第三十四章：Haproxy+Nginx+Tomcat实现动静分离页面

**一、Haproxy概述；**

**二、Haproxy原理实现；**

**三、Nginx、LVS、Haproxy对比；**

**四、Haproxy配置文件讲解；**

**五、案例：Haproxy+Nginx+Tomcat搭建高可用集群；**

**一、Haproxy概述；**

**概述：**Haproxy是一个开源的高性能的反向代理或者说是负载均衡服务软件之一，由C语言编写而成，支持会话保持、七层处理、健康检查、故障修复后自动加载、动静分离。HAProxy运行在当前的硬件上，完全可以支持数以万计的并发连接；

Haproxy软件引入了frontend，backend的功能，frontend（acl规则匹配）可以运维管理人员根据任意HTTP请求头做规则匹配，然后把请求定向到相关的backend（server pools等待前端把请求转过来的服务器组）。

**二、Haproxy原理实现；**

**代理模式：**

1.四层tcp代理：例如：可用于邮件服务内部协议通信服务器、Mysql服务等；

2.七层应用代理：例如：HTTP代理或https代理。在4层tcp代理模式下，Haproxy仅在客户端和服务器之间双向转发流量。但是在7层模式下Haproxy会分析应用层协议，并且能通过运行、拒绝、交换、增加、修改或者删除请求（request）或者回应（reponse）里指定内容来控制协议。

**四层代理：**

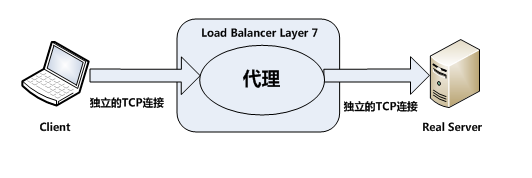
ISO参考模型中的第四层传输层。四层负载均衡也称为四层交换机，它主要是通过分析IP层及TCP/UDP层的流量实现的基于IP加端口的负载均衡。常见的基于四层的负载均衡器有LVS、F5等。以常见的TCP应用为例，负载均衡器在接收到第一个来自客户端的SYN请求时，会通过设定的负载均衡算法选择一个最佳的后端服务器，同时将报文中目标IP地址修改为后端服务器IP，然后直接转发给该后端服务器，这样一个负载均衡请求就完成了。从这个过程来看，一个TCP连接是客户端和服务器直接建立的，而负载均衡器只不过完成了一个类似路由器的转发动作。在某些负载均衡策略中，为保证后端服务器返回的报文可以正确传递给负载均衡器，在转发报文的同时可能还会对报文原来的源地址进行修改。整个过程下图所示。



**七层代理：**

ISO参考模型中的最高层第七层应用层。七层负载均衡也称为七层交换机，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。

这里仍以常见的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。纵观整个过程，七层负载均衡器在这种情况下类似于一个代理服务器。整个过程如下图所示。



**调度算法：**

balance roundrobin：表示简单的轮询，负载均衡基础算法  
balance static-rr：表示根据权重  
balance leastconn：表示最少连接者先处理  
balance source：表示根据请求源IP  
balance uri：表示根据请求的URI；  
balance url\_param：表示根据请求的URl参数来进行调度  
balance hdr(name)：表示根据HTTP请求头来锁定每一次HTTP请求；  
balance rdp-cookie(name)：表示根据据cookie(name)来锁定并哈希每一次TCP请求。

**三、Nginx、LVS、Haproxy对比；**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Nginx（七层代理）** | **LVS（四层代理）** | **Haproxy（四层、七层）** |
| **优势** | 1.跨平台性；  2.配置简单，容易上手；  3.非阻塞、高并发连接；  4.事件驱动：通信机制采用 epoll 模型，支持更大的并发连接  Master/Worker 结构；  5.内存消耗小；  6.内置的健康检查功能；  7.支持 GZIP 压缩；  8.稳定性高：用于反向代理，宕机的概率微乎其微；  9.会话保持：通过调度算法实现；  10.动静分离：基于location匹配正则； | 1.抗负载能力强，强大的四层代理，媲美F5硬件的四层代理；  2.对内存和 cpu 资源消耗比较低；  3.配置性比较低；  4.工作稳定，自身有完整的双机热备方案如 LVS+Keepalived；  5.无流量，LVS 只分发请求，而流量并不从它本身出去，这点保证了均衡器 IO 的性能不会受到大流量的影响。  6.会话保持； | 1.支持 Session 的保持，Cookie 的引导；同时支持通过获取指定的 url 来检测后端服务器的状态。  2.基于TCP协议的负载效率高于Nginx；  3.负载均衡策略非常多；  4.动静分离：支持acl规则匹配；  5.高效稳定；  6.节点健康检查：支持多种方式检测，如端口、url； |
| **劣势** | 1.适用性低：仅能支 持http、https 和 Email 协议，这样就在适用范围上面小些；  2.对后端服务器的健康检查，只支持通过端口来检测，不支持通过 ur l来检测；  3.不支持 Session 的直接保持，但能通过 ip\_hash 来解决；  4.不适用于高负载量的并发集群； | 1.健康检查：无法检查后端节点的健康情况；  2.动静分离：不支持正则，无法区分请求；  3.实现比较麻烦；  4.网络环境依赖性较大； | 1.扩展性差：添加新功能很费劲，对不断扩展的新业务，haproxy很难应对；  2.在四层代理时，仅支持tcp协议的代理； |

**总结：**

* 大型网站架构：对性能有严格要求的时候可以使用lvs或者硬件F5，就单纯从负载均衡的角度来说，lvs也许会成为主流，更适合现在大型的互联网公司；
* 中型网站架构：对于页面分离请求由明确规定，并且性能有严格要求时，可以使用haproxy；
* 中小型网站架构：比如日 PV 小于1000万，需要进行高并发的网站或者对网络不太严格的时候，可以使用nginx；

注解：页面访问量术语：

PV：指页面的浏览次数；

UV：指一天内访问某网站的人数，以（COOKIE）作为依据；

IP：指一天内访问该网站的IP地址数量；

QPS：每秒处理完的请求数量，计算公式：请求数量/平均响应时间；

**四、Haproxy配置文件讲解；**

* global：全局配置区域参数是进程级的，通常是和操作系统相关。这些参数一般只设置一次，如果配置无误，就不需要再次进行修改；



* defaults：配置默认参数，这些参数可以被用到frontend，backend，Listen组件；



* frontend：处理请求的虚拟节点，Frontend可以将匹配到本地区域的请求交给下边的backend；
* backend：后端服务集群的配置，是真实服务器，一个Backend对应一个或者多个实体服务器；
* ACL编写：

ACL语法：acl 名称 方法 -i [匹配的路径或文件] ##定义ACL，-i忽略大小写

use\_backend back名称 if ACL名称 ##指定符合ACL的流量，多个ACL之间和用空格表示，或用or表示

default\_backend 默认back名称 ##默认的模块

方法：hdr\_beg(host)：检测请求报文首部开头部分

hdr\_end(host)：检测请求报文首部结尾部分

hdr\_reg(host)：正则匹配

url\_sub：表示请求url包含什么字符串

url\_dir：便是请求url中存在哪些字符串作为部分地址路径

path\_beg：检测请求的URL是否匹配路径开头

path\_end：检测请求的URL是否匹配路径结尾

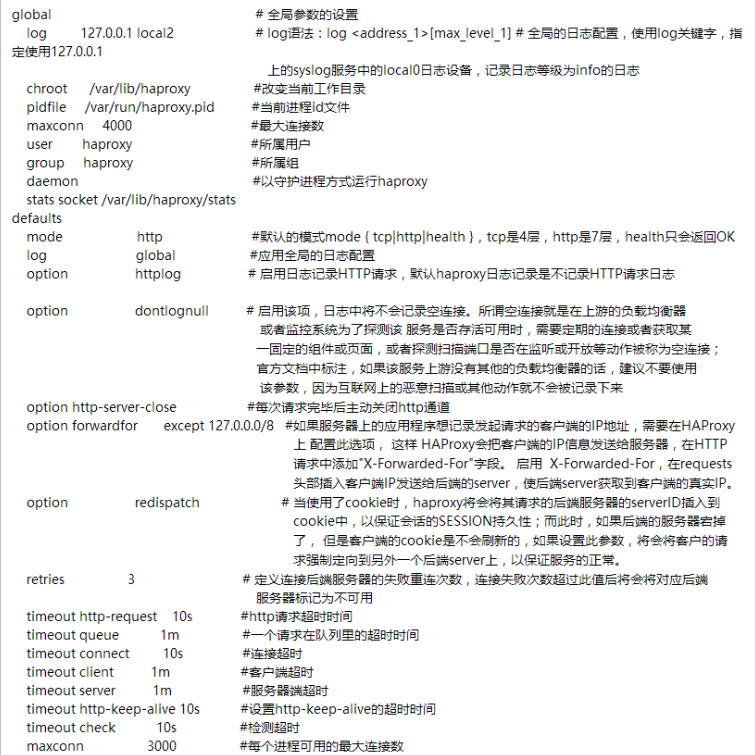
dst：目标地址

dst\_port：目标端口

src：源地址

src\_port：源端口

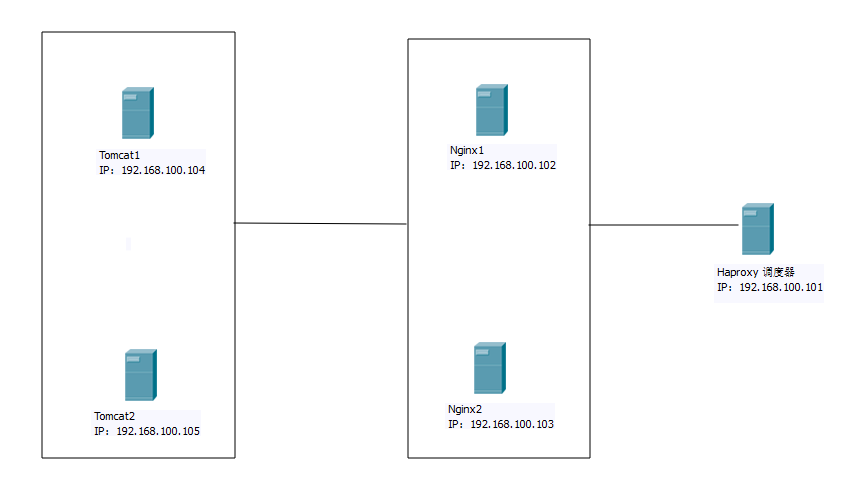
**配置文件样例：**





**五、案例：Haproxy+Nginx+Tomcat搭建高可用集群**

**案例拓扑：**



**案例环境：**

|  |  |  |  |
| --- | --- | --- | --- |
| 系统类型 | IP地址 | 主机名 | 所需软件 |
| Centos 7.4 1708 64bit | 192.168.100.101 | ha.linuxfan.cn | haproxy-1.4.24.tar.gz |
| Centos 7.4 1708 64bit | 192.168.100.102 | ng1.linuxfan.cn | nginx-1.12.2.tar.gz |
| Centos 7.4 1708 64bit | 192.168.100.103 | ng2.linuxfan.cn | nginx-1.12.2.tar.gz |
| Centos 7.4 1708 64bit | 192.168.100.104 | tm1.linuxfan.cn | jdk-8u171-linux-x64.tar.gz  apache-tomcat-9.0.10.tar.gz |
| Centos 7.4 1708 64bit | 192.168.100.105 | tm2.linuxfan.cn | jdk-8u171-linux-x64.tar.gz  apache-tomcat-9.0.10.tar.gz |

**案例步骤：**

* 搭建并配置nginx节点，准备网页，启动服务，测试节点（两台nginx配置相同，在此列出一台的配置）；
* 搭建并配置tomcat节点，准备网页，启动服务，测试节点（两台tomcat配置相同，在此列出一台的配置）；
* 安装Haproxy程序软件；
* 配置Haproxy服务；
* 启动Haproxy服务；
* 客户端访问测试验证集群；
* 在两台tomcat节点上传超市管理项目（两台tomcat配置相同，在此列出一台的配置）；
* 在tomcat1节点配置mysql数据库；
* 调整haproxy调度器的配置；
* 客户端测试访问验证集群；
* 将Haproxy的日志文件分离（便于查看）；
* 配置Haproxy服务的日志管理界面；
* 客户端访问测试Haproxy的日志管理界面；
* **搭建并配置nginx节点，准备网页，启动服务，测试节点（两台nginx配置相同，在此列出一台的配置）；**

[root@ng1 ~]# yum -y install pcre-devel zlib-devel

[root@ng1 ~]# useradd -M -s /sbin/nologin nginx

[root@ng1 ~]# tar zxvf nginx-1.12.2.tar.gz -C /usr/src/

[root@ng1 ~]# cd /usr/src/nginx-1.12.2/

[root@ng1 nginx-1.12.2]# ./configure --prefix=/usr/local/nginx --user=nginx --group=nginx --with-http\_stub\_status\_module

[root@ng1 nginx-1.12.2]# make && make install

[root@ng1 nginx-1.12.2]# cd

[root@ng1 ~]# ln -s /usr/local/nginx/sbin/nginx /usr/local/sbin/

[root@ng1 ~]# vi /usr/lib/systemd/system/nginx.service

[Unit]

Description=nginxapi

After=network.target

[Service]

Type=forking

PIDFile=/usr/local/nginx/logs/nginx.pid

ExecStart=/usr/local/nginx/sbin/nginx

ExecReload=kill -s HUP $(cat /usr/local/nginx/logs/nginx.pid)

ExecStop=kill -s QUIT $(cat /usr/local/nginx/logs/nginx.pid)

PrivateTmp=Flase

[Install]

WantedBy=multi-user.target

[root@ng1 ~]# echo "192.168.100.102" >>/usr/local/nginx/html/index.html

[root@ng1 ~]# systemctl start nginx

[root@ng1 ~]# systemctl enable nginx

[root@ng1 ~]# netstat -utpln |grep nginx

tcp 0 0 0.0.0.0:80 0.0.0.0:\* LISTEN 3538/nginx: master

* **搭建并配置tomcat节点，准备网页，启动服务，测试节点（两台tomcat配置相同，在此列出一台的配置）；**

[root@tm1 ~]# ls

apache-tomcat-9.0.10.tar.gz jdk-8u171-linux-x64.tar.gz

[root@tm1~]# rpm -qa |grep java

[root@tm1 ~]# tar zxvf jdk-8u171-linux-x64.tar.gz

[root@tm1 ~]# mv jdk1.8.0\_171/ /usr/local/java

[root@tm1 ~]# ls /usr/local/java

bin db javafx-src.zip lib man release THIRDPARTYLICENSEREADME-JAVAFX.txt

COPYRIGHT include jre LICENSE README.html src.zip THIRDPARTYLICENSEREADME.txt

[root@tm1 ~]# cat <<END >>/etc/profile

export JAVA\_HOME=/usr/local/java

export PATH=$PATH:/usr/local/java/bin

END

[root@tm1~]# source /etc/profile

[root@tm1 ~]# java -version

java version "1.8.0\_171"

Java(TM) SE Runtime Environment (build 1.8.0\_171-b11)

Java HotSpot(TM) 64-Bit Server VM (build 25.171-b11, mixed mode)

[root@tm1 ~]# tar zxvf apache-tomcat-9.0.10.tar.gz

[root@tm1 ~]# mv apache-tomcat-9.0.10 /usr/local/tomcat

[root@tm1 ~]# ls /usr/local/tomcat

bin conf lib LICENSE logs NOTICE RELEASE-NOTES RUNNING.txt temp webapps work

[root@tm1 ~]# vi /usr/local/tomcat/conf/server.xml

150 <Context docBase="/web/webapp" path="" reloadable="false"></Context>

:wq

[root@tm1 ~]# mkdir -p /web/webapp

[root@tm1 ~]# vi /web/webapp/index.jsp

<%@ page language="java" import="java.util.\*" pageEncoding="UTF-8"%>

<html>

<head>

<title>JSP TEST PAGE1 </title>

</head>

<body>

<% out.println("Welcome to test site;http://192.168.100.104");%>

</body>

</html>

[root@tm1 ~]# /usr/local/tomcat/bin/startup.sh ##启动apache-tomcat

[root@tm1 ~]# netstat -utpln |grep 8080

tcp 0 0 0.0.0.0:8080 0.0.0.0:\* LISTEN 14758/java

* **安装Haproxy程序软件；**

[root@ha ~]# yum -y install pcre-devel bzip2-devel ##安装依赖软件包

[root@ha ~]# tar zxvf haproxy-1.4.24.tar.gz -C /usr/src/

[root@ha ~]# cd /usr/src/haproxy-1.4.24/

[root@ha haproxy-1.4.24]# uname -r

3.10.0-693.5.2.el7.x86\_64

[root@ha haproxy-1.4.24# make TARGET=linux310 ##make编译，指定系统内核版本为3.10

[root@ha haproxy-1.4.24]# make install

[root@ha haproxy-1.4.24]# cd

* **配置Haproxy服务；**

[root@ha ~]# mkdir /etc/haproxy ##创建配置文件目录

[root@ha ~]# cp /usr/src/haproxy-1.4.24/examples/haproxy.cfg /etc/haproxy/

[root@ha ~]# vi /etc/haproxy/haproxy.cfg ##修改主配置文件如下

global

log 127.0.0.1 local0 info ##定义日志级别；

log 127.0.0.1 local1 notice

maxconn 4096 ##设定每个haproxy进程所接受的最大并发连接数

uid 99 ##指定运行服务的用户和组

gid 99

daemon ##指定运行模式为daemon，以守护进程的方式工作在后台

defaults

log global ##采取global中的日志配置

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

option httplog ##采用http日志格式记录日志

option dontlognull ##不记录健康检查的日志记录

option httpclose ##关闭保持连接

retries 3 ##检查节点最多失败次数

maxconn 2000 ##最大连接数，定义不得大于global中的值

contimeout 5000 ##连接超时时间，毫秒，在此期间，如若客户端与服务端无法成功建立连接，则断掉

clitimeout 50000 ##设置连接客户端发送数据时的成功连接最长等待时间，单位为毫秒，如若在这期间无法请求成功，则断掉

srvtimeout 50000 ##设置服务器端回应客户端数据发送的最长等待时间，如果在这期间还无法发送成功，则断掉

##################无分离页面需求的配置##############

#listen webcluster 0.0.0.0:80 ##指定haproxy服务监听地址和端口

# option httpchk GET /index.html ##指定http请求方法和默认文件

# balance roundrobin ##指定轮询调度算法

# server inst1 192.168.100.155:80 check inter 2000 fall 3 ##定义web节点，检测心跳频率，单位为毫秒，定义检查节点最多失败次数

# server inst2 192.168.100.156:80 check inter 2000 fall 3

##################有分离页面需求的配置##############

frontend http ##定义名称为http

bind \*:80 ##指定监听地址和端口

acl linuxfan1 hdr\_end(host) -i 192.168.100.101 ##指定类型为访问路径的域名，-i不区分大小写

acl linuxfan2 hdr\_end(host) -i www.linuxfan.cn

acl linuxfan3 path\_end -i .jsp .do .css .js ##指定请求文件格式为.jsp

#acl linuxfan3 hdr\_reg -i \.(css|png|jpg|jpeg|gif|ico|swf|xml|txt|pdf|do|jsp|js)$ ##调用正则表达式

acl linuxfan4 path\_end -i .html .css .png .jpg .jpeg .xml ##指定请求文件格式为.html

acl linuxfan5 path\_beg -i /WebRoot ##指定访问URL中的路径，如http://www.linuxfan.cn/WebRoot/index.jsp

use\_backend dongtai if linuxfan1 linuxfan3

use\_backend dongtai if linuxfan2 linuxfan3

use\_backend dongtai if linuxfan1 linuxfan5 linuxfan3

use\_backend dongtai if linuxfan2 linuxfan5 linuxfan3

default\_backend jingtai ##默认的请求使用backend dongtai

backend jingtai ##定义backend ：jingtai

mode http ##定义模式

balance roundrobin ##定义调度算法为轮询

server jingtai01 192.168.100.102:80 check inter 2000 fall 3 ##定义节点

server jingtai02 192.168.100.103:80 check inter 2000 fall 3

backend dongtai

mode http

balance roundrobin

server dongtai01 192.168.100.104:8080 check inter 2000 fall 3

server dongtai02 192.168.100.105:8080 check inter 2000 fall 3

* **启动Haproxy服务；**

[root@ha ~]# cp /usr/src/haproxy-1.4.24/examples/haproxy.init /etc/init.d/haproxy

[root@ha ~]# chmod +x /etc/init.d/haproxy

[root@ha ~]# ln -s /usr/local/sbin/haproxy /usr/sbin/

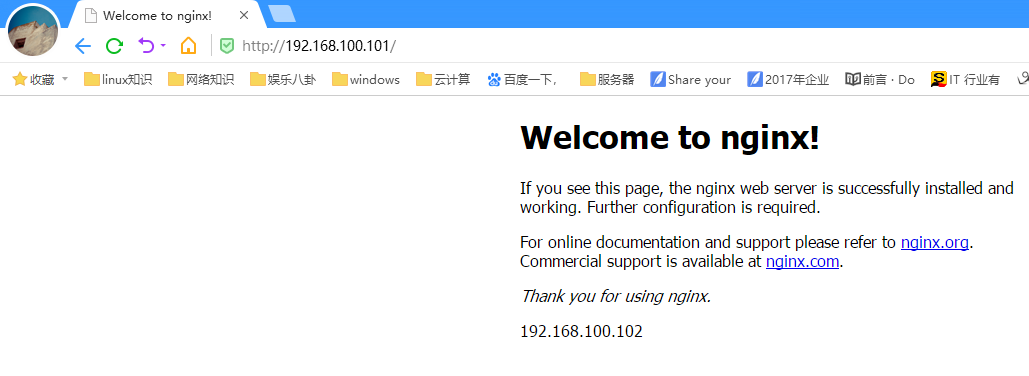
[root@ha ~]# /etc/init.d/haproxy start

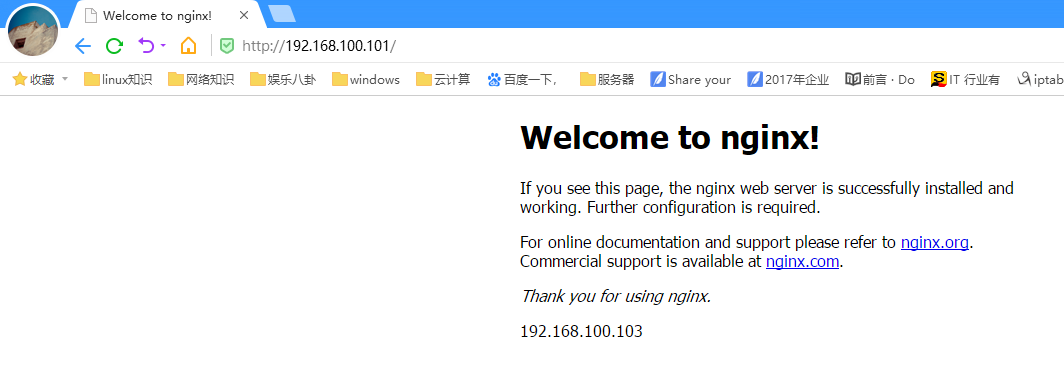
[root@ha ~]# netstat -utpln |grep haproxy

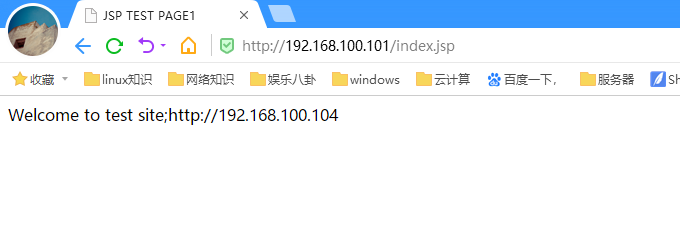
tcp 0 0 0.0.0.0:80 0.0.0.0:\* LISTEN 1264/haproxy

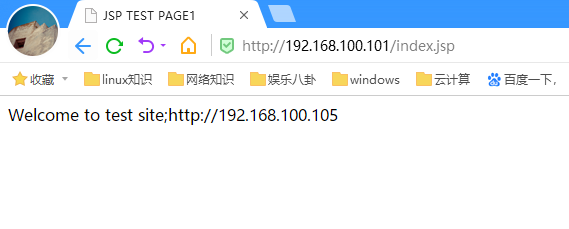
udp 0 0 0.0.0.0:52177 0.0.0.0:\* 1264/haproxy

* **客户端访问测试验证集群；**









* **在两台tomcat节点上传超市管理项目（两台tomcat配置相同，在此列出一台的配置）；**

[root@tm1 ~]# ls /usr/local/tomcat/webapps/WebRoot/ ##上传到服务器节点

calendar css error.jsp images index.jsp js jsp login.jsp META-INF WEB-INF

[root@tm1 ~]# vi /usr/local/tomcat/webapps/WebRoot/WEB-INF/classes/database.properties

driver=com.mysql.jdbc.Driver

#在和mysql传递数据的过程中，使用unicode编码格式，并且字符集设置为utf-8

url=jdbc:mysql://192.168.100.104:3306/smbms?useUnicode=true&characterEncoding=utf-8

user=linuxfan

password=123123

:wq

* **在tomcat1节点配置mysql数据库；**

[root@tm1 ~]# yum -y install mariadb-server mysql

[root@tm1 ~]# systemctl start mariadb

[root@tm1 ~]# systemctl enable mariadb

[root@tm1 ~]# mysqladmin -uroot password

[root@tm1 ~]# mysql -uroot -p123123<smbms\_db.sql

[root@tm1 ~]# mysql -uroot -p123123

MariaDB [(none)]> grant all on smbms.\* to 'linuxfan'@'192.168.100.104' identified by '123123';

MariaDB [(none)]> grant all on smbms.\* to 'linuxfan'@'192.168.100.105' identified by '123123';

MariaDB [(none)]> flush privileges;

MariaDB [(none)]> exit

* **调整haproxy调度器的配置；**

[root@ha ~]# vi /etc/haproxy/haproxy.cfg

...

use\_backend dongtai if linuxfan1 linuxfan3

use\_backend dongtai if linuxfan2 linuxfan3

use\_backend dongtai if linuxfan1 linuxfan5 linuxfan3

use\_backend dongtai if linuxfan2 linuxfan5 linuxfan3

use\_backend dongtai if linuxfan1 linuxfan5

use\_backend dongtai if linuxfan2 linuxfan5

...

backend dongtai

mode http

balance source

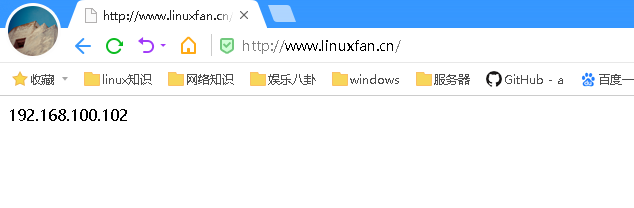
server dongtai01 192.168.100.104:8080 check inter 2000 fall 3

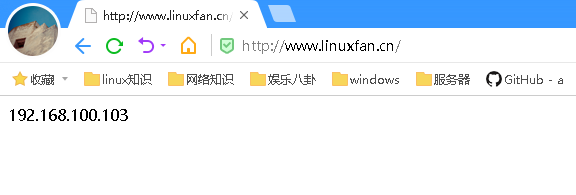
server dongtai02 192.168.100.105:8080 check inter 2000 fall 3

:wq

[root@ha ~]# /etc/init.d/haproxy restart

* **客户端测试访问验证集群；**









* **将Haproxy的日志文件分离（便于查看）；**

[root@ha ~]# vi /etc/rsyslog.conf

15 $ModLoad imudp

16 $UDPServerRun 514

##打开此配置项的注释，使rsyslog服务可以接受udp协议514端口的系统日志消息

74 local0.\* /var/log/haproxy/ha-info.log

75 local1.\* /var/log/haproxy/ha-notice.log

##在haproxy主配置文件中所指定的日志设备和日志级别将其输出到不同的文件中，以方便查看；

:wq

[root@ha ~]# vi /etc/sysconfig/rsyslog

SYSLOGD\_OPTIONS="-r -m 0 -c 2"

##-r 接收udp 514号端口的系统日志消息；-m 0 为日志消息添加时间等的标记 ；-c 2表示使用兼容模式

:wq

[root@ha ~]# systemctl restart rsyslog

[root@ha ~]# /etc/init.d/haproxy restart

[root@ha ~]# tail -f /var/log/haproxy/ha-info.log

Jul 27 18:14:44 localhost haproxy[1375]: 192.168.100.1:52965 [27/Jul/2018:18:14:44.167] http dongtai/dongtai01 0/0/0/2/3 200 349 - - ---- 0/0/0/0/0 0/0 "GET /index.jsp HTTP/1.1"

* **配置Haproxy服务的日志管理界面；**

[root@ha ~]# vi /etc/haproxy/haproxy.cfg ###在后边添加

listen status 0.0.0.0:8080 ##监控8080端口

stats enable ##开启stats的模块

stats uri /stats ##指定访问的url

stats auth admin:123456 ##指定登陆监控页面的用户名密码

stats realm (Haproxy\ statistic) ##定义页面的提示信息

[root@ha ~]# /etc/init.d/haproxy restart

* **客户端访问测试Haproxy的日志管理界面；**

