官网地址：

<http://nginx.org/en/docs/http/ngx_http_core_module.html#try_files>

<https://www.fcwys.cc/archives/255.html>

nginx负载均衡的4种方法

轮询（默认）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。

upstream backserver {

server 192.168.0.14;

server 192.168.0.15;

}

权重 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。

upstream backserver {

server 192.168.0.14 weight=3;

server 192.168.0.15 weight=7;

}

上面的策略会影响登录状态 每个请求按访问ip的hash结果分配，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。

Ip-hash

upstream backserver {

ip\_hash;

server 192.168.0.14:88;

server 192.168.0.15:80;

}

4、Fair看响应时间，响应时间短的优先分配

upstream backserver {

server server1;

server server2;

fair;

}

5、请求url\_hash，相同的请求地址分配到指定的机器 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，后端服务器为缓存时比较有效。

upstream backserver {

server squid1:3128;

server squid2:3128;

hash $request\_uri;

hash\_method crc32;

}

6、设置服务器状态来影响负载均衡。



nginx支持同时设置多组的负载均衡，用来给不用的server来使用。

正向代理和反向代理

<https://www.fcwys.cc/archives/254.html>

正向代理——需要借助响应的代理软件

在客户端需要配置代理服务器，和端口。客户端一般在内网，想访问外网资源，外网资源只有代理服务器能访问到。

用途：

访问原来无法访问的软件

可以做缓存，加速资源访问

对客户端访问授权，进行上网认证

代理软件可以记录用户访问记录，对外隐藏用户信息

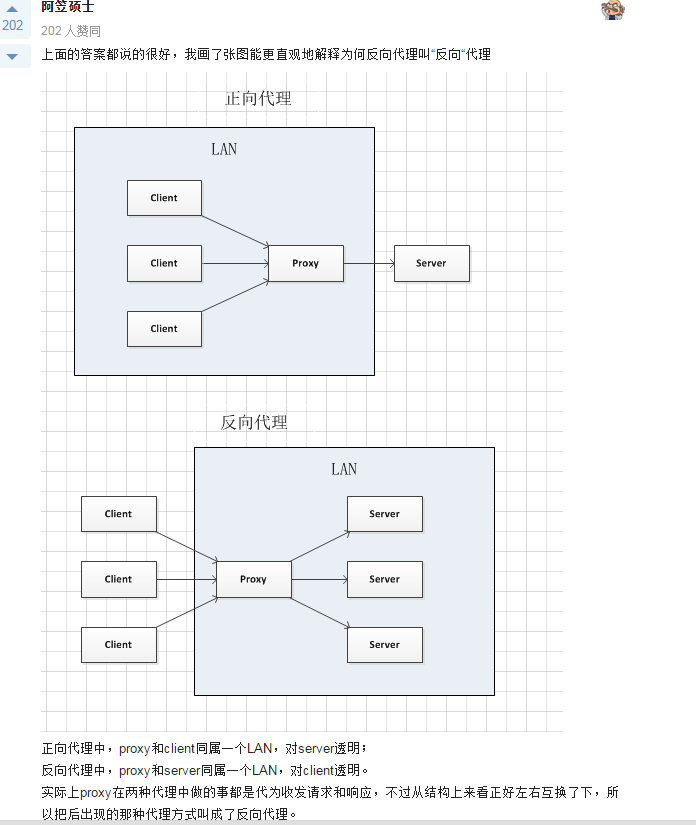
反向代理——也是需要代理软件

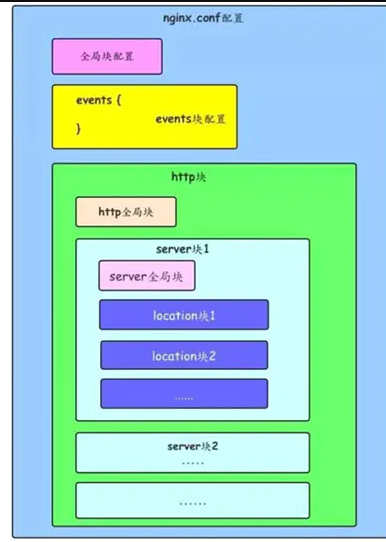
客户端访问不需要配置代理服务器地址和端口，代理服务器对客户端是无感知的。客户端在互联网上，访问的资源在某个内网上面，内网上的资源不对外公开，要能让在外网的客户端能访问到，可以通过一个和外网联通的服务器来提供服务。这个和外网联通的的服务器就是代理服务器，要实现代理功能，也需要相应的软件来实现。代理服务软件，除了转发请求，可以有很多功能，负载均衡、安全控制、流量过滤等等。

反向代理一般具有的功能

保证内网安全。

负载均衡





**main（全局设置）**：设置的指令将影响其他所有设置；  
**server（主机设置）**：指令主要用于指定主机和端口、  
**upstream（负载均衡服务器设置）**：指令主要用于负载均衡，设置一系列的后端服务器  
**location（URL匹配特定位置的设置）**：用于匹配网页位置。

server继承main，location继承server，upstream既不会继承其他设置也不会被继承。

主模块配置解释：

a、配置运行Nginx服务器用户（组）  
     如：**user nobody nobody;**   user是个主模块指令，指定Nginx Worker进程运行以及用户组。  
    *指令格式：user user [group];    
     user：指定可以运行Nginx服务器的用户；group：可选项，可以运行Nginx服务器的用户组。  
     如果user指令不配置或者配置为user nobody nobody，默认由nobody账户运行。*

    b、worker process数      
      如：**worker\_processes 2;**    woker\_processes是个主模块指令，制定了Nginx要开启的进程数。每个Nginx进程平均耗费10M~12M内存。建议指定和CPU的数量一致即可。  
      Nginx服务器实现并发处理服务的关键。  
      *指令格式：worker\_processes number | auto;  
      number : Nginx 进程最多可以产生的worker process 数。  
      auto ： Nginx 进程将自动检测*  
      在按照上面的配置格式配置了之后，假如上面的数目是2，那么启动Nginx服务器后，在后台主机上查看Nginx的进程情况，可以看到应该是有2个Nginx进程。

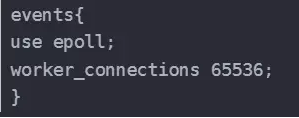
    c、错误日志的存放路径  
      如：**error\_log logs/error.log  notice;**  error\_log 是个主模块指令，用来定义全局错误日志文件。日志输出级别有debug，info，notice，warn，error，erit可供选择，其中，debug输出日志最为详细，而crit输出日志最少。  
     *指定格式：error\_log file  | stderr;  
     file : 日志输出到某个文件file  
     stderr : 日志输出到标准错误输出 （日志输出级别）。*

    d、Nginx进程PID存放路径  
       如：**pid logs/nginx.pid;**  pid是个主模块指令，用来指定进程pid的存储文件位置。  
       Nginx进程是作为系统守护进程在进行，需要在某个文件中保存当前运行程序的主进程号，Nginx支持该保存文件路径的定义。  
       *指令格式：pid file;  
       file：指定存放路径和文件名称。*  
        如果不指定，则默认置于路径 logs/nginx.pid

    e、worker\_rlimit\_nofile   
       如：**worker\_rlimit\_nofile 65535;** 用来绑定worker进程和CPU，Linux内核2.4 以上可用  
       指定进程可以打开的最多文件描述数目，理论值应该是最多打开文件数（ulimit -n ）与nginx进程数相除，但是Nginx分配请求并不是那么均匀，所以最好与ulimit -n 的值保持一致。  
       现在在Linux2.6 内核下开启文件打开数为65535，worker\_rlimit\_nofile 就相应应该填写65535.这是因为Nginx调度时请求到进程并不是那么均衡，所以假如填写10240，总并发量达到3-4万时就有进程超过10240了，这就会返回502。

在Linux2.6 内核下开启文件打开数为65535

 events 事件指令是设定Nginx的工作模式及连接数上限

**2:events块  
 ** events事件指令是设定Nginx的工作模式及连接数上限。每个配置选项的含义解释如下：  
*a、use  
如：use epoll;   
use是事件模块指令，用来指定Nginx的工作模式。Nginx支持的工作模式有select、poll、kqueue、epoll、rtsig和/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，不同的是epoll用在Linux平台上，而kqueue用在BSD系统中。对于Linux系统，epoll工作模式是首选。*  
  
*b、worker\_connections  
如：worker\_connections65536;*  
*work\_connections也是个事件模块指令，用于定义Nginx每个进程的最大连接数，默认是1024。*

### 3：http块

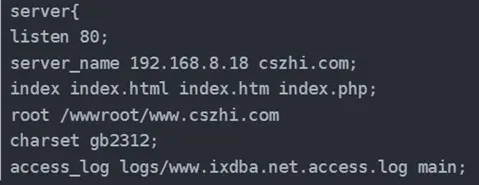
A、定义MIMI-Type  
如：include  mime.types;  
*指令格式:include file;*  
*该指令主要用于将其他的Nginx配置或第三方模块的配置引用到当前的主配文件中，减少主配置文件的复杂度。default\_type  application/octet-stream;  
default\_type*  
属于HTTP核心模块指令，这里设定默认类型为二进制流。也就是当文件类型未定义时使用这种方式，

B、自定义服务日志  
 log\_format 是Nginx的HttpLog模块指令，用于指定Nginx日志的输出日志。当然其中还有更为详细的配置，但是这里只做大略的研究学习，不过分深入细节。

C、允许sendfile方式传输文件  
*指令格式：sendfile  on;  
这个指令中的参数on是表示开启高效文件传输模式，默认是关闭状态（off），将tcp\_nopush和tcp\_nodelay两个指令设置为on用于防止网络阻塞；*

D、连接超时时间  
*指令格式：keepalive\_timeout number;  
如：keepalive\_timeout 65;  
  keepalive\_timeout  
 设置客户端连接保持活动的超时时间。在超过这个时间之后，服务器会关闭该连接；*

### 4、server块



server 块是对虚拟主机的配置，server标志定义虚拟主机开始，

A、listen用于指定虚拟主机的服务端口，  
B、server\_name 用来指定IP地址或域名，多个域名之间用空格分开。  
C、index用于设定访问的默认首页地址.。  
D、root指令用于指定虚拟主机的网页根目录，这个目录可以是相对路径，也可以是绝对路径。  
E、charset用于设置网页的默认编码格式。  
F、access\_log 用来指定虚拟主机的访问日志存放路径，最后的main 用于指定访问日志的输出格式。

### 5.location块

URL地址匹配是进行Nginx配置中最灵活的部分。 location支持正则表达式匹配，也支持条件判断匹配，用户可以通过location指令实现Nginx对动、静态网页进行过滤处理。使用location URL匹配配置还可以实现反向代理，用于实现PHP动态解析或者负载负载均衡。



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

语法：

rewrite    <regex>    <replacement>    [flag];

关键字      正则        替代内容          flag标记

rewrite /$1 break;

^/api/supply/(.\*)$ 匹配以 /api/supply/ 开始的路径，重定向到$1 就是url除去/api/supply/后面的部分

rewrite是实现URL重写的关键指令，根据regex（正则表达式）部分内容，重定向到replacement，结尾是flag标记。

关键字：其中关键字

正则：perl兼容正则表达式语句进行规则匹配

替代内容：将正则匹配的内容替换成replacement

flag标记：rewrite支持的flag标记

flag标记说明：

last  #本条规则匹配完成后，继续向下匹配新的location URI规则

break  #本条规则匹配完成即终止，不再匹配后面的任何规则

redirect  #返回302临时重定向，浏览器地址会显示跳转后的URL地址

permanent  #返回301永久重定向，浏览器地址栏会显示跳转后的URL地址

rewrite命令可以使用在这些标签里面 server,location,if

**例子：**

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

说明：

rewrite为固定关键字，表示开始进行rewrite匹配规则

regex部分是 ^/(.\*) ，这是一个正则表达式，匹配完整的域名和后面的路径地址

replacement部分是http://www.czlun.com/$1 $1，是取自regex部分()里的内容。匹配成功后跳转到的URL。

flag部分 permanent表示永久301重定向标记，即跳转到新的 http://www.czlun.com/$1 地址上

参考文档

<https://www.cnblogs.com/quwaner/p/8193186.html>

location

语法：**location [=|~|~\*|^~|@] pattern {……}**

location ^~/api/supply {}

**location区段**

通过指定模式来与客户端请求的URI相匹配，基本语法如下：**location [=|~|~\*|^~|@] pattern{……}**

1、没有修饰符 表示：必须以指定模式开始，如：

[复制代码](javascript:void(0);)

server {

　　server\_name baidu.com;

　　location /abc {

　　　　……

　　}

}

那么，如下是对的：  
http://baidu.com/abc  
http://baidu.com/abc?p1  
http://baidu.com/abc/  
http://baidu.com/abcde

[复制代码](javascript:void(0);)

2、=表示：必须与指定的模式精确匹配

[复制代码](javascript:void(0);)

server {

server\_name sish

　　location = /abc {

　　　　……

　　}

}

那么，如下是对的：

http://baidu.com/abc

http://baidu.com/abc?p1

如下是错的：

http://baidu.com/abc/

http://baidu.com/abcde

[复制代码](javascript:void(0);)

3、~ 表示：指定的正则表达式要区分大小写

[复制代码](javascript:void(0);)

server {

server\_name baidu.com;

　　location ~ ^/abc$ {

　　　　……

　　}

}

那么，如下是对的：

http://baidu.com/abc

http://baidu.com/abc?p1=11&p2=22

如下是错的：

http://baidu.com/ABC

http://baidu.com/abc/

http://baidu.com/abcde

[复制代码](javascript:void(0);)

4、~\* 表示：指定的正则表达式不区分大小写

[复制代码](javascript:void(0);)

server {

server\_name baidu.com;

location ~\* ^/abc$ {

　　　　……

　　}

}

那么，如下是对的：

http://baidu.com/abc

http://baidu..com/ABC

http://baidu..com/abc?p1=11&p2=22

如下是错的：

http://baidu..com/abc/

http://baidu..com/abcde

[复制代码](javascript:void(0);)

5、^~ 类似于无修饰符的行为，也是以指定模式开始，不同的是，如果模式匹配，  
那么就停止搜索其他模式了。  
6、@ ：定义命名location区段，这些区段客户端不能访问，只可以由内部产生的请  
求来访问，如try\_files或error\_page等

**查找顺序和优先级  
1：带有“=“的精确匹配优先  
2：没有修饰符的精确匹配  
3：正则表达式按照他们在配置文件中定义的顺序  
4：带有“^~”修饰符的，开头匹配  
5：带有“~” 或“~\*” 修饰符的，如果正则表达式与URI匹配  
6：没有修饰符的，如果指定字符串与URI开头匹配**

匹配顺序

1、带等号= location，精确匹配。

2、没带符号的location， 以匹配字符开始的uri，找出更长的匹配。例如：两个location /test; location /test/empty; uri=/test/empty/xxx，两个location都匹配的情况下，最终会执行location /test/empty。每一个uri匹配都会将所有的 没有符号的location 匹配一遍，选择最长匹配的执行。

3、带^~ 的location 匹配就停止继续匹配。没有最长匹配一说

4、带~\*的location，不区分大小写。带 ~的location，区分大小写。如果找到相应的匹配，则nginx停止搜索其他匹配；当没有正则表达式或者没有正则表达式被匹配的情况下，那么匹配程度最高的逐字匹配location会被使用

符号相同的，按配置文件中定义的顺序匹配

location = / {

# 只匹配"/".

[ configuration A ]

}

location / {

# 匹配任何请求，因为所有请求都是以"/"开始

# 但是更长字符匹配或者正则表达式匹配会优先匹配

[ configuration B ]

}

location ^~ /images/ {

# 匹配任何以 /images/ 开始的请求，并停止匹配 其它location

[ configuration C ]

}

location ~\* .(gif|jpg|jpeg)$ {

# 匹配以 gif, jpg, or jpeg结尾的请求.

# 但是所有 /images/ 目录的请求将由 [Configuration C]处理.

[ configuration D ]

}

参考文档：

<https://www.cnblogs.com/xiaoliangup/p/9175932.html>

比较好的博客

<https://www.nginx.cn/115.html>

location中的proxy\_pass   
里面是proxy\_pass的值以/结尾时nginx最后匹配的网址是的内容加proxy\_pass+（url去掉匹配字符串）  
里面是proxy\_pass不以/结尾时nginx最后匹配的网址是 proxy\_pass + url

proxy\_pass 就是反向代理

upstream 是负载均衡

# [nginx负载均衡（url\_hash）配置](https://www.cnblogs.com/zl0372/articles/nginx.html)

<https://www.cnblogs.com/zl0372/articles/nginx.html>

url\_hash 问题阐述

使用场景 服务拆分

特殊功能专用服务器

location / {

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

}

当用户请求 http://localhost/example 时，这里的 $uri 就是 /example。

try\_files 会到硬盘里尝试找这个文件。如果存在名为 /$root/example（其中 $root 是项目代码安装目录）的文件，就直接把这个文件的内容发送给用户。

显然，目录中没有叫 example 的文件。然后就看 $uri/，增加了一个 /，也就是看有没有名为 /$root/example/ 的目录。

又找不到，就会 fall back 到 try\_files 的最后一个选项 /index.php，发起一个内部 “子请求”，也就是相当于 nginx 发起一个 HTTP 请求到 http://localhost/index.html。

[复制代码](javascript:void(0);)

loaction / {

try\_files $uri @apache

}

loaction @apache{

proxy\_pass http://127.0.0.1:88

include aproxy.conf

}

[复制代码](javascript:void(0);)

try\_files方法让Ngxin尝试访问后面得$uri链接，并进根据@apache配置进行内部重定向。

当然try\_files也可以以错误代码赋值，如try\_files /index.php = 404 @apache，则表示当尝试访问得文件返回404时，根据@apache配置项进行重定向。

1.try\_files的语法规则：

　　　　格式1：**try\_files** file ... uri;  格式2：**try\_files** file ... =code;

　　　　可应用的上下文：server，location段

2.try\_files的语法解释：(先贴出官方的解释，楼主再解释下)

Checks the existence of files in the specified order and uses the first found file for request processing; the processing is performed in the current context. The path to a file is constructed from the fileparameter according to the [root](http://nginx.org/en/docs/http/ngx_http_core_module.html#root) and [alias](http://nginx.org/en/docs/http/ngx_http_core_module.html#alias) directives. It is possible to check directory’s existence by specifying a slash at the end of a name, e.g. “$uri/”. If none of the files were found, an internal redirect to the uri specified in the last parameter is made.

　　关键点1：按指定的file顺序查找存在的文件，并使用第一个找到的文件进行请求处理

　　关键点2：查找路径是按照给定的root或alias为根路径来查找的

　　关键点3：如果给出的file都没有匹配到，则重新请求最后一个参数给定的uri，就是新的location匹配

　　关键点4：如果是格式2，如果最后一个参数是 = 404 ，若给出的file都没有匹配到，则最后返回404的响应码

 3.举例说明：

location /images/ {  
 root /opt/html/;

try\_files $uri $uri/ /images/default.gif;   
}  
比如 请求 127.0.0.1/images/test.gif 会依次查找 1.文件/opt/html/images/test.gif 2.文件夹 /opt/html/images/test.gif/下的index文件 3. 请求127.0.0.1/images/default.gif  
  
4.其他注意事项  
1.try-files 如果不写上 $uri/，当直接访问一个目录路径时，并不会去匹配目录下的索引页 即 访问127.0.0.1/images/ 不会去访问 127.0.0.1/images/index.html

三.

其他用法：

location / {

try\_files /system/maintenance.html

$uri $uri/index.html $uri.html

@mongrel;

}

location @mongrel {

proxy\_pass http://mongrel;

}

以上中若未找到给定顺序的文件，则将会交给location @mongrel处理（相当于匹配到了@mongrel来匹配）

tryfiles参考文档：

<https://www.cnblogs.com/jedi1995/p/10900224.html>

index index.html; 默认路径

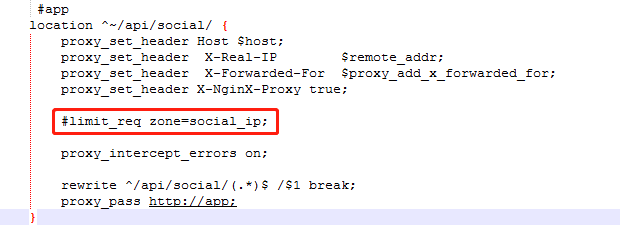
显示声明一个default server listen 80 default\_server;

当域名或则ip没有匹配所有的server的请求，用这个default server来处理

如果没有声明 default server。配置文件第一个声明的server 即为default server



**可能是因为nginx并没有支持 http\_limit\_conn\_zone模块**



停止nginx

#nginx -s quit

#nginx -s stop

注：stop是快速停止nginx，可能并不保存相关信息；quit是完整有序的停止nginx，并保存相关信息。

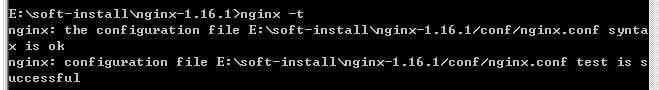
启动nginx

#nginx

#start nginx 在windows环境下不行，linux环境待验证

修改配置文件，检测语法

#nginx -t



修改配置文件重启

#nginx -s reload

查看nginx版本

#nginx -v

查看详细信息

#nginx -V

重新打开日志文件

#nginx -s reopen

ps 表头信息

ps -ef | head -1;ps -ef | grep nginx

find / -name "nginx.conf"

find / -name "nginx.conf"

windows下面查看进程

tasklist | findstr nginx

windows下面查看端口信息

netstat -ano | findstr 8080

windows 下面杀掉进程

tasklist /PID 进程号 -F -T

nginx配置限制每个ip连接数不能超过多少，在整个http，某些server，location下面

limit\_conn\_zone $binary\_remote\_addr zone=addr:10m;

server {

location /download/ {

limit\_conn addr 1;

}

limit\_conn\_zone $binary\_remote\_addr zone=perip:10m;

limit\_conn\_zone $server\_name zone=perserver:10m;

server {

...

limit\_conn perip 10;

limit\_conn perserver 100;

}

限制每个ip 请求的数量

limit\_req\_zone $binary\_remote\_addr zone=one:10m rate=1r/s;

server {

location /search/ {

limit\_req zone=one burst=5;

}

limit\_req\_zone $binary\_remote\_addr zone=perip:10m rate=1r/s;

limit\_req\_zone $server\_name zone=perserver:10m rate=10r/s;

server {

...

limit\_req zone=perip burst=5 nodelay;

limit\_req zone=perserver burst=10;

}

#白名单外的IP默认值为1，指定公司的IP的值为0，1都会受限制，map指令是将$limited值为1的，也就是受限制的IP，映射为客户端IP,将值为0的，也就是白名单IP，映射为空的字符串

geo $limited {

default 1;

122.224.234.0/24 0;

}

map $limited $limit {

1 $binary\_remote\_addr;

0 "";

}

#针对用户IP做限制,limit\_conn\_zone和limit\_req\_zone指令对于键为空值的将会被忽略，从而实现对于列出来的IP不做限制

#每个IP并发连接不超过200个

limit\_conn\_zone $limit zone=TotalConnLimitZone:10m;

limit\_conn TotalConnLimitZone 200;

#每个IP每秒处理30个请求

limit\_req\_zone $limit zone=ConnLimitZone:10m rate=30r/s;

nodelay： 限制每秒处理一个请求，如果同意时间来了多个请求，这种情况，多个请求会被处理，而忽略每秒只处理一个请求的限制。

directive error\_page

Syntax: error\_log file [level];

Default: error\_log logs/error.log error;

Context: main, http, mail, stream, server, location

debug, info, notice, warn, error, crit, alert, emerg

stderr——standard error file

系统日志syntax

error\_log syslog:server=192.168.1.1 debug;

access\_log syslog:server=unix:/var/log/nginx.sock,nohostname;

access\_log syslog:server=[2001:db8::1]:12345,facility=local7,tag=nginx,severity=info combined;

缓存日志syntax

error\_log memory:32m debug;