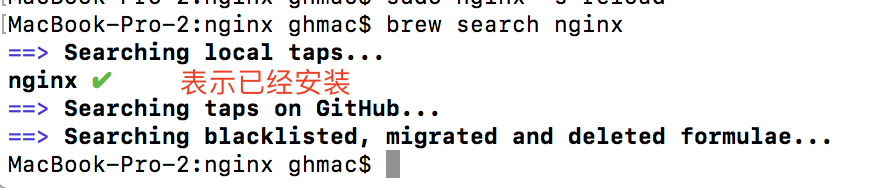
Nginx教程

1. **下载安装**

**Mac版：**

打开终端，输入brew search nginx



没有安装则输入brew install nginx

系统会自动安装，

安装目录：

/usr/local/Cellar/nginx

配置文件目录：

/usr/local/etc/nginx

成功后，在终端输入nginx –V，可以查看版本号



**Ubuntu版：**

安装Nginx前必须安装依赖，如果不安装依赖在我们执行make命令的时候会出现如下错误：

make: \*\*\* No rule to make target ` ‘, needed by xxx. Stop.

步骤1：安装依赖

以下所有的操作都在/home目录下进行

安装openssl，版本自己选

#下载

wget https://www.openssl.org/source/openssl-1.1.0e.tar.gz

#解压

tar zxvf openssl-1.1.0e.tar.gz

#进目录

cd openssl-1.1.0e/

#执行配置

./config

#安装

make & make install

安装zlib

wget https://downloads.sourceforge.net/project/libpng/zlib/1.2.11/zlib-1.2.11.tar.gz

tar zxvf zlib-1.2.11.tar.gz

cd zlib-1.2.11/

./configure

make & make install

安装pcre

wget https://ftp.pcre.org/pub/pcre/pcre-8.40.tar.gz

tar zxvf pcre-8.40.tar.gz

cd pcre-8.40/

./configure

make & make install

执行完上述操作，依赖就安装好了

步骤2：安装Nginx

下载nginx，在http://nginx.org/en/download.html上可以获取

cd /home

wget http://nginx.org/download/nginx-1.11.10.tar.gz

tar zxvf nginx-1.11.10.tar.gz

进入了nginx包的安装目录

然后创建一个nginx的安装目录，本人创建在/home下

cd /home

mkdir nginx

再返回安装包目录,并执行配置命令

cd /home/nginx-1.11.10/

#执行下面命令进行配置（注意观察下面的目录，就是刚才的依赖包目录和创建的nginx待安装目录）

./configure \

--sbin-path=/home/nginx/nginx \

--conf-path=/home/nginx/nginx.conf \

--pid-path=/home/nginx/nginx.pid \

--with-http\_ssl\_module \

--with-pcre=/home/pcre-8.40 \

--with-zlib=/home/zlib-1.2.11 \

--with-openssl=/home/openssl-1.1.0e

#安装

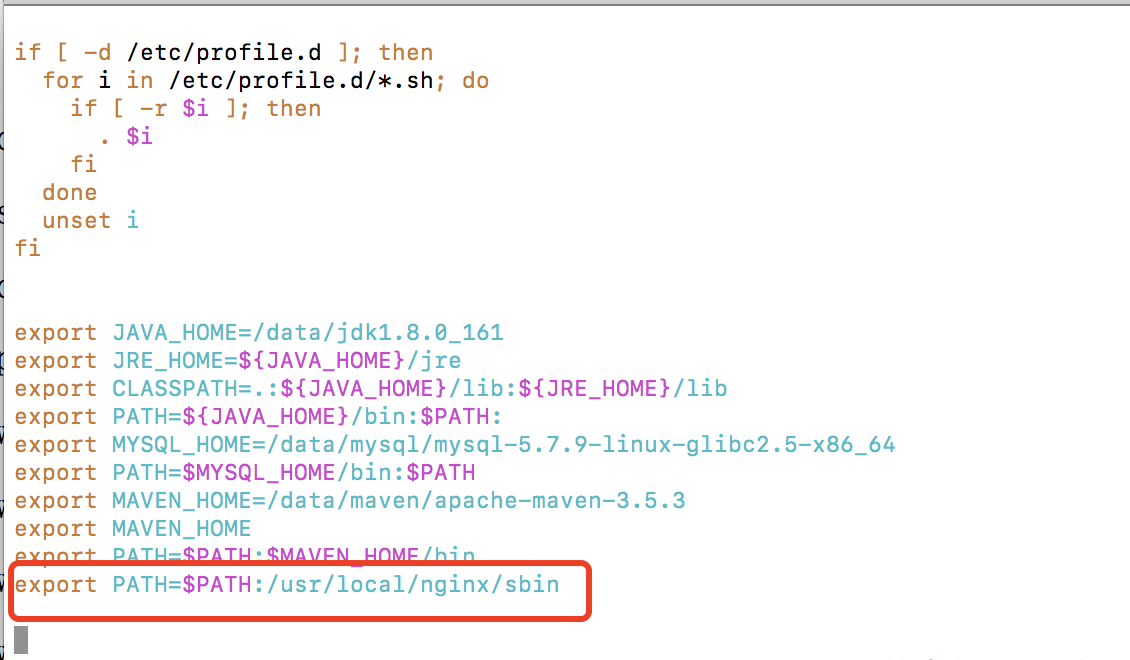
make $ make install

执行完毕就好了。

如果想在任意目录下执行Nginx命令，需要配置环境变量

vim /etc/profile

添加如下记录



或者在~/.bash\_profile文件添加如上记录也可以

还有一个需要注意的地方：如果我们安装好后，Nginx配置https后，运行报错：

nginx: [emerg] the "ssl" parameter requires ngx\_http\_ssl\_module in /usr/local/nginx/conf/nginx.conf:37

这是因为在安装Nginx的时候未开启ssl模块

解决方法: 原因也很简单，nginx缺少http\_ssl\_module模块，编译安装的时候带上--with-http\_ssl\_module配置就行了，但是现在的情况是我的nginx已经安装过了，怎么添加模块，其实也很简单，往下看： 做个说明：我的nginx的安装目录是/usr/local/nginx这个目录，我的源码包在/usr/local/src/nginx-1.6.2目录

切换到源码包：

cd /usr/local/src/nginx-1.11.3

查看nginx原有的模块

/usr/local/nginx/sbin/nginx -V

在configure arguments:后面显示的原有的configure参数如下：

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

那么我们的新配置信息就应该这样写：

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

运行上面的命令即可，等配置完，运行命令

make

这里不要进行make install，否则就是覆盖安装

然后备份原有已安装好的nginx(可以不备份)

cp /usr/local/nginx/sbin/nginx /usr/local/nginx/sbin/nginx.bak

然后将刚刚编译好的nginx覆盖掉原有的nginx（这个时候nginx要停止状态）

cp ./objs/nginx /usr/local/nginx/sbin/

然后启动nginx，仍可以通过命令查看是否已经加入成功

或者我们在安装的时候，configure的时候加上如下命令：

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

**centos版：**

安装Nginx时必须先安装相应的编译工具

yum –y install gcc gcc-c++ autoconf automake

yum –y install zlib zlib-devel openssl openssl-devel pcre-devel

zlib:nginx提供gzip模块，需要zlib支持

openssl：Nginx提供ssl功能

pcre:支持地址重写rewrite功能

1. **Nginx做负载均衡**

描述：Nginx做负载均衡，两个Tomcat服务器，一个Nginx，根据权重将客户请求分发到相应的Tomcat服务器上。

Nginx配置：打开Nginx配置文件nginx.conf，然后重新启动nginx,启动Tomcat服务器1和Tomcat服务器2进行测试



负载均衡策略：

轮询：这种是默认的策略，把每个请求按顺序逐一分配到不同的server，如果server挂掉，能自动剔除。

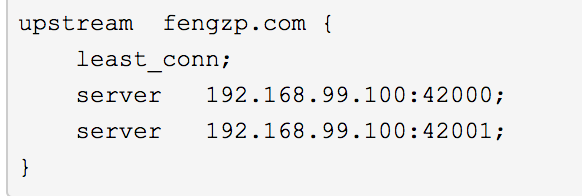
upstream www.zbc.com {

server 192.168.31.64:8082 weight=1;

server 192.168.31.64:8084 weight=2;

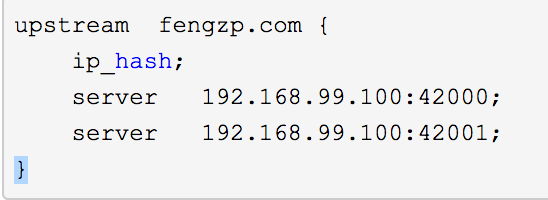
}

最少连接：把请求分配到连接数最少的server



权重：使用weight来指定server访问比率，weight默认是1。上面配置的server2访问的比例是server1的两倍。

ip\_hash：每个请求会按照访问ip的hash值分配，这样同一客户端连续的Web请求都会被分发到同一server进行处理，可以解决session的问题。如果server挂掉，能自动剔除。



ip\_hash可以和weight结合使用。

后续问题：

如果关了tomcat1，再多次刷新页面，接下来出现的就会都是tomcat2的页面，但是时而快时而慢。这其中原因是当如果nginx将请求转发到tomcat2时，服务器会马上跳转成功，但是如果是转到tomcat1，因为tomcat1已经关闭了，所以会出现一段等待响应过程的过程，要等它失败后才会转到tomcat2。

而这个等待响应的时间我们是可以配置的。

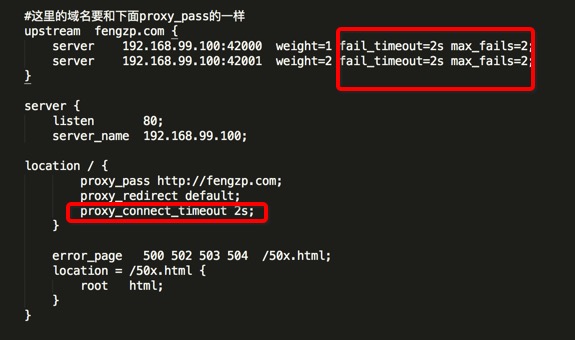
这个时间由以下3个参数控制：

proxy\_connect\_timeout：与服务器连接的超时时间，默认60s

fail\_timeout：当该时间内服务器没响应，则认为服务器失效，默认10s

max\_fails：允许连接失败次数，默认为1

等待时间 = proxy\_connect\_timeout + fail\_timeout \* max\_fails

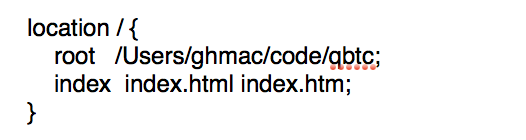


如果我这样配置的话，只需等待6秒就可以了。

1. **Nginx启动加载前端页面工程**

Mac版：

如下配置，前端页面地址，启动Nginx，就可以访问了。

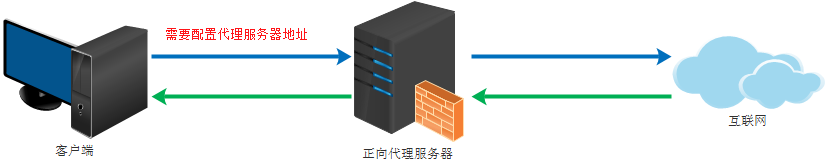


配置https



1. **Nginx反向代理，https**

**正向代理：**



正向代理,也就是传说中的代理,他的工作原理就像一个跳板,简单的说, 我是一个用户，我访问不了某网站，但是我能访问一个代理服务器，这个代理服务器呢,他能访问那个我不能访问的网站，于是我先连上代理服务器,告诉他我需要那个无法访问网站的内容，代理服务器去取回来,然后返回给我。从网站的角度，只在代理服务器来取内容的时候有一次记录，有时候并不知道是用户的请求，也隐藏了用户的资料，这取决于代理告不告诉网站。

正向代理 是一个位于客户端和原始服务器(origin server)之间的服务器，为了从原始服务器取得内容，客户端向代理发送一个请求并指定目标(原始服务器)，然后代理向原始服务器转交请求并将获得的内容返回给客户端。客户端必须要进行一些特别的设置才能使用正向代理。

**反向代理：**

客户端是无感知代理的存在的，反向代理对外都是透明的，访问者者并不知道自己访问的是一个代理。因为客户端不需要任何配置就可以访问。

反向代理（Reverse Proxy）实际运行方式是指以代理服务器来接受internet上的连接请求，然后将请求转发给内部网络上的服务器，并将从服务器上得到的结果返回给internet上请求连接的客户端，此时代理服务器对外就表现为一个服务器。

反向代理的作用：保证内网的安全，可以使用反向代理提供WAF功能，阻止web攻击，大型网站，通常将反向代理作为公网访问地址，Web服务器是内网。

场景描述：系统是前后端分离，前端页面放在一个服务器，后端的服务根据功能放在不同的服务器，后端是通过ajax请求数据。

实现:

安装一个Nginx，做前端页面容器，同时根据url匹配不同规则，分发请求到不同后端服务器。比如有web1服务器是做权限模块服务的，web2服务器是做交易模块的，配置Nginx通过不同的url请求规则来分发到web1或者web2。另外一个要求是，在Nginx配置HTTPS证书，这样通过Nginx分发的请求都是带https的，不用在web1和web2服务器上再配置证书。

流程：用户通过服务前端服务器地址，获取到前端页面，对前端页面进行操作，前端发送ajax请求到Nginx服务器，Nginx服务器根据配好的url匹配规则进行匹配，然后将请求分发到匹配上的具体服务器进行请求执行。

注：Nginx服务器的地址会配置在前端页面。

配置Nginx的配置文件nginx.conf如下

server {

listen 443;

server\_name zbc.com www.zbc.com \*.zbc.com; #域名

ssl on; #开启ssl认证

ssl\_certificate /Users/ghmac/Documents/zbc.com.crt; #证书地址

ssl\_certificate\_key /Users/ghmac/Documents/\_.zbc.com.key; #证书的key

ssl\_session\_timeout 5m; #超时时间

ssl\_ciphers ECDHE-RSA-AES128-GCM-SHA256:ECDHE:ECDH:AES:HIGH:!NULL:!aNULL:!MD5:!ADH:!RC4;

ssl\_protocols TLSv1 TLSv1.1 TLSv1.2;

ssl\_prefer\_server\_ciphers on;

# Load configuration files for the default server block.

#include /usr/local/etc/nginx/\*.conf;

location /{ #配置前端页面地址，访问<https://www.zbc.com>的时候会先展示前端页面

root /Users/ghmac/code/qbtc/dist;

index index.html index.htm;

#前端使用vue，配置如下时，进入vue子页面在刷新的时候才不会出现404.

try\_files $uri $uri/ /index.html;

}

#根据不同的ajax请求url规则，转发请求到具体的服务器，访问[https://www.zbc.com/web/\*](https://www.zbc.com/web/*)的ajax请求会转发到<http://192.168.31.76:8081>这个服务器。前端配置地址时必须也加上https

location /web {

proxy\_pass http://192.168.31.76:8081/web;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

//加上该配置，在后端服务器日志里面记录的是远端真实的主机，而不是Nginx代理服务器

proxy\_set\_header Host $http\_host;

proxy\_set\_header X-Forwarded-Proto https;

proxy\_redirect off;

proxy\_connect\_timeout 240;

proxy\_send\_timeout 240;

proxy\_read\_timeout 240;

}

#此服务器是websocket服务器，因此配置会与上面有稍许不同，要不然没法建立websocket连接，前端的配置也要改变，必须是wss://www.zbc.com:8081这样的。

location /websocket {

proxy\_pass http://192.168.31.64:8084/websocket;

proxy\_http\_version 1.1;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection "upgrade";

}

location /json {

proxy\_pass http://192.168.31.64:8084/json;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header Host $http\_host;

proxy\_set\_header X-Forwarded-Proto https;

proxy\_redirect off;

proxy\_connect\_timeout 240;

proxy\_send\_timeout 240;

proxy\_read\_timeout 240;

}

error\_page 404 /404.html;

location = /40x.html {

}

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

location = /50x.html {

}

}

1. **Nginx，vue刷新子页面404**

在Nginx配置文件nginx.conf上加一行配置

location / {

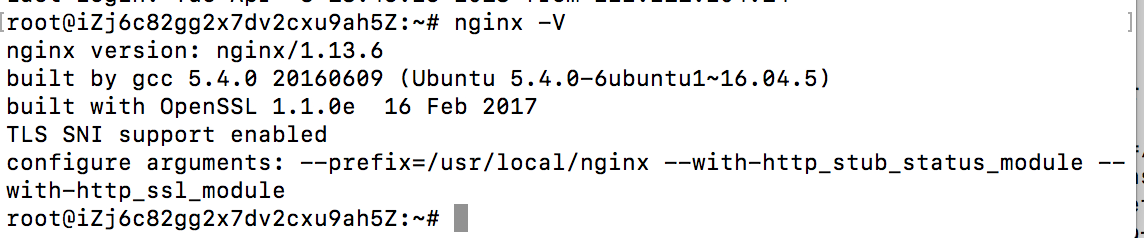
try\_files $uri $uri/ /index.html;

}

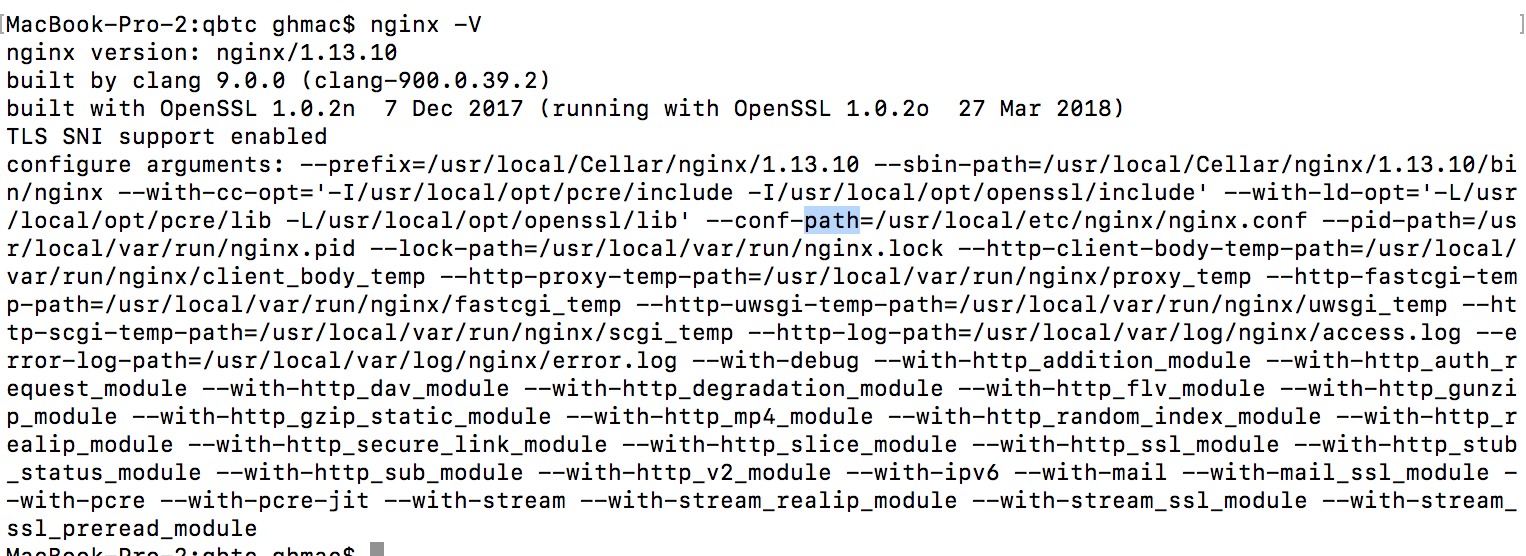
1. **Nginx参数详解**

查看Nginx参数情况，在命令行输入：nginx –V

ubuntu系统下：



Mac系统下：



可以看到mac系统下比Ubuntu系统下的参数多很多，其中每个代表的意思如下。

--prefix= 指向安装目录

--sbin-path 指向（执行）程序文件（nginx）

--conf-path= 指向配置文件（nginx.conf）

--error-log-path= 指向错误日志目录

--pid-path= 指向pid文件（nginx.pid）

--lock-path= 指向lock文件（nginx.lock）（安装文件锁定，防止安装文件被别人利用，或自己误操作。）

--user= 指定程序运行时的非特权用户

--group= 指定程序运行时的非特权用户组

--with-http\_stub\_status\_module 启用ngx\_http\_stub\_status\_module支持（获取nginx自上次启动以来的工作状态）

1. **Nginx常用命令**

nginx 启动 在mac下如果启动不了，没有权限，加上sudo nginx

nginx –s stop 停止

nginx –s reload 重启

nginx –t 检查Nginx配置是否ok

nginx –t –c /usr/local/etc/nginx/nginx.conf:-c指定配置文件路径



sudo kill –QUIT 1895(进程号) 平滑停止：nginx处理完手里未做完的事再停止

sudo kill –TERM 1895(进程号) 快速停止：立即停止服务正在处理的所有网络请求

kill -9 (进程id)

1. **配置虚拟主机**

准备站点

我们站点统一放到/data/site下，每个站点根目录名称都和域名相同,具体如下。

新建a.ttlsa.com的站点根目录

# mkdir -p /data/site/a.ttlsa.com

# mkdir -p /data/site/a.ttlsa.com

新建a站的首页index.html

# cat /data/site/a.ttlsa.com/index.html

this is a.ttlsa.com!

新建b.ttlsa.com站点根目录

# mkdir -p /data/site/b.ttlsa.com

# mkdir -p /data/site/b.ttlsa.com

新建b站首页index.html,内容如this is b.ttlsa.com！

# cat /data/site/b.ttlsa.com/index.html

this is b.ttlsa.com!

新建日志文件目录

# mkdir -p /data/logs/nginx

我们统一讲日志存放到/data/logs下,这边是存放nginx日志,所以nginx日志保持在当前的nginx目录下.日志统一存放相对来说比较规范（如果你不习惯,你可以按自己的方式来做）

配置nginx虚拟主机

增加nginx主配置文件nginx.conf

先配置nginx日志格式，在nginx.conf找到如下内容，并且将#注释标志去掉

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

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

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

配置nginx主配置文件

# vim /usr/local/nginx-1.5.1/conf/nginx.conf

server{

server\_name a.ttlsa.com;

listen 80;

root /data/site/a.ttlsa.com;

access\_log /data/logs/nginx/a.ttlsa.com-access.log main;

location /

{

}

}

server{

server\_name b.ttlsa.com;

listen 80;

root /data/site/b.ttlsa.com;

access\_log /data/logs/nginx/b.ttlsa.com-access.log main;

location /

{

}

}

配置讲解:

server{}：配置虚拟主机必须有这个段。

server\_name：虚拟主机的域名，可以写多个域名，类似于别名，比如说你可以配置成

server\_name b.ttlsa.com c.ttlsa.com d.ttlsa.com，这样的话，访问任何一个域名，内容都是一样的

listen 80，监听ip和端口，这边仅仅只有端口，表示当前服务器所有ip的80端口，如果只想监听127.0.0.1的80，写法如下：

listen 127.0.0.1:80

root /data/site/b.ttlsa.com：站点根目录，你网站文件存放的地方。注：站点目录和域名尽量一样，养成一个好习惯

access\_log /data/logs/nginx/b.ttlsa.com-access.log main：访问日志

location /{} 默认uri,location具体内容后续讲解,大家关注一下.

重启并打开站点

nginx -t 检查nginx配置是否ok，命令如下：

# /usr/local/nginx-1.5.1/sbin/nginx -t

nginx: the configuration file /usr/local/nginx-1.5.1/conf/nginx.conf syntax is ok

nginx: configuration file /usr/local/nginx-1.5.1/conf/nginx.conf test is successful

如果看到以上两行ok和successful就表示配置问题，那接下来我们启动nginx

启动nginx

访问a.ttlsa.com、b.ttlsa.com（我这边DNS已经解析到了192.168.1.201，在测试的情况下，我们可以通过版本hosts即可），绑定host方法如下：

讲如下内容增加到C:\Windows\System32\Drivers\etc\hosts

192.168.1.201 a.ttlsa.com

192.168.1.201 b.ttlsa.com

以上是windows绑定hosts方式，如下是linux方式

echo "192.168.1.201 a.ttlsa.com

192.168.1.201 b.ttlsa.com" >> /etc/hosts

使用浏览器访问这两个站点。我这边使用curl来访问。

[root@ns conf]# curl http://a.ttlsa.com

this is a.ttlsa.com! //a站点内容

[root@ns conf]# curl http://b.ttlsa.com

this is b.ttlsa.com! //b站点内容

1. **nginx配置http自动跳转到https**

server {

listen 80;

server\_name \*.zbc.com;

rewrite ^(.\*)$ https://$host$1 permanent;

}

完成配置如图：



1. **基础介绍**

Nginx是一个高性能的Web和反向代理服务器,它具有很多非常优越的特性:

作为 Web 服务器：相比Apache，Nginx使用更少的资源，支持更多的并发连接，体现更高的效率，这点使 Nginx 尤其受到虚拟主机提供商的欢迎。能够支持高达 50,000 个并发连接数的响应，实际测试可以达到3万左右，按照这样计算，每天可以处理亿次访问量，感谢 Nginx 为我们选择了epoll and kqueue 作为开发模型.Apache使用的是select模型

作为负载均衡服务器：Nginx 既可以在内部直接支持 Rails和PHP，也可以支持作为HTTP代理服务器对外进行服务。Nginx用C编写,不论是系统资源开销还是CPU使用效率都比Perlbal要好的多。

作为邮件代理服务器: Nginx同时也是一个非常优秀的邮件代理服务器（最早开发这个产品的目的之一也是作为邮件代理服务器），Last.fm 描述了成功并且美妙的使用经验。

Nginx安装非常的简单，配置文件 非常简洁（还能够支持perl语法），Bugs非常少的服务器:Nginx启动特别容易，并且几乎可以做到7\*24不间断运行，即使运行数个月也不需要重新启动。你还能够在不间断服务的情况下进行软件版本的升级。

内存消耗小：Nginx+PHP服务器在3万并发连接下，开启10个Nginx进程消耗150M内存，开启64个PHP-cgi进程消耗128M内存，使用webbench做压力测试，在3万并发量下速度依然很快

成本低廉：相对于F5 BIG-IP、netscaler等硬件负载均衡交换机动辄10多万甚至几十万，可以免费使用，并且可以用于商业用途。

淘宝在Nginx的基础上开发了一个比Nginx功能更强大的叫Tengine

功能：

nginx还可以重写地址，比如我们网站之前的地址是[www.123.com](http://www.123.com)，我们新的网址已经变成了[www.345.com](http://www.345.com)，当用户在浏览器输入[www.123.com](http://www.123.com)时，Nginx接收到该地址可以直接跳转到[www.345.com](http://www.345.com)，这叫地址重写的功能。

控制客户端缓存以及缓存时间，比如看视频，刚开始网速很快，马上缓存一部分视频，然后用户先看着，后面就限制网速，慢慢加载剩余的部分

反向代理：

负载均衡：

阻塞模型：

非阻塞模型：

同步：

异步：

1. **nginx配置文件nginx.conf解析**

**main:**定义user、group、worker\_process、error\_log等

**events**：定义事件驱动相关的配置

**server:**定义虚拟主机

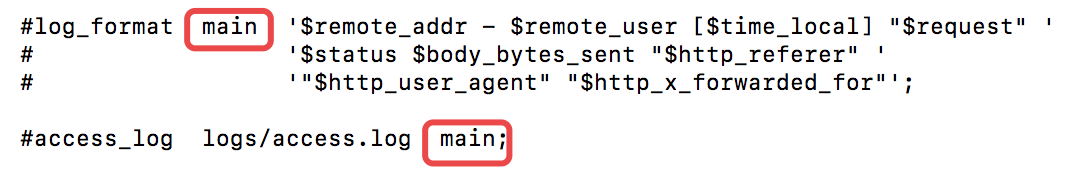
**location：**定义URI的访问属性，比如后端服务器地址等

**proxy\_pass:**定义反向代理

**worker\_processes**:nginx的worker线程的个数，一般情况下都是跟机器的cpu核心数相同。

一个http下面可以配置多个server，server也就是虚拟主机，一个server下面又可以配置多个location

**worker\_connections**：最大连接数也就是并发量，默认是1024，一般会配置的更大

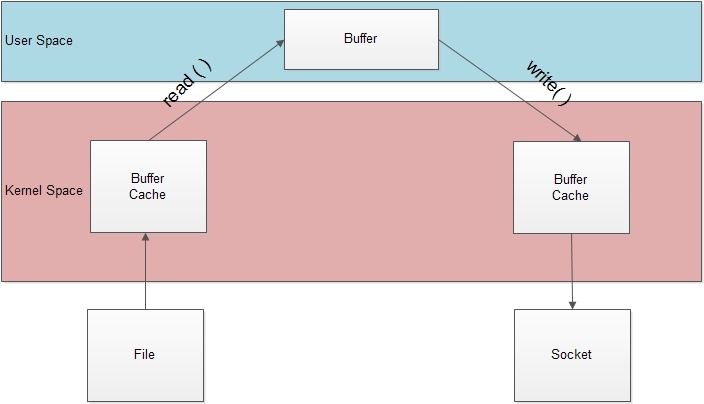


**log\_format**：定义日志的输出格式，main是格式的别名

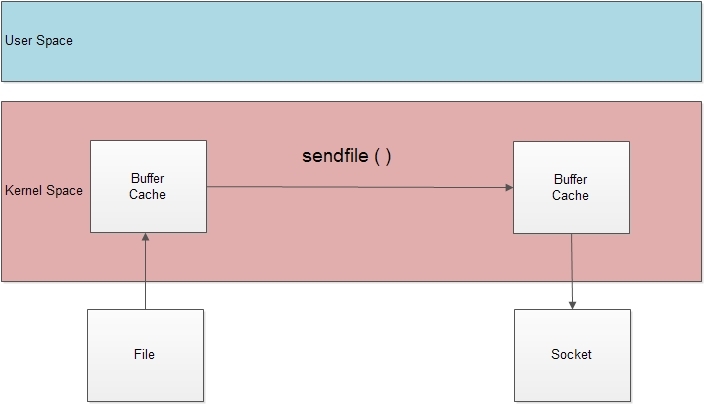
**access\_log**：日志的输出地址，其中main就是格式的别名（跟上面必须对应上）。

**keepalive\_timeout**：连接超时时间，默认是65s，超过会断掉当前的http连接。

**sendfile**: Linux kernel 2.2之前，（如图）读写数据基本都是使用read系统调用和write系调用，以nginx来说如果一个请求建立，从磁盘的文件到网络连接之间会通过硬件（DMA）---内核层---用户层多次读写系统来完成文件数据的复制传输：从内核层用read系统调用读到用户层，再从用户层用write系统调用写到内核层，每一次用户层到内核层的进行一次上下文转换，这种代价是非常昂贵的。甚至在没有数据变化时这种复制尤其显得多余。如果nginx接受大量并发请求，这种系统调用就会非常频繁，服务器的性能就会下降。



在Linux kernel2.2版本之后出现了一种叫做“零拷贝(zero-copy)”系统调用机制，目前很多应用服务器如apache、samba、nginx都支持sendfile。



如图所示，nginx在支持了sendfile系统调用后，避免了内核层与用户层的上下文切换（content swith）工作，大大减少了系统性能的开销。

**listen**:虚拟主机监听的端口号

listen 80 default\_server:定义默认的服务器

listen 192.168.0.1:80 :指定IP的端口

server\_name：虚拟主机名称

**location相关用法：**

**location**：配置地址目录

语法：location [=|~|~\*|^~] /uri/{...}

默认值：no

使用字段：server

这个参数根据uri的不同需求来进行配置，可以使用字符串与正则表达式匹配，如果要使用正则表达式，你必须指定下列前缀

~：区分大小写

~\*：不区分大小写

^~：禁止表达式匹配

=：精确匹配

//以.php结尾的请求转发给服务器127.0.0.1:80

location ~ \.php${

proxy\_pass <http://127.0.0.1>:80;

}



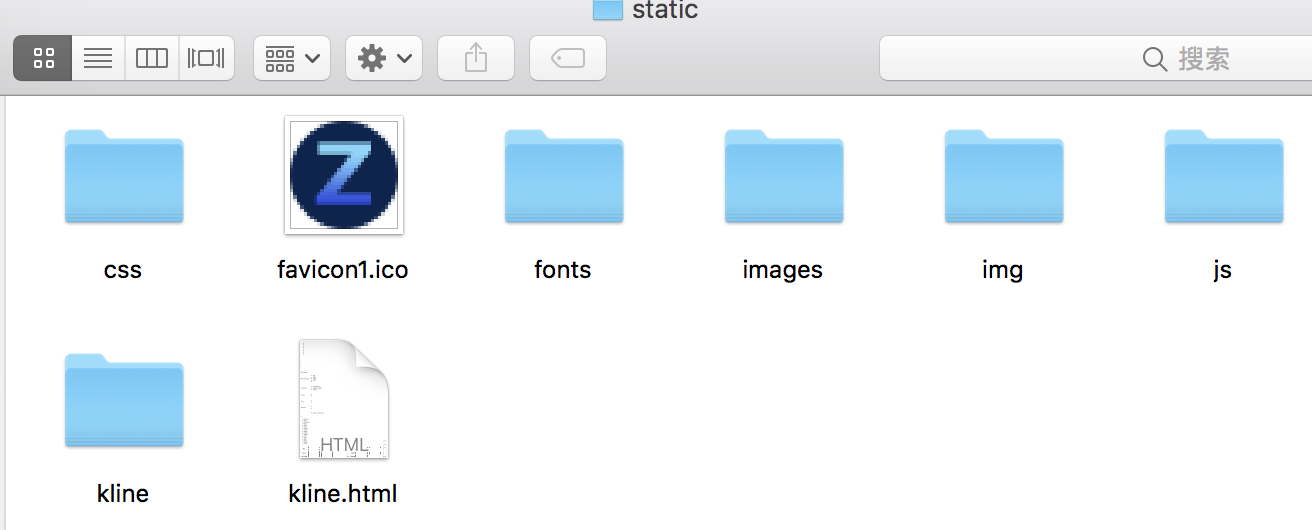
location>**root**:加载前端页面的路径，如果不指定路径，则默认去找/usr路径下的html文件夹

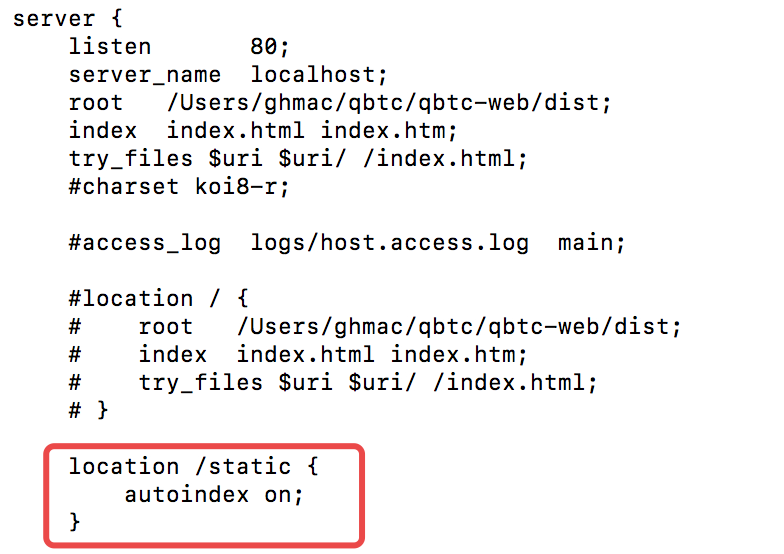
**索引和别名：**

location>**index**:索引文件

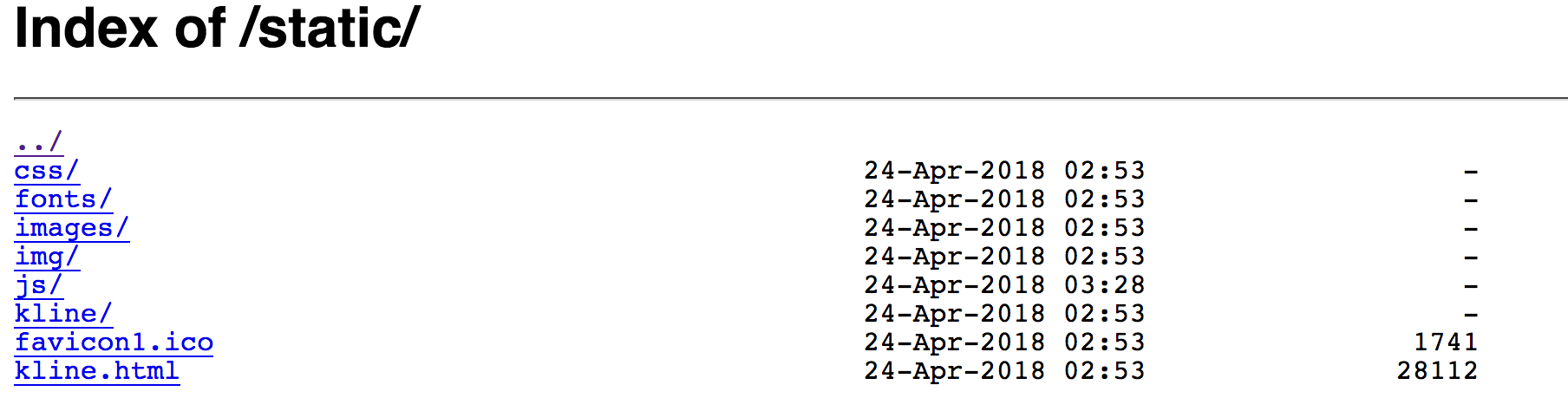
error\_page 404 /404.html:配置404出错页面

location>**autoindex**:列出该目录下的内容,值on表示开启



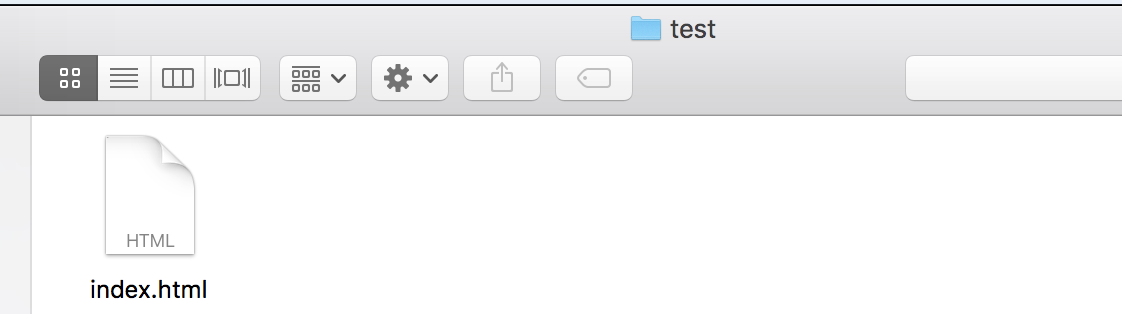


当我们在浏览器输入http://localhost/static/的时候，页面会展示/data目录下的所有文件



location>**alias**:别名

test文件夹下的index.html文件



nginx.conf配置文件的配置如下

location /abc {

alias /Users/ghmac/qbtc/qbtc-web/test/;

}



此时内容就是test文件夹下的index.html文件的内容。

**状态检查：**

location /nginx\_status {

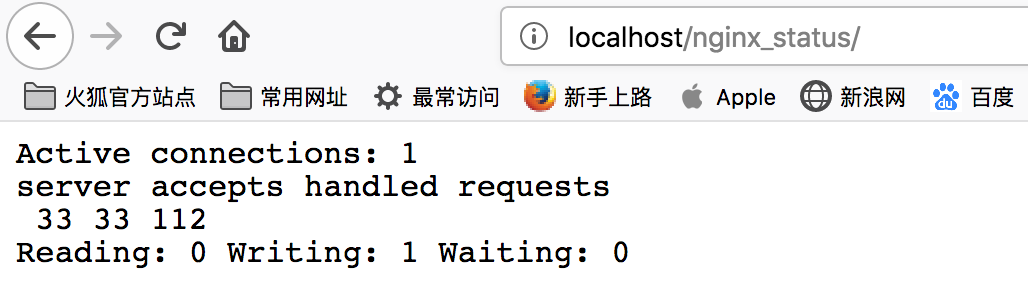
stub\_status on;

access\_log off;

}

nginx\_status这个名词可以随便取

配置完后，重新启动Nginx，地址栏输入如下地址即可看到Nginx的状态



注：一般这个地址都会加上密码权限访问，这种地址一般不会允许其他人访问，因此最好加上权限认证模块。

**系统自带参数说明：**

$remote\_addr, $http\_x\_forwarded\_for 记录客户端IP地址

$remote\_user 记录客户端用户名称

$request 记录请求的URL和HTTP协议

$status 记录请求状态

$body\_bytes\_sent 发送给客户端的字节数，不包括响应头的大小； 该变量与Apache模块mod\_log\_config里的“%B”参数兼容。

$bytes\_sent 发送给客户端的总字节数。

$connection 连接的序列号。

$connection\_requests 当前通过一个连接获得的请求数量。

$msec 日志写入时间。单位为秒，精度是毫秒。

$pipe 如果请求是通过HTTP流水线(pipelined)发送，pipe值为“p”，否则为“.”。

$http\_referer 记录从哪个页面链接访问过来的

$http\_user\_agent 记录客户端浏览器相关信息

$request\_length 请求的长度（包括请求行，请求头和请求正文）。

$request\_time 请求处理时间，单位为秒，精度毫秒； 从读入客户端的第一个字节开始，直到把最后一个字符发送给客户端后进行日志写入为止。

$time\_iso8601 ISO8601标准格式下的本地时间。

$time\_local 通用日志格式下的本地时间。

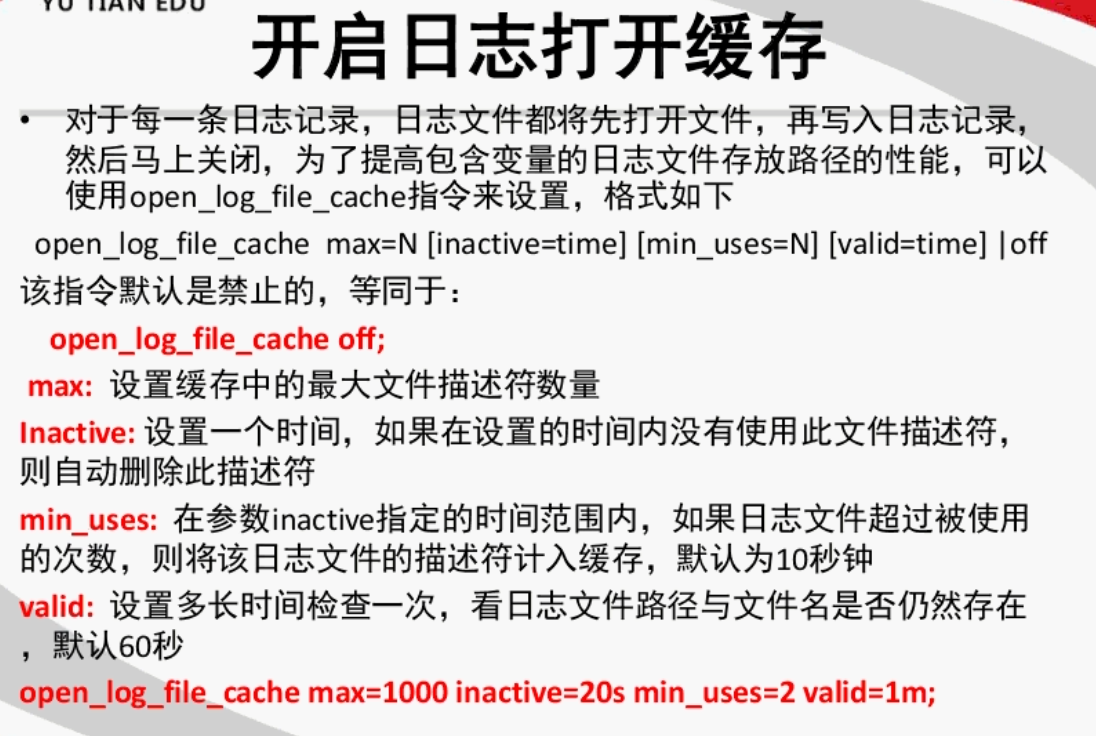
**日志管理：**

我们可以为每个server甚至每个location定义一个access.log

access\_log /data/log/$server\_name.log mylogformat buffer=32k

从0.7.4之后Nginx支持变量，一般建议不要使用变量，因为每次系统都要去计算一次，耗费资源

buffer=32k:表示设置内存缓冲区的大小





**压缩功能：**

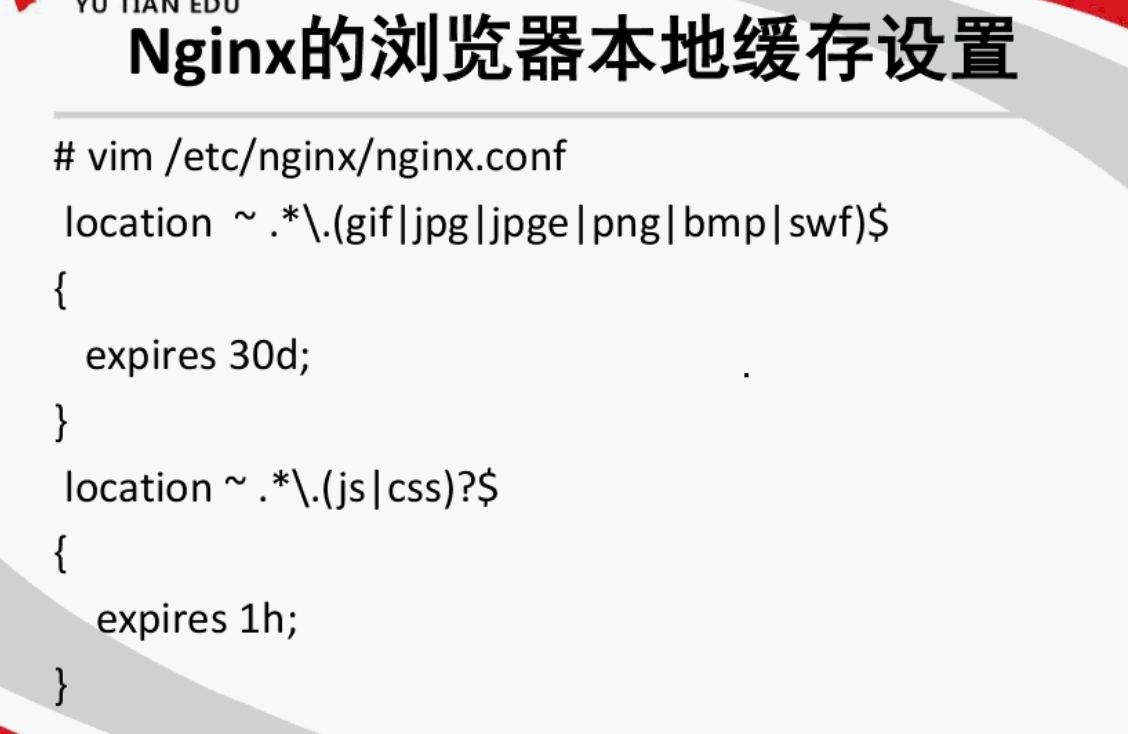


压缩开启后，如果配置不好，可能会降低缓存的命中率，一定要配置好

gzip\_proxied:可以配置哪些要压缩，哪些不压缩来解决缓存命中率问题。

**缓存设置：**

一般情况下静态页面我们会缓存的时间比较长，动态网页一般不缓存或者缓存的时间比较短。



**反向代理加入缓存：**

在http下面加入如下配置：

proxy\_cache\_path /Users/ghmac/nginx\_cache levels=1:2 keys\_zone=one:10m max\_size=1g;

然后在反向代理的location里面加入如下配置：

proxy\_cache one;

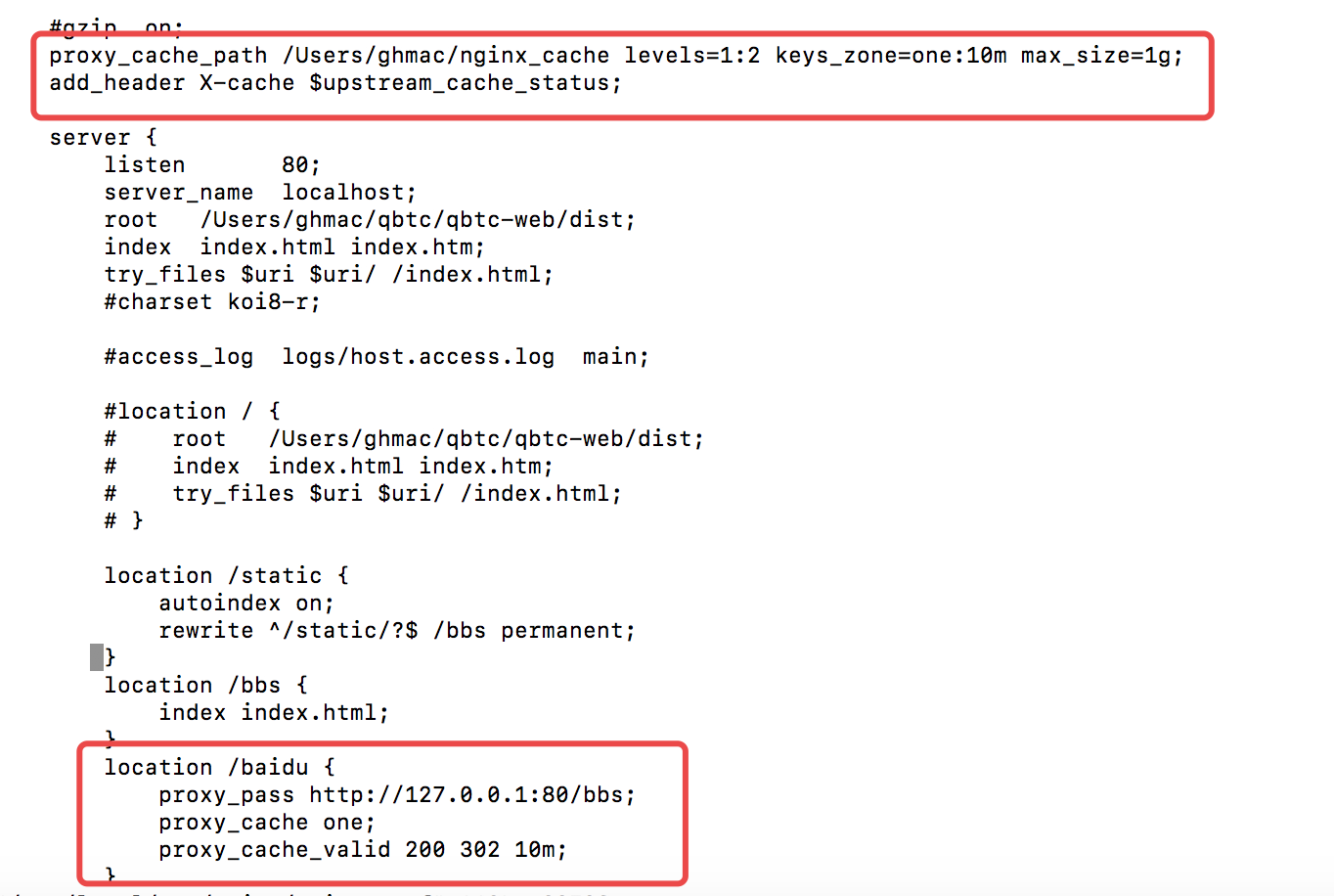
proxy\_cache\_valid 200 302 10m;

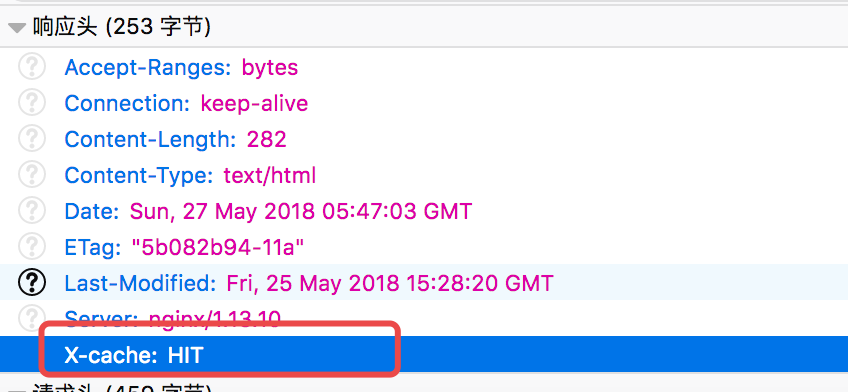
注意：此处的one就是proxy\_cache\_path里面的one，levels目录级别1：2表示两级目录，一级目录字符数1，二级目录字符数2；max\_size表示最大缓存容量，这里设置的为1g。

如何验证：我们可以在server里面加入配置

add\_header X-cache $upstream\_cache\_status;

这样在浏览器请求的时候，响应参数里面就会有X-cache字段，如果为Hit表示缓存命中。

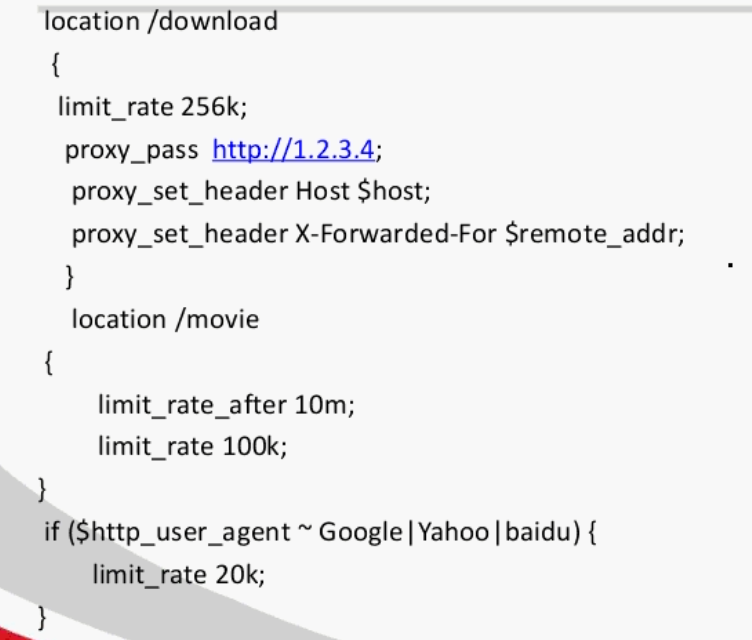






**设置限速：**

比如看视频或者下载东西一般都会限速。

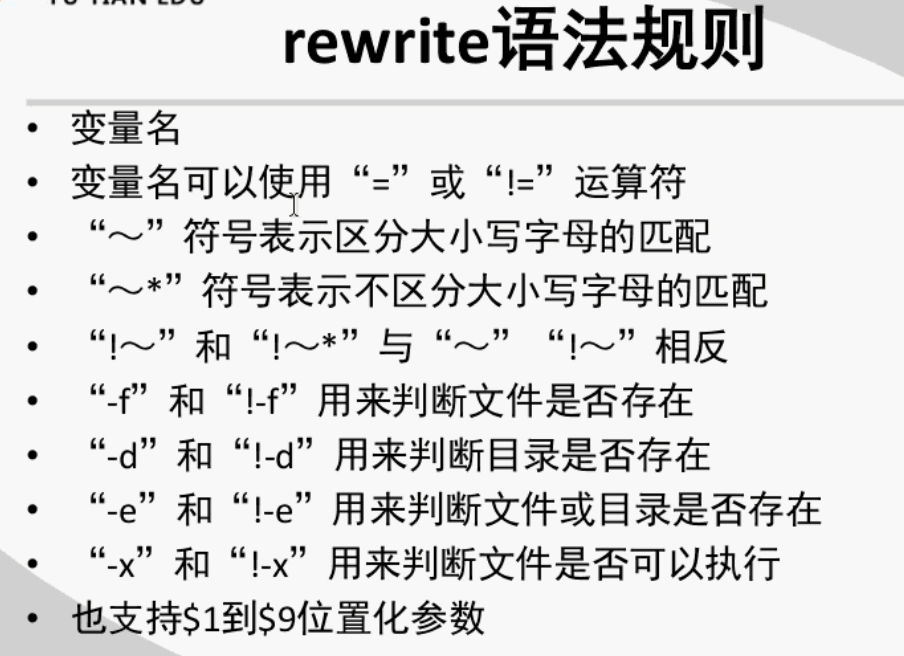


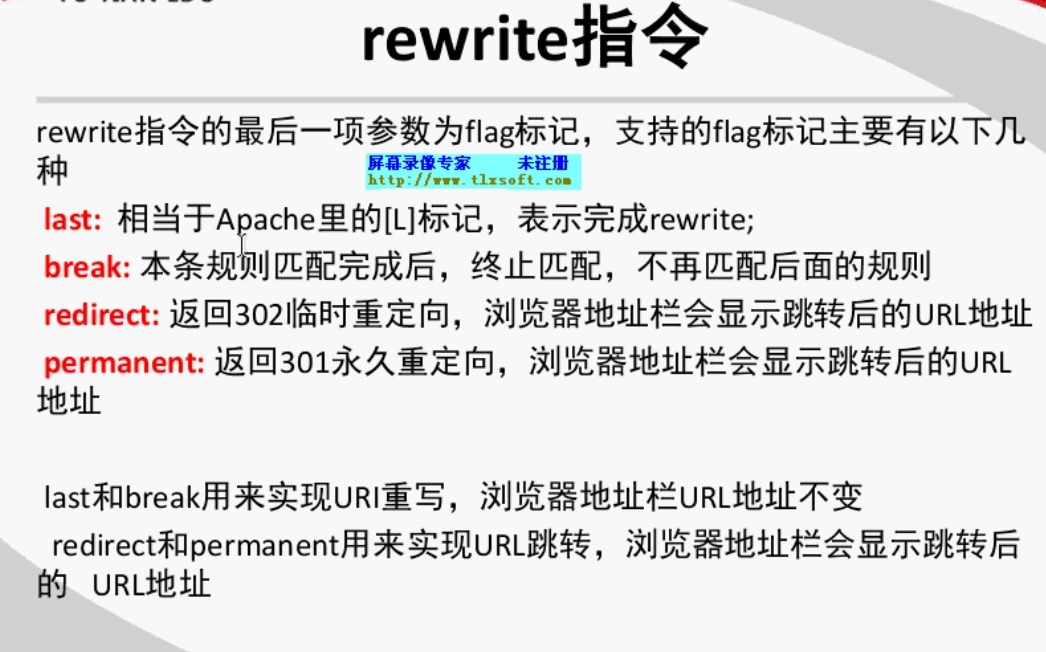
1. **地址重写**

支持url和uri的重写rewrite，Nginx的rewrite规则采用PCRE perl兼容正则表达式的语法进行规则匹配，如果需要Nginx的rewrite功能，在编译Nginx之前，需要编译安装PCRE库

url:是Uniform Resource Location的缩写，比如<http://www.zbc.com/user/login>

uri：由一个通过通用资源标识符Universal Resource Identifier进行定位。





if($http\_user\_agent ~ MSIE){

rewrite ^(.\*)$ /msie/$1 break;

}

$http\_user\_agent:浏览器的代理

~：忽略大小写

MSIE：浏览器的代理等于MSIE

^:开头

$:以什么结尾

(.\*)：任意字符

/msie/$1:如果你请求的是<http://localhost/index.html>，则会跳转到<http://localhost/msie/index.html,$1>也就是index.html

if(!-f$request\_filename){

rewrite ^/img/(.\*)$ /site/$host/images/$1 last;

}

如果你请求的$request\_filename文件不存在，则跳转到/site/$host/images目录下

return指令：

示例：如果访问的URL以.sh，.bash结尾，则返回状态码403

location ~ .\*\.(sh|bash)?${

return 403;

}

location里面可以编写if语句

location /{

proxy\_pass <http://172.16.100.6/>;

if($request\_method = “PUT”){

proxy\_pass http://172.16.100.7/;

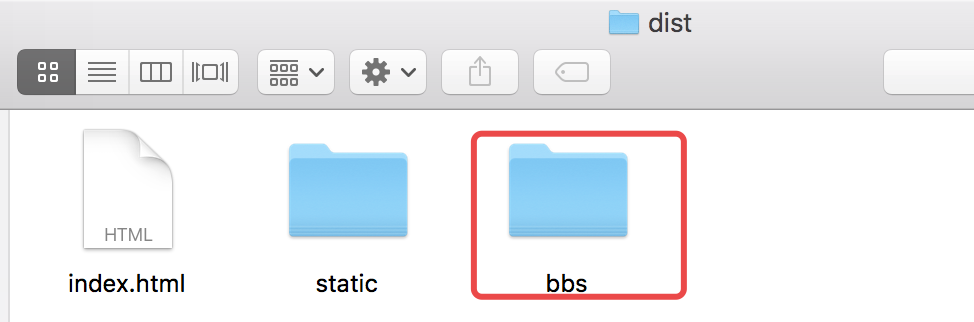
}

}

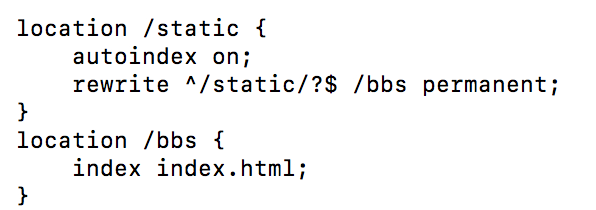
注：在location下面rewrite特别是last的时候，容易造成请求循环，此时最好设置为break而不是last。

**示例1**：

在根目录文件夹dist下面，新建bbs文件夹，在bbs下面新建index.html，index.html内容就是bbs。

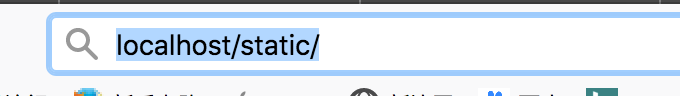


修改nginx.conf配置文件，曾经输入<http://localhost/static/>会展示static文件夹下的文件列表，现在要让这个地址跳转到<http://localhost/bbs/>



效果如下：在地址栏<http://localhost/static/>会自动变成<http://localhost/bbs/>

回车前：



回车后：



**防止盗链：**

盗链：内容不在自己服务器上，而通过技术手段，绕过别人放广告有利益的最终页，直接在自己有广告有利益的页面上向最终用户提供内容。常常是一些名不经传的小网站来盗取一些有实力的大网站的地址（比如一些音乐、图片、软件的下载地址）然后放置在自己的网站中，通过这种方法盗取大网站的空间和流量。

防盗链解决方法：

location ~\*\.(gif|jpg|png|swf|flv)${

valid\_referers none blocked [www.zbc.com](http://www.zbc.com) \*.zbc.com;

if($invalid\_referer){

rewrite ^/(.\*) http://www.zbc.com/block.html;

}

}

none：表示浏览器直接输入

blocked:

解析：如果请求的是gif、jpg、png、swf、flv文件，且referer不是我们自己的网站[www.zbc.com](http://www.zbc.com)过来的，则所有的这个请求都跳转到盗链的页面block.html.

**域名跳转：**

所有对[www.abc.com](http://www.abc.com)的访问，redirect到[www.test.com](http://www.test.com)

server{

listen 80;

server\_name [www.test.com](http://www.test.com);

index index.html index.php;

root /export/home/www;

if($host=”www.abc.com”){

rewrite ^/(.\*)$ <http://www.test.com/$1> permanent;

}

}

1. **aa**

aaa