# Nginx模块简介

Apache

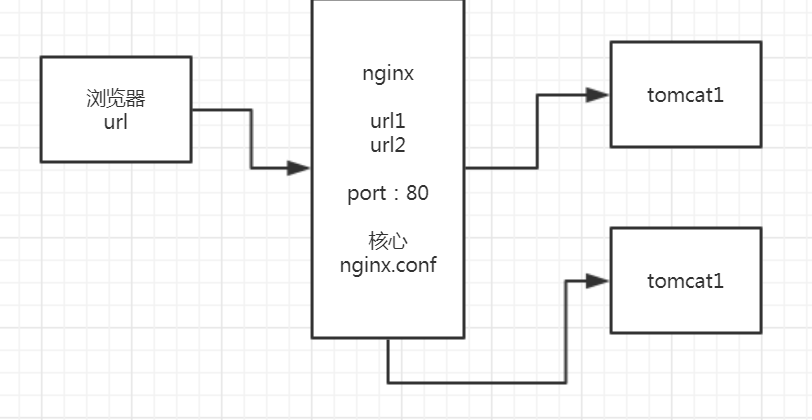
Apache仍然是时长占用量最高的web服务器，据最新数据统计，市场占有率目前是50%左右。主要优势在于一个是比较早出现的一个Http静态资源服务器，同时又是开源的。所以在技术上的支持以及市面上的各种解决方案都比较成熟。Apache支持的模块非常丰富。

Nginx

Nginx是俄罗斯人编写的一款高性能的HTTP和反向代理服务器，在高连接并发的情况下，它能够支持高达50000个并发连接数的响应，但是内存、CPU等系统资源消耗却很低，运行很稳定。目前Nginx在国内很多大型企业都有应用，据最新统计，Nginx的市场占有率已经到33%左右了。而Apache的市场占有率虽然仍然是最高的，但是是呈下降趋势。而Nginx的势头很明显。选择Nginx的理由也很简单：第一，它可以支持5W高并发连接；第二，内存消耗少；第三，成本低，如果采用F5、NetScaler等硬件负载均衡设备的话，需要大几十万。而Nginx是开源的，可以免费使用并且能用于商业用途

# Nginx工作原理及安装配置

## 一、nginx结构示意图

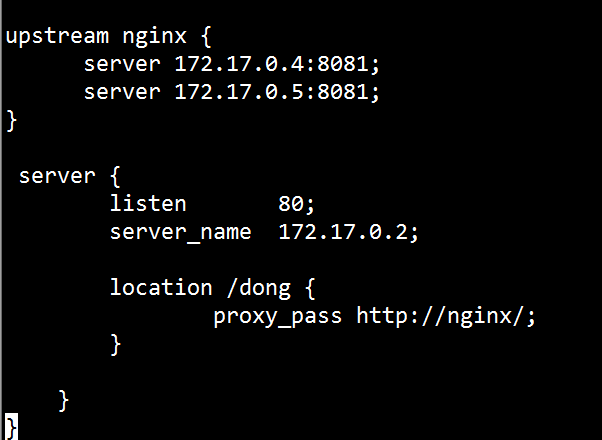


1、nginx上面记录后台两台tomcat1和tomcat2的访问路径。

2、nginx监听系统port（如80端口），当有浏览器url请求到达时，nginx根据此url结构信息，将请求转发至后台两个tomcat

3、转发至后台tomcat的策略，是一个负载均衡里的配置策略。

4、以上三个步骤，均是在核心文件nginx.conf中配置（如下面这个配置信息）。



5、配置中的upstream里，记录了tomcat1和tomcat2的地址

6、配置中的server里，监听了80端口

7、配置中的location /dong，是分析url的结构（路径是/dong时），是这种结构的，就进入请求转发命令proxy\_pass.

8、proxy\_pass是代理转发命令，将请求传递到 upstream（名字为nginx）配置对应的模块。

9、upstream按负载策略（默认轮流两个地址），把请求发到配置的两后台tomcat地址

## nginx在系统架构（网关入口）中的作用

1、路由功能（与微服务对应）：域名/路径，进行路由选择后台服务器, 横向拓展，集群

2、负载功能（与高并发高可用对应）：对后台服务器集群进行负载, 横向拓展，提升高并发性能.

3、静态服务器（比tomcat性能高很多）：在mvvm模式中，充当文件读取职责

总结：实际使用中，这三项功用，会混合使用。比如先分离动静，再路由服务，再负载机器

1、代理：客户端自己请求出现困难。客户请了一个代理，来代自己做事，就叫代理。

    比如代理律师，代购，政府机关办事的代理人等等。

2、反向代理，服务端推出的一个代理招牌。**反向代理**是[代理服务器](https://baike.baidu.com/item/%E4%BB%A3%E7%90%86%E6%9C%8D%E5%8A%A1%E5%99%A8/97996)的一种。服务器根据客户端的请求，从其关系的一组或多组后端[服务器](https://baike.baidu.com/item/%E6%9C%8D%E5%8A%A1%E5%99%A8)（如[Web服务器](https://baike.baidu.com/item/Web%E6%9C%8D%E5%8A%A1%E5%99%A8" \t "_blank)）上获取资源，然后再将这些资源返回给客户端，客户端只会得知反向代理的IP地址，而不知道在代理服务器后面的服务器簇的存在[1]。

与前向代理不同，前向代理作为客户端的代理，将从互联网上获取的资源返回给一个或多个的客户端，服务端（如Web服务器）只知道代理的IP地址而不知道客户端的IP地址；而反向代理是作为服务器端（如Web服务器）的代理使用，而不是客户端。客户端借由前向代理可以间接访问很多不同互联网服务器（簇）的资源，而反向代理是供很多客户端都通过它间接访问不同后端服务器上的资源，而不需要知道这些后端服务器的存在，而以为所有资源都来自于这个反向代理服务器。

## nginx安装

**源码编译方式：**

安装make：yum -y install autoconf automake make

安装g++: yum -y install gcc gcc-c++

#一般系统中已经装了了make和g++，无须再装

yum -y install pcre pcre-devel

yum -y install zlib zlib-devel

yum install -y openssl openssl-devel

#安装nginx依赖的库

直接下载

wget  <http://nginx.org/download/nginx-1.9.15.tar.gz>

tar -zxvf nginx-1.9.0.tar.gz

cd nginx-1.9.0

./configure   --prefix=/usr/local/nginx --with-http\_stub\_status\_module --with-http\_ssl\_module

#配置

#--prefix指定安装目录

#--with-http\_ssl\_module安装https模块

#creating objs/Makefile 代表编译成功

make && make install

#make编译

#make install安装

**yum方式：**

**先执行yum扩展源**

1. centos6：yum install epel-release -y   #yum扩展源
2. yum install nginx -y

# Nginx常用命令管理及升级

**启停命令：**

./nginx -c nginx.conf的文件。如果不指定，默认为NGINX\_HOME/conf/nginx.conf

./nginx -s stop  停止

./nginx -s quit退出

./nginx -s reload 重新加载nginx.conf

**发送信号的方式**

kill -QUIT  进程号 安全停止

kill -TERM  进程号 立即停止

**Nginx会按需同时运行多个进程：**

一个主进程(master)和几个工作进程(worker)，配置了缓存时还会有缓存加载器进程(cache loader)和缓存管理器进程(cache manager)等。

所有进程均是仅含有一个线程，并主要通过“共享内存”的机制实现进程间通信。

主进程以root用户身份运行，而worker、cache loader和cache manager均应以非特权用户身份（user配置项）运行。

**主进程主要完成如下工作：**

1. 读取并验正配置信息；

2. 创建、绑定及关闭套接字；

3. 启动、终止及维护worker进程的个数；

4. 无须中止服务而重新配置工作特性；

5. 重新打开日志文件；

**worker进程主要完成的任务包括：**

1. 接收、传入并处理来自客户端的连接；

2. 提供反向代理及过滤功能；

3. nginx任何能完成的其它任务

# Nginx配置文件精讲

## 全局属性的配置

#user nobody; #主模块命令， 指定Nginx的worker进程运行用户以及用户组，默认由nobody账号运行。

worker\_processes 8;#指定Nginx要开启的进程数。

worker\_rlimit\_nofile 100000; #worker进程的最大打开文件数限制

#error\_log logs/error.log;

#error\_log logs/error.log notice;

#error\_log logs/error.log info;

#pid logs/nginx.pid;

events {

use epoll; //默认

worker\_connections 51200;//线程读文件的限制数

}

/\*

**以上这块配置代码是对nginx。**

  user :主模块命令， 指定Nginx的worker进程运行用户以及用户组，默认由nobody账号运行。

  worker\_processes: 指定Nginx要开启的进程数。

  error log:用来定义全局错设日志文件的路径和日志名称。

日志输出级别有debug，info，notice，warn，error，crit 可供选择，其中debug输出日志最为详细，面crit（严重）输出日志最少。默认是error

  pid: 用来指定进程id的存储文件位置。

  event：设定nginx的工作模式及连接数上限，

其中参数use用来指定nginx的工作模式（这里是epoll，epoll是多路复用IO(I/O Multiplexing)中的一种方式）,

nginx支持的工作模式有select ,poll,kqueue,epoll,rtsig,/dev/poll。

其中select和poll都是标准的工作模式，kqueue和epoll是高效的工作模式，对于linux系统，epoll是首选。

    worker\_connection是设置nginx每个进程最大的连接数，默认是1024，所以nginx最大的连接数max\_client=worker\_processes \* worker\_connections。

进程最大连接数受到系统最大打开文件数的限制，需要设置ulimit。

\*/

## http服务器相关属性的设置

http {

include mime.types; 主模块命令，对配置文件所包含文件的设定，减少主配置文件的复杂度，相当于把部分设置放在别的地方，然后在包含进来，保持主配置文件的简洁

default\_type application/octet-stream; 默认文件类型，当文件类型未定义时候就使用这类设置的。

#log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" ' 指定nginx日志的格式

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

# '"$http\_user\_agent" "$http\_x\_forwarded\_for"';

#access\_log logs/access.log main;

sendfile on; 开启高效文件传输模式（zero copy 方式），避免内核缓冲区数据和用户缓冲区数据之间的拷贝。

#tcp\_nopush on; 开启TCP\_NOPUSH套接字（sendfile开启时有用）

#keepalive\_timeout 0; 客户端连接超时时间 单位是s

keepalive\_timeout 65;

#gzip on; 设置是否开启gzip模块

## server段虚拟主机的配置

1、每一个server，对应一个监听port端口，一个ip/域名

2、nginx里面可配置多个server，多个upstream

3、每个server配置多个location

4、一般配置方式：

在/etc/nginx/con.d/目录下，放置多个server配置文件，每个配置文件里仅含一个server配置。

然后在主配置nginx.conf中 ： include 文件夹下所以server配置 ==== jsp里的include

server {

listen 80; 虚拟主机的服务端口

server\_name localhost; 用来指定ip或者域名，多个域名用逗号分开

#charset koi8-r;

location / {

#地址匹配设置，支持正则匹配，也支持条件匹配，这里是默认请求地址，用户可以location命令对nginx进行动态和静态网页过滤处理

root html; 虚拟主机的网页根目录

index index.html index.htm; 默认访问首页文件

}

#error\_page 404 /404.html;

# redirect server error pages to the static page /50x.html

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

location = /50x.html {

root html;

}

}

root/alias时， index命令

url 以 /结尾时，如：http://location.enjoy.com/static/，指的是一个目录，nginx认为用户没有指定文件

url 不是以 / 结尾，http://location.enjoy.com/static，认为它是个文件，尝试打开这个文件，此时index命令不启用

## upstream/location/rewrite功用

upstream --- 用以配置负载的策略，nginx自带的有：轮询/权重/ip\_hash。特殊需求可用第三方策略（使用较少）。

location:----用以匹配url信息结构，配足即进入对应逻辑块。是一套匹配逻辑的小型语法，核心是做url匹配。

rewrite：----用以修改url信息（ip+port部分无法改）的path部分，做重定向。与servlet类似，有页面重定向和内部重定向两类。

proxy\_pass----用以转发命令。是实际的转发执行者。它可将流程转给upstream模块。

## location语法规则：

location [=|~|~\*|^~] /uri/ {… }

最常用：

location = / =表示精准匹配

location ^~ /static/ ^~ uri以某个字符开头

location ~\* \.(gif|png|css|js)$ ~\*正则匹配

首先匹配 =，其次匹配^~,其次是按文件中顺序的正则匹配，最后是交给 /通用匹配。当有匹配成功时候，停止匹配，按当前匹配规则处理请求。

|  |  |
| --- | --- |
| 符号 | 含义 |
| = | = 开头表示精确匹配 |
| ^~ | ^~开头表示uri以某个常规字符串开头，理解为匹配 url路径即可。nginx不对url做编码，因此请求为/static/20%/aa，可以被规则^~ /static/ /aa匹配到（注意是空格） |
| ~ | ~ 开头表示区分大小写的正则匹配 |
| ~\* | ~\* 开头表示不区分大小写的正则匹配 |
| !~和!~\* | !~和!~\*分别为区分大小写不匹配及不区分大小写不匹配的正则 |
| / | 用户所使用的代理（一般为浏览器） |
|  |  |

Rewrite:

proxy\_pass:反向代理 http://或https://开头

root:指定路径

alias:别名,指向目录路径.

## nginx执行location流程示例

1、通过url中【ip/域名+port】匹配server之后，携带后面path?params进入location逻辑

2、location匹配后，path = 匹配中的path1 + 剩余path2



如：http://process.enjoy.com /nginx/enjoy = 命中路径 /getInfo = 剩余路径

3、如这条url：

http://process.enjoy.com/dynamic/nginx/enjoy/getInfo里面/dynamic，是被location匹配中的path1，而tomcat的路径不能有/ dynamic，nginx转发时必须中传path2，舍弃path1

4、root命令情况

看url---------http://process.enjoy.com /static/a.html如下图

location匹配后，path1=/static，path2=/a.html。

root方式是在目标html文件夹内，查找路径path1+path2=/static/a.html的文件。

5、看url---------http://process.enjoy.com /target /a.html如下图



location匹配后，path1=/target，path2=/a.html。

alias方式是在目标html/static/文件夹内，查找路径path2= /a.html的文件。

6、root/alias 什么命令，目录 是怎么配的

1、是声明，不是命令;

相当于java中：string rootPath = "/usr/local/html"; //就是根目录预置变量

rootPath = 初始值; ======/usr/local/nginx/html

nginx安装的时候，有一个目录就html目录（此目录就是用做初始值用）。

7、当在生产放文件，会是其它目录。不写root，会找错文件夹，定会错

root尽量都用绝对路径

## location实操

a、location匹配分出path1和path2

http://process.enjoy.com/static/a.html ----- path == /static/a.html

与 location /static 命中：path1 = /static ，剩余path2 = /a.html



b、代理转发path。

分两种，静态文件时，转给static处理模块 == 参数path

动态代理时，转给后台web服务器 == 参数path

root转整个path1+path2

alias只转path2

root方式.

http://process.enjoy.com/static/a.html ---- 命中/static， path1=/static ，path2= /a.html

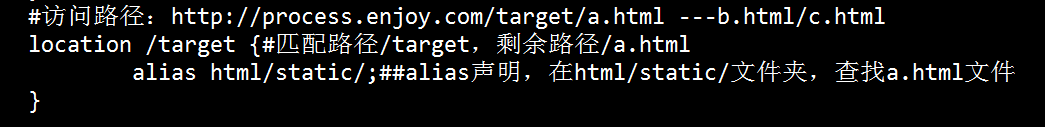
转整个path1+path2 给 静态文件处理模块



alias方式.

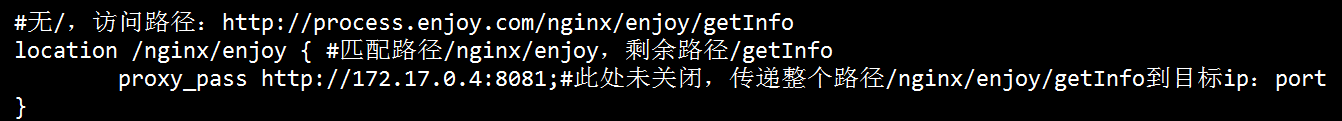
http://process.enjoy.com/target/a.html ---- 命中/ target， path1=/ target，path2= /a.html

转整个path2 给 静态文件处理模块



c、nginx中内容生成，有两种：静态文件处理模块 和 proxy\_pass转的第三方

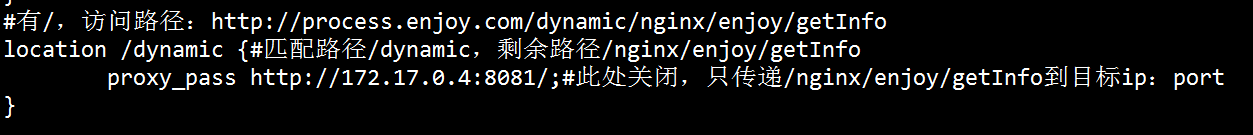
http://process.enjoy.com/nginx/enjoy/getInfo，命中/nginx/enjoy，path1=/nginx/enjoy，path2=/getInfo。 proxy\_pass http://172.17.0.4:8081;未关闭，此处传 path1+path2



http://process.enjoy.com/dynamic/nginx/enjoy/getInfo，命中/dynamic,

path1=/dynamic,path2=/nginx/enjoy/getInfo。

proxy\_pass http://172.17.0.4:8081/;有关闭，此处只传path2



**proxy\_pass的关闭与不关闭。针对ip+port之后有没有/**

d、内容生成部分，根据前面传过来信息，生成内容，返回页面。response功能

location运算符，优先级处理

1、只有 = 是精匹配，不会被正则覆盖

2、普通匹配，即使是完全匹配，也不是精准匹配

e、echo命令：**它是一个内容生成模块（content阶段）里的命令---echo出来的值，就是页面内容**

**echo命令，不是nginx默认自带模块，是个第三方模块**

**echo模块安装：**

**下载：wget** [**https://github.com/openresty/echo-nginx-module/archive/v0.60.tar.gz**](https://github.com/openresty/echo-nginx-module/archive/v0.60.tar.gz)

解压：tar -zxvf [**v0.60.tar.gz**](https://github.com/openresty/echo-nginx-module/archive/v0.60.tar.gz)

配置：

./configure  --add-module=/usr/local/src/echo-nginx-module-0.60  --with-http\_ssl\_module --with-debug

编译&安装：

make && make install

server {

listen       80;

server\_name  location.enjoy.com;

location /a {

        alias :html;

}

location /b/a {

        rewrite ^/  /b.html break;

        root   html/static/;

}

location /b/d/a {

        rewrite ^/  /d.html break;

        root   html/static/;

}

location ^~/b/c/a {

        rewrite ^/  /d.html break;

        root   html/static/;

}

location ~ /b/d {

        rewrite ^/  /c.html break;

        root   html/static/;

}

location ~ /b/d/a {

        rewrite ^/  /a.html break;

        root   html/static/;

}

}

## Rewrite的使用

rewrite功能就是，使用nginx提供的全局变量或自己设置的变量，

结合正则表达式和标志位实现url重写以及重定向。

rewrite只能放在server{},location{},if{}中，

并且只能对域名后边的除去传递的参数外的字符串起作用，

例如http://seanlook.com/a/we/index.**jsp**?id=1&u=str 只对/a/we/index.php重写。

语法rewrite regex replacement [flag];

rewrite ^/rout/(.\*)  $1.html break;

**rewrite regex replacement [flag]; flag=【break/last/redirect/permanent】**

regex 是正则表达式

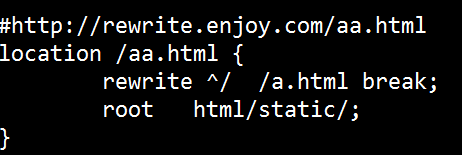
replacement 是替换值，新值

flag -- 是处理标志

例：

http://rewrite.enjoy.com/aa.html --命中 /aa.html ,path=aa.html

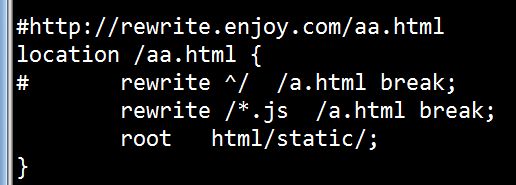
本来应该在html/static文件夹下查找aa.html

 正则 ^/ 表示：总是匹配

若rewrite正则匹配命中，则修改为path = /a.html（若不命中呢，不执行）。

变成了在html/static文件夹下查找a.html

e、rewrite不命中，则rewrite更换path运作不会发生

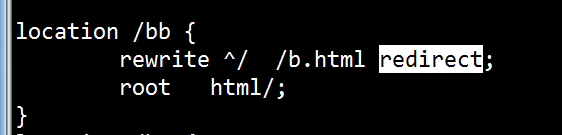


----相对servlet里面，是什么命令。内部重定向。外部人，浏览器不知道的

flag=【break/last/redirect/permanent/不写】含义。可以没有的。

/redirect/permanent，如果rewrite命中，发页面重定向

#http://rewrite.enjoy.com/b.html



会发生页面重定向

301/302重定向，对浏览器的记忆有区别

页面重定向，像servlet页面重定向一样，response返回。return;

#break/last 内部重定向，换path值

break标记，停止执行后续命令。for（）{break}

last标记，会引发location重新匹配。for（）{continue;}

ps ： 不要轻易使用last标签

当有flag值的时候，rewrite层面的命令会中断。last会引发location重匹配

当没有flag值的时候，rewrite还会往下走，最后一个rewrite覆盖前面的。再引发location重匹配

## 负载均衡配置

1、轮询（默认）

upstream nginx {

server 172.17.0.4:8081;

server 172.17.0.5:8081;

}

每个请求按时间顺序逐一分配到不同的后端服务器，如果后端服务器down掉，能自动剔除。

2、weight

指定轮询几率，weight和访问比率成正比，用于后端服务器性能不均的情况。down 暂时不参与负载

例如：

upstream nginx {

server 172.17.0.4:8081 weight=2;

server 172.17.0.5:8081 weight=1;

}

2、ip\_hash

每个请求按访问ip的hash结果分配，这样同一客户端的请求总是发往同一个后端服务器，可以解决session的问题。

例如：

upstream nginx {

ip\_hash;

server 172.17.0.4:8081;

server 172.17.0.5:8081;

}

## If语句

Nginx的内置变量:

$host：请求中的主机头(Host)字段，如果请求中的主机头不可用或者空，则为处理请求的server名称

$http\_HEADER ： HTTP请求头中的内容，HEADER为HTTP请求中的内容转为小写，

-变为\_(破折号变为下划线)，例如：$http\_user\_agent(Uaer-Agent的值)

$remote\_addr 客户端的IP地址。

$remote\_port 客户端的端口。

$request\_method 这个变量是客户端请求的动作，通常为GET或POST。

$request\_uri 这个变量等于包含一些客户端请求参数的原始URI

$scheme 所用的协议，比如http或者是https

$server\_name 服务器名称。

$server\_port 请求到达服务器的端口号。

$server\_protocol 请求使用的协议，通常是HTTP/1.0或HTTP/1.1。

$uri 请求中的当前URI(不带请求参数，参数位于$args)

|  |  |
| --- | --- |
| = ,!= | 比较的一个变量和字符串。 |
| ~， ~\* | 与正则表达式匹配的变量，如果这个正则表达式中包含 |
| -f，!-f | 检查一个文件是否存在。 |
| -d, !-d | 检查一个目录是否存在。 |
| -e，!-e | 检查一个文件、目录、符号链接是否存在。 |
| -x， !-x | 检查一个文件是否可执行。 |

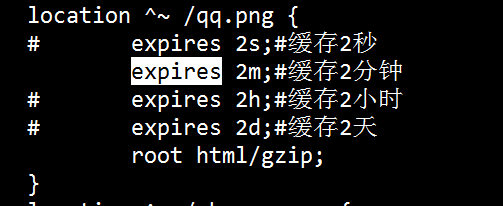
静态资源： location ~ /rex/.\*\.(htm|js|css)$

域名校验：if ( $http\_origin ~ [http://(.\*).enjoy.com)](http://(.*).enjoy.com)/)

浏览器校验：if ($http\_user\_agent ~ Firefox)

## 缓存

expires命令：过期时间



## 压缩

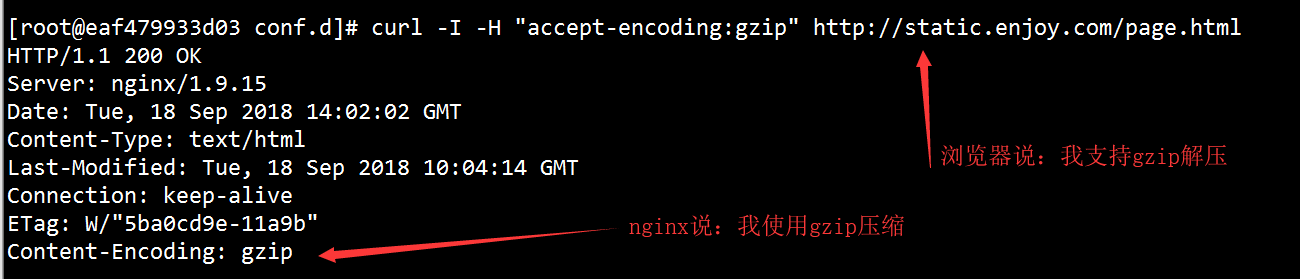
带宽资源很贵

--- /html/js/css压缩，/images不压缩

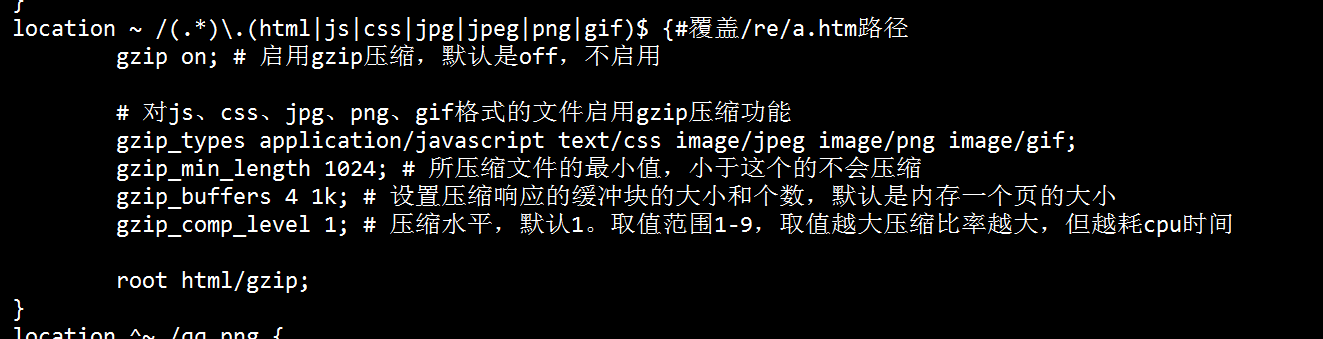
过程：nginx压缩 ----》网络传输 ---》chrome解压（压缩和解压消耗cpu）

1、浏览器携带支持的解压方式

2、浏览器与nginx的交互



nginx配置



gzip on;

　　gzip\_min\_length 1k;

　　gzip\_buffers 4 32k;

　　gzip\_http\_version 1.1;

　　gzip\_comp\_level 5;

　　gzip\_types text/plain application/x-javascript text/css application/xml;

　　gzip\_vary on;

## 隐藏服务器信息

在http模块下加入：

server\_tokens off; #隐藏服务器信息

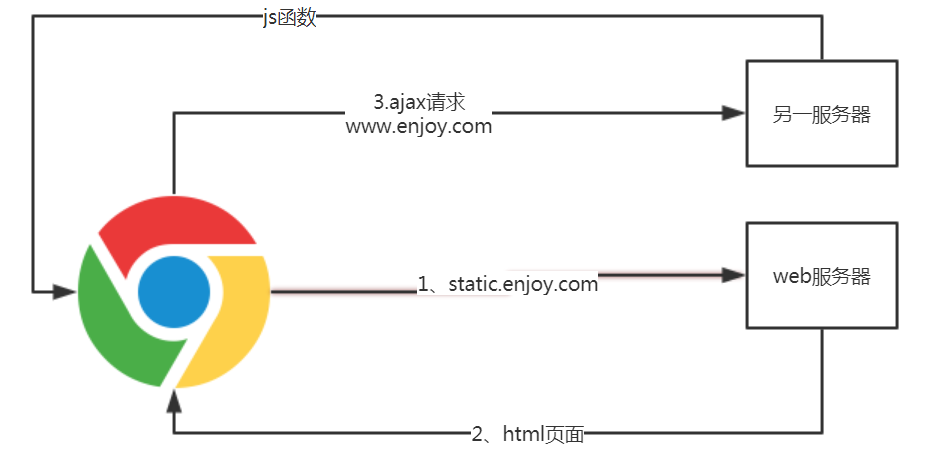
/application/nginx/sbin/nginx -t

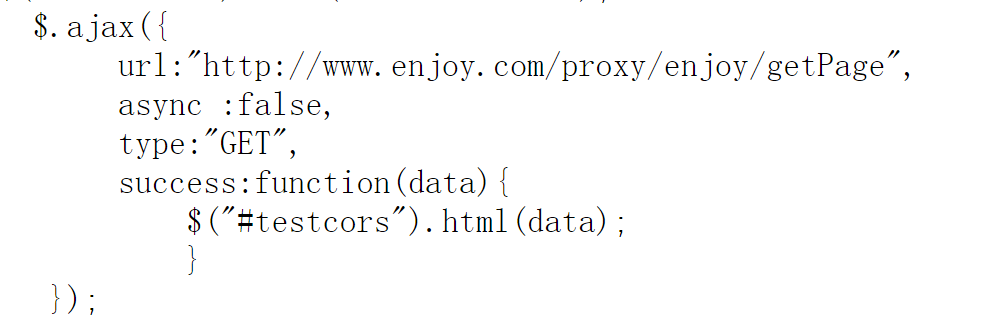
/application/nginx/sbin/nginx -s reload

# 实战线上Nginx多站点配置

# 跨域处理

问题由来：浏览器拒绝执行其它域名下的ajax运作





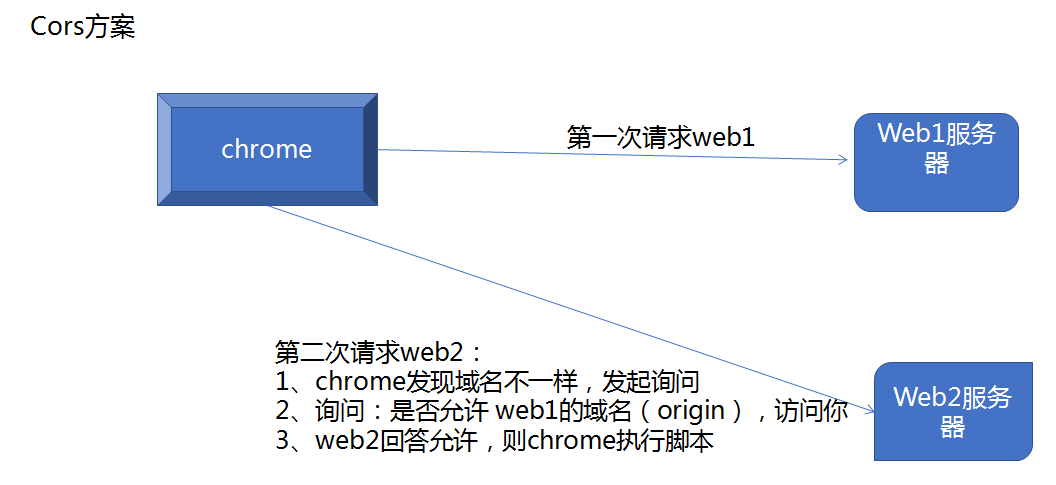
如上图：chrome首次使用域名static.enjoy.com加载html页面------->然后在页面内由ajax方式向域名www.enjoy.com发起请求。

此时问题出现：chrome拒绝执行ajax请求得到的返回值。

此问题常见解决方案：

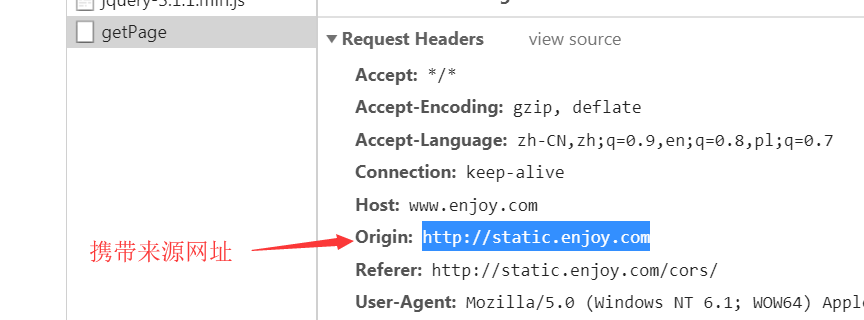
1、最常用的是，jsonp。此方案需要前后端共同协作来解决。

2、cors跨域，此方式非常优雅，是w3c组织制定的解决方案。为目前主流方案。方案流程如下图：



案例：

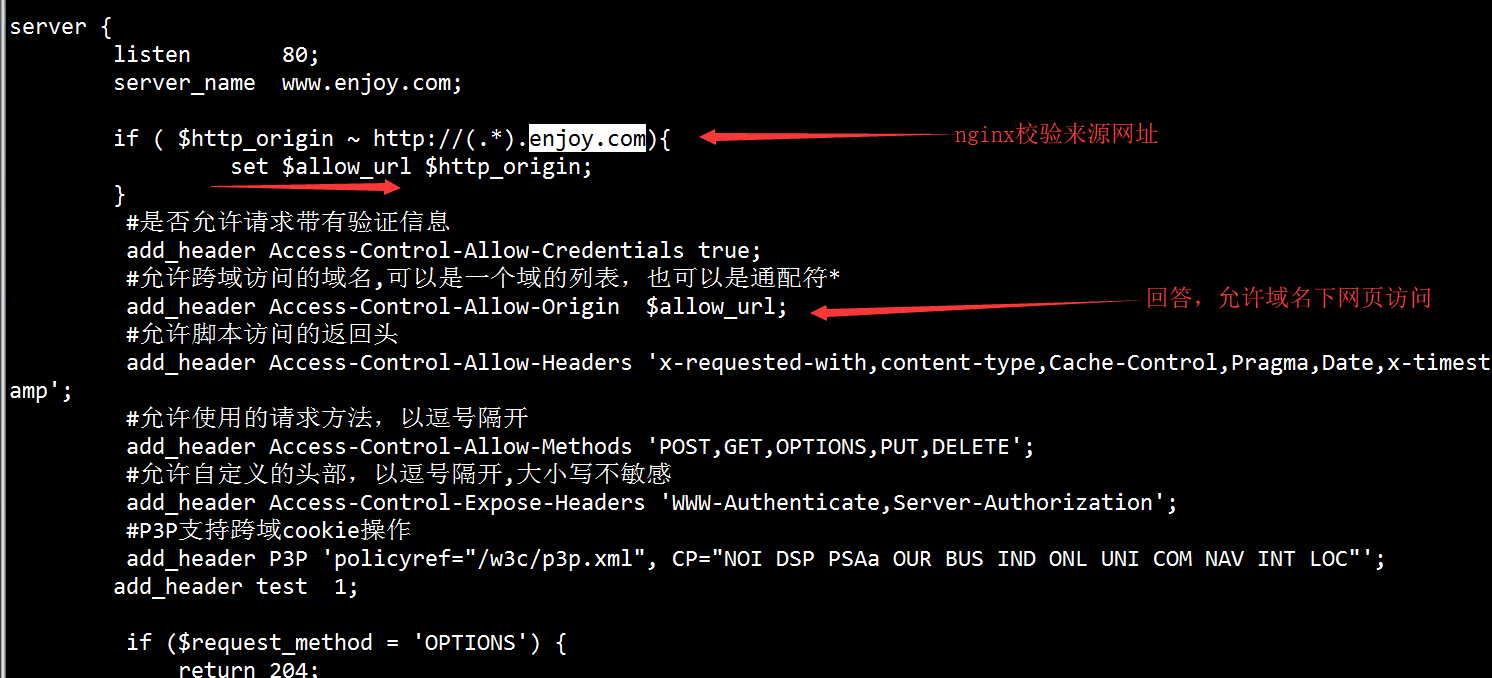
a、当chrome发现ajax请求的网址，与当前主域名不一致（跨域）时，会在请求header中追加值页面主域名值，即：origin = http://static.enjoy.com



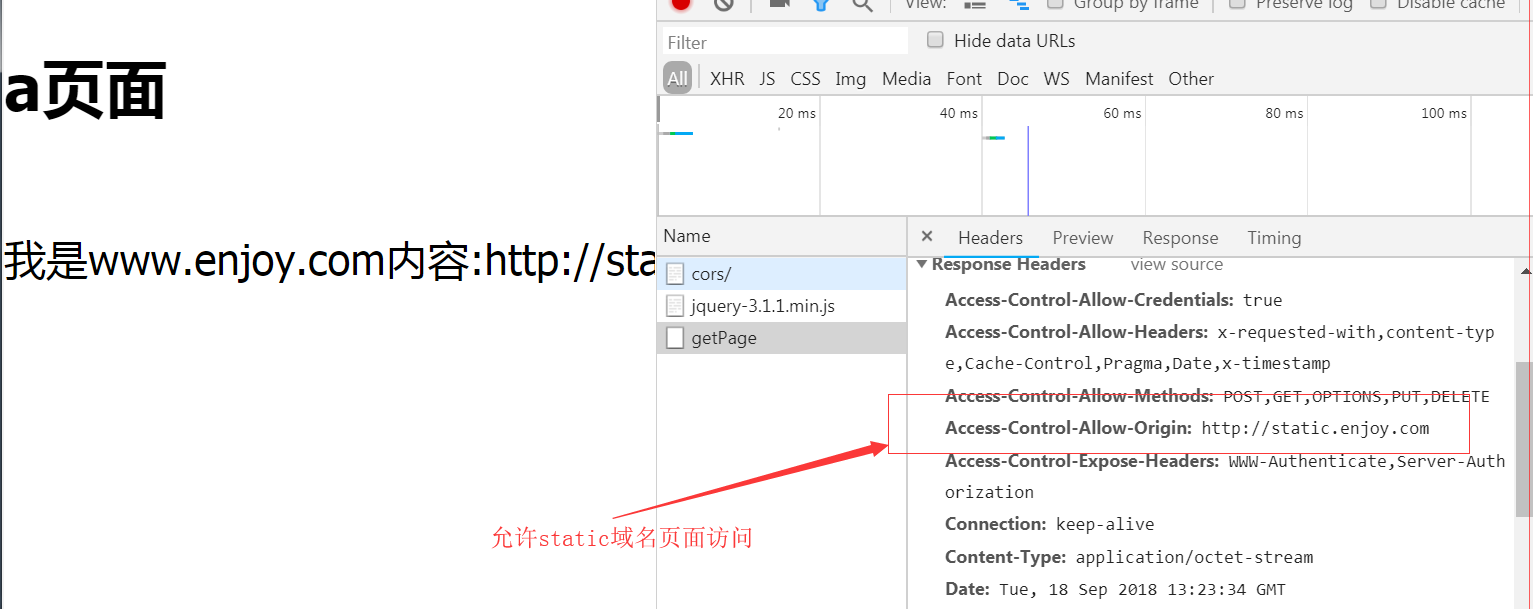
b、nginx在接收到ajax请求时，会查看origin值，即请求我的网址是谁？

此处使用正则来校验，即：只要是enjoy.com下的网址，都允许访问我

返回信息时，nginx追加header值：access-control-allow-origin = static.enjoy.com（回答浏览器，static域名网址可以访问我）



c、chrome收到ajax返回值后，查看返回的header中access-control-allow-origin的值，发现其中的值是static.enjoy.com,正是当前的页面主域名。这是允许访问，于是执行ajax返回值内容。（ps：若此处access-control-allow-origin不存在，或者值不是static域名，chrome就拒绝执行返回值）



# Nginx日志分析及脚本编写

**日志格式**

打开nginx.conf配置文件：vi /usr/local/nginx/conf/nginx.conf

日志部分内容：

#access\_log  logs/access.log  main;

日志生成的到Nginx根目录logs/access.log文件，默认使用“main”日志格式，也可以自定义格式。

**默认“main”日志格式：** 参数明细表：

|  |  |
| --- | --- |
| $remote\_addr | 客户端的ip地址(代理服务器，显示代理服务ip) |
| $remote\_user | 用于记录远程客户端的用户名称（一般为“-”） |
| $time\_local | 用于记录访问时间和时区 |
| $request | 用于记录请求的url以及请求方法 |
| $status | 响应状态码，例如：200成功、404页面找不到等。 |
| $body\_bytes\_sent | 给客户端发送的文件主体内容字节数 |
| $http\_user\_agent | 用户所使用的代理（一般为浏览器） |
| $http\_x\_forwarded\_for | 可以记录客户端IP，通过代理服务器来记录客户端的ip地址 |
| $http\_referer | 可以记录用户是从哪个链接访问过来的 |

查看日志命令tail -f /usr/local/nginx/logs/access.log

# Nginx日志切割案例

etc/init.d/rsyslog start #系统日志，如不开启，看不到定时任务日志

/etc/rc.d/init.d/crond start #定时任务开启

**编写sh：**

#!/bin/bash

#设置日志文件存放目录

LOG\_HOME="/usr/local/nginx/logs/"

#备分文件名称

LOG\_PATH\_BAK="$(date -d yesterday +%Y%m%d%H%M)"

#重命名日志文件

mv ${LOG\_HOME}/access.log ${LOG\_HOME}/access.${LOG\_PATH\_BAK}.log

mv ${LOG\_HOME}/error.log ${LOG\_HOME}/error.${LOG\_PATH\_BAK}.log

#向nginx主进程发信号重新打开日志

kill -USR1 `cat ${LOG\_HOME}/nginx.pid`

**配置定时任务 cron：**

\*/1 \* \* \* \* /usr/local/nginx/sbin/logcut.sh

**kill** 是向进程发送信号的命令。

**Nginx的信号**

1）、TERM、INT 快速关闭

2）、QUIT从容关闭

3）、HUP平滑重启，重新加载配置文件

4）、USR1 重新打开日志文件

5）、USR2 平滑升级可执行程序

KILL 9 强制终止，直接杀

# Nginx防盗链案例配置

目的：

1、让资源只能在我的页面内显示

2、不能单独来取或者下载

流程：

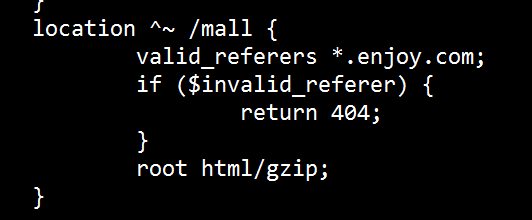
1、chrome以url1首次请求web服务器，得到html页面。

2、chrome再次发起url2资源请求，携带referers = url1。（注意，是url1，不是本次的url2）

3、nginx校验referers值，决定是否允许访问。

4、下面是nginx校验referers值的过程：

valid\_referers：匹配域名白名单，如果不匹配，把内置变量$invalid\_referers置为1，进入if块，返回404



#Preventing hot linking of images and other file types

location ~\* ^.+\.(jpg|png|swf|flv|rar|zip)$ {

valid\_referers none blocked \*.etiantian.org etiantian.org;

if ($invalid\_referer) {

rewrite ^/ http://bbs.etiantian.org/img/nolink.gif;

}

root html/www;

}

# Nginx日常运维及故障解决

# Nginx构建安全HTTPS架构实战

a、交互过程：

浏览器使用公钥，把数据加密后，再发送出去

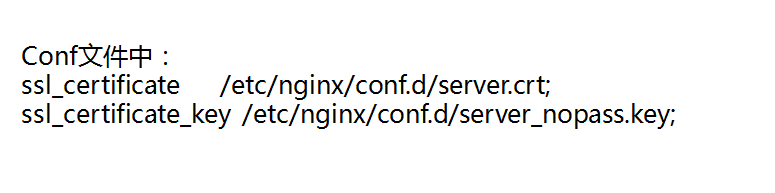
经过公钥加密内容，只有有私钥的人（nginx服务器）才看得懂。

b、nginx配置https的时候，需要两个东西：（此两样需购买）

一个key，私钥。放在nginx服务器里面，仅此一份

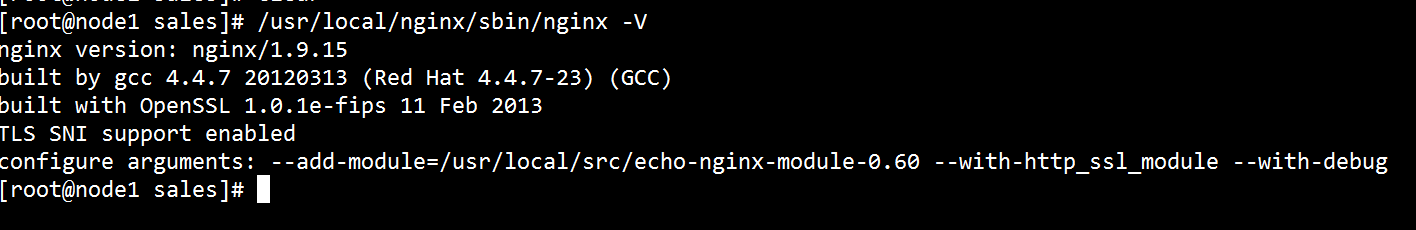
一个证书，公钥，供浏览器去下载。

配置方式如下图：



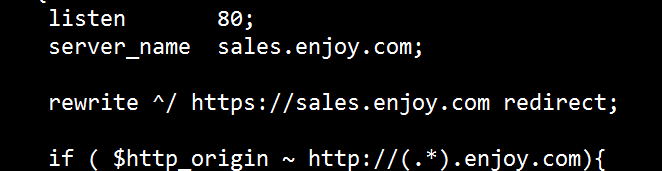
c、环境要求：

先查看nginx版本信息，命令：nginx -V



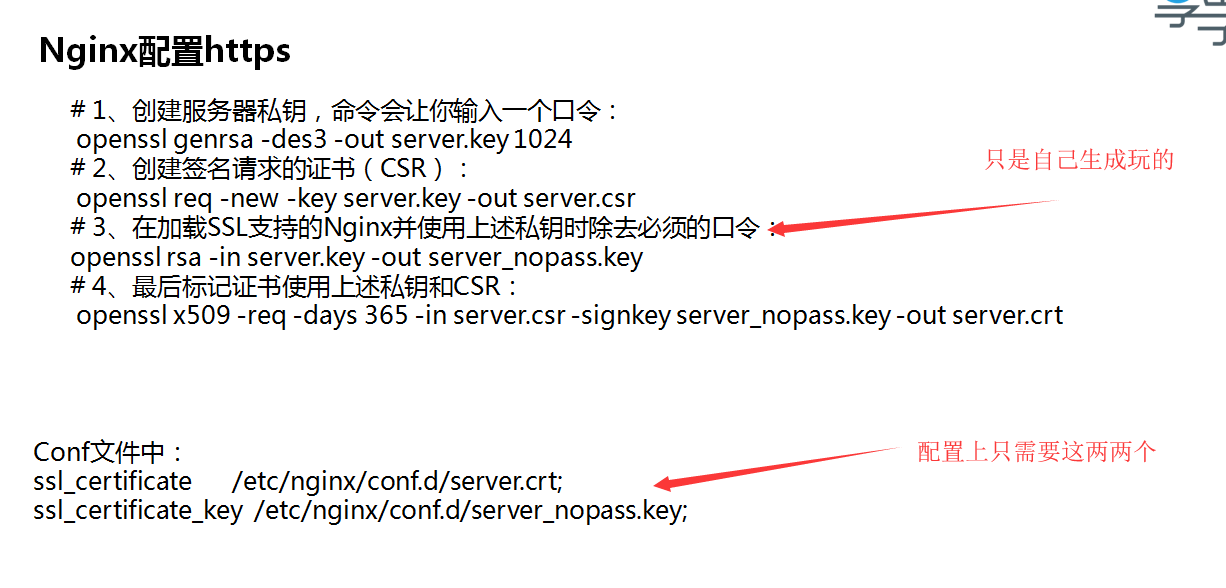
有--with-http\_ssl\_module,表示安装了https模块。否则需要重新安装。

d、配置好https方式之后，对http请求重定向



3、在https里，ajax请求普通http，会报错（原因：会降低安全）

4、我们自己也可以通过以下命令，生成证书和私钥



server

{

listen 443;

server\_name hongbao.tianya.cn;

ssl on;

ssl\_certificate /usr/local/nginx/ssl/server.pem;

ssl\_certificate\_key /usr/local/nginx/ssl/server.key;

ssl\_session\_timeout 5m;

ssl\_protocols TLSv1 TLSv1.1 TLSv1.2;

ssl\_ciphers AESGCM:ALL:!DH:!EXPORT:!RC4:+HIGH:!MEDIUM:!LOW:!aNULL:!eNULL;

ssl\_prefer\_server\_ciphers on;

location / {

root /www/app/hongbao.tianya.cn;

include /usr/local/nginx/conf/proxy.conf;

proxy\_pass http://localhost:8080;

proxy\_set\_header Protocol "https";

proxy\_set\_header Ty\_Remote\_Port $remote\_port;

}

}

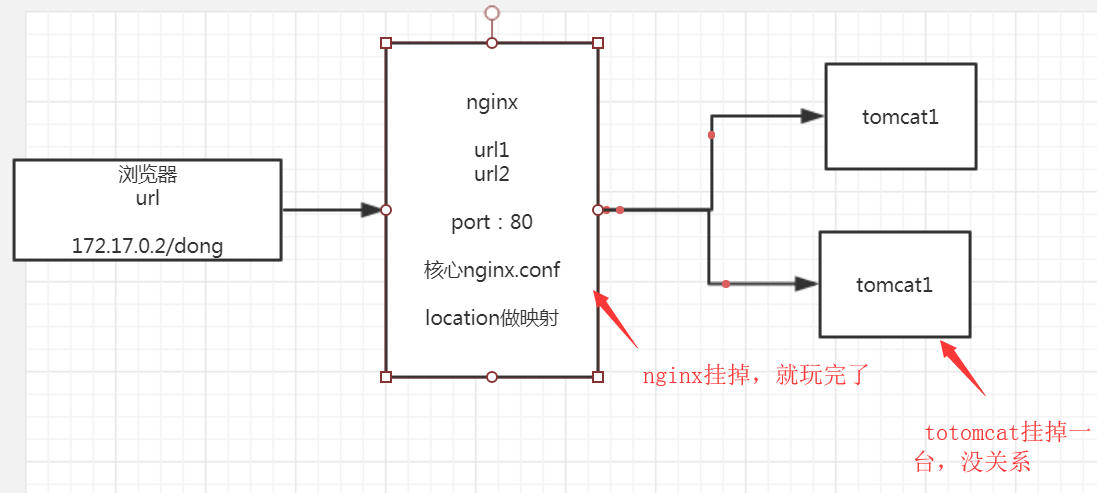
# 企业实战Nginx+tomcat动静分离架构

# Nginx+keepalived集群架构

# Nginx\_keepalived双主架构

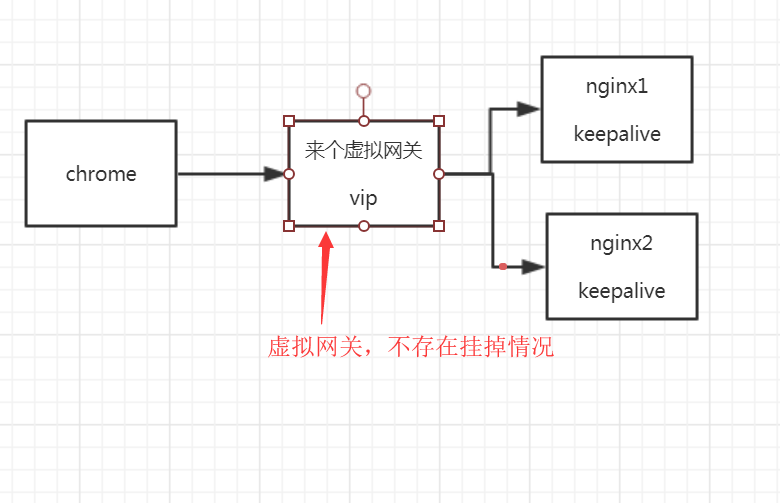
<https://blog.51cto.com/superpcm/2095395>

a、nginx解决tomcat高可用的思路，是前面加一层负载服务nginx。这种做法，总会有一个前端负载层存在宕机可能，是死循环



b、keepalived来解决。

lvs 思想来解决：



keepalived的思路，由 2台服务器软件虚拟出来一台 虚拟网关vip，

此vip由两台机器共同协商生成。当有一台机器宕机时，另一台机器一样能维持vip。这保证了，只要两台机器不同时宕机，vip就存在

c、keepalived的安装（不能用docker，docker中无法正常工作）

下载地址：https://pan.baidu.com/s/1G7sLL-YkZGSMu8G76yz1Rw 密码：adbw。

对应centos6系统

keepalived安装步骤：

A〉./configure --prefix=/data/program/keepalived --sysconf=/etc ##因为keepalive启动时候会默认读取/etc/keepalived/keepalived.conf

B〉make && make install

d、keepalived的配置：

打开/etc/keepalived/keepalived.conf，只需要配置如下一段。（其它是多余配置，删除）



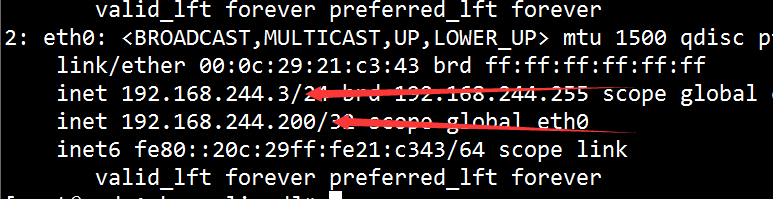
主备两个配置只有   priority **150和**state**MASTER不一致**

priority **100** state **BACKUP**

重启



e、ip addr命令查看网卡



eth0的网卡上，出现两个ip，244.3是真实ip，244.200是虚拟vip

f、重复c、d、e步骤，再装一台机器。主备两台机器的配置，只有两个配置差异：

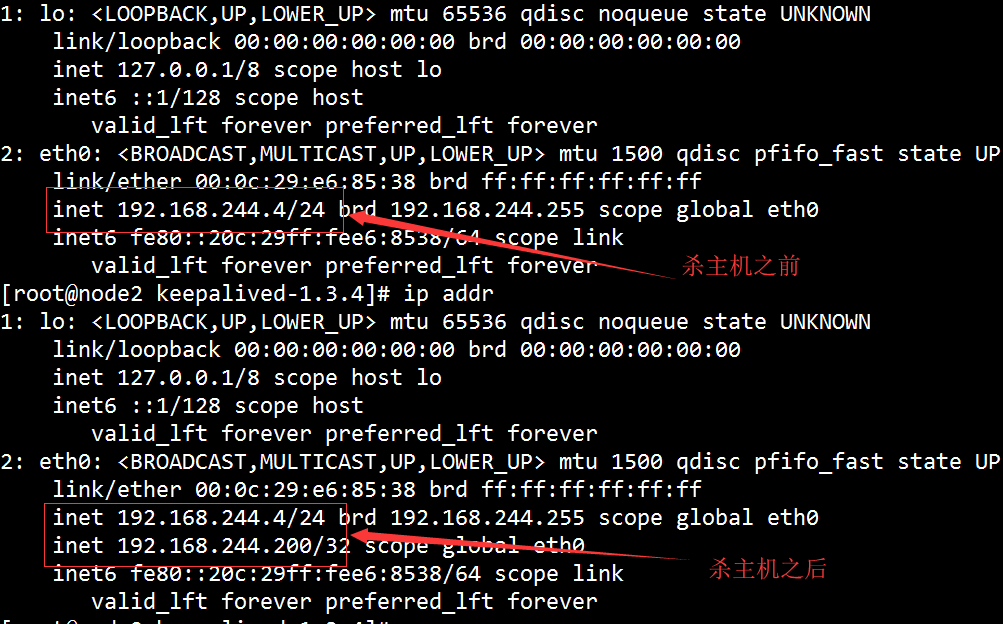
state BACKUP # 指定keepalived的角色，MASTER为主，BACKUP为备

priority 100 # 优先级，数值越大，处理请求的优先级越高。备用数字小些

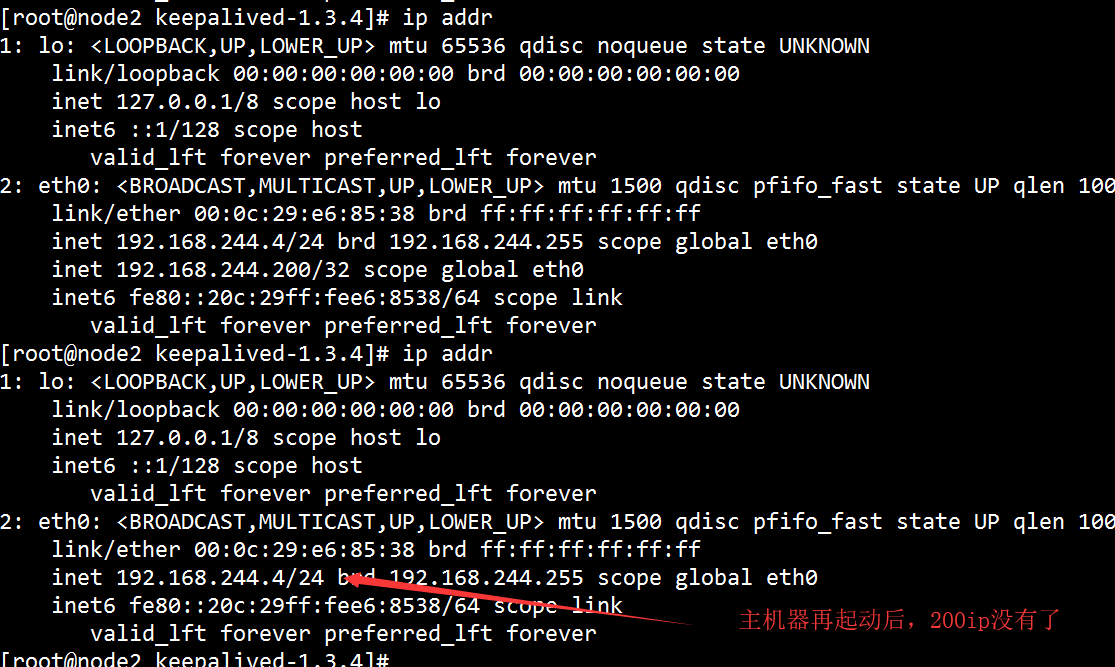
g、测试主备keepalived

正常情况下，vip是在主机器一端的，备用机器244.4上不会出现vip。如下图上半部分。

此时，如果我们杀掉主机器上keepalived，备用机器上会出现vip。如下图下半部分。



h、主机器keepalived再启动后，备机中vip = 200消失了，如下图



j、使用keepalived来监控nginx

编辑一个sh监控脚本，sh脚本：

#!/bin/bash

A=`ps -C nginx --no-header |wc -l`

if [ $A -eq 0 ];then

/usr/local/nginx/sbin/nginx #重启nginx

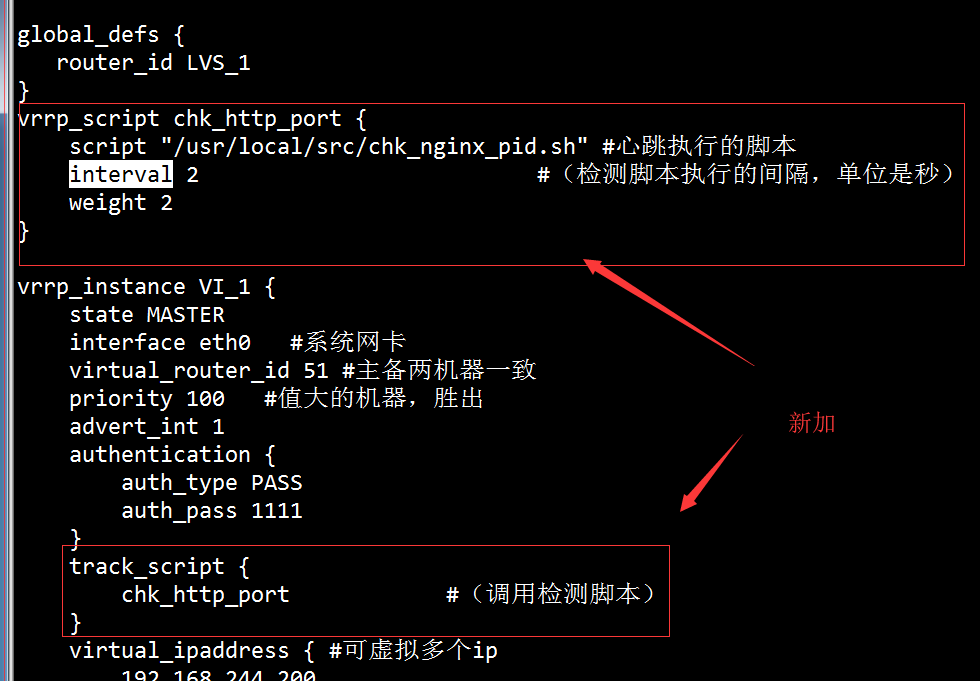
if [ `ps -C nginx --no-header |wc -l` -eq 0 ];then #nginx重启失败，则停掉keepalived服务，进行VIP转移

killall keepalived #杀掉，vip就漫游到另一台机器

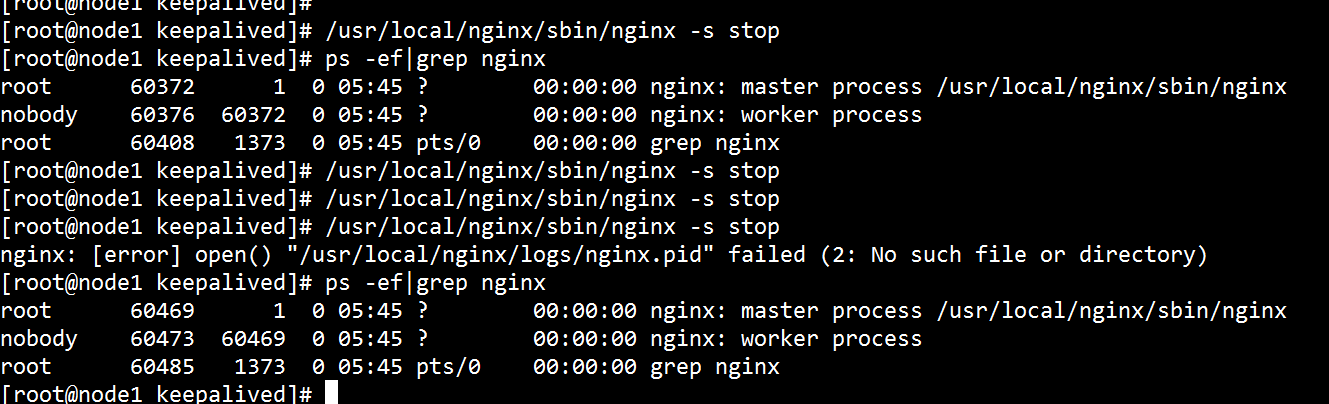
fi

fi

在配置文件中加入以下两处配置：



k、重启keepalived，测试监控效果，如下图操作：



nginx已变成不死鸟

**注意:**

1、如一台机器的keepalived死了，vip就漫游到活机器上去

2、https是在浏览器与nginx之间，nginx与tomcat之间的调用，还是用http

3、世界上没有nginx集群，因为没有必要。而且，keepalived在实际工作中，基本不用

4、有一个域名绑定多ip的东东，它已经实现这种高可用

5、keepalived也能与其它服务绑定做高可用，比如tomcat

6、死了还可以用，就是高可用。keepalived就想要达死了还能用

7、keepalived主备之间相互发送心跳检测，如果有一个死掉，另一个活的得不到回应了，此时启动vip到自己机器上。如果相互之间是通的，比较priority的优先级，值大的机器绑vip