**HAproxy 做负载均衡实验**

**一．实验**

1.1．实验服务器

210.21.48.69 图片服务器1

210.21.48.76 图片服务器2

210.21.48.75 为haproxy服务器。

1.2 实验目的

在启动3台服务器以后，使用haproxy做7层的负载均衡，并验证均衡算法的正确性。算法1：动态轮询(roundrobin)。 算法二：最少连接（lc）

二．实验步骤：

2.1 在76服务器上使用httpd搭建图片服务器。

0 yum install httpd

1002 whereis httpd

1003 cd /etc/httpd/conf

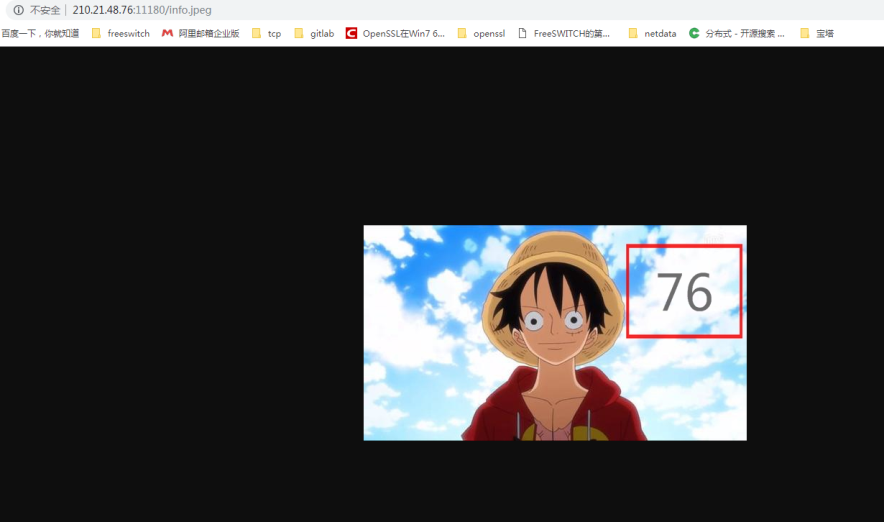
1009 vi httpd.conf

1015 setenforce 0

1016 systemctl start httpd

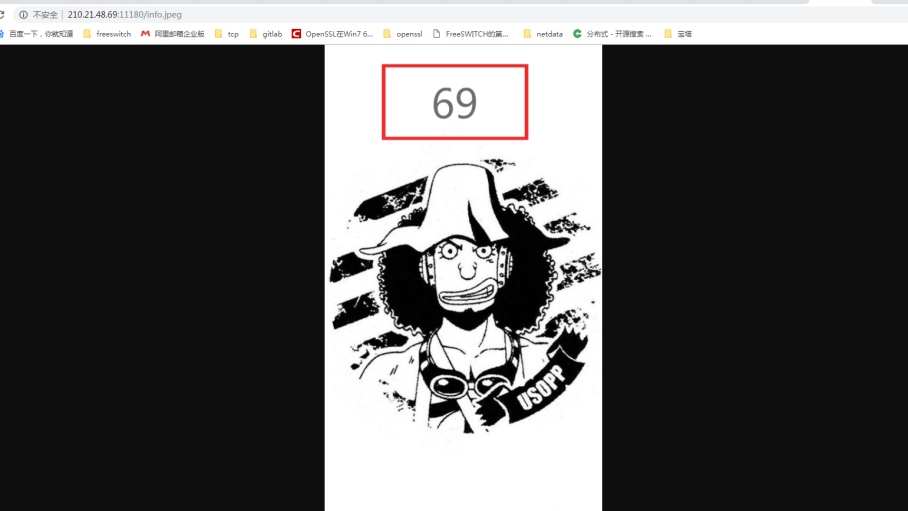
1018 netstat -anput|grep 11180

配置完成以后，在浏览器访问如下显示：http://210.21.48.76:11180/info.jpeg



2.2 在69服务器上完成httpd搭建图片服务器。

配置完成以后，在浏览器访问如下显示：http://210.21.48.69:11180/info.jpeg

****

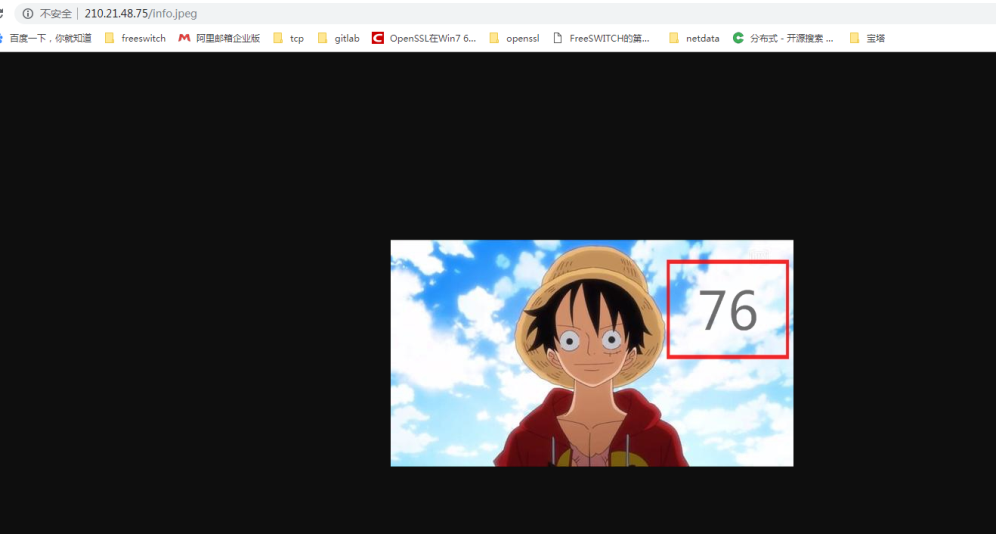
2.3 在75上配置haproxy。

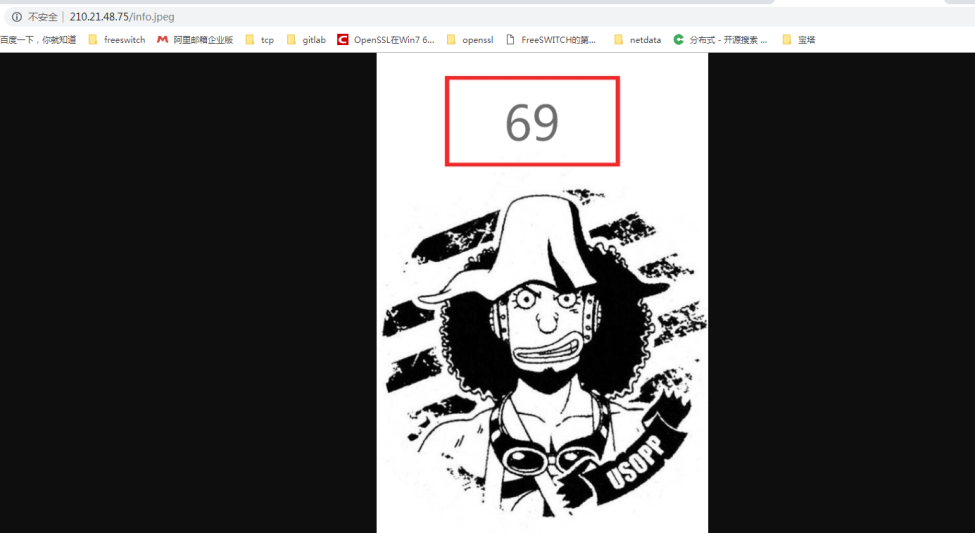


三．实验结果

3.1采用roundrobin 方式进行服务配置

在访问 <http://210.21.48.75/info.jpeg> 访问资源时，如果第一次访问了76服务器，f5s刷新就会访问69，从而实现了不同用户访问同一个地址，实际访问了不同的web服务器。达到负载均衡的效果。





3.2

四．haproxy配置字段详解

global # 全局参数的设置

log 127.0.0.1 local2 # log语法：log <address\_1>[max\_level\_1] # 全局的日志配置，使用log关键字，

指定使用127.0.0.1

上的syslog服务中的local0日志设备，记录日志等级为info的日志

chroot /var/lib/haproxy #改变当前工作目录

pidfile /var/run/haproxy.pid #当前进程id文件

maxconn 4000 #最大连接数

user haproxy #所属用户

group haproxy #所属组

daemon #以守护进程方式运行haproxy

stats socket /var/lib/haproxy/stats

defaults

mode http #默认的模式mode { tcp|http|health }，tcp是4层，http是7层，health只会返回OK

log global #应用全局的日志配置

option httplog # 启用日志记录HTTP请求，默认haproxy日志记录是不记录HTTP请求日志

option dontlognull # 启用该项，日志中将不会记录空连接。所谓空连接就是在上游的负载均衡器

或者监控系统为了探测该 服务是否存活可用时，需要定期的连接或者获取某

一固定的组件或页面，或者探测扫描端口是否在监听或开放等动作被称为空连接；

官方文档中标注，如果该服务上游没有其他的负载均衡器的话，建议不要使用

该参数，因为互联网上的恶意扫描或其他动作就不会被记录下来

option http-server-close #每次请求完毕后主动关闭http通道

option forwardfor except 127.0.0.0/8 #如果服务器上的应用程序想记录发起请求的客户端的IP地址，需要在HAProxy

上 配置此选项， 这样 HAProxy会把客户端的IP信息发送给服务器，在HTTP

请求中添加"X-Forwarded-For"字段。 启用 X-Forwarded-For，在requests

头部插入客户端IP发送给后端的server，使后端server获取到客户端的真实IP。

option redispatch # 当使用了cookie时，haproxy将会将其请求的后端服务器的serverID插入到

cookie中，以保证会话的SESSION持久性；而此时，如果后端的服务器宕掉

了， 但是客户端的cookie是不会刷新的，如果设置此参数，将会将客户的请

求强制定向到另外一个后端server上，以保证服务的正常。

retries 3 # 定义连接后端服务器的失败重连次数，连接失败次数超过此值后将会将对应后端

服务器标记为不可用

timeout http-request 10s #http请求超时时间

timeout queue 1m #一个请求在队列里的超时时间

timeout connect 10s #连接超时

timeout client 1m #客户端超时

timeout server 1m #服务器端超时

timeout http-keep-alive 10s #设置http-keep-alive的超时时间

timeout check 10s #检测超时

maxconn 3000 #每个进程可用的最大连接数

frontend main \*:80 #监听地址为80

acl url\_static path\_beg -i /static /images /javascript /stylesheets

acl url\_static path\_end -i .jpg .gif .png .css .js

use\_backend static if url\_static

default\_backend my\_webserver #定义一个名为my\_app前端部分。此处将对于的请求转发给后端

backend static #使用了静态动态分离（如果url\_path匹配 .jpg .gif .png .css .js静态文件则

访问此后端）

balance roundrobin #负载均衡算法（#banlance roundrobin 轮询，balance source 保存session值，

支持static-rr，leastconn，first，uri等参数）

server static 127.0.0.1:80 check #静态文件部署在本机（也可以部署在其他机器或者squid缓存服务器）

backend my\_webserver #定义一个名为my\_webserver后端部分。PS：此处my\_webserver只是一个

自定义名字而已，但是需要与frontend里面配置项default\_backend 值相一致

balance roundrobin #负载均衡算法

server web01 172.31.2.33:80 check inter 2000 fall 3 weight 30 #定义的多个后端

server web02 172.31.2.34:80 check inter 2000 fall 3 weight 30 #定义的多个后端

server web03 172.31.2.35:80 check inter 2000 fall 3 weight 30 #定义的多个后端