upstream backendweb {

ip\_hash; --在上个例子的基础上只加这一句;

server 192.168.224.11 weight=1 max\_fails=2 fail\_timeout=1s;

server 192.168.224.12 weight=1 max\_fails=2 fail\_timeout=1s;

}



# systemctl reload nginx

客户端验证

访问http://192.168.2.19/1.txt

nginx的ip\_hash的意思是,如果一个客户端的访问被调度到其中一台后台服务器,那么同一个源IP来的

访问都只会被调到这个后台服务器；这里测试时，如果都用同一个网段的内网IP来做客户端测试，可能会

都只转到一个后台（因为nginx的hash算法是按网段来算的，如果是公网不同网段的客户端IP就不一样了）

对于nginx的upstrem算法总结:

1,round-robin 轮循（平均分配）

2,weight 权重（人为地分配权重，用于后台服务器性能不均的情况）

3,fair 响应时间（按后台的响应时间来分配，需要第三方模块，但如果后台服务器都在内网，就没太大必要使用这种算法了）

4,url\_hash 按访问url的hash结果来分配请求，使每个url定向到同一个后端服务器，后端服务器为多台缓存时比较有效，提高缓存命中率（后面例子会讲）

5,ip\_hash 在负载均衡的基础上加上会话保持（优点是配置方便，缺点是不能完全的负载均衡）

**Nginx 做集群架构**

!!!!! 需要拆掉以前的环境，在新的虚拟机开始做！！！！

client 192.168.2.x

|

| 192.168.2.19(模拟网站公网ip，整个架构的域名假设为server.example.com )

nginx 反向代理

｜ 192.168.224.10

｜

－－－－－－－－－－－

｜ ｜ 命中 hit 直接返回

动态程序文件.php ｜ |

｜ squid（web加速，缓存静态文件或图片)

直接找web ｜ |

－－－－ | 没命中 miss 找后端web去取

｜ | 192.168.224.12

lnmp <---- |

192.168.224.11

!!!!! 需要拆掉以前的环境，在新的虚拟机开始做！！！！

实验前准备：

1,所有机器配置主机名并在/etc/hosts里互相绑定主机

2,关闭firewalld,selinux

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

第一大步：在上图中的lnmp服务器上安装并配置后面的网站

1.在192.168.224.11安装lnmp相关的rpm包

yum install epel-release -y

yum -y install nginx

yum install mariadb mariadb-server php php-mysql php-gd libjpeg\\* php-ldap php-odbc php-pear php-xml php-xmlrpc php-mbstring php-bcmath php-mhash php-fpm php-pecl-zendopcache nginx -y

Note: php-pecl-zendopcache需要epel

systemctl restart mariadb.service

systemctl enable mariadb.service

systemctl status mariadb.service

vim /etc/php-fpm.d/www.conf

12 listen = /var/run/php-fpm/fastcgi.socket

31 listen.owner = nginx

32 listen.group = nginx --socket文件的权限设置。用户与组和跑nginx服务的用户一致，避免权限问题（如果前面使用的是tcp/ip的方式，这里就注释就好)

39 user = nginx --用户与组和跑nginx服务的用户一致，避免权限问题

41 group = nginx

218 php\_flag[display\_errors] = on --打开php错误显示功能，取消注释

# chown nginx.nginx /var/run/php-fpm/

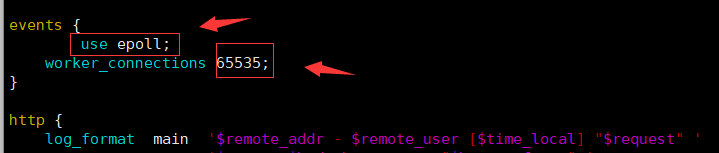
启动php-fpm服务

# systemctl start php-fpm.service

# systemctl status php-fpm.service

# systemctl enable php-fpm.service

# vim /etc/nginx/nginx.conf



user nginx;

worker\_processes auto;

error\_log /var/log/nginx/error.log;

pid /run/nginx.pid;

include /usr/share/nginx/modules/\*.conf;

events {

use epoll;

worker\_connections 65535;

}

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;

include /etc/nginx/conf.d/\*.conf;

server {

listen 80 default\_server;

listen [::]:80 default\_server;

server\_name 192.168.224.11;

root /usr/share/nginx/html;

index index.php index.html;

include /etc/nginx/default.d/\*.conf;

location ~ \.php$ {

fastcgi\_pass unix:/var/run/php-fpm/fastcgi.socket;

fastcgi\_index index.php;

fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;

include fastcgi\_params;

}

error\_page 404 /404.html;

location = /40x.html {

}

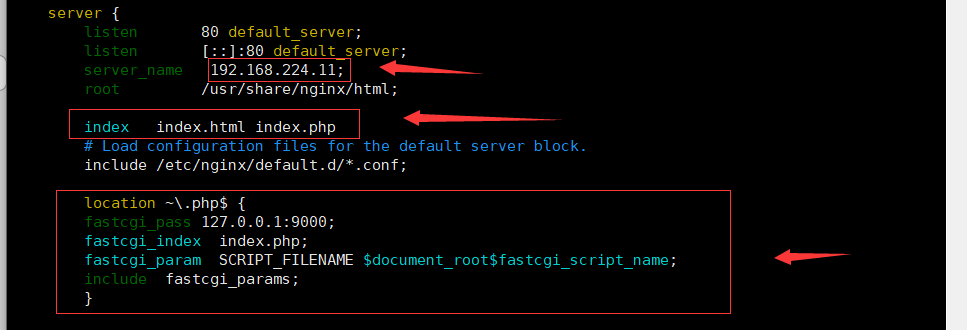
error\_page 500 502 503 504 /50x.html;

location = /50x.html {

}

}

}



启动nginx服务

# systemctl start nginx.service --80端口不要被其它服务（如httpd）占用了

# systemctl enable nginx.service

# systemctl status nginx.service

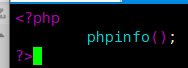
vim /usr/share/nginx/html/index.php

php测试代码 index.php

<?php

phpinfo();

?>



访问看看:

<http://192.168.224.11/index.php>

1,解压discuz到nginx家目录

wget

http://download.comsenz.com/DiscuzX/3.2/Discuz\_X3.2\_SC\_UTF8.zip

# mkdir /usr/share/nginx/html/discuz/

# rz通过u盘把软件包传过来

# unzip Discuz\_X3.3\_SC\_UTF8.zip -d /usr/share/nginx/html/discuz/

# cd /usr/share/nginx/html/discuz/

# mv upload/\* .

# rm upload/ -rf

3,环境检查这一步，有些目录和文件权限需要修改(下面直接使用简单方式全改成nginx的owner和group)

# chown nginx.nginx /usr/share/nginx/html/discuz -R

4,mariadb数据库授权

# mysql

MariaDB [(none)]> create database lnmp\_discuz; --创建一个库，用于存放将要安装的discuz论坛的表

MariaDB [(none)]> grant all on lnmp\_discuz.\* to 'lnmpdiscuz'@'localhost' identified by '123'; --授权一个用户，用于discuz论坛程序连接mysql

MariaDB [(none)]> flush privileges;

5,http://192.168.224.11/discuz/

填上对应的数据库地址,库,用户,密码。开始安装

On web page wizard:

选择：全新安装 Discuz! X (含 UCenter Server)

数据库名:lnmp\_discuz

数据库用户名：lnmpdiscuz

数据库密码：123

管理员密码：123

重复密码：123

6,访问http://192.168.224.11/discuz/forum.php 测试论坛

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

第二大步:在上图中的nginx服务器上安装并配置nginx

1,安装nginx

# yum install nginx -y

2,配置文件如下

# cat /etc/nginx/nginx.conf |grep -v '#'

user nginx;

worker\_processes auto;

error\_log /var/log/nginx/error.log;

pid /run/nginx.pid;

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;

include /etc/nginx/conf.d/\*.conf;

upstream squid {

server 192.168.224.12;

}

upstream web {

server 192.168.224.11;

}

server {

listen 80 default\_server;

listen [::]:80 default\_server;

server\_name 192.168.2.19; ----------替换为实际的名字

root /usr/share/nginx/html;

include /etc/nginx/default.d/\*.conf;

location ~ .\*\.php$ {

proxy\_pass http://web;

proxy\_set\_header Host $host;

proxy\_set\_header X-Forwarded-For $remote\_addr;

}

location ~ .\*\.(html|htm|gif|jpeg|jpg|css|js|png|swf)$ {

proxy\_pass http://squid;

proxy\_set\_header Host $host;

proxy\_set\_header X-Forwarded-For $remote\_addr;

}

location / {

proxy\_pass http://squid;

proxy\_set\_header Host $host;

proxy\_set\_header X-Forwarded-For $remote\_addr;

}

error\_page 404 /404.html;

location = /40x.html {

}

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

location = /50x.html {

}

}

}

# systemctl restart nginx

# lsof -i:80

第三大步:在上图中的squid服务器上安装并配置squid

1，安装squid

# yum install squid -y

2,配置squid主配置文件

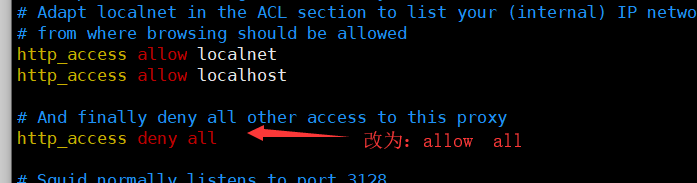
# vim /etc/squid/squid.conf

http\_access allow localnet

http\_access allow localhost

http\_access deny all #把这一行前面的全删除，再把这行修改成允许所有。注意！有多个重复的http\_access，都要删除

http\_access allow all



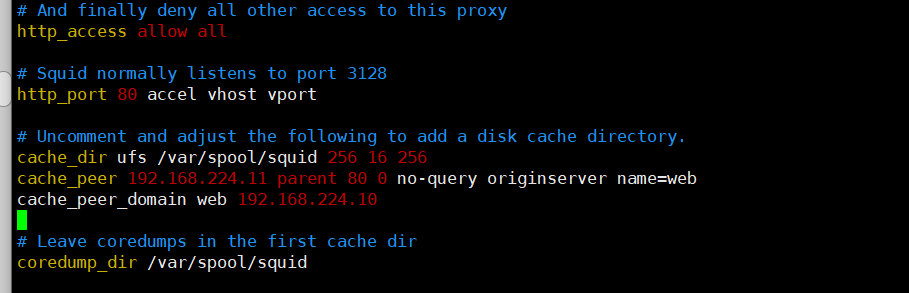
http\_port 80 accel vhost vport #修改成支持反向代理模式，端口也为80与nginx的配置对应(这里如果用3128也可以，nginx和这里对应的端口也要改成3128，并且后面清缓存的http://192.168.2.19/static/image/common/logo.png要改成http://192.168.2.19:3128/static/image/common/logo.png)

cache\_dir ufs /var/spool/squid 256 16 256 --打开缓存目录的定义这一句

cache\_peer 192.168.224.11 parent 80 0 no-query originserver name=web --192.168.224.11 是 lnmp server

--server.example.com就是我现在模拟的整个网站架构的域名

cache\_peer\_domain web 192.168.2.19 #加上这三句,表示代理后台的lnmp的80端口;server.example.com为网站的域名,192.168.2.19为我这个架构最前端的nginx的IP



3,启动squid

# yum install openssl -y -- 需要安装OpenSSL，否则无法启动squid

在/etc/hosts中配置nginx的外部网卡IP和域名的绑定

# systemctl restart squid

# lsof -i:80

/usr/bin/squidclient -p 80 mgr:objects | grep png -- 查询目前所有缓存的资料

第四大步:验证

=========不需要，已经在虚拟机的快照中配置好了==========

在客户端机器192.168.2.x上首先绑定静态DNS

--用于模拟DNS，如果不绑定，也可以直接使用公网IP192.168.2.19来访问，因为在squid里配置了（cache\_peer\_domain web server.example.com 和 cache\_peer\_domain web 192.168.2.19 两句)

cat /etc/hosts

192.168.2.19 server.example.com --IP要为前端nginx的IP，名字为这个网站的域名要和squid里的cache\_peer\_domain web server.example.com要对应

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

1,在客户端用firefox访问http://server.example.com/或http://192.168.2.19/是可以正常看到我的lnmp安装的discuz论坛



2,在客户端使用下面的命令验证discuz论坛的一个logo,可以看到在squid上命中的信息

# curl -I http://server1.com/discuz/static/image/common/logo.png

HTTP/1.1 200 OK

Server: nginx/1.8.0

Date: Mon, 23 Nov 2015 08:10:09 GMT

Content-Type: image/png

Content-Length: 4425

Connection: keep-alive

Last-Modified: Tue, 09 Jun 2015 02:21:12 GMT

ETag: "55764d98-1149"

Accept-Ranges: bytes

Age: 3227

X-Cache: HIT from squid.cluster.com

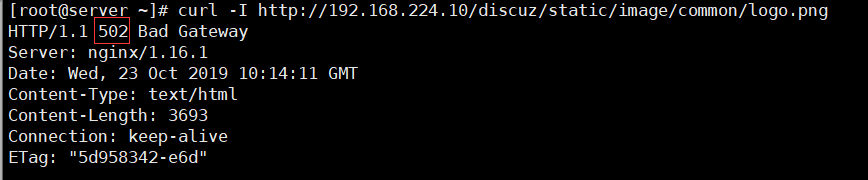
X-Cache-Lookup: HIT from squid.cluster.com

Via: 1.0 squid.cluster.com (squid/3.1.10)



3,关闭squid,在客户端用firefox访问,会发现整个网站都没有图片(静态的元素)

用curl -I http://server.example.com/discuz/static/image/common/logo.png来验证也会报错



因为我的架构里只有一台squid,再次启动squid后,一切又恢复正常

4,关于squid手动清缓存

# vim /etc/squid/squid.conf

acl purge\_admin src 127.0.0.1 ---设定管理员为purge\_admin 从本地可以清除缓存

acl purge method PURGE

http\_access allow purge\_admin purge --允许purge\_admin执行purge

http\_access deny all purge ---默认禁止所有用户操作清除缓存

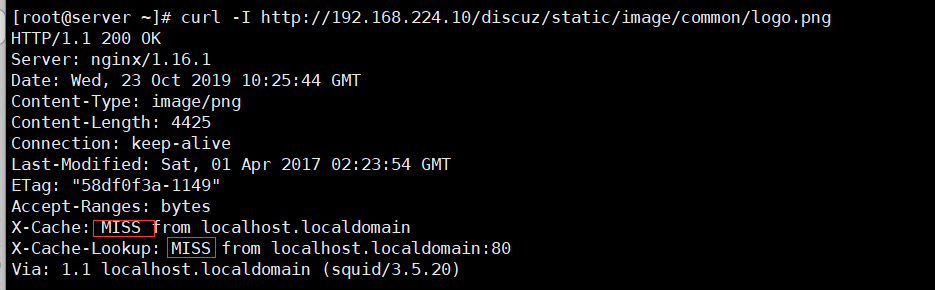
# systemctl restart squid

最基本的清除一条缓存的操作,必须要在squid本机执行

# squidclient -m PURGE -h 127.0.0.1 -p 80 http://192.168.224.10/discuz/static/image/common/logo.png

-- -h参数后只能接127.0.0.1;-p 80是squid的监听端口;最后的路径就是客户端访问的路径

清除缓存后再到客户端访问，就变成miss了



如果要批量清除squid,可以使用下面的脚本(你需要修改成自己对应的路径)

# vim /tmp/purge\_squid.sh

#!/bin/bash

squidcache\_path="/var/spool/squid/"

squidclient\_path="/usr/bin/squidclient"

grep -a -r $1 $squidcache\_path/\* | strings | grep ^"http" | while read url

do

$squidclient\_path -h 127.0.0.1 -m PURGE -p 80 $url > /dev/null 2>&1

echo "$url被清除"

done

--注意：脚本的squidcache\_path修改成你对应的缓存目录，squidclient\_path修改成squidclient命令的路径；-h 127.0.0.1是因为我做了acl限制的，所以只能在squid本机上清除

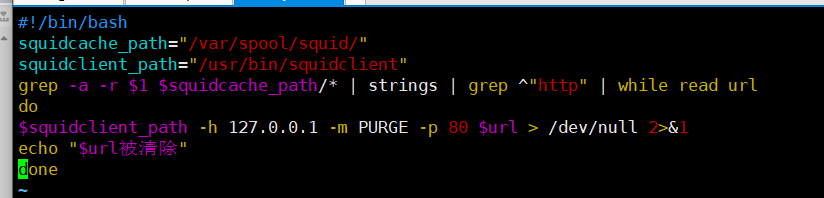
批量清除的方法:

sh /tmp/purge\_squid.sh .txt --表示清除所有的.txt结尾的缓存

sh /tmp/purge\_squid.sh . --表示清除所有缓存

sh /tmp/purge\_squid.sh /aaa/ --表示url里有/aaa/路径就清掉缓存

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



在上面的架构基础上多加一台squid2(我这里IP为192.168.224.13),

client 192.168.2.x

|

| 192.168.2.19

nginx

| 192.168.224.10

|

|------------|---------------

| | |

| squid1 squid2

| 192.168.224.12 192.168.224.13

|------------|

|

|

lnmp

192.168.224.11

以下练习在Squid2进行：

# yum install openssl -y

# yum install squid -y

在squid2中安装squid软件，配置文件与squid服务器中的配置保持一致/etc/squid/squid.conf

vim /etc/squid/squid.conf

-----------

http\_access allow all

acl purge\_admin src 127.0.0.1

acl purge method PURGE

http\_access allow purge\_admin purge

http\_access deny all purge

# Squid normally listens to port 3128

http\_port 80 accel vhost vport

# Uncomment and adjust the following to add a disk cache directory.

cache\_dir ufs /var/spool/squid 256 16 256

cache\_peer 192.168.224.11 parent 80 0 no-query originserver name=web

cache\_peer\_domain web server.example.com

cache\_peer\_domain web 192.168.224.10

# Leave coredumps in the first cache dir

coredump\_dir /var/spool/squid

#

# Add any of your own refresh\_pattern entries above these.

#

refresh\_pattern ^ftp: 1440 20% 10080

refresh\_pattern ^gopher: 1440 0% 1440

refresh\_pattern -i (/cgi-bin/|\?) 0 0% 0

refresh\_pattern . 0 20% 4320

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

重启2台squid服务器上的squid服务

systemctl restart squid

做法，在nginx配置要修改为下面一段

upstream squid {

server 192.168.224.12 weight=1 max\_fails=2 fail\_timeout=3s;

server 192.168.224.13 weight=1 max\_fails=2 fail\_timeout=3s;

}

# systemctl restart nginx

在客户端用curl -I去测试多个不同的文件请求，看缓存情况,如:

curl -I http://server.example.com/discuz/static/image/common/logo.png

curl -I http://server.example.com/discuz/static/image/feed/task\_b.png

curl -I http://server.example.com/discuz/static/image/feed/album\_b.png

curl -I http://server.example.com/discuz/static/image/feed/portal\_b.png

curl -I http://server.example.com/discuz/static/image/feed/wall\_b.png

测试结果为:第一次squid1,第二次squid2,第三次squid1...以此类推(round-robin)

但这个做法的缺点为:比如同一个url的请求，连续访问，它也会RR轮循给squid1和squid2，这样会造成两个squid重复缓存。

改进的做法为:使用nginx的url\_hash的算法，把同一个url的请求只给同一个后台squid，以提高缓存命中率。如果要做这个改进的话，只需要把nginx的配置再修改成如下:

upstream squid {

hash $request\_uri;

server 192.168.224.12 weight=1 max\_fails=2 fail\_timeout=3s;

server 192.168.224.13 weight=1 max\_fails=2 fail\_timeout=3s;

}

# systemctl restart nginx

再次测试:

结果为:新的请求仍然会RR轮循调给squid1和squid2，但已经请求过的地址再次被请求，会调给第一次调的squid，提高缓存命中率。

查询缓存

/usr/bin/squidclient -p 80 mgr:objects | grep png

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

在上面的架构中，把squid去掉，由nginx又做反向代理，又做缓存

nginx做缓存需要一个另外一个软件(ngx\_cache\_purge)

下载的网址为:wget http://labs.frickle.com/files/ngx\_cache\_purge-2.3.tar.gz

wget http://nginx.org/download/nginx-1.8.1.tar.gz

wget http://nginx.org/download/nginx-1.12.2.tar.gz #这个也可以

架构图，在上面做的基础上把squid或

client 192.168.2.x

|

| 192.168.2.19

nginx 反向代理加缓存

| 192.168.224.10

|

lnmp

192.168.224.11

192.168.224.10 虚拟机在上一个实验的基础上继续做。但是需要停掉nginx（恢复快照也可以，但是要

自己创建nginx用户，useradd -s /usr/sbin/nologin nginx

# systemctl stop nginx

# systemctl disable nginx

第一步:

先把squid停掉

安装nginx

使用源码版本编译

软件包在笔记目录下/lnmp\_soft/

nginx-1.8.0.tar.gz

ngx\_cache\_purge-2.3.tar.gz

# yum install pcre-devel zlib-devel gcc openssl-devel -y

复制压缩包到本地

tar xf nginx-1.8.0.tar.gz -C /usr/src/

tar xf ngx\_cache\_purge-2.3.tar.gz -C /usr/src/

cd /usr/src/nginx-1.8.0/

./configure --prefix=/usr/local/nginx --user=nginx --group=nginx --with-http\_gzip\_static\_module --with-http\_stub\_status\_module --add-module=../ngx\_cache\_purge-2.3/

如果上一步执行报错需要安装gcc: yum -y install gcc

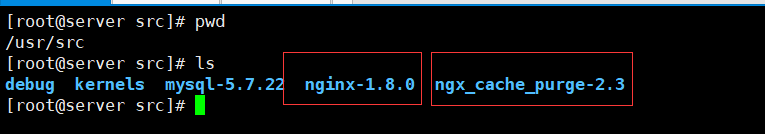
配置源码nginx 不加缓存模块功能

./configure --user=nginx --group=nginx --prefix=/usr/local/nginx --with-http\_stub\_status\_module --with-http\_ssl\_module --with-http\_realip\_module --with-http\_gzip\_static\_module

# make

# make install

使用--add-module=../ngx\_cache\_purge-2.3/参数加上缓存模块的功能，两个目录是同级目录（从编译的路径可以看出来）



第二步：

修改nginx主配置文件

# vim /usr/local/nginx/conf/nginx.conf

user nginx;

worker\_processes 2;

error\_log logs/error.log info;

pid logs/nginx.pid;

events {

worker\_connections 65535;

use epoll;

}

http {

include mime.types;

default\_type application/octet-stream;

log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

'$status $body\_bytes\_sent "$http\_referer" '

'"$http\_user\_agent" "$upstream\_cache\_status"';

sendfile on;

tcp\_nopush on;

keepalive\_timeout 65;

gzip on;

proxy\_temp\_path /usr/local/nginx/proxy\_temp\_dir 1 2;

proxy\_cache\_path /usr/local/nginx/proxy\_cache\_dir/cache levels=1:2 keys\_zone=cache:100m inactive=1d max\_size=10g;

upstream web {

server 192.168.224.11 weight=1 max\_fails=2 fail\_timeout=30s;

}

server {

listen 80;

server\_name 192.168.2.19;

access\_log logs/host.access.log main;

location / {

proxy\_pass http://web;

proxy\_set\_header Host $host;

proxy\_set\_header X-Forwarded-For $remote\_addr;

proxy\_cache cache;

proxy\_cache\_key $host$uri$is\_args$args;

proxy\_cache\_valid 200 304 10m;

add\_header X-Cache '$upstream\_cache\_status from $host';

expires 1d;

}

location ~ .\*\.(php|cgi)$ {

proxy\_pass http://web;

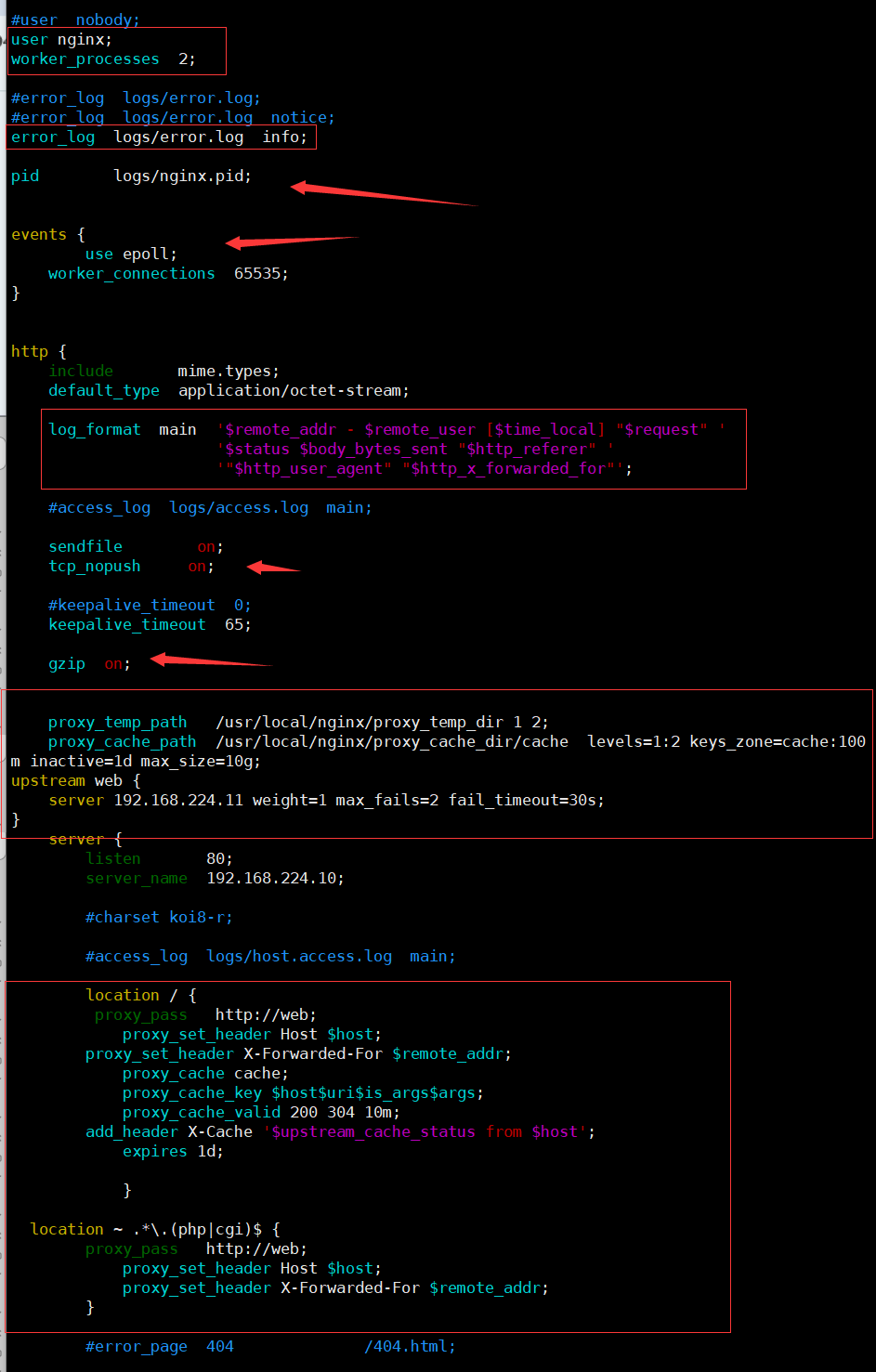
proxy\_set\_header Host $host;

proxy\_set\_header X-Forwarded-For $remote\_addr;

}

}

}



上面的配置参数说明

1、http段设置。

proxy\_temp\_path /usr/local/nginx/proxy\_temp\_dir; --设置临时目录

proxy\_cache\_path /usr/local/nginx/proxy\_cache\_dir/cache levels=1:2 keys\_zone=cache:100m inactive=1d max\_size=10g;

--keys\_zone=cache:100m 表示这个zone名称为cache，分配的内存大小为100MB

--/usr/local/nginx/proxy\_cache\_dir/cache 表示cache这个zone的文件要存放的目录

--levels=1:2 表示缓存目录的第一级目录是1个字符，第二级目录是2个字符，即/usr/local/nginx/proxy\_cache\_dir/cache/a/1b这种形式

--inactive=1d 表示这个zone中的缓存文件如果在1天内都没有被访问，那么文件会被cache manager进程删除掉

--max\_size=10g 表示这个zone的硬盘容量为10GB

2、server段设置

proxy\_cache cache; --设置缓存共享内存区块，也就是keys\_zone名称

proxy\_cache\_key $host$uri$is\_args$args; --设置缓存key

proxy\_cache\_valid 200 304 10m; --设置http状态码为200,304缓存时间为10分钟

add\_header X-Cache '$upstream\_cache\_status from $host'; --$upstream\_cache\_status表示资源缓存的状态，有HIT MISS EXPIRED三种状态

expires 1d; --设置失期时间，为1天

保存主配置文件后，建立对应的缓存目录

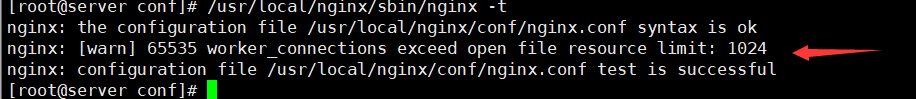
# mkdir /usr/local/nginx/proxy\_cache\_dir/cache -p

# ls /usr/local/nginx/proxy\_cache\_dir/cache

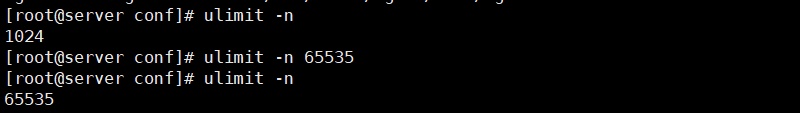
启动nginx

/usr/local/nginx/sbin/nginx

/usr/local/nginx/sbin/nginx -t

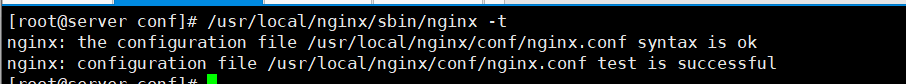


如果遇到句柄限制就执行 ulimit -n 65535



参考：停止服务 /usr/local/nginx/sbin/nginx -s stop

警告信息没有了



创建软连接，可以方便启动

ln -s /usr/local/nginx/sbin/nginx /usr/local/sbin/nginx

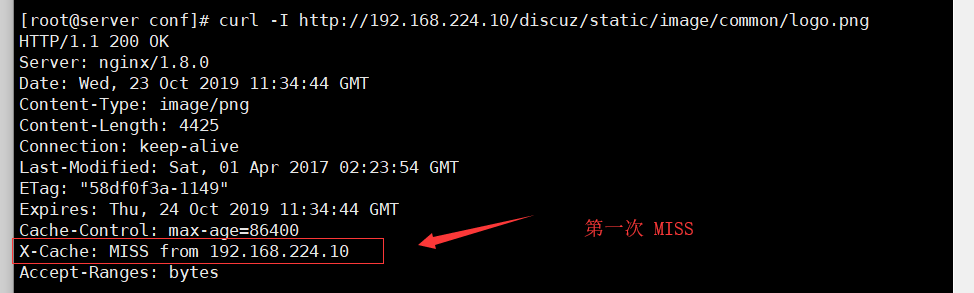
nginx -s quit 停止nginx

第三大步：

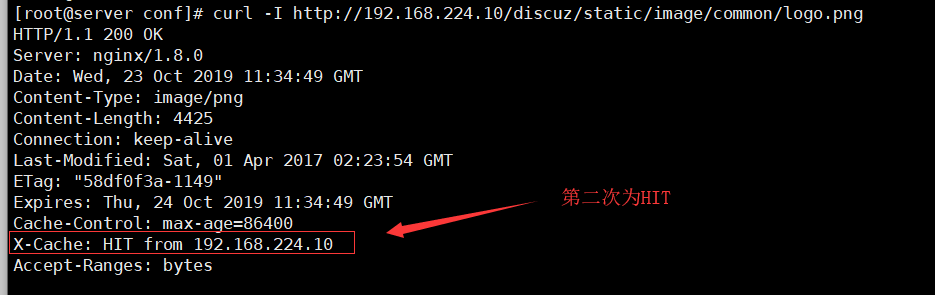
客户端测试

1，使用下面的命令访问

＃ curl -I http://192.168.224.10/discuz/static/image/common/logo.png



# curl -I http://192.168.224.10/discuz/static/image/common/logo.png



2，在客户端用户firefox访问http://192.168.224.10/discuz,可以访问整个discuz论坛

在nginx上查看缓存目录，会看到很多子目录（缓存都在这些目录里)

ls /usr/local/nginx/proxy\_cache\_dir/cache



3,nginx的缓存清除

在nginx服务器写一个脚本，如下

# vim /tmp/purge\_nginx\_cache.sh

#!/bin/bash

cachedir=/usr/local/nginx/proxy\_cache\_dir/cache

grep -ra $1 $cachedir |grep $1 | awk -F':' '{print $1,$3}'|while read cacheurl url

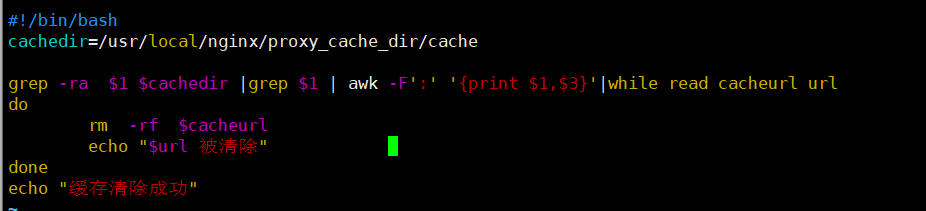
do

rm -rf $cacheurl

echo "$url 被清除"

done

echo "缓存清除成功"



清除方法为

bash /tmp/purge\_nginx\_cache.sh .png$ --清除所有的.png结尾的缓存

手动搜索缓存文件，txt后面要有$符号

grep -ra .png$ /usr/local/nginx/proxy\_cache\_dir/cache/