**Nginx配置详解**

#nginx进程,一般设置为和cpu核数一样  
worker\_processes 4;      #或者auto

#错误日志存放目录   
error\_log  /data1/logs/error.log  crit;    
#运行用户/组，默认即是nginx，可不设置  
user nginx   nginx；      
#进程pid存放位置  
pid        /application/nginx/nginx.pid;#最大文件打开数（连接），可设置为系统优化后的ulimit -HSn的结果  
worker\_rlimit\_nofile 51200;

**Events事件**

use epoll;    #epoll是多路复用IO(I/O Multiplexing)中的一种方式,但是仅用于linux2.6以上内核,可以大大提高nginx的性能

worker\_connections 1024;  #;单个后台worker process进程的最大并发链接数

**Http设置**

Include mime.types;设置mime类型，类型由mime.type文件定义default\_type application/octet-stream; #默认文件类型

#limit模块，可防范一定量的DDOS攻击  
#用来存储session会话的状态，如下是为session分配一个名为one的10M的内存存储区，限制了每秒只接受一个ip的一次请求 1r/s  
  limit\_req\_zone $binary\_remote\_addr zone=one:10m rate=1r/s;  
  limit\_conn\_zone $binary\_remote\_addr zone=addr:10m;  
  include       mime.types;  
  default\_type  application/octet-stream;

#第三方模块lua防火墙  
    lua\_need\_request\_body on;  
    #lua\_shared\_dict limit 50m;  
    lua\_package\_path "/application/nginx/conf/waf/?.lua";  
    init\_by\_lua\_file "/application/nginx/conf/waf/init.lua";  
    access\_by\_lua\_file "/application/nginx/conf/waf/access.lua";  
  
 #设定请求缓存      
  server\_names\_hash\_bucket\_size 128;  
  client\_header\_buffer\_size 512k;  
  large\_client\_header\_buffers 4 512k;  
  client\_max\_body\_size 100m;

  **#隐藏响应header和错误通知中的版本号**  
  server\_tokens off;  
  #开启高效传输模式     
  sendfile on;#senfile指令指定nginx是否调用 sendfile函数来输出文件，对于普通的应用必须设为 on，如果用来进行下载等应用磁盘IO负载应用，可以设置为off，以平衡磁盘与网络I/O处理速度，降低系统的uptime

 #激活tcp\_nopush参数可以允许把httpresponse header和文件的开始放在一个文件里发布，  
  积极的作用是减少网络报文段的数量  
  tcp\_nopush     on;  
  #激活tcp\_nodelay，内核会等待将更多的字节组成一个数据包，从而提高I/O性能  
  tcp\_nodelay on;

 #FastCGI相关参数：为了改善网站性能：减少资源占用，提高访问速度

fastcgi\_connect\_timeout 300;  
fastcgi\_send\_timeout 300;  
fastcgi\_read\_timeout 300;  
fastcgi\_buffer\_size 64k;  
fastcgi\_buffers 4 64k;  
fastcgi\_busy\_buffers\_size 128k;  
fastcgi\_temp\_file\_write\_size 128k;

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

#连接超时时间，单位是秒  
  keepalive\_timeout 60;

Log\_format日志

普通模式：

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 /usr/local/nginx/access/access.log main buffer=32k;

#日志文件存放路径

Error\_log /usr/local/nginx/error/error.log error; #错误日志文件存放路径

|  |  |
| --- | --- |
| **字段** | **说明** |
| $remote\_addr | 客户端地址 |
| $remote\_user | 客户端用户名称 |
| $time\_local | 访问时间和时区 |
| $request | 请求的URI和HTTP协议 |
| $http\_host | 请求地址，即浏览器中你输入的地址（IP或域名） |
| $status | HTTP请求状态 |
| $upstream\_status | Upstream状态 |
| $body\_bytes\_sent | 发送给客户端的文件内容大小 |
| $http\_user\_agent | 用户终端浏览器等信息 |
| $ssl\_protocol | Ssl协议版本 |
| $ssl\_cipher | 交换数据中的算法 |
| $upstream\_addr | 后台upstrean的地址，即真正提供服务的主机地址 |
| $request\_time | 整个请求总时间 |
| $upstream\_response\_time | 请求过程中，upstream响应时间 |
| $http\_referer | Url跳转来源 |

**Server虚拟主机设置**

Listen 80； #监听端口

 server\_name  www.abc.com abc.com;    #虚拟主机名称

 index index.html index.htm index.php;    #首页排序，能够识别的文件类型

Root /var/www/html； #网站根目录

 error\_page 500 502 404 /templates/kumi/phpcms/404.html;  #错误页面

Charset utf-8；#编码识别

**SSL相关**

Listen 443 ssl；#监听端口443

ssl on; #开启ssl

ssl\_certificate /ls/app/nginx/conf/mgmtxiangqiankeys/server.crt; #服务的证书  
ssl\_certificate\_key /ls/app/nginx/conf/mgmtxiangqiankeys/server.key; #服务端key  
ssl\_client\_certificate /ls/app/nginx/conf/mgmtxiangqiankeys/ca.crt; #客户端证书  
ssl\_session\_timeout 5m; #session超时时间  
ssl\_verify\_client on; # 开户客户端证书验证   
ssl\_protocols SSLv2 SSLv3 TLSv1; #指定用于启动特定的加密协议，nginx1.1.13和1.0.12版本后默认是ssl\_protocols SSLv3 TLSv1 TLSv1.1 TLSv1.2 TLSv1与TLSv1.2要确保OpenSSL>=1.0.1,SSLv3 现在还有很多地方在用但是有不少被攻击的漏洞

ssl\_ciphers ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP; #加密算法，不同的浏览器所支持的套件（和顺序）可能会不同。这里指定的是OPENSSL库能够识别的写法，你可以通过openssl -v cipher ‘RC4:HIGH:!aNULL:!MD5’（后面是你所指定的套件加密算法）来看所支持的算法。

　　ssl\_prefer\_server\_ciphers on; #启动加密算法

ssl\_session\_cache shared:SSL:10m;#设置ssl/tls会话缓存的类型和大小。如果设置了这个参数一般是sheared，buildin可能会参数内存碎片，默认是none，和off差不多，停用缓存。如shared:SSL:10m表示我所有的nginx工作进程共享ssl会话缓存，官网介绍说1m可以存放越4000个sessions。

**Location 标签**

deny all；#禁止访问

fastcgi\_pass  127.0.0.1:9000;  #抛给本机的9000端口

  fastcgi\_index index.php; #设置动态页面

  allow   219.237.222.30 ;#允许访问的ip

 expires      30d;#客户端缓存30天

**负载均衡，反向代理相关**

proxy\_pass [http://server反向代理到](http://server;反向代理到upstream)***[upstream](http://server;反向代理到upstream) server1{}模块***

#后端的Web服务器可以通过X-Forwarded-For获取用户真实IP

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

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

**proxy\_redirect off指定修改被代理服务器返回的响应头中的location头域跟refresh头域数值**

**proxy\_max\_temp\_file\_size 0; #**这是每个请求能用磁盘上临时文件最大大小。这些当上游响应太大不能装配到缓冲区时被创建。

**proxy\_connect\_timeout 90; #nginx跟后端服务器连接超时时间(代理连接超时)**

**proxy\_send\_timeout 90; #后端服务器数据回传时间(代理发送超时) / 在连接断开之前两次发送至upstream server的写操作的最大间隔时长；**

**proxy\_read\_timeout 90; #连接成功后，后端服务器响应时间(代理接收超时)/ 在连接断开之前两次从接收upstream server接收读操作的最大间隔时长；**

**proxy\_buffer\_size 4k; #设置代理服务器（nginx）保存用户头信息的缓冲区大小**

**proxy\_buffers 4 32k; #proxy\_buffers缓冲区，网页平均在32k以下的话，这样设置**

**proxy\_busy\_buffers\_size 64k; #高负荷下缓冲大小（proxy\_buffers\*2）**

**proxy\_temp\_file\_write\_size 64k; #设定缓存文件夹大小，大于这个值，将从upstream服务器传**

**client\_max\_body\_size #上传文件大小最大限制**

client\_header\_buffer\_size 4k;

客户端请求头部的缓冲区大小。这个可以根据你的系统分页大小来设置，一般一个请求头的大小不会超过1k，不过由于一般系统分页都要大于1k，所以这里设置为分页大小。

分页大小可以用命令getconf PAGESIZE 取得。

client\_body\_buffer\_size 512k;

如果把它设置为比较大的数值，例如256k，那么，无论使用firefox还是IE浏览器，来提交任意小于256k的图片，都很正常。如果注释该指令，使用默认的client\_body\_buffer\_size设置，也就是操作系统页面大小的两倍，8k或者16k，问题就出现了。

无论使用firefox4.0还是IE8.0，提交一个比较大，200k左右的图片，都返回500 Internal Server Error错误

proxy\_max\_temp\_file\_size 0;

设置最大的缓存大小，0位不缓存

**缓存设置**

nginx的http\_proxy模块，可以实现类似于Squid的缓存功能。Nginx对客户已经访问过的内容在Nginx服务器本地建立副本，这样在一段时间内再次访问该数据，就不需要通过Ｎginx服务器再次向后端服务器发出请求，所以能够减少Ｎginx服务器与后端服务器之间的网络流量，减轻网络拥塞，同时还能减小数据传输延迟，提高用户访问速度。同时，当后端服务器宕机时，Nginx服务器上的副本资源还能够回应相关的用户请求，这样能够提高后端服务器的鲁棒性。

proxy\_cache\_path /usr/local/nginx/cache levels=1:2 keys\_zone=my\_cache:10m max\_size=10g inactive=60m use\_temp\_path=off;

Poxy\_cache此缓存设置应该写在http模块里

/usr/local/nginx/cache 缓存文件存放地址

**Levels=1:2**

默认所有缓存文件都放在同一个/usr/local/nginx/cache下，但会影响缓存性能，因此通常会在/usr/local/nginx/cache下面建立子目录用来分别存放不同的文件。假设levels=1:2，Nginx为将要缓存的资源生成key为f4cd0fbc769e94925ec5540b6a4136d0,那么key的最后一位0，以及倒数第2-3位6d作为两级的子目录，也就是该资源最终会被缓存到/usr/local/nginx/cache/0/6d目录中

**Keys\_zone=my\_cache**

在共享内存中设置一块存储区域来存放缓存的key和metadata（类似使用次数）这样nginx可以快速判断一个request是否命中或者未命中缓存，1m可以存储8000个key 10m可以存储80000个key

**max\_size**

最大的ceche空间，如果不指定，会使用所有的disk space ，当达到配额后，会删除最少使用的cache文件

**Inactive=60m**

未被访问文件在缓存中保留的时间，本配置中如果60分钟未被访问则不论状态是否为expired，缓存控制程序会删掉文件。inactive默认是10分钟。需要注意的是，inactive和expired配置项的含义是不同的，expired只是缓存过期，但不会被删除，inactive是删除指定时间内未被访问的缓存文件

**use\_temp\_path**

如果为off，则nginx会将缓存文件直接写入指定的cache文件中，而不是使用temp\_path存储，official建议为off，避免文件在不同文件系统中不必要的拷贝

location / {

...

proxy\_cache\_use\_stale error timeout http\_500 http\_502 http\_503 http\_504;

}

**proxy\_cache\_use\_stale**

增强站点容错能力

源站有问题时，nginx可以通过proxy\_cache\_use\_stale指令开启容错能力，即使用缓存内容来响应客户端的请求。

**proxy\_set\_header Host $host;**

自定义http header头，用于发送给后端真实服务器。

**proxy\_cache cache\_one;**

启用proxy cache，并指定key\_zone。另外，如果proxy\_cache off表示关闭掉缓存。

**proxy\_cache\_valid**

为不同的HTTP返回状态码的资源设置不同的缓存时长。

**proxy\_cache\_valid 200 304 12h;**

**为响应码是200和304的资源，设置缓存时长为12小时**

**proxy\_cache\_valid 301 302 1m;**

**为响应码是301和302的资源，设置缓存时长为1分钟**

**proxy\_cache\_valid any 1m；**

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

nginx对缓存的资源会设置一个key，NGINX生成的键的默认格式是类似于上面的NGINX变量的MD5哈希值:$scheme$proxy\_host$request\_uri，实际的算法有些复杂。 为了改变变量（或其他项）作为基础键，可以使用proxy\_cache\_key命令。例如，

**proxy\_cache\_purge**

缓存清理指令

**Gzip压缩**

在ngixn服务器配置文件中可以通过配置Gzip的使用，可以配置http块，server块或者location块中设置，nginx服务器可以通过ngx\_http\_gzip\_module模块，ngx\_http\_static\_module模块，ngx\_http\_static\_module模块功能ngx\_http\_gunzip\_\_module模块对这些指令进行分析和处理。

1. **Ngx\_http\_gzip\_modu*le模块处理指令***
2. ***gzip：***

*用于打开或者关闭gzip压缩功能，用法如下****：***

gzip on|off #默认为off

1. **gzip\_buffers：**

用于指定gzip功能压缩文件时使用的缓存空间的大小，语法结构为：

Gzip\_buffers number | size;

Number #指定Nginx服务器需要向服务器申请的缓存空间个数

Size #指定每个缓存空间的大小，从nginx0.7.8开始，默认number\*size的值为128，其中size取系统中内存页一页的大小，为4k或者8k等

Gzip\_buffers 32 4k | 12 8k;

**3.gzip\_comp\_level**

*指定gzip压缩级别，级别为1-9，默认为1，级别越高压缩率越高即压缩后的文件越小，但占用cpu，压缩率越小压缩后的文件越大，但是节约cpu。*

*Gzip\_comp\_level 4; #推销先设置为中间值，比如4或5*

***4.gzip\_disbale***

*针对不同客户端发起的请求进行有选择的打开或关闭gzip命令，后跟浏览器的名称，用法如下：*

*Gzip\_disable msie[1-6]\. 3使用正则表达式可以配置UC字符串中MSIE 1-6浏览器，这三个浏览器在发起请求后nginx将不进行数据包的压缩。5、gzip\_http\_verson*

*早期的一些浏览器或者http客户端可能不支持gzip压缩，因此有客户有时候会看到乱码，那是应为客户端无法进行解压，因此可以根据不同的http协议版本进行选择性的打开或关闭gzip功能，用法如下：*

*gzip\_http\_version 1.0 | 1.1; #对使用htto协议1.0和1.1的请求进行压缩*

***6、gzip\_min\_length***

*设置最少压缩的数据包大小，小于此值的数据包将不被压缩，因为大量的小的数据压缩后没有明显的效果还占用了CPU的资源，因此可以设置最少多大的数据才进行压缩，用法如下：*

*gzip\_min\_length 1024; #默认为20，建议设置位1KB，如果设置为0则表示压缩全部数据包*

***7、gzip\_proxine***

*在Nginx 服务器作为反向代理的时候有效，用于设置nginx 服务器是否对后端返回的结果进行gzip压缩，用法如下：*

*gzip\_proxied off | any; #off为关闭，any为压缩所有后端服务器返回的数据。*

***8、gzip\_types***

*设置Nginx服务器可以根据响应页面的类型进行选择性的打开或关闭gzip功能，用法如下：*

*gzip\_types text/plain application/x-javascript text/css application/xml; #一般情况下如此设置*

***9、gzip\_vary***

*用于设置是否在使用gzip功能时发送带有"vary:Accept-Encoding" 头域的响应头部，该头域的主要功能时要告诉客户端数据已经在服务器进行了压缩，默认设置为off，用法如下：*

*gzip\_vary on| off; #默认为off*

*还可以使用Nginx 配置的add\_header指令强制在Nginx服务器的响应头部添加“Vary:Accept-Encoding”也可以实现相同的效果。*

***二：ngx\_http\_gzip\_static\_module 模块：***

*该模块辅助用于搜索和发送经过gzip功能压缩的数据，这些数据以\*.gz作为后缀名存储在服务器上，如果客户端请求的数据之前被压缩过，并且客户端的浏览器支持压缩，将直接范湖压缩后的数据，该模块使用的是静态编码，在http响应头部包含content-length头域来指明报文的长度，用于服务器可以确定*

*响应数据的长度的情况，而ngx\_http\_gzip\_module使用chunked编码动态压缩，主要用于服务器无法确定响应数据长度的情况，比如较大文件的下载等情形，此时就要实时生成数据的长度，用法与ngx\_http\_gzip\_module一样，如下：*

***1、gzip\_static***

*gzip\_static off | on | always; #on为开启并检查客户端浏览器是否中吃gzip压缩功能，off为关闭，always一直发送gzip压缩文件，而不检查浏览器是否支持gzip压缩*

*注：这是一个可选模块，如果要使用，必须在编译nginx时添加--with-http\_gzip\_static\_module编译，与该模块有关的 指令与ngx\_http\_gzip\_module的模块使用方法是一致的，本模块有gzip\_static、gzip\_http\_version、gzip\_proxied、gzip\_disable和gzip\_vary等。*

***三：ngx\_http\_gunzip\_module 模块处理的两个指令***

***1、gunzip***

*设置Nginx服务器对不支持gzip的客户端返回解压后的数据，如果客户的浏览器支持压缩还仍然返回压缩的后的数据，及可以同时响应支持gzip的浏览器进程压缩，有可以响应不支持gzip的浏览器。*

*gunzip on|off ；#on为打开off为关闭*

***2、gunzip\_buffers*** *#用于设置nginx服务器解压gzip文件时候使用的缓存空间大小的，用法如下：*

*gunzip\_buffers number size； #number为nginx服务器向系统向系统申请缓存空间的个数，size为每个空间的大小，单位为k，默认情况下number \* size的大小为128k，其中size 的值取系统内存页一页的大小为4KB或者8KB即可，用法如下：*

*gunzip\_buffers 32 4k | 16 8k; #与ngx\_http\_gzip\_module模块下的使用方法一致。*

***反向代理，负载均衡***

***设置负载均衡服务器列表***

***upstream [名称]***

***{***

*Ip\_hash;当用户再访问时，会将该请求通过哈希算法，自动定位到该服务器。这样每个访客固定访问一个后端服务器，可以解决session的问题。*

*server 192.168.52.128 weigth 8；*

#weigth参数表示权值，权值越高被分配到的几率越大，轮询的加强版

*#fail\_timeout 等待请求的目标服务器发送响应的时长。*

*#backup 用于fallback的目的，所有服务器故障时才启动此服务器*

*#down 手动标记其不再处理如何请求*

#fair 顾名思义，公平地按照后端服务器的响应时间（rt）来分配请求，响应时间短即rt小的后端服务器优先分配请求。Nginx本身不支持fair，必须下载Nginx的upstream\_fair模块。

#least\_conn（最少连接数）

# consistent\_hash $request\_uri; 一致性hash

***}***

*-----------------------------------------------------------------------------------------------------------------*

*附：完整的nginx.conf配置文件*

*user nginx nginx;*

*worker\_processes auto;*

*#error\_log logs/error.log;*

*#error\_log logs/error.log notice;*

*#error\_log logs/error.log info;*

*#pid logs/nginx.pid;*

*events {*

*worker\_connections 1024; #单个后台worker process进程的最大并发链接数*

*}*

*http {*

*include mime.types; #设定mime类型,类型由mime.type文件定义*

*default\_type application/octet-stream;*

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

*'fwf[$http\_x\_forwarded\_for] tip[$http\_true\_client\_ip] '*

*'$upstream\_addr $upstream\_response\_time $request\_time '*

*'$http\_host $request '*

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

*'"$http\_accept\_language" "$http\_user\_agent" ';*

*#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 logs/access.log main;*

*sendfile on; #sendfile 指令指定 nginx 是否调用 sendfile 函数（zero copy 方式）来输出文件，对于普通应用，*

*#必须设为 on,如果用来进行下载等应用磁盘IO重负载应用，可设置为 off，以平衡磁盘与网络I/O处理速度，降低系统的uptime.*

*#将tcp\_nopush和tcp\_nodelay两个指令设置为on用于防止网络阻塞*

*tcp\_nopush on;*

*tcp\_nodelay on;*

*#配置nginx代理服务器哦缓存来减少响应时间*

*#keepalive\_timeout 0;*

*keepalive\_timeout 65; #连接超时时间*

*proxy\_cache\_path /usr/local/nginx/cache levels=1:2 keys\_zone=cache\_one:500m inactive=1d max\_size=30g;*

*proxy\_temp\_path /usr/local/nginx/temp;*

*gzip on; #开启gzip压缩*

*#设定请求缓冲*

*client\_header\_buffer\_size 128k;*

*large\_client\_header\_buffers 4 128k;*

*proxy\_buffer\_size 64k;*

*proxy\_buffers 8 64k;*

*fastcgi\_buffer\_size 128k;*

*fastcgi\_buffers 4 128k;*

*#设定负载均衡的服务器列表*

*upstream server1 {*

*#weigth参数表示权值，权值越高被分配到的几率越大*

*# #本机上的Squid开启3128端口*

*# ip\_hash; #当用户再次访问时，会将该请求通过哈希算法，自动定位到该服务器。这样每个访客固定访问一个后端服务器，可以解决session的问题。*

*#server 192.168.52.129:8080 weight=5;*

*#server 192.168.52.129:80 weight=1;*

*server 192.168.52.134:8080;*

*#max\_fails：最大失败连接次数，失败连接的超时时长由fail\_timeout指定；*

*#fail\_timeout：等待请求的目标服务器发送响应的时长；*

*#backup：用于fallback的目的，所有服务均故障时才启动此服务器；*

*#down：手动标记其不再处理任何请求*

*}*

*upstream server2 {*

*server 192.168.52.134:8080;*

*}*

*server {*

*listen 192.168.52.134:80;*

*server\_name www.web1.com;*

*index index.php index.html;*

*charset UTF-8;*

*#charset koi8-r;*

*access\_log /web1/log/access.log ;*

*error\_log /web1/log/error.log error;*

*location / {*

*index index.php;*

*try\_files $uri $uri/ /index.php?$args;*

*proxy\_cache cache\_one;*

*proxy\_pass http://server1;*

*#以下是一些反向代理的配置可删除.*

*proxy\_redirect off;*

*#后端的Web服务器可以通过X-Forwarded-For获取用户真实IP*

*proxy\_set\_header Host $host;*

*proxy\_set\_header X-Real-IP $remote\_addr;*

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

*proxy\_next\_upstream error timeout invalid\_header http\_500 http\_502 http\_503 http\_504;*

*client\_max\_body\_size 10m;*

*client\_body\_buffer\_size 128k;*

*proxy\_max\_temp\_file\_size 0;*

*proxy\_connect\_timeout 90; #nginx跟后端服务器连接超时时间(代理连接超时)*

*proxy\_send\_timeout 90; #后端服务器数据回传时间(代理发送超时) / 在连接断开之前两次发送至upstream server的写操作的最大间隔时长；*

*proxy\_read\_timeout 90; #连接成功后，后端服务器响应时间(代理接收超时)/ 在连接断开之前两次从接收upstream server接收读操作的最大间隔时长；*

*proxy\_buffer\_size 4k; #设置代理服务器（nginx）保存用户头信息的缓冲区大小*

*proxy\_buffers 4 32k; #proxy\_buffers缓冲区，网页平均在32k以下的话，这样设置*

*proxy\_busy\_buffers\_size 64k; #高负荷下缓冲大小（proxy\_buffers\*2）*

*proxy\_temp\_file\_write\_size 64k; #设定缓存文件夹大小，大于这个值，将从upstream服务器传*

*}*

*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 {*

*}*

*location /backend {*

*proxy\_pass http://server1/backend;*

*#以下是一些反向代理的配置可删除.*

*proxy\_redirect off;*

*# #后端的Web服务器可以通过X-Forwarded-For获取用户真实IP*

*proxy\_set\_header Host $host;*

*proxy\_set\_header X-Real-IP $remote\_addr;*

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

*proxy\_next\_upstream error timeout invalid\_header http\_500 http\_502 http\_503 http\_504;*

*client\_max\_body\_size 10m;*

*client\_body\_buffer\_size 128k;*

*proxy\_max\_temp\_file\_size 0;*

*proxy\_connect\_timeout 90; #nginx跟后端服务器连接超时时间(代理连接超时)*

*proxy\_send\_timeout 90; #后端服务器数据回传时间(代理发送超时) / 在连接断开之前两次发送至upstream server的写操作的最大间隔时长；*

*#*

*proxy\_read\_timeout 90; #连接成功后，后端服务器响应时间(代理接收超时)/ 在连接断开之前两次从接收upstream server接收读操作的最大间隔时长；*

*proxy\_buffer\_size 4k; #设置代理服务器（nginx）保存用户头信息的缓冲区大小*

*proxy\_buffers 4 32k; #proxy\_buffers缓冲区，网页平均在32k以下的话，这样设置*

*proxy\_busy\_buffers\_size 64k; #高负荷下缓冲大小（proxy\_buffers\*2）*

*proxy\_temp\_file\_write\_size 64k; #设定缓存文件夹大小，大于这个值，将从upstream服务器传*

*#location ~ \.php$ {*

*# proxy\_pass http://127.0.0.1;*

*#}*

*}*

*# pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000*

*#*

*#location ~\*/{*

*#proxy\_pass http://server1;*

*# fastcgi\_index index.php;*

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

*# include fastcgi\_params;*

*# }*

*# deny access to .htaccess files, if Apache's document root*

*# concurs with nginx's one*

*#*

*location ~ /\.ht {*

*deny all;*

*}*

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

*{*

*proxy\_pass http://server1;*

*proxy\_set\_header Host $host;*

*proxy\_cache cache\_one;*

*proxy\_cache\_valid 200 304 12h;*

*proxy\_cache\_valid 301 302 1m;*

*proxy\_cache\_valid any 1m;*

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

*}*

*}*

*server {*

*listen 192.168.52.134:8080;*

*server\_name www.web1.com;*

*root /var/www/html/newscms/web;*

*index index.php index.html;*

*charset UTF-8;*

*#charset koi8-r;*

*access\_log /web1/log/access.log ;*

*error\_log /web1/log/error.log error;*

*location / {*

*index index.php;*

*try\_files $uri $uri/ /index.php?$args;*

*#以下是一些反向代理的配置可删除.*

*proxy\_redirect off;*

*#后端的Web服务器可以通过X-Forwarded-For获取用户真实IP*

*proxy\_set\_header Host $host;*

*proxy\_set\_header X-Real-IP $remote\_addr;*

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

*proxy\_next\_upstream error timeout invalid\_header http\_500 http\_502 http\_503 http\_504;*

*client\_max\_body\_size 10m;*

*client\_body\_buffer\_size 128k;*

*proxy\_max\_temp\_file\_size 0;*

*proxy\_connect\_timeout 90; #nginx跟后端服务器连接超时时间(代理连接超时)*

*proxy\_send\_timeout 90; #后端服务器数据回传时间(代理发送超时) / 在连接断开之前两次发送至upstream server的写操作的最大间隔时长；*

*proxy\_read\_timeout 90; #连接成功后，后端服务器响应时间(代理接收超时)/ 在连接断开之前两次从接收upstream server接收读操作的最大间隔时长；*

*proxy\_buffer\_size 4k; #设置代理服务器（nginx）保存用户头信息的缓冲区大小*

*proxy\_buffers 4 32k; #proxy\_buffers缓冲区，网页平均在32k以下的话，这样设置*

*proxy\_busy\_buffers\_size 64k; #高负荷下缓冲大小（proxy\_buffers\*2）*

*proxy\_temp\_file\_write\_size 64k; #设定缓存文件夹大小，大于这个值，将从upstream服务器传*

*}*

*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 {*

*}*

*rewrite /backend /backend/index.php break;*

*#last：重写完成后停止对当前URI在当前location中后续的其它重写操作，而后对新的URI启动新一轮（从第一个开始）重写检查；提前重启新一轮循环。*

*#break：重写完成后停止对当前URI在当前location中后续的其它重写操作，而后直接跳转至重写规则配置块之后的其它配置；结束循环，建议在location中使用。*

*#redirect：临时重定向，重写完成后以临时重定向方式直接返回重写后生成的新URI给客户端，由客户端重新发起请求；不能以http://或https://开头，使用相对路径，状态码： 302。*

*#permanent：重写完成后以永久重定向方式直接返回重写后生成的新URI给客户端，由客户端重新发起请求，状态码：301。*

*# proxy the PHP scripts to Apache listening on 127.0.0.1:80*

*#*

*#location ~ \.php$ {*

*# proxy\_pass http://127.0.0.1;*

*#}*

*location ~ ^/(images|javascript|js|css|flash|media|static)/ {*

*#过期30天，静态文件不怎么更新，过期可以设大一点，如果频繁更新，则可以设置得小一点。*

*expires 30d;*

*}*

*# pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000*

*#*

*location ~\*\.php$ {*

*fastcgi\_pass 127.0.0.1:9000;*

*fastcgi\_index index.php;*

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

*include fastcgi\_params;*

*}*

*rewrite /backend /backend/index.php break;*

*# deny access to .htaccess files, if Apache's document root*

*# concurs with nginx's one*

*#*

*location ~ /\.ht {*

*deny all;*

*}*

*}*

*# HTTPS server*

*#*

*#server {*

*# listen 443 ssl;*

*# server\_name localhost;*

*# ssl\_certificate cert.pem;*

*# ssl\_certificate\_key cert.key;*

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

*# ssl\_session\_timeout 5m;*

*# ssl\_ciphers HIGH:!aNULL:!MD5;*

*# ssl\_prefer\_server\_ciphers on;*

*# location / {*

*# root html;*

*# index index.html index.htm;*

*# }*

*#}*

*}*