<https://github.com/haproxy/haproxy/tags>

# 1、部署lua

#yum install libtermcap-devel ncurses-devel libevent-devel readline-devel

#curl -R -O http://www.lua.org/ftp/lua-5.4.4.tar.gz

#tar zxf lua-5.4.4.tar.gz

#cd lua-5.4.4

#make all test

# lua -v #当前系统版本

Lua 5.1.4 Copyright (C) 1994-2008 Lua.org, PUC-Rio

# #/usr/local/src/lua-5.4.4/src/lua -v

Lua 5.4.4 Copyright (C) 1994-2022 Lua.org, PUC-Rio

# 2、部署haproxy

#yum install -y systemd-devel

#tar xf haproxy-2.6.0.tar.gz

#cd haproxy-2.6.0

#make ARCH=x86\_64 TARGET=linux-glibc USE\_PCRE=1 USE\_OPENSSL=1 USE\_ZLIB=1 \

USE\_SYSTEMD=1 USE\_CPU\_AFFINITY=1 USE\_LUA=1 LUA\_INC=/usr/local/src/lua-5.4.4/src/ \

LUA\_LIB=/usr/local/src/lua-5.4.4/src/ PREFIX=/usr/local/haproxy

#make install PREFIX=/usr/local/haproxy

# cp haproxy /usr/sbin/

#开启https #USE\_OPENSSL=1

#指定为systemd模式 #USE\_SYSTEMD=1

#可指定安装目录 #PREFIX=/usr/local/haproxy

#默认为/usr/local/sbin/

#mkdir /usr/local/haproxy/{etc,logs,run}

# chown nobody /usr/local/haproxy/ -R

# 3、加入service

|  |
| --- |
| cat > /usr/lib/systemd/system/haproxy.service << EOF  [Unit]  Description=HAProxy Load Balancer  After=syslog.target network.target  [Service]  ExecStartPre=/usr/local/haproxy/sbin/haproxy -f /usr/local/haproxy/etc/haproxy.cfg -c -q  ExecStart=/usr/local/haproxy/sbin/haproxy -Ws -f /usr/local/haproxy/etc/haproxy.cfg -p /usr/local/haproxy/run/haproxy.pid  ExecReload=/bin/kill -USR2 $MAINPID  [Install]  WantedBy=multi-user.target  EOF |

# 4、配置**haproxy.cfg**

**# vim /usr/local/haproxy/etc/haproxy.cfg**

|  |
| --- |
| global  log 127.0.0.1 local0  #log 127.0.0.1 local1 notice  #log loghost local0 info  maxconn 4096  chroot /usr/local/haproxy  uid 99  gid 99  daemon #以后台形式运行haproxy  #nbproc 1 #启动1个haproxy实例。# #工作进程数量(CPU数量) ，实际工作中，应该设置成和CPU核心数一样。 这样可以发挥出最大的性能。2.5版本后不再支持，需要注释。  pidfile /usr/local/haproxy/run/haproxy.pid #将所有进程写入pid文件  #debug #调试错误时用  #quiet #安静  defaults  log global  log 127.0.0.1 local3 #日志文件的输出定向。产生的日志级别为local3. 系统中local1-7，用户自己定义  mode http #工作模式，所处理的类别,默认采用http模式，可配置成tcp作4层消息转发  option httplog #日志类别，记载http日志  option httpclose #每次请求完毕后主动关闭http通道,haproxy不支持keep-alive,只能模拟这种模式的实现  option dontlognull #不记录空连接，产生的日志  option forwardfor #如果后端服务器需要获得客户端真实ip需要配置的参数，可以从Http Header中获得客户端ip  option redispatch #当serverid对应的服务器挂掉后，强制定向到其他健康服务器  retries 2 #2次连接失败就认为服务器不可用，主要通过后面的check检查  maxconn 2000 #最大连接数  balance roundrobin #负载均衡算法  stats uri /haproxy-stats #haproxy 监控页面的访问地址 # 可通过 http://localhost:80/haproxy-stats 访问  timeout connect 5000 #连接超时时间。 单位：ms 毫秒  timeout client 50000 #客户端连接超时时间  timeout server 50000 #服务器端连接超时时间  mode http  option httpchk GET /index.html #健康检测#注意实际工作中测试时，应该下载某一个页面来进行测试，因此这个页面应该是个小页面，而不要用首页面。这里是每隔一秒检查一次页面。  frontend http #前端配置，http名称可自定义  bind 0.0.0.0:80 #发起http请求80端口，会被转发到设置的ip及端口  default\_backend http\_back #转发到后端 写上后端名称  backend http\_back #后端配置，名称上下关联  server s1 192.168.31.43:8081 weight 3 check #后端的主机 IP &权衡  server s2 192.168.31.43:8082 weight 3 check #后端的主机 IP &权衡  #server node1 192.168.179.131:8081 check inter 2000 rise 3 fall 3 weight 30  # inter 2000 健康检查时间间隔2秒  # rise 3 检测多少次才认为是正常的  # fall 3 失败多少次才认为是不可用的  # weight 30 权重 |

关于nbproc 1

[ALERT] (11351) : config : parsing [/usr/local/haproxy/etc/haproxy.cfg:10] : nbproc is not supported any more since HAProxy 2.5. Threads will automatically be used on multi-processor machines if available.

检查配置文件语法

#haproxy -c -f /usr/local/haproxy/etc/haproxy.cfg

**关于负载均衡算法**

#source 根据请求源IP

#static-rr 根据权重

#leastconn 最少连接者先处理

#uri 根据请求的uri

#url\_param 根据请求的url参数

#rdp-cookie 据据cookie(name)来锁定并哈希每一次请求

#hdr(name) 根据HTTP请求头来锁定每一次HTTP请求

#roundrobin 轮询方式

**使用nobody用户运行haproxy**

# id nobody

uid=99(nobody) gid=99(nobody) groups=99(nobody) #id 为99

# 5、启动haproxy

#systemctl daemon-reload

#systemctl start haproxy.service

#systemctl status haproxy.service

# systemctl enable haproxy

# 6、配置日志收集

**vim /etc/rsyslog.conf**

$ModLoad imudp #取消注释

$UDPServerRun 514 #取消注释

$ModLoad imtcp #取消注释

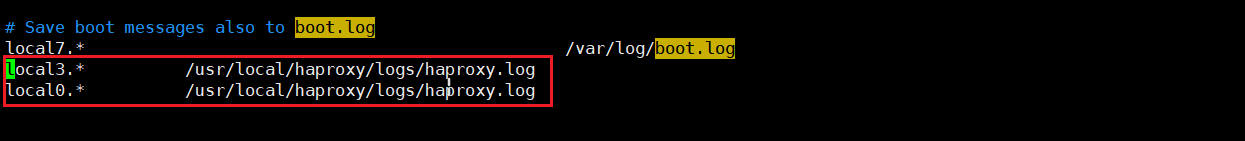
$InputTCPServerRun 514 #取消注释

local7.\* /var/log/boot.log #下面添加两行

local3.\* /usr/local/haproxy/logs/haproxy.log

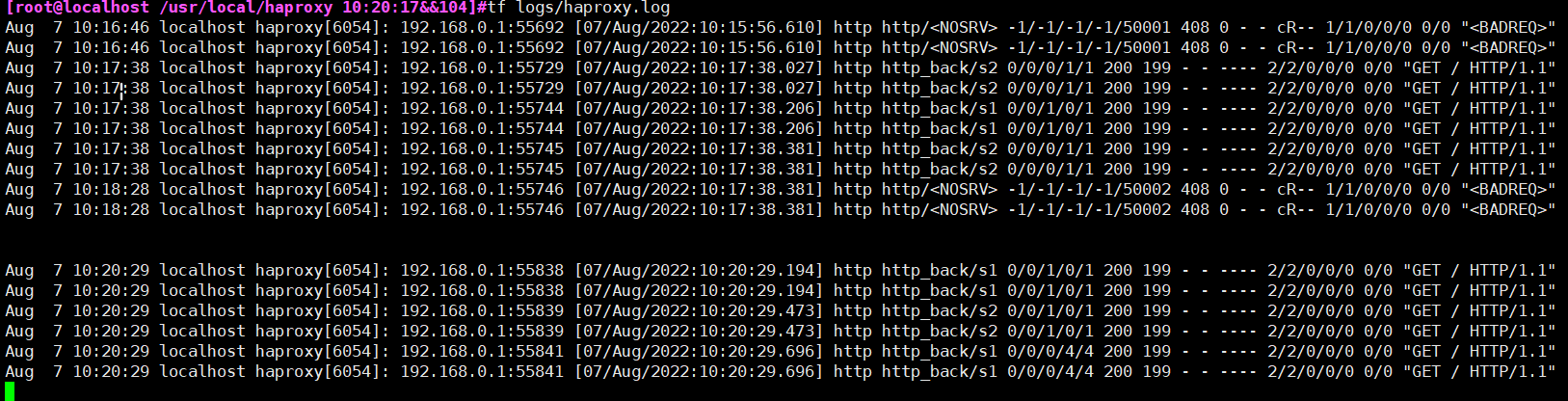
local0.\* /usr/local/haproxy/logs/haproxy.log





**# systemctl restart rsyslog**

**# 访问查看日志**



# 7验证

通过docker跑两个nginx

#docker pull nginx

#docker run -d --name ng1 -p 8081:80 nginx

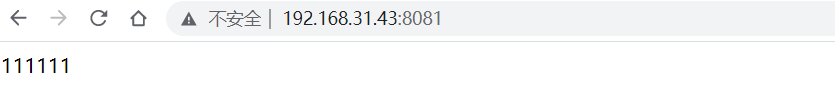
#docker run -d --name ng2 -p 8082:80 nginx

#docker exec -it ng1 bash -c "echo 111111 > /usr/share/nginx/html/index.html"

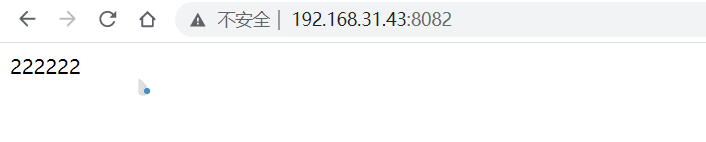
#docker exec -it ng2 bash -c "echo 222222 > /usr/share/nginx/html/index.html"

## 7.1 Nginx单独访问

<http://192.168.31.43:8081/>

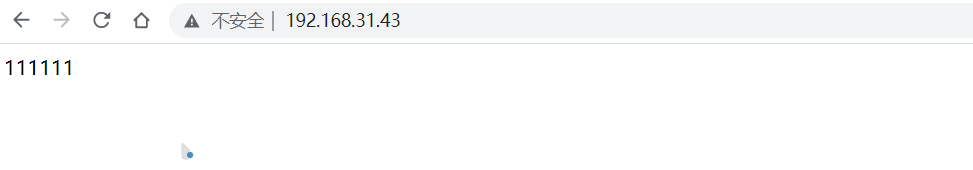


<http://192.168.31.43:8082/>



## 7.2通过haproxy 访问

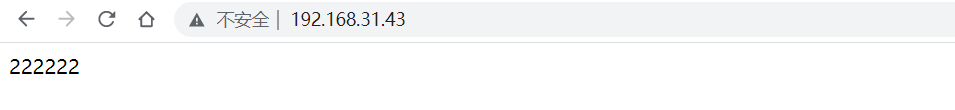
<http://192.168.31.43/>



## 7.3 停止ng1

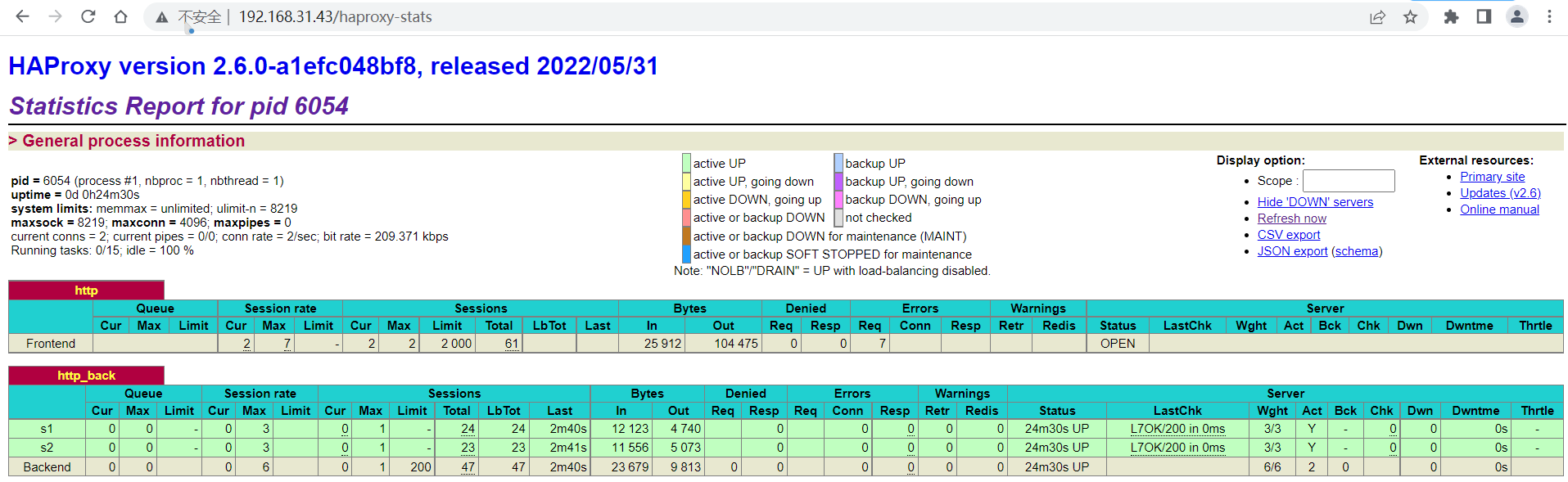
#docker stop ng1

Haproxy 只会访问到ng2



## 7.4 Haproxy集群状态

<http://192.168.31.43/haproxy-stats>



# 参考url

<https://blog.csdn.net/martinlinux/article/details/115620051>

<https://github.com/haproxy/haproxy/blob/master/INSTALL>