**正向代理**

正向代理类似一个跳板机，代理访问外部资源。比如我是一个用户，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。

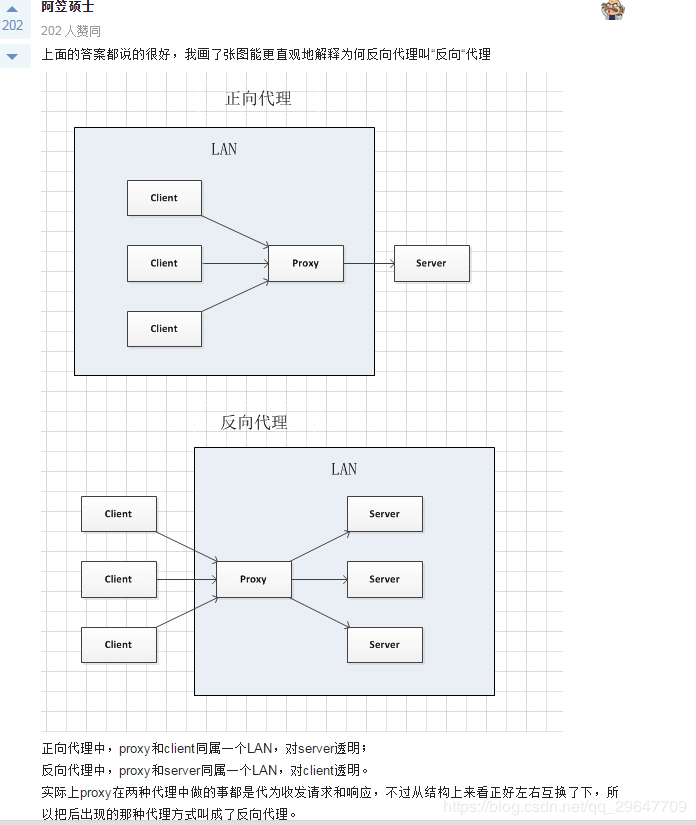
**反向代理**

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：

（1）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。

（2）负载均衡，通过反向代理服务器来优化网站的负载



正向代理就是我知道要干嘛，知道通过代理服务器去访问谁，反向代理他不知道目标服务器是哪个

Nginx 重定向

hosts文件新增：

127.0.0.1 8080.itmayiedu.com

127.0.0.1 8081.itmayiedu.com

Nginx.conf

server {

listen 80;

server\_name 8080.itmayiedu.com;

location / {

proxy\_pass http://127.0.0.1:8080;

index index.html index.htm;

}

}

server {

listen 80;

server\_name 8081.itmayiedu.com;

location / {

proxy\_pass http://127.0.0.1:8081;

index index.html index.htm;

}

}

**Index指令详解**：<https://blog.csdn.net/qq_32331073/article/details/81945134>

基本内容（中文文档和官方文档都可见）：

该指令后面可以跟多个文件，用空格隔开；

如果包括多个文件，Nginx会根据文件的枚举顺序来检查，直到查找的文件存在；

文件可以是相对路径也可以是绝对路径，绝对路径需要放在最后；

文件可以使用变量$来命名；

index index.$geo.html index.0.html /index.html;

1该指令拥有默认值，index index.html ，即，如果没有给出index，默认初始页为index.html

核心内容（中文文档没有或一笔带过，而官方文档作详细解释）：

Nginx给了三种方式来选择初始页，三种方式按照顺序来执行：ngx\_http\_random\_index\_module 模块，从给定的目录中随机选择一个文件作为初始页，而且这个动作发生在 ngx\_http\_index\_module 之前，注意：这个模块默认情况下没有被安装，需要在安装时提供配置参数 --with-http\_random\_index\_module；

ngx\_http\_index\_module 模块，根据index指令规则来选择初始页；

ngx\_http\_autoindex\_module 模块，可以使用指定方式，根据给定目录中的文件列表自动生成初始页，这个动作发生在 ngx\_http\_index\_module之后，即只有通过index指令无法确认初始页，此时启用后的自动生成模块才会被使用。

切记，index指令并不是查到文件之后，就直接拿来用了。它的实际工作方式是：

如果文件存在，则使用文件作为路径，发起内部重定向。直观上看上去就像再一次从客户端发起请求，Nginx再一次搜索location一样。

既然是内部重定向，域名+端口不发生变化，所以只会在同一个server下搜索。

同样，如果内部重定向发生在proxy\_pass反向代理后，那么重定向只会发生在代理配置中的同一个server。

**负载均衡策略**

1、 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;

}

2、指定权重

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=10;

server 192.168.0.15 weight=10;

}

3、IP绑定 ip\_hash

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。

upstream backserver {

ip\_hash;

server 192.168.0.14:88;

server 192.168.0.15:80;

}

4、fair（第三方）

按后端服务器的响应时间来分配请求，响应时间短的优先分配。

upstream backserver {

server server1;

server server2;

fair;

}

5、url\_hash（第三方）

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;

}

使用举例：

server {

listen 80;

server\_name 8080.chai.com;

location / {

proxy\_pass http://backserver;

index index.html index.htm;

}

}

upstream backserver {

server 127.0.0.1:8080 weight=5;

server 127.0.0.1:8081 weight=20;

}

宕机轮询配置规则

server {

listen 80;

server\_name 8080.chai.com;

location / {

proxy\_pass http://backserver;

index index.html index.htm;

proxy\_connect\_timeout 1;

proxy\_send\_timeout 1;

proxy\_read\_timeout 1;

}

}

解决网站跨越问题

server {

listen 80;

server\_name 8080.chai.com;

location /activiti-explorer{

proxy\_pass http://127.0.0.1:8080/activiti-explorer;

index index.html index.htm;

}

location /B {

proxy\_pass http://127.0.0.1:8081/B;

index index.html index.htm;

}

}

配置防盗链 ？？？

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

valid\_referers blocked http://www.itmayiedu.com www.itmayiedu.com;

if ($invalid\_referer) {

return 403;

}

}

限制请求次数

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

server {

listen 80;

server\_name 8080.chai.com;

location / {

limit\_req zone=cs burst=3 nodelay;

proxy\_pass http://127.0.0.1:8081/;

index index.html index.htm;

}

}

`limit\_req\_zone`命令设置了一个叫one的共享内存区来存储请求状态的特定键值，在上面的例子中是客户端IP($binary\_remote\_addr)。location块中的`limit\_req`通过引用one共享内存区来实现限制访问/login.html的目的

<https://blog.csdn.net/keketrtr/article/details/75315330>

