**Haproxy笔记**

一、HAProxy简介

（1）HAProxy 是一款提供高可用性、负载均衡以及基于TCP（第四层）和HTTP（第七层）应用的代理软件，支持虚拟主机，双机热备，负载均衡，它是免费、快速并且可靠的一种解决方案。 HAProxy特别适用于那些负载特大的web站点，这些站点通常又需要会话保持或七层处理。HAProxy完全可以支持数以万计的 并发连接。并且它的运行模式使得它可以很简单安全的整合进您当前的架构中， 同时可以保护你的web服务器不被暴露到网络上，而且拥有对服务器节点健康检查功能（相当于keepalived健康检查），配置简单，维护方便。

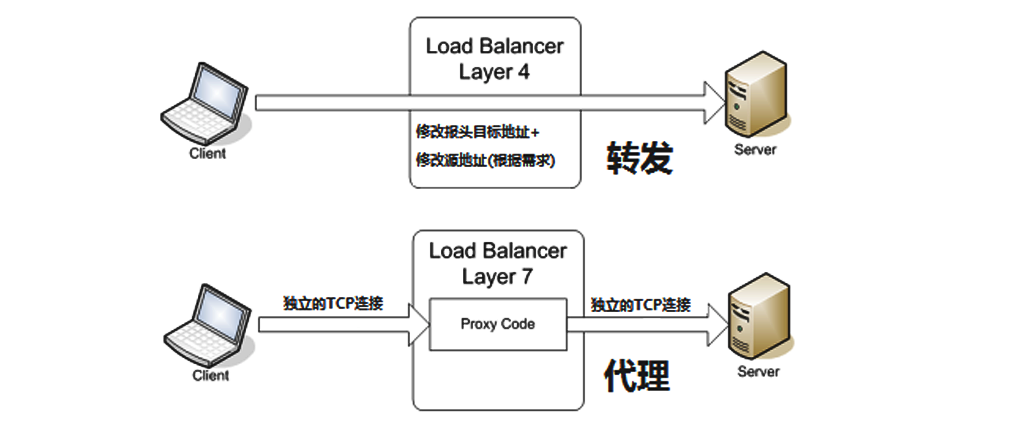
**四层和七层负载均衡的区别**

**四层负载均衡也称为四层交换机：**  
　　所谓的四层就是ISO参考模型中的第四层。四层负载均衡也称为四层交换机，它主要是 通过分析IP层及TCP/UDP层的流量实现的基于IP加端口的负载均衡。常见的基于四层的负载均衡器有LVS、F5等。

以常见的TCP应用为例，负载均衡器在接收到第一个来自客户端的SYN请求时，会通过设定的负载均衡算法选择一个最佳的后端服务器，同时将报文中目标IP地址修改为后端服务器IP，然后直接转发给该后端服务器，这样一个负载均衡请求就完成了。  
 从这个过程来看，一个TCP连接是客户端和服务器直接建立的，而负载均衡器只不过完成了一个类似路由器的转发动作。在某些负载均衡策略中，为保证后端服务器返回的报文可以正确传递给负载均衡器，在转发报文的同时可能还会对报文原来的源地址进行修改。

**七层负载均衡器也称为七层交换机：**

位于 OSI 的最高层，即应用层，此时负载均衡器支持多种应用协议，常见的有 HTTP、FTP、SMTP 等。七层负载均衡器可以根据报文内容，再配合负载均衡算法来选择后端服务器，因此也称为“内容交换器”。  
　　比如，对于 Web 服务器的负载均衡，七层负载均衡器不但可以根据“IP+端口”的方式进行负载分流，还可以根据网站的 URL、访问域名、浏览器类别、语言等决定负载均衡的策略。  
　　例如，有两台 Web 服务器分别对应中英文两个网站，两个域名分别是 A、B，要实现访问 A 域名时进入中文网站，访问 B 域名时进入英文网站，这在四层负载均衡器中几乎是无法实现的，而七层负 载均衡可以根据客户端访问域名的不同选择对应的网页进行负载均衡处理。  
　　常见的七层负载均衡器有 HAproxy、Nginx 等。



**HAProxy支持两种主要的代理模式:**

TCP： 即4层 （大多用于邮件服务器、内部协议通信服务器等）。在4层模式 下，HAProxy仅在客户端和服务器之间转发双向流量。

HTTP： 即7层模式，HAProxy会分析协议，并且能通过允许、拒绝、交换、增加、修改或者\*\*\*请求 (request)或者回应(response)里指定内容来控制协议，这种操作要基于特定规则。（新的1.3之后的版本引入了frontend,backend指令；frontend根据任意HTTP请求头内容做规则匹配,然后把请求定向到相关的backend.） [服ruang特按的 --- 白可按的]

以TCP应用为例，由于负载均衡器要获取到报文的内容，因此只能先代替后端服务器和客户端建立连接，接着，才能收到客户端发送过来的报文内容，然后再根据该报文中特定字段加上负载均衡器中设置的负载均衡算法来决定最终选择的内部服务器。

对比四层负载均衡和七层负载均衡运行的整个过程，可以看出，

七层负载均衡模式下，负载均衡器与客户端及后端的服务器会分别建立一次TCP连接  
四层负载均衡模式下， 仅建立一次TCP连接。  
由此可知，七层负载均衡对负载均衡设备的要求更高，而七层负载均衡的处理能力也必然低于四层模式的负载均衡。

**总结haproxy主要优点：**

一、免费开源，稳定性也是非常好，这个可通过我做的一些小项目可以看出来，单Haproxy也跑得不错，稳定性可以与LVS相媲美；

二、根据官方文档，HAProxy可以跑满10Gbps-New benchmark of HAProxy at 10 Gbps using Myricom's 10GbE NICs (Myri-10G PCI-Express)，这个作为软件级负载均衡，也是比较惊人的；

三、HAProxy可以作为[MySQL](http://lib.csdn.net/base/14)、邮件或其它的非web的负载均衡，我们常用于它作为MySQL(读)负载均衡；

四、自带强大的监控服务器状态的页面，

五、HAProxy支持虚拟主机。

**开始安装haproxy**

官网下载：https://www.haproxy.org/download/

wget https://www.haproxy.org/download/1.8/src/haproxy-1.8.19.tar.gz

wget https://www.haproxy.org/download/1.7/src/haproxy-1.7.11.tar.gz

wget https://www.haproxy.org/download/1.6/src/haproxy-1.6.14.tar.gz

wget https://www.haproxy.org/download/1.5/src/haproxy-1.5.19.tar.gz

wget <https://www.haproxy.org/download/1.4/src/haproxy-1.4.27.tar.gz>

**net.ipv4.ip\_forward = 1 基于NAT模式的负载均衡器都需要打开转发功能**

[ ! -f haproxy-1.6.14.tar.gz ] && wget https://www.haproxy.org/download/1.6/src/haproxy-1.6.14.tar.gz

tar zxvf haproxy-1.6.14.tar.gz

cd haproxy-1.6.14

yum -y install gcc gcc-c++

make TARGET=linux2628 ARCH=X86\_64

make PREFIX=/server/application/haproxy-1.6.14 install

ln -s /server/application/haproxy-1.6.14/ /server/application/haproxy

[ -z "`grep "net.ipv4.ip\_forward" /etc/sysctl.conf`" ] && echo net.ipv4.ip\_forward = 1 >>/etc/sysctl.conf

sysctl -p

cd /server/application/haproxy

mkdir -p bin conf logs var/run var/chroot

useradd -s /sbin/nologin -M haproxy

说明：

net.ipv4.ip\_forward 开启转发功能

net.ipv4.ip\_nonlocal\_bind = 1 解决haproxy端没有VIP时，启动haproxy不能bind的问题

**配置文件段介绍：**

haproxy 配置中分成五部分内容，分别如下：

global #全局配置参数段，主要控制haproxy启动前的进程及系统相关设置

defaults： #配置默认参数，如果frondend,backend,listen没有设置，则使用defaults段的配置

frontend： #用来匹配接收客户端所有请求的域名，uri等，并针对不同的匹配，做不同的请求处理，frondend规则直接调用后端backend

backend： #后端服务集群的配置，是真实服务器，以及对后端服务器的一些权重，队列，连接数等选项配置

Listen #Fronted和backend的组合体

配置日志不记录到单独文件中而不记录到/var/log/messages：

echo local0.\* /server/application/haproxy/logs/haproxy.log>>/etc/rsyslog.conf

tail -1 /etc/rsyslog.conf

sed -i s#'SYSLOGD\_OPTIONS=""'#'SYSLOGD\_OPTIONS="-c 2 -m 0 -r -x"'#g /etc/sysconfig/rsyslog

tail -1 /etc/sysconfig/rsyslog

sed -i s/'#$ModLoad imudp'/'$ModLoad imudp'/g /etc/rsyslog.conf

sed -i s/'#$UDPServerRun 514'/'$UDPServerRun 514'/g /etc/rsyslog.conf

systemctl restart rsyslog

修改大约54行 这里是为了不让日志记录两份，如果不加如local0.none 还会把日志记录到var/log/messages里面

\*.info;mail.none;authpriv.none;cron.none;local0.none /var/log/messages

**注意：一定要重启rsyslog 一定要确保路径都正确，运维90%的故障都来自于自己的不认真**

-r:打开接受外来日志消息的功能,其监控514 UDP端口;  
-x:关闭自动解析对方日志[服务器](https://www.baidu.com/s?wd=%E6%9C%8D%E5%8A%A1%E5%99%A8&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)的FQDN信息,这能避免DNS不完整所带来的麻烦;  
-m:修改syslog的内部mark消息写入间隔时间(0为关闭),例如240为每隔240分钟写入一次"--MARK--"信息

启动脚本

[root@localhost /application/haproxy/conf]# cat ../bin/haproxy

#!/bin/bash

BASE="/server/application/haproxy"

PROG=$BASE/sbin/haproxy

PIDFILE=$BASE/var/run/haproxy.pid

CONFFILE=$BASE/conf/haproxy.conf

case "$1" in

start)

$PROG -f $CONFFILE -q -D

;;

status)

if [ ! -f $PIDFILE ];then

echo "haproxy pid not found"

exit 1

fi

for pid in $(cat $PIDFILE);do

kill -0 $pid

RETVAL="$?"

if [ ! "$RETVAL" = "0" ];then

echo "process $pid died"

exit 1

fi

done

echo "haproxy is running"

;;

restart)

$PROG -f $CONFFILE -q -sf $(cat $PIDFILE)

;;

check)

$PROG -f $CONFFILE -c

;;

stop)

kill -9 $(cat $PIDFILE)

;;

\*)

echo "USAGE: $0 start|restart|status|check|stop"

exit 1

;;

esac

TCP代理ssh登陆

[root@localhost /application/haproxy/conf]# cat haproxy.conf

global

chroot /server/application/haproxy/var/chroot

daemon

user haproxy

group haproxy

log 127.0.0.1:514 local0 warning

pidfile /server/application/haproxy/var/pid

maxconn 4000

nbproc 8

defaults

log global

retries 3

option redispatch

listen www

bind \*:80

mode tcp

balance roundrobin

server web01 10.10.10.20:20 check port 20 inter 5000 fall 5

**注意：启动的时候80端口不能被占用**

启动haproxy  
-f 指定配置文件  
-c 检查语法  
-D 启动进程  
-q 不显示提示信息  
-p pid文件

../sbin/haproxy -f ./haproxy.conf -c #检查语法

../sbin/haproxy -f ./haproxy.conf -D #-D以deamon模式启动

ps -ef|grep haproxy.conf

ssh -p80 10.10.10.3 #访问本机的80端口、登陆到了10.10.10.20的20端口

至此：haproxy TCP四层代理测试完成

**加入开机自启动**

echo "/server/application/haproxy-1.6.14/bin/haproxy start" >>/etc/rc.local

chmod +x /etc/rc.local

**http-7层代理：------Listen**

global

chroot /application/haproxy/var/chroot

daemon

user haproxy

group haproxy

log 127.0.0.1:514 local0 warning

pidfile /application/haproxy/var/run/haproxy.pid

maxconn 4000

nbproc 1

defaults

log global

mode http

retries 3

option redispatch

timeout connect 5000ms

timeout client 50000ms

timeout server 50000ms

#ulimit -n 65535

option abortonclose

option dontlognull

option httpclose

listen linyaohong

bind \*:80

mode http

stats enable

stats hide-version

stats uri /admin?status

stats realm Private lands

stats auth admin:111111

balance roundrobin

option httpclose

option forwardfor

cookie SERVERID insert indirect

timeout connect 15s

timeout client 15s

timeout server 15s

#option httpchk

#option httpchk GET /test/get.html

#option httpchk HEAD /test/index.html HTTP/1.1\r\nHost:www.linyaohong.com

option httpchk HEAD /test/index.html HTTP/1.0

server www01 10.10.10.20:80 cookie A check port 80 inter 5000 rise 3 fall 5

#server www01 10.10.10.20:80 cookie A check port 80 inter 5000 rise 3 fall 5

server www02 10.10.10.21:80 cookie A check port 80 inter 5000 rise 3 fall 5

#server www03 10.10.10.21:80 maxconn 2048 weight 8 check port 80 inter 5000 rise 3 fall 5

配置文件详解

**global**

**--------------------------------------------------------------------------------------------------------------------------------**

修改haproxy的工作目录至指定目录并在放弃权限前执行chroot()操作，可以提升haproxy安全级别

chroot /application/haproxy/var/chroot

以守护进程的方式运行

daemon

指定运行haproxy的用户和组

user haproxy

group haproxy

全局日志配置，log关键字，指定只用127.0.0.1上的rsyslog服务中的local0日志设备，

记录日志等级为emerg|alert|crit|err|warning|notice|info|debug，生产环境中最低warning以上

log 127.0.0.1:514 local0 warning

定义haproxy的pid文件文件，方便管理和查看

pidfile /application/haproxy/var/run/haproxy.pid

定义haproxy的最大连接数

maxconn 4000

设置haproxy启动的进程数，该值的设置应该和服务器的CPU核心数一致，即常见的2颗8核心CPU的服务器，即共有16核心，  
则可以将其值设置为：<=16 ，创建多个进程数，可以减少每个进程的任务队列，但是过多的进程数也可能会导致进程的崩溃。这里我设置为16

nbproc 8

设置最大打开的文件描述符数，在1.4的官方文档中提示，该值会自动计算，所以不建议进行设置

#ulimit -n 65536

**defaults**

---------------------------------------------------------------------------------------------------------------------------------

继承global中log的定义

log global

mode语法：mode {http|tcp|health} 。http是七层模式，tcp是四层模式，health是健康检测，返回OK

mode http

定义连接后端服务器的失败重连次数，连接失败此处超过 此值 后，将后端服务器标记为不可用

retries 3

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。

option redispatch

设置成功连接到一台服务器的最长等待时间，默认单位是毫秒，老版本的haproxy使用 contimeout 5000

timeout connect 5000ms

#设置连接客户端发送数据时的成功连接最长等待时间，默认单位是毫秒，老版本haproxy使用 clitimeout 3000

timeout client 50000ms

设置服务器端回应客户度数据发送的最长等待时间，默认单位是毫秒，老版本haproxy使用 srvtimeout 3000

timeout server 50000ms

当服务器负载很高的时候，自动结束掉当前队列处理比较久的链接

option abortonclose

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

option dontlognull

使用该参数，每处理完一个request时，haproxy都会去检查http头中的Connection的值，如果该值不是close，haproxy将会将其删除，如果该值为空将会添加为：Connection: close。  
使每个客户端和服务器端在完成一次传输后都会主动关闭TCP连接。与该参数类似的另外一个参数是“option forceclose”，该参数的作用是强制关闭对外的服务通道，因为有的服务器端收到Connection: close时，也不会自动关闭TCP连接，  
如果客户端也不关闭，连接就会一直处于打开，直到超时

option httpclose

--------------------------------------------------------------------------------------------------------------------------------

定义一个名为status的部分，可以在listen指令指定的区域中定义匹配规则和后端服务器ip，相当于需要在其中配置frontend，backend的功能。一般做tcp转发比较合适，不用太多的规则匹配

**listen linyaohong**

定义监听的套接字

bind :80

定义为HTTP模式

mode http

开启web功能

stats enable

隐藏版本号

stats hide-version

设置状态页面的uri为/admin?stauts

stats uri /admin?status

设置状态页面认证时的提示内容

stats realm Private lands

设置状态页面认证的用户和密码，如果要设置多个，另起一行写入即可

stats auth admin:111111

设置haproxy的调度算法：roundrobin|static-rr|leastconn|first|source|uri|url\_param|hdr <name>

balance roundrobin

default已做介绍

option httpclose

让后端RS记录用户访问真实IP，而不是记录haproxy的IP，后端日志格式也需要设置forwardfor

option forwardfor

允许向cookie插入SERVERID，每台服务器的SERVERID可在下面用cookie关键字定义

cookie SERVERID insert indirect

如下已经在default介绍：

timeout connect 15s

timeout client 15s

timeout server 15s

健康检查：

由于基于uri的方式做健康检查相对来说也比较容易配置，推荐使用基于URL的健康检查、可以用http://10.0.0.8/check.html的方式，也可以使用基子域名的 http://www.etiantian.ora/chcck.html 的方式。

实际生产环境中，最重要最关键的是，我们设置的check.html能否真正的代表rs 占的业兰状态，即：访问check.htm正常,整个业务是否能正常，这个是需要运维以及开发人员分析。

对于用户体验比较高的业务，可请开发人员设置更加深入的健康检查文件，例如: check.php或check.jsp,这个健康检查可以深入到数据库，存储及各个接口，如果之一出现异常就摘除，如何做还是需要去思考的，工作中不是做的越高级越好，根据业务需求达到要求就是可以。

对于用户体验比较高的业务，除了健康检查文件更深入外，健康检查的频率、次数 都要相应调整。例如：1秒检查一次，一次连不上就摘除RS、未必是是最好的策略，尤其是服务器节点少的时候，可能会导致hapraxy后端瞬间全部被摘除的问题发生。

option httpchk #此检查方法不准，如果遇到403也是正常状态

option httpchk GET /test/get.html #推荐

option httpchk HEAD /test/index.html HTTP/1.0 #推荐 或者 HTTP/1.1

option httpchk HEAD /test/index.html HTTP/1.1\r\nHost:www.linyaohong.com #推荐 或者 HTTP/1.1

**HEAD和GET都可以**

--------------------------------------------------------------------------------------------------------------------------------

**server行：**

check 80 #代表对80端口进程检查、也可以直接写check，但是check不可省略，标记对后端RS进行健康检查

inter 5000 fall5 #表示每5秒检查一次,总共检查5次，如果有问题就会摘除掉出问题的机器

如果结尾不加inter 5000 fall5 则默认：2秒检查一次，检查3次，如果有问题就会摘除掉出问题的机器

rise 3 #检查3次OK的话，认为机器复活，并加入集群组中

maxconn 2048 #最大连接数

weight 8 #权重

backup #备机、做高可用也可以（mysql web服务）， 设置2个server 当第一个挂了 ，第二个接管服务

maxqueue #队列限制 不常用

minconn #最小连接数 不常用

server www01 10.10.10.20:80 cookie A check port 80 inter 5000 rise 3 fall 5

#server www01 10.10.10.20:80 cookie A check port 80 inter 5000 rise 3 fall 5

server www02 10.10.10.21:80 cookie A check port 80 inter 5000 rise 3 fall 5 backup

#server www03 10.10.10.21:80 maxconn 2048 weight 8 check port 80 inter 5000 rise 3 fall 5

设置了backup的时候，默认第一个backup会优先，设置option allbackups后，所有备份服务器权重一样

当正常提供服务器的机器全部挂掉以后， backup机器全部一起启动，如果不加，会一个个启动

option allbackups

haproxy 301跳转

frontend √

acl 123\_deamon hdr(host) -i test.linyaohong.com

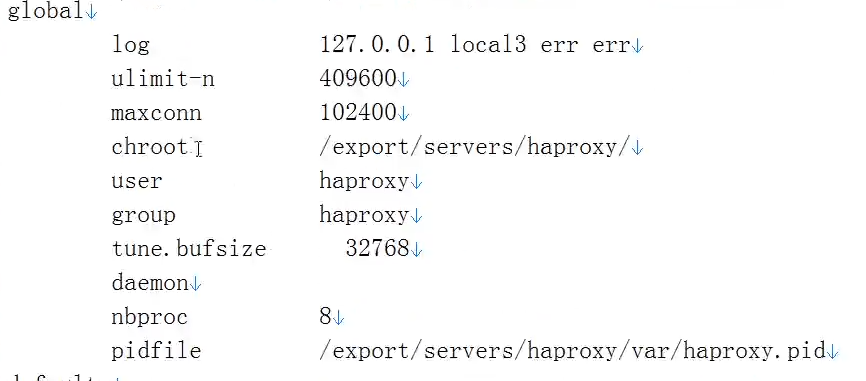
redirect prefix https://www.baidu.com code 301 if 123\_deamon

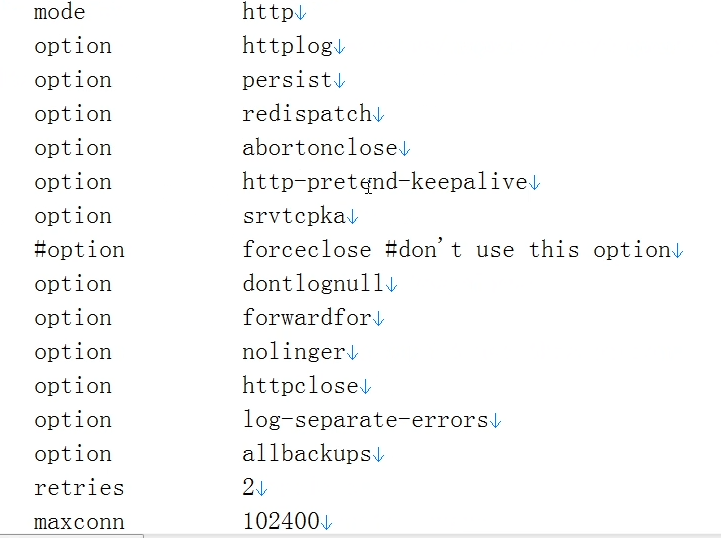
总结： 确保haproxy对后端的超时时间---大于RS自身向其后端请求的超时时间之和

调整健康检查频率

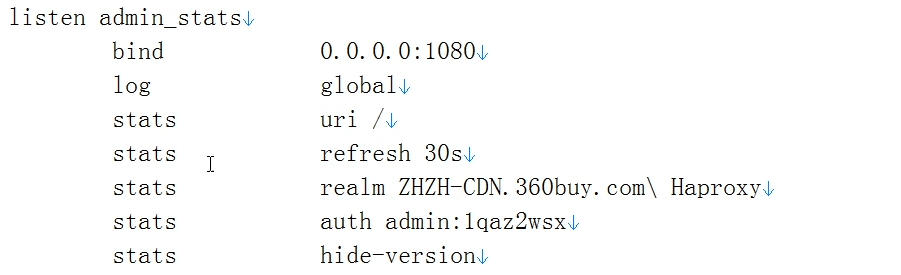
日志级别的控制、以及日记记录

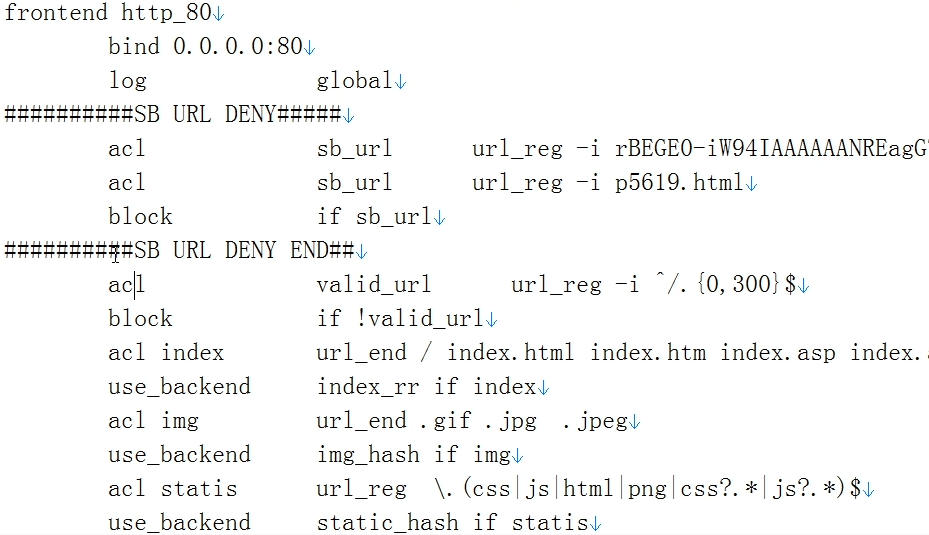
京东：



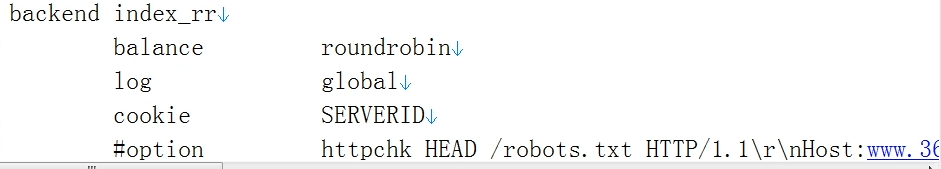


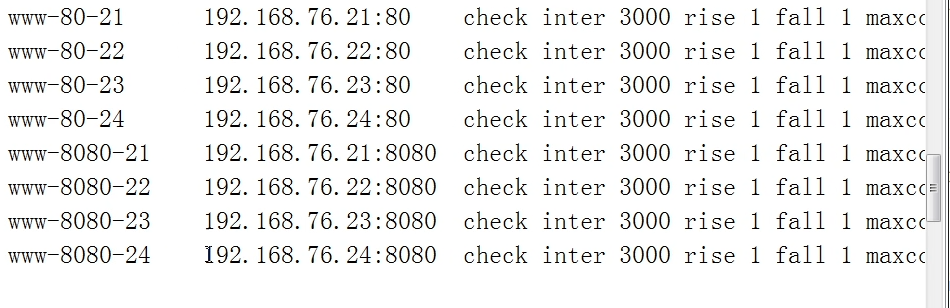


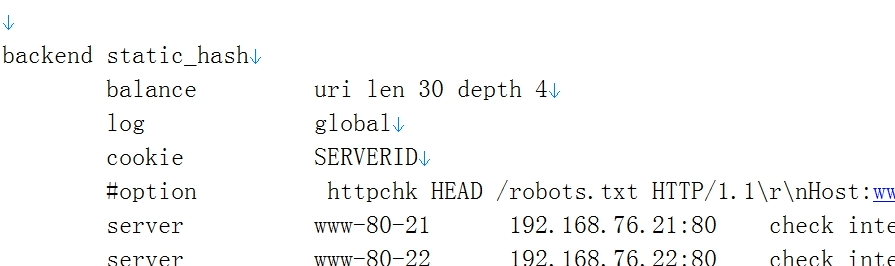




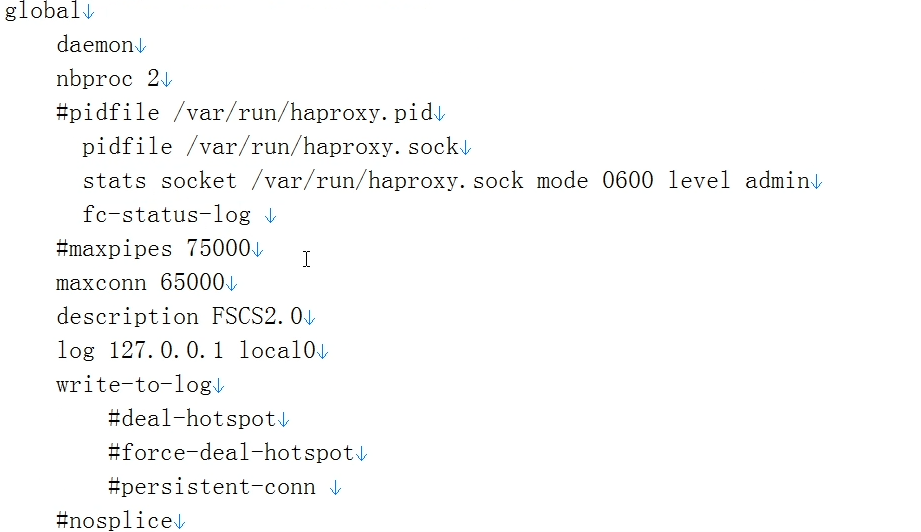








蓝汛：

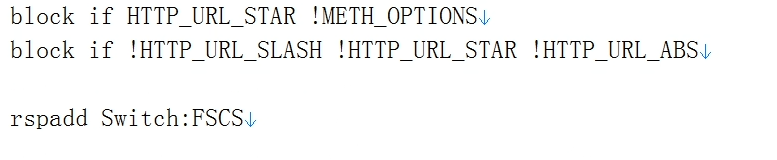


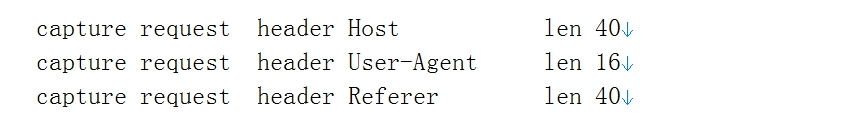
default

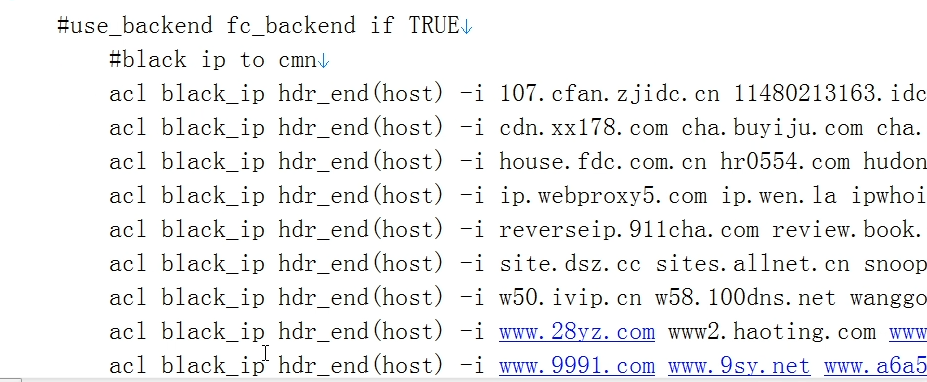


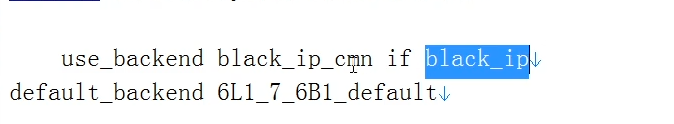


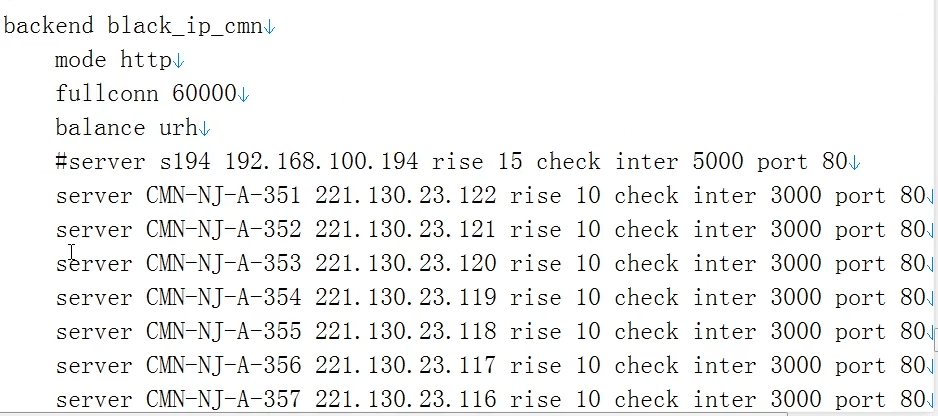


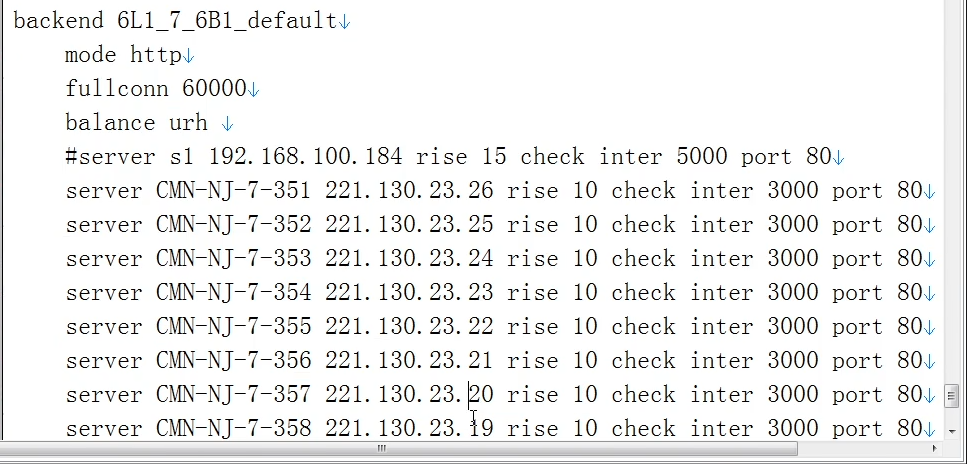




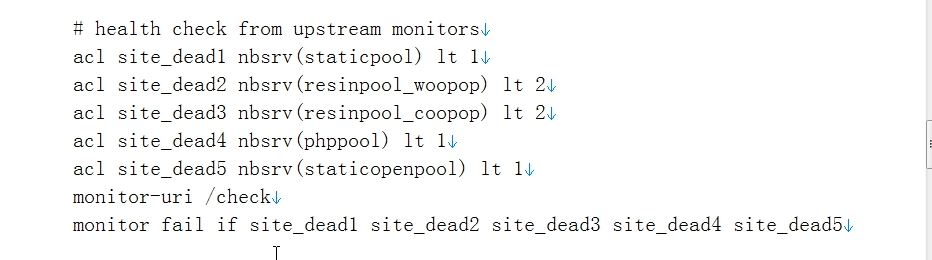




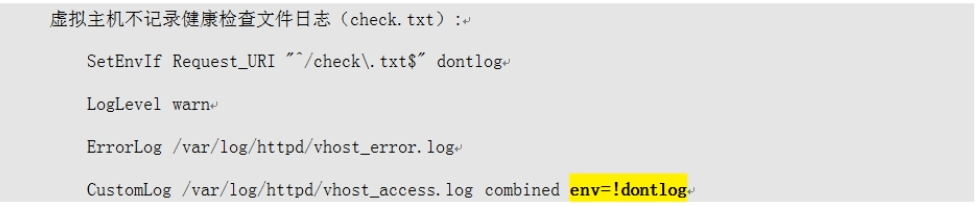


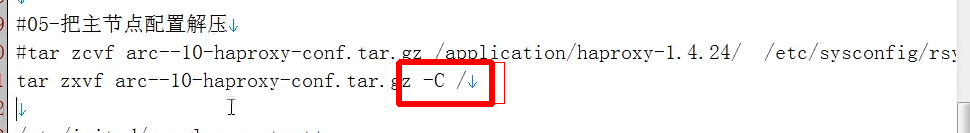


下图配置：当挂掉几个正常服务器的机器后，启动backup ，（暂时不理解，估计也J8用不到，）

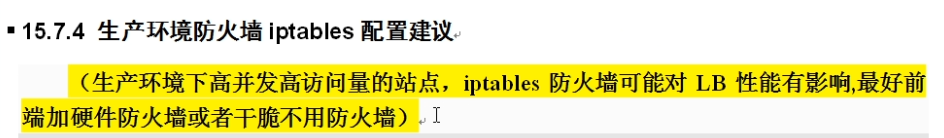












for n in `echo {1..100}` ; do curl http://10.10.10.25;sleep 1;done