**[Nginx正反向代理、负载均衡等功能实现配置](http://blog.csdn.net/hiyun9/article/details/51602428)**

**系统环境：**

VirtualBox Manager

Centos6.4

nginx1.10.0

IP对应的机器名：

      IP                机器名               角色名

10.0.0.139      [elk]                     client

10.0.0.136      [lvs-master]        nginx server

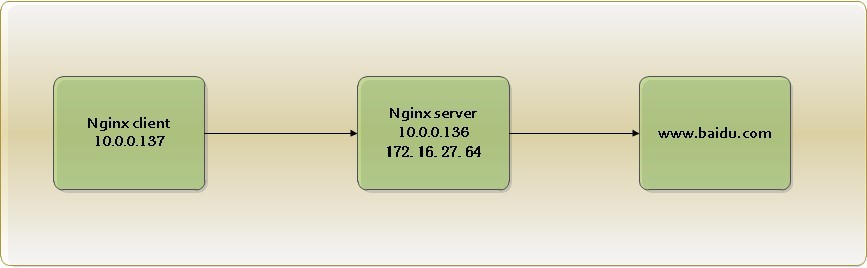
10.0.0.137      [kvm]                   web server 1

10.0.0.111      [lvs-backup]        web server 2

**一、正向代理**

概念这里不在介绍，可以参考此文http://my.oschina[**.NET**](http://lib.csdn.net/base/dotnet)/yoyoko/blog/147414。

**1.1 环境介绍**



**1.2 配置介绍**

Nginx server:(内网地址：10.0.0.136，外网地址：172.16.27.64)

使用VirtualBox Manager虚拟出双网卡。

**[plain]** [view plain](http://blog.csdn.net/hiyun9/article/details/51602428) [copy](http://blog.csdn.net/hiyun9/article/details/51602428)

[root@lvs-master conf.d]# ifconfig

eth0      Link encap:Ethernet  HWaddr 08:00:27:30:56:99

          inet addr:10.0.0.136  Bcast:10.255.255.255  Mask:255.0.0.0

          inet6 addr: fe80::a00:27ff:fe30:5699/64 Scope:Link

          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1

          RX packets:891978 errors:0 dropped:0 overruns:0 frame:0

          TX packets:9509 errors:0 dropped:0 overruns:0 carrier:0

          collisions:0 txqueuelen:1000

          RX bytes:81841095 (78.0 MiB)  TX bytes:13339058 (12.7 MiB)

eth1      Link encap:Ethernet  HWaddr 08:00:27:55:4C:72

          inet addr:172.16.27.64  Bcast:172.16.27.255  Mask:255.255.255.0

          inet6 addr: fe80::a00:27ff:fe55:4c72/64 Scope:Link

          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1

          RX packets:913671 errors:0 dropped:0 overruns:0 frame:0

          TX packets:22712 errors:0 dropped:0 overruns:0 carrier:0

          collisions:0 txqueuelen:1000

          RX bytes:109369858 (104.3 MiB)  TX bytes:1903855 (1.8 MiB)

lo        Link encap:Local Loopback

          inet addr:127.0.0.1  Mask:255.0.0.0

          inet6 addr: ::1/128 Scope:Host

          UP LOOPBACK RUNNING  MTU:16436  Metric:1

          RX packets:36222 errors:0 dropped:0 overruns:0 frame:0

          TX packets:36222 errors:0 dropped:0 overruns:0 carrier:0

          collisions:0 txqueuelen:0

          RX bytes:3899937 (3.7 MiB)  TX bytes:3899937 (3.7 MiB)

**[plain]** [view plain](http://blog.csdn.net/hiyun9/article/details/51602428) [copy](http://blog.csdn.net/hiyun9/article/details/51602428)

[root@lvs-master conf.d]# cat zxproxy.conf

server {

    listen       80;   #监听的端口

    server\_name  10.0.0.136;  #server的内容地址，与client需要网络互通

    resolver 172.16.5.1;   #DNS，这个是DNS，访问外网

    location / {

           proxy\_pass http://$http\_host$request\_uri;   #$http\_host和$request\_uri是nginx系统变量，不需要替换，保持原样

                }

Nginx client:

只有一个内网网卡，通过访问Nginx server去访问internet，其实翻墙、肉鸡、之类的俗称就是这个原理。

**[plain]** [view plain](http://blog.csdn.net/hiyun9/article/details/51602428) [copy](http://blog.csdn.net/hiyun9/article/details/51602428)

[root@kvm ~]# ifconfig

eth0      Link encap:Ethernet  HWaddr 08:00:27:72:8C:3B

          inet addr:10.0.0.137  Bcast:10.255.255.255  Mask:255.0.0.0

          inet6 addr: fe80::a00:27ff:fe72:8c3b/64 Scope:Link

          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1

          RX packets:1462448 errors:0 dropped:0 overruns:0 frame:0

          TX packets:21130 errors:0 dropped:0 overruns:0 carrier:0

          collisions:0 txqueuelen:1000

          RX bytes:145119904 (138.3 MiB)  TX bytes:2814635 (2.6 MiB)

lo        Link encap:Local Loopback

          inet addr:127.0.0.1  Mask:255.0.0.0

          inet6 addr: ::1/128 Scope:Host

          UP LOOPBACK RUNNING  MTU:16436  Metric:1

          RX packets:60800 errors:0 dropped:0 overruns:0 frame:0

          TX packets:60800 errors:0 dropped:0 overruns:0 carrier:0

          collisions:0 txqueuelen:0

          RX bytes:4831102 (4.6 MiB)  TX bytes:4831102 (4.6 MiB)

[root@kvm ~]# wget www.baidu.com

--2016-06-08 13:02:08--  http://www.baidu.com/

正在解析主机 www.baidu.com... 失败：域名解析暂时失败。     #无法访问百度

wget: 无法解析主机地址 “www.baidu.com”

[root@kvm ~]# export http\_proxy=http://10.0.0.136:80 #设定环境变量，指定代理服务器的ip及端口

[root@kvm ~]# wget www.baidu.com      #可以成功访问百度了

--2016-06-08 13:08:15--  http://www.baidu.com/

正在连接 10.0.0.136:80... 已连接。

已发出 Proxy 请求，正在等待回应... 200 OK

长度：未指定 [text/html]

正在保存至: “index.html.1”

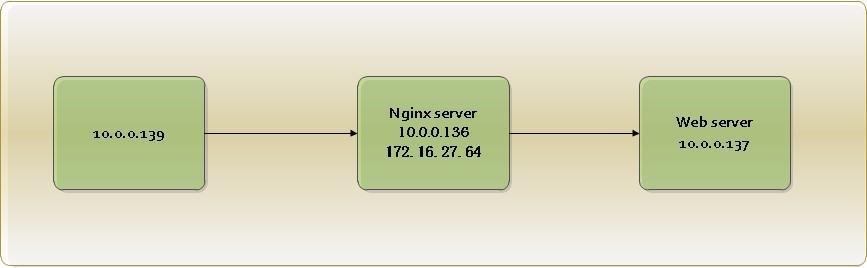
    [ <=>                                                  ] 99,762      --.-K/s   in 0.07s

2016-06-08 13:08:16 (1.36 MB/s) - “index.html.1” 已保存 [99762]

**二、反向代理**

介绍文章同正向代理

**2.1 环境介绍**



1.下面来看下**[测试](http://lib.csdn.net/base/softwaretest" \o "软件测试知识库" \t "_blank)**页面:

**[python]** [view plain](http://blog.csdn.net/hiyun9/article/details/51602428) [copy](http://blog.csdn.net/hiyun9/article/details/51602428)

[root@kvm ~]# yum install httpd

[root@kvm ~]# echo "<html>10.0.0.137</html>" > /var/www/html/index.html

[root@lvs-backup ~]# yum install httpd

[root@lvs-backup~]# echo "<html>10.0.0.111</html>" > /var/www/html/index.html

2.看下效果：

**[plain]** [view plain](http://blog.csdn.net/hiyun9/article/details/51602428) [copy](http://blog.csdn.net/hiyun9/article/details/51602428)

[root@lvs-backup html]# curl 10.0.0.111

<html>

10.0.0.111

</html>

[root@lvs-backup html]# curl 10.0.0.137

<html>

10.0.0.137

</html>

##都成功了，我们进行下一步。

**2.2 配置介绍**

**[plain]** [view plain](http://blog.csdn.net/hiyun9/article/details/51602428) [copy](http://blog.csdn.net/hiyun9/article/details/51602428)

1. [root@lvs-master conf.d]# ls         #nginx目录下的配置文件
2. zxproxy.conf
3. [root@lvs-master conf.d]# cp zxproxy.conf fxproxy.conf  #复制一份，之前是正向代理，现在是反向代理
4. [root@lvs-master conf.d]# mv zxproxy.conf zxproxy.conf.bak

**[python]** [view plain](http://blog.csdn.net/hiyun9/article/details/51602428) [copy](http://blog.csdn.net/hiyun9/article/details/51602428)

[root@lvs-master conf.d]# cat fxproxy.conf

server {

    listen       80;

    server\_name  10.0.0.136; #根据环境介绍，nginx server ip

    location / {

           proxy\_pass http://10.0.0.137; #被代理的服务器ip

                }

#proxy\_pass: proxy\_pass URL

#默认值：NO

#使用字段：location，location中的if字段

#这个参数设置被代理服务器的地址和被映射的URL，地址可以使主机名、域名、IP加端口的模式，如：

#proxy\_pass    http://192.168.1.6:8099/linuxtone/;

[root@lvs-master conf.d]# service nginx restart #重启加载配置

看下结果：

**[plain]** [view plain](http://blog.csdn.net/hiyun9/article/details/51602428) [copy](http://blog.csdn.net/hiyun9/article/details/51602428)

#先登录到实验环境中的clinet机上，ip如下：

[root@elk ~]# ifconfig

eth0      Link encap:Ethernet  HWaddr 08:00:27:3D:40:40

          inet addr:10.0.0.139  Bcast:10.255.255.255  Mask:255.0.0.0

          inet6 addr: fe80::a00:27ff:fe3d:4040/64 Scope:Link

          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1

          RX packets:2618345 errors:0 dropped:0 overruns:0 frame:0

          TX packets:247926 errors:0 dropped:0 overruns:0 carrier:0

          collisions:0 txqueuelen:1000

          RX bytes:336182790 (320.6 MiB)  TX bytes:35145157 (33.5 MiB)

lo        Link encap:Local Loopback

          inet addr:127.0.0.1  Mask:255.0.0.0

          inet6 addr: ::1/128 Scope:Host

          UP LOOPBACK RUNNING  MTU:16436  Metric:1

          RX packets:177352 errors:0 dropped:0 overruns:0 frame:0

          TX packets:177352 errors:0 dropped:0 overruns:0 carrier:0

          collisions:0 txqueuelen:0

          RX bytes:26547640 (25.3 MiB)  TX bytes:26547640 (25.3 MiB)

[root@elk ~]# curl 10.0.0.136    #访问反向代理服务器

<html>

10.0.0.137

</html>

#我们看到访问代理服务器，结果被转发到了web server1上。

#接下来我们分别看下nginx-server和web-server1的日志：

nginx-server：

[root@lvs-master ~]# tail /var/log/nginx/access.log

10.0.0.139- - [08/Jun/2016:15:35:43 +0800] "GET / HTTP/1.1" 200 26 "-" "curl/7.19.7

(x86\_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.19.1 Basic ECC zlib/1.2.3 libidn/1.18 libssh2/1.4.2" "-"

web-server：

[root@kvm httpd]# tail /var/log/httpd/access\_log

10.0.0.136 - - [08/Jun/2016:15:21:12 +0800] "GET / HTTP/1.0" 200 26 "-" "curl/7.19.7

(x86\_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.19.1 Basic ECC zlib/1.2.3 libidn/1.18 libssh2/1.4.2"

##我们看到nginx-server上的nginx的日志，显示访问的用户是10.0.0.139也就是我们环境的clinet，

#而web-server上显示的ip是10.0.0.136，也就是nginx-server。

#说白了反向代理，对客户来说nginx-server就是真正的服务器，实际上，当用户访问nginx-server的时候，会将请求转发到

#web-server1上，然后web-server1将请求的结果发给nginx-server，然后由ngin小-server将请求的结果转交给用户。

#在web-server上看到的都是代理的ip，能不能也看到真实用户的ip呢？

[root@lvs-master conf.d]# cat fxproxy.conf

server {

    listen       80;

    server\_name  10.0.0.136;                            #根据环境介绍，nginx server ip

    location / {

           proxy\_pass http://10.0.0.137;                #被代理的服务器ip

           proxy\_set\_header  X-Real-IP  $remote\_addr;    #多了这行

                 }

**[plain]** [view plain](http://blog.csdn.net/hiyun9/article/details/51602428) [copy](http://blog.csdn.net/hiyun9/article/details/51602428)

[root@lvs-master conf.d]# service nginx restart

[root@kvm ~]# tail /var/log/httpd/access\_log

10.0.0.136 - - [08/Jun/2016:16:10:53 +0800] "GET / HTTP/1.0" 200 26 "-" "curl/7.19.7

(x86\_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.19.1 Basic ECC zlib/1.2.3 libidn/1.18 libssh2/1.4.2"

#改了之后还是显示的是代理服务器的ip,我们去web-server上修改下配置

[root@kvm ~]# vim /etc/httpd/conf/httpd.conf

LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined

LogFormat "%h %l %u %t \"%r\" %>s %b" common

LogFormat "%{Referer}i -> %U" referer

LogFormat "%{User-agent}i" agent

#修改为：(%h指的的访问的主机，现在改为访问的真实主机ip)

LogFormat "%{X-Real-IP}i</span> %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined

LogFormat "%h %l %u %t \"%r\" %>s %b" common

LogFormat "%{Referer}i -> %U" referer

LogFormat "%{User-agent}i" agent</span>

**[plain]** [view plain](http://blog.csdn.net/hiyun9/article/details/51602428) [copy](http://blog.csdn.net/hiyun9/article/details/51602428)

[root@kvm ~]# service httpd restart

停止 httpd：                                               [确定]

正在启动 httpd：                                           [确定]

[root@kvm ~]# tail /var/log/httpd/access\_log

10.0.0.136 - - [08/Jun/2016:16:10:53 +0800] "GET / HTTP/1.0" 200 26 "-" "curl/7.19.7

(x86\_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.19.1 Basic ECC zlib/1.2.3 libidn/1.18 libssh2/1.4.2"

<span style="color:#FF0000;">10.0.0.139</span> - - [08/Jun/2016:16:16:01 +0800] "GET / HTTP/1.0" 200 26 "-" "curl/7.19.7

(x86\_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.19.1 Basic ECC zlib/1.2.3 libidn/1.18 libssh2/1.4.2"

#已经变成了真实的访问地址

代理多个web服务器：

**[plain]** [view plain](http://blog.csdn.net/hiyun9/article/details/51602428) [copy](http://blog.csdn.net/hiyun9/article/details/51602428)

[root@lvs-master conf.d]# cat fxproxy.conf

server {

    listen       80;

    server\_name  10.0.0.136;

    location / {

           proxy\_pass http://10.0.0.137;

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

                }

    location /web2 {                            #多加个location

           proxy\_pass http://10.0.0.111;

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

                }

[root@lvs-backup ~]# cd /var/www/html/   #进入10.0.0.111这个web-server2

[root@lvs-backup html]# mkdir web

[root@lvs-backup html]# echo "<html>10.0.0.111</html>" > index.html

#我们去client上访问试试：

[root@elk ~]# curl 10.0.0.136/web2/

<html>

10.0.0.111

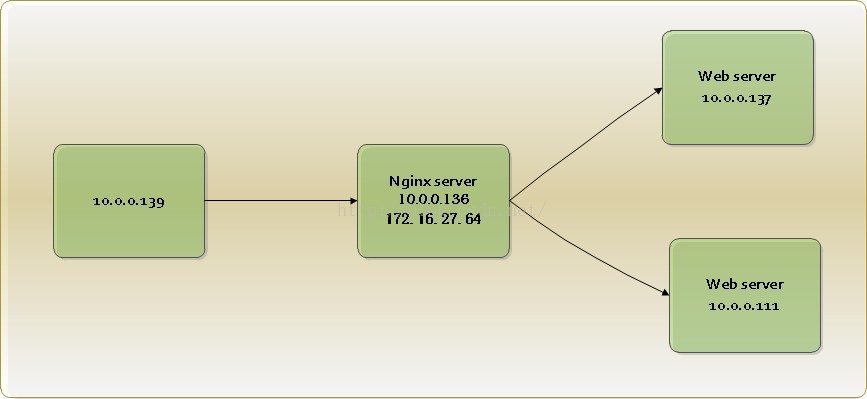
</html>

#访问成功

**三、负载均衡**

负载均衡实现的方式有很多，常用的lvs四层负载均衡，nginx是七层负载均衡，可以网上查询相关资料。

**3.1 环境介绍**



**3.2 配置介绍**

1.upstream是Nginx的HTTP Upstream模块，这个模块通过一个简单的调度算法来实现客户端IP到后端服务器的负载均衡。在上面的设定中，通过upstream指令指定了一个负载均衡器的名称1.2.3.4。这个名称可以任意指定，在后面需要用到的地方直接调用即可。

2.Nginx的负载均衡模块目前支持4种调度**[算法](http://lib.csdn.net/base/datastructure" \o "算法与数据结构知识库" \t "_blank)**，下面进行分别介绍，其中后两项属于第三方调度算法。

* 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，如果后端某台服务器宕机，故障系统被自动剔除，使用户访问不受影响。Weight 指定轮询权值，Weight值越大，分配到的访问机率越高，主要用于后端每个服务器性能不均的情况下。
* ip\_hash。每个请求按访问IP的hash结果分配，这样来自同一个IP的访客固定访问一个后端服务器，有效解决了动态网页存在的session共享问题。
* fair。这是比上面两个更加智能的负载均 衡算法。此种算法可以依据页面大小和加载时间长短智能地进行负载均衡，也就是根据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。Nginx本身 是不支持fair的，如果需要使用这种调度算法，必须下载Nginx的upstream\_fair模块。
* url\_hash。此方法按访问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，可以进一步提高后端缓存服务器的效率。Nginx本身是不支持url\_hash的，如果需要使用这种调度算法，必须安装Nginx 的hash软件包。

3.upstream 支持的状态参数

在HTTP Upstream模块中，可以通过server指令指定后端服务器的IP地址和端口，同时还可以设定每个后端服务器在负载均衡调度中的状态。常用的状态有：

* down，表示当前的server暂时不参与负载均衡。
* backup，预留的备份机器。当其他所有的非backup机器出现故障或者忙的时候，才会请求backup机器，因此这台机器的压力最轻。
* max\_fails，允许请求失败的次数，默认为1。当超过最大次数时，返回proxy\_next\_upstream 模块定义的错误。
* fail\_timeout，在经历了max\_fails次失败后，暂停服务的时间。max\_fails可以和fail\_timeout一起使用。

注，当负载调度算法为ip\_hash时，后端服务器在负载均衡调度中的状态不能是weight和backup。

我们来看下具体配置：

**[python]** [view plain](http://blog.csdn.net/hiyun9/article/details/51602428) [copy](http://blog.csdn.net/hiyun9/article/details/51602428)

[root@lvs-master conf.d]# cat ../nginx.conf

http {

    include       /etc/nginx/mime.types;

    default\_type  application/octet-stream;

    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  /var/log/nginx/access.log  main;

    sendfile        on;

    #tcp\_nopush     on;

    keepalive\_timeout  65;

    #gzip  on;

upstream 1.2.3.4 {

    server 10.0.0.111:80;

    server 10.0.0.137:80;

    }

    include /etc/nginx/conf.d/\*.conf;

}

[root@lvs-master conf.d]# cat slb.confserver

{

location / {

     proxy\_pass http://1.2.3.4; proxy\_set\_header X-Real-IP $remote\_addr;

           }

#注，upstream是定义在server{ }之外的，不能定义在server{ }内部。定义好upstream之后，用proxy\_pass引用一下即可。

4.测试结果

**[python]** [view plain](http://blog.csdn.net/hiyun9/article/details/51602428) [copy](http://blog.csdn.net/hiyun9/article/details/51602428)

[root@elk ~]# curl 10.0.0.136

<html>

10.0.0.111

</html>

[root@elk ~]# curl 10.0.0.136

<html>

10.0.0.137

</html>

[root@elk ~]# curl 10.0.0.136

<html>

10.0.0.111

</html>

#结果是server1,2交替出现，说明默认是轮询方式的负载均衡。

5.健康检查

一般健康检查都需要搞个keepalived，但nginx也有相应的参数可以设置。

* max\_fails，允许请求失败的次数，默认为1。当超过最大次数时，返回proxy\_next\_upstream 模块定义的错误。
* fail\_timeout，在经历了max\_fails次失败后，暂停服务的时间。max\_fails可以和fail\_timeout一起使用，进行健康状态检查。

**[python]** [view plain](http://blog.csdn.net/hiyun9/article/details/51602428) [copy](http://blog.csdn.net/hiyun9/article/details/51602428)

[root@lvs-master conf.d]# cat ../nginx.conf

http {

    include       /etc/nginx/mime.types;

    default\_type  application/octet-stream;

    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  /var/log/nginx/access.log  main;

    sendfile        on;

    #tcp\_nopush     on;

    keepalive\_timeout  65;

    #gzip  on;

    upstream 1.2.3.4 {

    server 10.0.0.111:80 weight=1 max\_fails=2 fail\_timeout=2;

    server 10.0.0.137:80 weight=1 max\_fails=2 fail\_timeout=2;

    }

    include /etc/nginx/conf.d/\*.conf;

    }

[root@lvs-master conf.d]# service nginx restart

6.测试下结果

**[python]** [view plain](http://blog.csdn.net/hiyun9/article/details/51602428) [copy](http://blog.csdn.net/hiyun9/article/details/51602428)

[root@kvm httpd]# service httpd stop  #关闭web-server1服务

[root@elk ~]# curl 10.0.0.136

<html>

10.0.0.111

</html>

[root@elk ~]# curl 10.0.0.136

<html>

10.0.0.111

</html>

#现在只能访问web-server2了。

[root@kvm httpd]# service httpd start   #打开web-server1服务

[root@elk ~]# curl 10.0.0.136

<html>

10.0.0.111

</html>

[root@elk ~]# curl 10.0.0.136

<html>

10.0.0.137

</html>

[root@elk ~]# curl 10.0.0.136

<html>

10.0.0.111

</html>

7.ip\_hash的负载均衡

**[python]** [view plain](http://blog.csdn.net/hiyun9/article/details/51602428) [copy](http://blog.csdn.net/hiyun9/article/details/51602428)

[root@lvs-master conf.d]# cat ../nginx.conf

upstream 1.2.3.4 {

    ip\_hash;

    server 10.0.0.111:80 weight=1 max\_fails=2 fail\_timeout=2;

    server 10.0.0.137:80 weight=1 max\_fails=2 fail\_timeout=2;

    }

[root@lvs-master conf.d]# service nginx restart

停止 nginx：                                               [确定]

正在启动 nginx：                                           [确定]

[root@elk ~]# curl 10.0.0.136

<html>

10.0.0.137

</html>

[root@elk ~]# curl 10.0.0.136

<html>

10.0.0.137

</html>

#配置这种负载均衡后，>每个请求按访问IP的hash结果分配，这样来自同一个IP的访客固定访问一个后端服务器，

#有效解决了动态网页存在的session共享问题。（一般电子商务网站用的比较多）