**Nginx企业级优化与防盗链**

# 一、配置Nginx隐藏版本号

在生产环境中，需要隐藏Nginx等服务的版本信息，以避免安全风险

实验前提前默认安装nginx-1.14.2

[root@localhost nginx-1.14.2]# curl -I 192.168.200.111

HTTP/1.1 200 OK

Server: nginx/1.14.2 #显示了网站服务的软件为nginx，版本1.14.2

Date: Sat, 05 Mar 2022 10:40:14 GMT

Content-Type: text/html

Content-Length: 612

Last-Modified: Sat, 05 Mar 2022 10:39:25 GMT

Connection: keep-alive

ETag: "62233ddd-264"

Accept-Ranges: bytes

[root@localhost nginx-1.14.2]# grep nobody /usr/local/nginx/conf/nginx.conf

#user nobody; #默认用户nobody

[root@localhost nginx-1.14.2]# ps aux|grep nginx|grep -v grep

root 14010 0.0 0.0 20544 608 ? Ss 18:39 0:00 nginx: master process nginx

nobody 14011 0.0 0.1 23072 1628 ? S 18:39 0:00 nginx: worker process

显示了nginx进程的执行用户是nobody

[root@localhost nginx-1.14.2]# ll /usr/local/nginx/

总用量 0

drwx------ 2 nobody root 6 3月 5 18:39 client\_body\_temp

drwxr-xr-x 2 root root 333 3月 5 18:39 conf

drwx------ 2 nobody root 6 3月 5 18:39 fastcgi\_temp

drwxr-xr-x 2 root root 40 3月 5 18:39 html

drwxr-xr-x 2 root root 58 3月 5 18:39 logs

drwx------ 2 nobody root 6 3月 5 18:39 proxy\_temp

drwxr-xr-x 2 root root 19 3月 5 18:39 sbin

drwx------ 2 nobody root 6 3月 5 18:39 scgi\_temp

drwx------ 2 nobody root 6 3月 5 18:39 uwsgi\_temp

1、修改源码包

**[root@localhost ~]# yum -y install pcre-devel zlib-devel openssl-devel gcc gcc-c++ make**

**[root@localhost ~]# useradd -M -s /sbin/nologin nginx**

**[root@localhost ~]# tar xf nginx-1.14.2.tar.gz -C /usr/src/**

**[root@localhost ~]# cd /usr/src/nginx-1.14.2/**

**[root@localhost nginx-1.14.2]# vim src/core/nginx.h**

**13 #define NGINX\_VERSION "7.0.0 "**

**14 #define NGINX\_VER "IIS/" NGINX\_VERSION**

**[root@localhost nginx-1.14.2]# ./configure --prefix=/usr/local/nginx --user=nginx --group=nginx && make && make install**

**[root@localhost nginx-1.14.2]# echo hello nginx >/usr/local/nginx/html/index.html**

**[root@localhost ~]# /usr/local/nginx/sbin/nginx**

[root@localhost nginx-1.14.2]# curl -I 192.168.200.111

HTTP/1.1 200 OK

Server: iis/7.0.0 #已经更改为iis7.0.0了

Date: Sat, 05 Mar 2022 11:03:09 GMT

Content-Type: text/html

Content-Length: 612

Last-Modified: Sat, 05 Mar 2022 11:02:22 GMT

Connection: keep-alive

ETag: "6223433e-264"

Accept-Ranges: bytes

2、修改配置文件

**[root@localhost ~]# vim /usr/local/nginx/conf/nginx.conf**

**keepalive\_timeout 65;**

**server\_tokens off; #**放在http段中

**[root@localhost ~]# killall -HUP nginx**

**[root@localhost ~]# curl -I http://192.168.200.111**

**HTTP/1.1 200 OK**

**Server: nginx #不显示具体版本号**

**Date: Thu, 17 Nov 2016 16:14:19 GMT**

**Content-Type: text/html**

**Content-Length: 612**

**Last-Modified: Thu, 17 Nov 2016 16:05:27 GMT**

**Connection: keep-alive**

**ETag: "582dd547-264"**

**Accept-Ranges: bytes**

# 二、修改Nginx用户与组

Nginx运行时进程需要有用户与组身份的支持，以实现对网站文件读取时进行访问控制。Nginx默认使用nobody用户账号与组账号，一般也要进行修改。

Linux 系统为了安全，很多操作和服务的运行都不是运行在 root 用户下面的，而是一个专用的 ID ，这个 ID 一般就是 nobody ，这样就可以把每个服务运行的情况隔离出来。保证不会因为服务器程序的问题而让服务器程序成了黑客的直接操作源（黑客拿下了服务器程序，也仅仅是 nobody 用户而不是 root 用户）。同时也不会影响其他用户的数据。nobody不是用来跑服务，而是用来占坑，权限会有限制，所以一般安装程序时要指定各个程序的用户组并设置相应的权限，一个程序的攻破不会影响到其他程序

**1、编译安装时指定（前面安装步骤已经指定用户，无需再重新编译安装）**

**[root@localhost ~]# useradd -M -s /sbin/nologin nginx**

**[root@localhost nginx-1.14.2]# ./configure --prefix=/usr/local/nginx --user=nginx --group=nginx && make && make install**

**2、修改配置文件**

**[root@localhost ~]# vim /usr/local/nginx/conf/nginx.conf**

**2 user nginx nginx;**

**[root@localhost ~]# killall -HUP nginx**

[root@localhost nginx-1.14.2]# ps aux|grep nginx

root 17995 0.0 0.1 20544 1344 ? Ss 19:03 0:00 nginx: master process /usr/local/nginx/sbin/nginx

nginx 18129 0.0 0.1 23076 1732 ? S 19:07 0:00 nginx: worker process

#worker process执行用户为nginx了

三、配置Nginx网页缓存时间

当Nginx将网页数据返回给客户端后，可设置资源在客户端缓存的时间，以方便客户端在日后进行相同内容的请求时直接返回，以避免重复请求，加快了访问速度，一般针对静态网页进行设置，对动态网页不用设置缓存时间。可在Windows客户端中使用fiddler查看网页缓存时间。

**设置方法：可修改配置文件，在http段、或server段、或者location段加入对特定内容的过期参数。**

**[root@localhost ~]# vim /usr/local/nginx/conf/nginx.conf**

**49 location ~ \.(gif|jpg|jpeg|png|bmp|ico)$ {**

**50 expires 1d; #指定图片格式文件缓存时间为1天（86400秒）**

**51 }**

**52 location ~ .\*\.(js|css)$ {**

**53 expires 1h;**

**54 }**

**[root@localhost ~]# killall -HUP nginx**

**[root@localhost ~]# ll /usr/local/nginx/html/**

**total 124**

**-rw-r--r-- 1 root root 537 Nov 18 00:05 50x.html**

**-rw-r--r-- 1 root root 643 Nov 18 00:21 index.html**

**-rw-r--r-- 1 root root 116309 Nov 14 11:54 linux.png**

**[root@localhost html]# curl -I 192.168.200.111/linux.png**

**HTTP/1.1 200 OK**

**Server: nginx**

**Date: Sat, 05 Mar 2022 11:32:50 GMT**

**Content-Type: image/png**

**Content-Length: 116309**

**Last-Modified: Sat, 05 Mar 2022 11:30:41 GMT**

**Connection: keep-alive #启用了连接保持时间**

**ETag: "622349e1-1c655"**

**Expires: Sun, 06 Mar 2022 11:32:50 GMT**

**Cache-Control: max-age=86400 #缓存时间86400（1d）**

**Accept-Ranges: bytes**

# 四、实现Nginx的日志切割

Nginx日志：

**[root@localhost ~]# ls -l /usr/local/nginx/logs/**

**总用量 12**

**-rw-r--r-- 1 root root 3584 3月 31 10:02 access.log #访问日志**

**-rw-r--r-- 1 root root 1575 3月 31 10:02 error.log #错误日志**

**-rw-r--r-- 1 root root 5 3月 31 09:27 nginx.pid**

编写日志切割脚本

**[root@localhost ~]# vim /opt/cut\_nginx\_log.sh**

**#!/bin/bash**

**# cut\_nginx\_log.sh**

**datetime=$(date -d "-1 day" "+%Y%m%d")**

**log\_path="/usr/local/nginx/logs"**

**pid\_path="/usr/local/nginx/logs/nginx.pid"**

**[ -d $log\_path/backup ] || mkdir -p $log\_path/backup**

**if [ -f $pid\_path ]**

**then**

**mv $log\_path/access.log $log\_path/backup/access.log-$datetime**

**kill -USR1 $(cat $pid\_path)**

**find $log\_path/backup -mtime +30 | xargs rm -f**

**else**

**echo "Error,Nginx is not working" | tee -a /var/log/messages**

**fi**

**[root@localhost ~]# chmod +x /opt/cut\_nginx\_log.sh**

**[root@localhost ~]# crontab -e**

**0 0 \* \* \* /opt/cut\_nginx\_log.sh**

**[root@localhost src]# date 03052359.50 #模拟时间到0点0分前10秒**

**2022年 03月 05日 星期六 23:59:50 CST**

**[root@localhost opt]# curl 192.168.200.111 #访问网站，产生日志文件内容**

**[root@localhost src]# ll /usr/local/nginx/logs/**

**-rw-r--r-- 1 nginx root 0 3月 6 00:00 access.log**

**drwxr-xr-x 2 root root 33 3月 6 00:00 backup #生成了backup**

**[root@localhost src]# ll /usr/local/nginx/logs/backup/**

**-rw-r--r-- 1 nginx root 0 3月 6 2022 access.log-20220305**

**保存了一个带时间标记的日志文件**

**[root@localhost src]# date**

**2022年 03月 06日 星期日 00:00:35**

**[root@localhost ~]# killall -9 nginx**

**[root@localhost ~]# rm -rf /usr/local/nginx/logs/nginx.pid**

**[root@localhost ~]# /opt/cut\_nginx\_log.sh**

**Error,Nginx is not working**

**[root@localhost ~]# tail -1 /var/log/messages**

**Error,Nginx is not working**

面试题讨论：

说明Nginx的访问日志记录在access.log文件中。

1. 如果将access.log重命名为a.log时，新产生的日志写到哪儿？为什么？

答案：日志将会写入到a.log文件中，因为重命名后文件的Inode没有变化，文件系统是根据Inode查找文件的。

1. 如果此时将Nginx服务重启后，新产生的日志写到哪儿？为什么？

答案：日志将会写入到新的access.log文件中，因为重启时会加载Nginx的配置文件，配置文件中是通过文件名指定日志的，所以会创建新的日志。

# 五、配置Nginx实现连接超时

保持连接（长连接）

在进行HTTP连接前要先建立TCP连接（TCP 3次握手），再建立HTTP连接，当HTTP资源请求结束后，会断开HTTP连接，再断开TCP连接（TCP 4次挥手）。

很多情况下用户访问网站并不是只访问一个资源，可能会打开很多页面，访问很多资源，如果每个资源的访问都这么繁琐，将会造成用户访问慢，服务器压力过大的问题。

解决如上问题的最好办法是开启网站的保持连接功能。在企业网站中，为了避免同一个客户长时间占用连接，造成服务器资源浪费，可以设置相应的连接超时参数，实现控制连接访问时间。

* **keepalived\_timeout：**设置连接保持超时时间，一般可只设置该参数，默认为65秒，可根据网站的情况设置，或者关闭，可在http段、server段、或者location段设置。

**[root@localhost ~]# vim /usr/local/nginx/conf/nginx.conf**

**34 keepalive\_timeout 65;**

**[root@localhost ~]# killall -HUP nginx**

# 六、更改Nginx工作进程数

在高并发场景，需要启动更多的nginx工作进程以保证快速访问，以处理用户的请求，避免造成阻塞。实验前先将cpu核心数设置为2个以匹配此实验

**修改配置文件的worker\_processes参数，一般设置为CPU的核数**

**[root@localhost ~]# grep 'core id' /proc/cpuinfo | uniq | wc -l**

**2**

**[root@localhost ~]# vim /usr/local/nginx/conf/nginx.conf**

**3 worker\_processes 2;**

**[root@localhost ~]# killall -HUP nginx**

**[root@localhost ~]# ps aux | grep nginx | grep -v grep**

**root 4431 0.0 0.2 45040 1160 ? Ss 00:50 0:00 nginx: master process /usr/local/nginx/sbin/nginx**

**nginx 4432 0.0 0.3 45492 1844 ? S 00:50 0:00 nginx: worker process**

**nginx 4433 0.0 0.3 45492 1756 ? S 00:50 0:00 nginx: worker process**

**配置CPU亲和性：**

**默认Nginx的多个进程可能跑在一颗CPU核心上，可以分配不同的进程给不同的CPU核心处理，充分利用硬件多核多CPU。在一台4核物理服务器，可以进行下面的配置，将进程进行分配。**

**worker\_cpu\_affinity 0001 0010 0100 1000;**

# 七、配置Nginx实现网页压缩功能

Nginx的ngx\_http\_gzip\_module压缩模块提供了对文件内容压缩的功能，允许nginx服务器将输出内容发送到客户端之前进行压缩，以节约网站带宽，提升用户的访问体验，模块默认已经安装。

**[root@localhost ~]# vim /usr/local/nginx/conf/nginx.conf**

**38 gzip on; //开启gzip压缩输出**

**39 gzip\_min\_length 1k; //用于设置允许压缩的页面最小字节数**

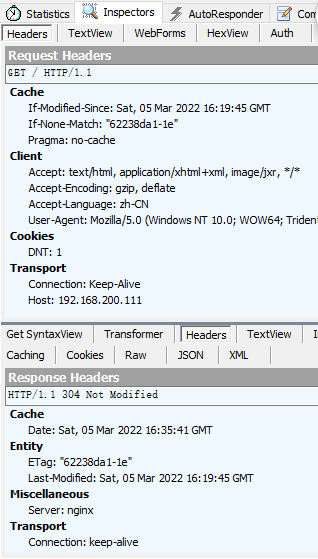
**40 gzip\_buffers 4 16k; //表示申请4个单位为16k的内存作为压缩结果流缓存，默认值是申请与原始数据大小相同的内存空间来储存gzip压缩结果**

**41 gzip\_http\_version 1.1; //设置识别http协议版本，默认是1.1**

**42 gzip\_comp\_level 2; //gzip压缩比，1-9等级**

**43 gzip\_types text/plain text/javascript application/x-javascript text/css text/xml application/xml application/xml+rss; //压缩类型，是就对哪些网页文档启用压缩功能**

**[root@localhost ~]# killall -HUP nginx**



# 八、配置Nginx实现防盗链功能

Nginx防盗链功能也非常强大，在默认情况下只需要进行简单的配置，即可实现防盗处理

## 实验环境：

**资源主机** [**www.source.com**](http://www.source.com) **192.168.200.111**

**盗链主机** [**www.steal.com**](http://www.steal.com) **192.168.200.112**

**准备工作**

**配置dns服务器，确保能正确解析此两个域名**

**[root@localhost html]# rpm -ivh /media/Packages/bind-9.9.4-61.el7.x86\_64.rpm**

**[root@localhost yum.repos.d]# cat /etc/named.conf**

**options { directory "/var/named"; };**

**zone "source.com" IN {**

**type master;**

**file "source";**

**};**

**zone "steal.com" IN {**

**type master;**

**file "steal";**

**};**

**[root@localhost named]# pwd**

**/var/named**

**[root@localhost named]# ll source steal**

**-rw-r----- 1 root named 165 3月 8 15:10 source**

**-rw-r----- 1 root named 165 3月 8 15:11 steal**

**[root@localhost named]# tail -2 source**

**NS www.source.com.**

**www A 192.168.200.111**

**[root@localhost named]# nslookup www.source.com**

**Server: 192.168.200.111**

**Address: 192.168.200.111#53**

**Name: www.source.com**

**Address: 192.168.200.111**

**[root@localhost named]# nslookup www.steal.com**

**Server: 192.168.200.111**

**Address: 192.168.200.111#53**

**Name: www.steal.com**

**Address: 192.168.200.112**

## 1、建立资源主机的测试主页

**[root@localhost ~]# vim /usr/local/nginx/html/index.html**

**<html>**

**<head>**

**<title>source page</title>**

**</head>**

**<body>**

**<h1>www.source.com</h1>**

**<img src="linux.png" />**

**</body>**

**</html>**

**[root@localhost ~]# ls /usr/local/nginx/html/**

**50x.html error.jpg index.html linux.png**

**客户机测试：（注意要关闭服务器的防火墙）**



## 2、建立盗链主机的测试主页

**[root@localhost ~]# vim /usr/local/nginx/html/index.html**

**<html>**

**<head>**

**<title>steal page</title>**

**</head>**

**<body>**

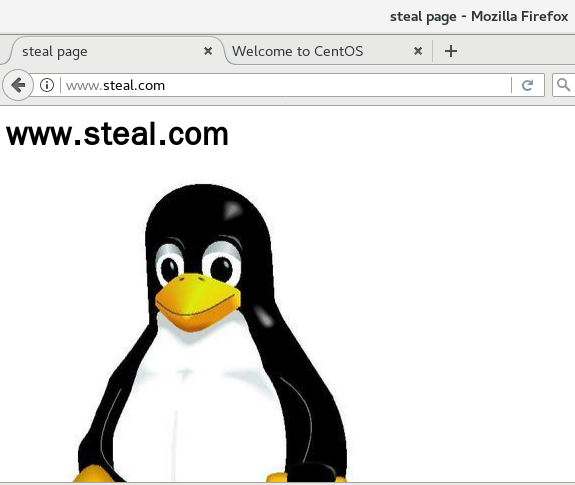
**<h1>www.steal.com</h1>**

**<img src="http://www.source.com/linux.png" />**

**</body>**

**</html>**

## 客户机测试：



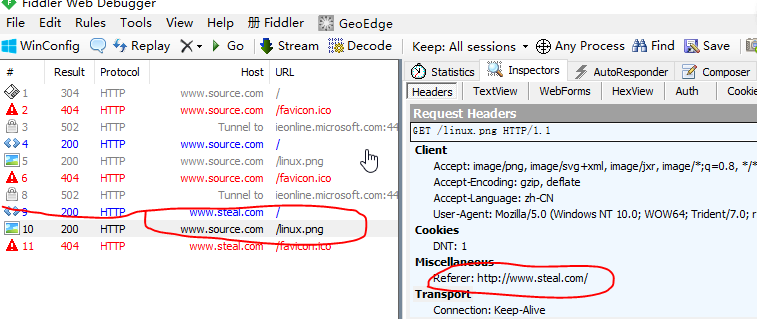
[**www.steal.com**](http://www.steal.com)**成功通过链接盗用**[**www.source.com**](http://www.source.com)**的图片，占用了source的带宽和资源**

## 3、资源主机设置防盗链

配置说明：

**valid\_referers** 设置信任网站

* **none** 浏览器中referer（Referer是header的一部分，当浏览器向web[服务器](http://baike.baidu.com/view/899.htm)发送请求的时候，一般会带上Referer，告诉服务器我是从哪个页面链接过来的，服务器基此可以获得一些信息用于处理）为空的情况，就直接在浏览器访问图片
* **blocked** referer不为空的情况，但是值被代理或防火墙删除了，这些值不以http://或https://开头，一般不是从外部来的



**[root@localhost ~]# ls /usr/local/nginx/html**

**error.jpg index.html linux.jpg**

**[root@localhost ~]# vim /usr/local/nginx/conf/nginx.conf**

**location ~\* \.(wma|wmv|asf|mp3|jpg|gif|png|swf|flv)$ {**

**valid\_referers none blocked \*.source.com source.com;**

**if ($invalid\_referer) {**

**return 403;**

**}**

**}**

**[root@localhost ~]# killall -HUP nginx**

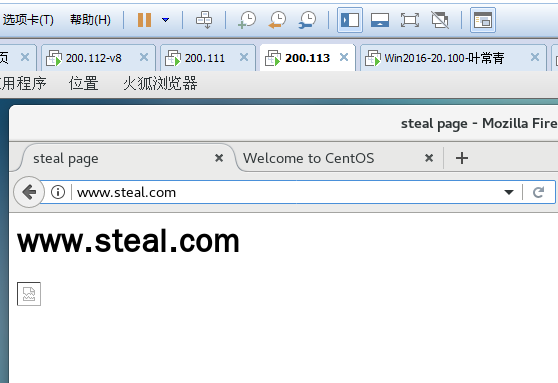
第一行：wma|wmv|asf|mp3|jpg|gif|png|swf|flv表示对这些后缀的文件实行防盗链

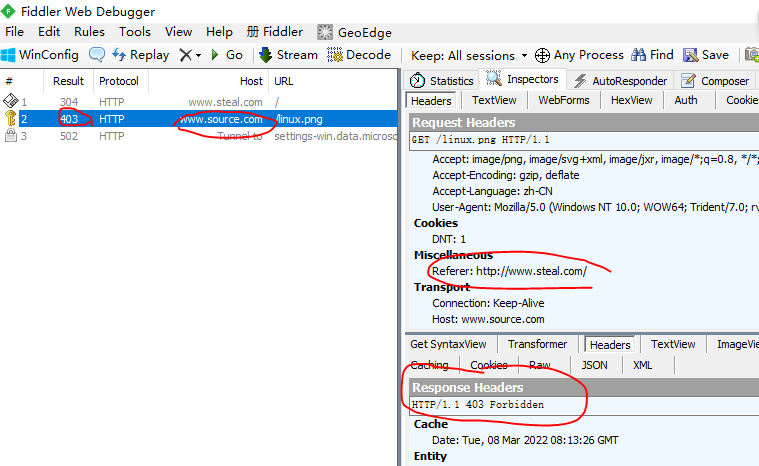
第二行：none blocked \*.source.com source.com; //不区分大小写

表示referers信息中匹配none blocked \*.source.com source.com (\*代表任何，任何的二级域名)

if{}里面内容的意思是，如果链接不是来自第二行指定的就强制跳转到403错误页面，当然直接返回404也是可以的，也可以是图片。

## 客户测试：





**[root@localhost html]# echo > /usr/local/nginx/logs/access.log**

**清空日志，访问后再查看日志，可以看到403代码信息**

**[root@localhost html]# cat /usr/local/nginx/logs/access.log**

**192.168.200.113 - - [08/Mar/2022:16:10:58 +0800] "GET /linux.png HTTP/1.1" 403 162 "http://www.steal.com/"**

# 九、Nginx 为目录添加访问控制

用户访问控制：使用apache 的 htpasswd 命令来创建密码文件

**[root@localhost ~]# yum -y install httpd-tools**

**[root@localhost ~]# htpasswd -c /usr/local/nginx/.htpasswd benet**

**New password:**

**Re-type new password:**

**Adding password for user benet**

**[root@localhost ~]# vim /usr/local/nginx/conf/nginx.conf**

**location / {**

**auth\_basic "Nginx access";**

**auth\_basic\_user\_file /usr/local/nginx/.htpasswd;**

**….**

**}**

客户端地址访问控制:

**[root@localhost ~]# vim /usr/local/nginx/conf/nginx.conf**

**location / {**

**auth\_basic "Nginx access";**

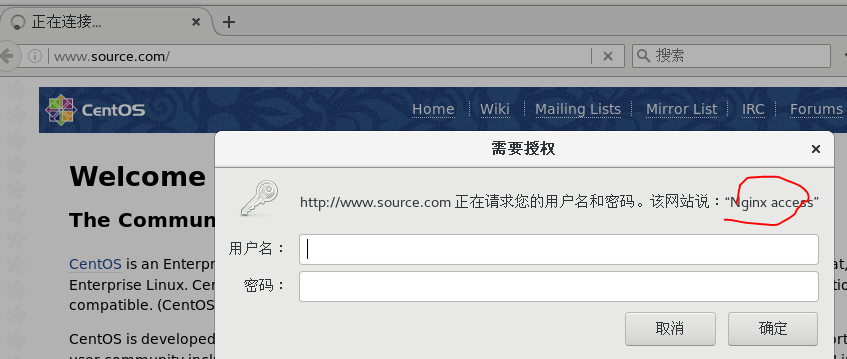
**auth\_basic\_user\_file /usr/local/nginx/.htpasswd;**

**allow 192.168.200.113;**

**deny all;**

**}**

**客户端访问测试：从192.168.200.113访问将弹出对话框验证，其他主机直接拒绝**



# 十、自定义错误页面

**error\_page 403 404 /404.html;**

**location = /404.html {**

**root html;**

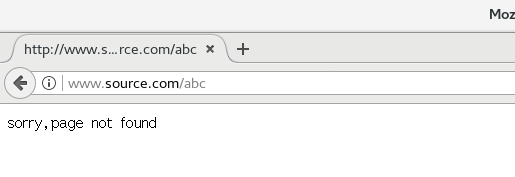
**}**

**[root@localhost html]# echo "Sorry,Page Not Found" > /usr/local/nginx/html/404.html**

**[root@localhost html]# killall -HUP nginx**

**浏览器访问** [**http://192.168.200.111/abc**](http://192.168.200.111/abc)

**返回结果： Sorry,Page Not Found**



**#访问不存在的目录显示404报错文本信息**

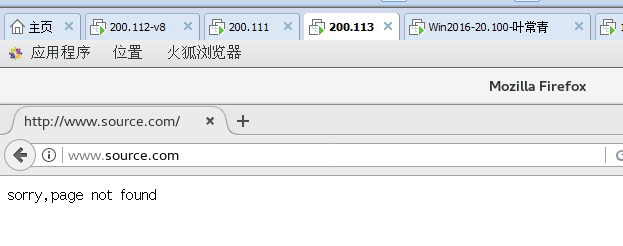
**[root@localhost html]# mv index.html index.html1 #将主页改名**

**[root@localhost html]# ls**

**404.html 50x.html index.html1 linux.png**

**[root@localhost html]# pwd**

**/usr/local/nginx/html**



**#无主页时显示403报错文本信息**

# 十一、自动索引

**location / {**

**root html;**

**index index.html index.htm;**

**autoindex on;**

**}**

**[root@localhost html]# killall -HUP nginx**

# 

**#添加自动索引后，访问不到主页就显示目录下的所有文件列表，不再显示403报错信息**

# 十二、目录别名功能：

**location /abc {**

**alias /var/www/html;**

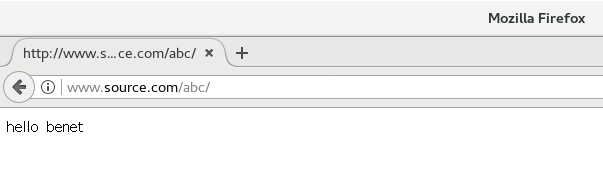
**index index.html;**

**}**

**[root@localhost html]# killall -HUP nginx**

**[root@localhost html]# mkdir /var/www/html**

**[root@localhost html]# echo hello benet >/html/index.html**



**#服务器根目录下并没有abc目录，由于做了目录别名，访问到了/var/www/html目录下，而不会再显示404报错信息了**

# 十三、通过UA实现手机端和电脑端的分离

实现nginx区分pc和手机访问不同的网站，是物理上完全隔离的两套网站(一套移动端、一套pc端),这样带来的好处pc端和移动端的内容可以不一样，移动版网站不需要包含特别多的内容，只要包含必要的文字和较小的图片，这样会更节省流量。有好处当然也就会增加困难，难题就是你需要维护两套环境，并且需要自动识别出来用户的物理设备并跳转到相应的网站，当判断错误时用户可以自己手动切换回正确的网站。

有两套网站代码，一套PC版放在/usr/local/nginx/html/web，一套移动版放在/usr/local/nginx/html/mobile。只需要修改nginx的配置文件，nginx通过UA来判断是否来自移动端访问，实现不同的客户端访问不同内容。

**location / {**

**#默认PC端访问内容**

**root /usr/local/nginx/html/web;**

**#如果是手机移动端访问内容**

**if ( $http\_user\_agent ~ "(MIDP)|(WAP)|(UP.Browser)|(Smartphone)|(Obigo)|(Mobile)|(AU.Browser)|(wxd.Mms)|(WxdB.Browser)|(CLDC)|(UP.Link)|(KM.Browser)|(UCWEB)|(SEMC-Browser)|(Mini)|(Symbian)|(Palm)|(Nokia)|(Panasonic)|(MOT-)|(SonyEricsson)|(NEC-)|(Alcatel)|(Ericsson)|(BENQ)|(BenQ)|(Amoisonic)|(Amoi-)|(Capitel)|(PHILIPS)|(SAMSUNG)|(Lenovo)|(Mitsu)|(Motorola)|(SHARP)|(WAPPER)|(LG-)|(LG/)|(EG900)|(CECT)|(Compal)|(kejian)|(Bird)|(BIRD)|(G900/V1.0)|(Arima)|(CTL)|(TDG)|(Daxian)|(DAXIAN)|(DBTEL)|(Eastcom)|(EASTCOM)|(PANTECH)|(Dopod)|(Haier)|(HAIER)|(KONKA)|(KEJIAN)|(LENOVO)|(Soutec)|(SOUTEC)|(SAGEM)|(SEC-)|(SED-)|(EMOL-)|(INNO55)|(ZTE)|(iPhone)|(Android)|(Windows CE)|(Wget)|(Java)|(curl)|(Opera)" )**

**{**

**root /usr/local/nginx/html/mobile;**

**}**

**index index.html index.htm;**

**}**

## 实验模拟：不同浏览器访问到不同的页面:

**[root@localhost html]# mkdir firefox msie**

**[root@localhost html]# echo "hello,firefox" > firefox/index.html**

**[root@localhost html]# echo "hello,msie" > msie/index.html**

**location / {**

**if ($http\_user\_agent ~ Firefox) {**

**root /usr/local/nginx/html/firefox;**

**}**

**if ($http\_user\_agent ~ MSIE|Mozilla) {**

**root /usr/local/nginx/html/msie;**

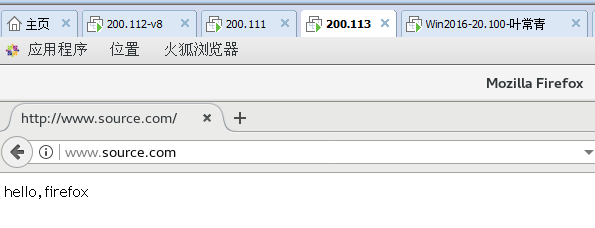
**}**

**index index.html index.htm;**

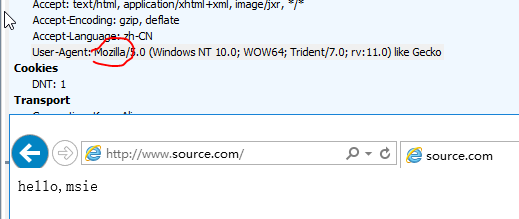
**}**

**#将原来location /的内容替换为上面内容**

**[root@localhost html]# killall -HUP nginx**



**#在linux客户端的火狐浏览器里显示firefox内容**



#在windows的ie浏览器里显示msie内容，360浏览器也显示这个内容

# 十四、nginx平滑升级版本

## 一、nginx平滑升级概述

随着网站并发访问量越来越高，nginx web 服务器也越来越流行，nginx 版本换代越来越频繁，1.15.2版本的nginx更新了许多新功能，生产环境中版本升级必然的，但是线上业务不能停，此时nginx的升级就是运维的重要工作了。

## 二、nginx平滑升级原理

多进程模式下的请求分配方式

Nginx默认工作在多进程模式下，即主进程（master process）启动后完成配置加载和端口绑定等动作，fork出指定数量的工作进程（worker process），这些子进程会持续监听端口的文件描述符（fd），并通过在该描述符上添加监听事件来接受连接（accept）。

信号的接收和处理

Nginx主进程在启动完成后会进入等待状态，负责响应各类系统消息，如HUP、USR2等。

Nginx信号简介

主进程支持的信号

QUIT: 等待工作进程结束后再退出

KILL: 强制终止进程

HUP: 重新加载配置文件，使用新的配置启动工作进程，并逐步关闭旧进程。

USR1: 重新打开日志文件

USR2: 启动新的主进程，实现热升级

WINCH: 逐步关闭工作进程

工作进程支持的信号

QUIT: 等待请求处理结束后再退出

## 三、nginx平滑升级实战

### 1.查看旧版nginx的编译参数

[root@localhost logs]# nginx -V

nginx version: nginx/1.14.2

built by gcc 4.8.5 20150623 (Red Hat 4.8.5-28) (GCC)

configure arguments: --prefix=/usr/local/nginx --user=nginx --group=nginx

### 2.编译新版本Nginx源码包，安装路径需与旧版一致，注意：不要执行make install

**[root@localhost ~]# tar xf nginx-1.15.9.tar.gz -C /usr/src/**

**[root@localhost ~]# cd /usr/src/nginx-1.15.9/**

**[root@localhost nginx-1.15.9]# ./configure --prefix=/usr/local/nginx --user=nginx --group=nginx && make**

### 3.备份二进制文件，用新版本的替换

**[root@localhost nginx-1.15.9]# mv /usr/local/nginx/sbin/nginx /usr/local/nginx/sbin/nginx.old**

**[root@localhost nginx-1.15.9]# ls**

**auto CHANGES.ru configure html Makefile objs src**

**CHANGES conf contrib LICENSE man README**

**[root@localhost nginx-1.15.9]# cp objs/nginx /usr/local/nginx/sbin/**

### 4.确保配置文件无报错

**[root@localhost nginx-1.15.9]# /usr/local/nginx/sbin/nginx -t**

**nginx: the configuration file /usr/local/nginx/conf/nginx.conf syntax is ok**

**nginx: configuration file /usr/local/nginx/conf/nginx.conf test is successful**

### 5.发送USR2信号

向主进程（master）发送USR2信号，Nginx会启动一个新版本的master进程和对应工作进程，和旧版一起处理请求

**[root@localhost ~]# ps aux | grep nginx | grep -v grep**

**root 4108 0.0 0.2 45028 1152 ? Ss 16:58 0:00 nginx: master process /usr/local/nginx/sbin/nginx**

**nginx 4109 0.0 0.4 45456 2012 ? S 16:58 0:00 nginx: worker process**

**[root@localhost ~]# kill -USR2 4108**

**[root@localhost ~]# ps aux | grep nginx | grep -v grep**

**root 4108 0.0 0.2 45028 1316 ? Ss 16:58 0:00 nginx: master process /usr/local/nginx/sbin/nginx**

**nginx 4109 0.0 0.4 45456 2012 ? S 16:58 0:00 nginx: worker process**

**root 6605 0.5 0.6 45196 3364 ? S 17:02 0:00 nginx: master process /usr/local/nginx/sbin/nginx**

**nginx 6607 0.0 0.3 45624 1756 ? S 17:02 0:00 nginx: worker process**

### 6.发送WINCH信号

向旧的Nginx主进程（master）发送WINCH信号，它会逐步关闭自己的工作进程（主进程不退出），这时所有请求都会由新版Nginx处理

**[root@localhost ~]# kill -WINCH 4108**

**[root@localhost ~]# ps aux | grep nginx | grep -v grep**

**root 4108 0.0 0.2 45028 1320 ? Ss 16:58 0:00 nginx: master process /usr/local/nginx/sbin/nginx**

**root 6605 0.0 0.6 45196 3364 ? S 17:02 0:00 nginx: master process /usr/local/nginx/sbin/nginx**

**nginx 6607 0.0 0.3 45624 1756 ? S 17:02 0:00 nginx: worker process**

注意：回滚步骤，发送HUP信号

如果这时需要回退继续使用旧版本，可向旧的Nginx主进程发送HUP信号，它会重新启动工作进程， 仍使用旧版配置文件。然后可以将新版Nginx进程杀死（使用QUIT、TERM、或者KILL）

### 7.发送QUIT信号

升级完毕，可向旧的Nginx主进程（master）发送（QUIT、TERM、或者KILL）信号，使旧的主进程退出

**[root@localhost ~]# kill -QUIT 4108**

**[root@localhost ~]# ps aux | grep nginx | grep -v grep**

**root 6605 0.0 0.6 45196 3364 ? S 17:02 0:00 nginx: master process /usr/local/nginx/sbin/nginx**

**nginx 6607 0.0 0.4 45624 2056 ? S 17:02 0:00 nginx: worker process**

### 8.验证nginx版本号，并访问测试

**[root@localhost nginx-1.15.9]# /usr/local/nginx/sbin/nginx -v**

**nginx version: nginx/1.15.9**

**#升级时网站没有停，原网站数据可以继续访问**

# 附录：相关资源信息查看

**查看CPU信息：/proc/cpuinfo文件**

**[root@localhost ~]# cat /proc/cpuinfo**

**processor : 0 #**CPU线程数

**vendor\_id : GenuineIntel**

**cpu family : 6**

**model : 61**

**model name : Intel(R) Core(TM) i5-5300U CPU @ 2.30GHz #**CPU的名字及其编号、主频

**stepping : 4**

**microcode : 0x2a**

**cpu MHz : 2294.691**

**cache size : 3072 KB**

**physical id : 0 #**物理CPU个数编号，0是第一个，1是第二个

**siblings : 1**

**core id : 0 #**CPU核心数编号

**cpu cores : 1**

**apicid : 0**

**查看内存信息：free**

[**root@docker ~]# free -m**

**total used free shared buff/cache available**

**Mem: 7822 3565 484 107 3772 3812**

**Swap: 0 0 0**

**查看磁盘使用情况:df**

**[root@localhost ~]# df -hT**

**文件系统 类型 容量 已用 可用 已用% 挂载点**

**/dev/mapper/centos-root xfs 37G 5.2G 32G 14% /**

**devtmpfs devtmpfs 471M 0 471M 0% /dev**

**tmpfs tmpfs 488M 0 488M 0% /dev/shm**

**tmpfs tmpfs 488M 8.1M 480M 2% /run**

**tmpfs tmpfs 488M 0 488M 0% /sys/fs/cgroup**

**/dev/sr0 iso9660 4.2G 4.2G 0 100% /media/cdrom**

**/dev/sda1 xfs 1014M 157M 858M 16% /boot**

**mii-tool，ethtool：用于查看、管理物理的网络接口的状态,还可以配置网卡需要的协商方式**

**[root@bogon ~]# mii-tool ens33**

**ens33: negotiated 1000baseT-FD flow-control, link ok**