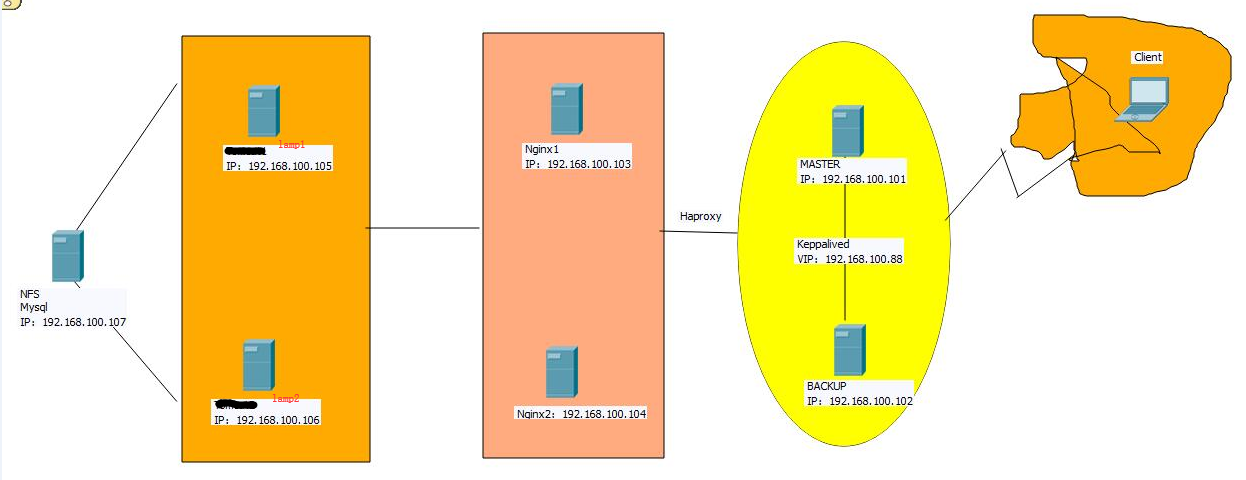
案例：Haproxy+Keepalived+Nginx+Lamp+Nfs实现高可用集群

**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 | ld1.linuxfan.cn | haproxy-1.4.24.tar.gz keepalived-1.2.13.tar.gz |
| Centos 7.4 1708 64bit | 192.168.100.102 | ld2.linuxfan.cn | haproxy-1.4.24.tar.gz keepalived-1.2.13.tar.gz |
| Centos 7.4 1708 64bit | 192.168.100.103 | ng1.linuxfan.cn | nginx-1.12.2.tar.gz |
| Centos 7.4 1708 64bit | 192.168.100.104 | ng2.linuxfan.cn | nginx-1.12.2.tar.gz |
| Centos 7.4 1708 64bit | 192.168.100.105 | lamp1.linuxfan.cn | httpd、mariadb-server、mysql、php、php-mysql（rpm包） |
| Centos 7.4 1708 64bit | 192.168.100.106 | lamp2.linuxfan.cn | httpd、mariadb-server、mysql、php、php-mysql（rpm包） |
| Centos 7.4 1708 64bit | 192.168.100.107 | st.linuxfan.cn | mariadb-server、rpcbind、nfs、mysql、  discuz\_7.2\_full\_sc\_utf8.zip |
|  | 192.168.100.88 | VIP地址  www.linuxfan.cn |  |

**案例思路：**

**案例重点：**

|  |
| --- |
| **Cookie与Session区别：**  **Cookie：**（储存在用户本地终端上的数据），Cookie 可以翻译为“小甜品，小饼干” ，Cookie 在网络系统中几乎无处不在，当我们浏览以前访问过的网站时，网页中可能会出现 ：你好 XXX，这会让我们感觉很亲切，就好像吃了一个小甜品一样。这其实是通过访问主机中的一个文件来实现的，这个文件就是 Cookie。  Cookie 在计算机中是个存储在浏览器目录中的文本文件，当浏览器运行时，存储在 RAM 中发挥作用 （此种 Cookies 称作 Session Cookies），一旦用户从该网站或服务器退出，Cookie 可存储在用户本地的硬盘上 （此种 Cookies 称作 Persistent Cookies） [3]  。  通常情况下，当用户结束浏览器会话时，系统将终止所有的 Cookie。当 Web 服务器创建了Cookies 后，只要在其有效期内，当用户访问同一个 Web 服务器时，浏览器首先要检查本地的Cookies，并将其原样发送给 Web 服务器。这种状态信息称作“Persistent Client State HTTP Cookie” ，简称为 Cookies [3]  。  **Session：**计算机术语，Session是指一个终端用户与交互系统进行通信的时间间隔，通常指从注册进入系统到注销退出系统之间所经过的时间。以及如果需要的话，可能还有一定的操作空间。  具体到Web中的Session指的就是用户在浏览某个网站时，从进入网站到关闭浏览器所经过的这段时间，也就是用户浏览这个网站所花费的时间。因此从上述的定义中我们可以看到，Session实际上是一个特定的时间概念。  **Cookie与Session区别：**  HTTP协议中，客户端请求服务端是一种无状态的连接-每次请求都是独立的请求。  **Cookie:**  如我们在网上购物的时候， 开始的时候先登录，然后选中商品加入到自己的购物车-其在客户单与服务器端的动作如下：  客户端—–（request:包含登录信息）——->服务器端  客户端<—（response:登录成功与否）——-服务器端  客户端—–（request:购物车里的信息）—–>服务器端  客户端<—（response:添加成功与否）—— -服务器端  这里关键是在第二次请求加入购物车的时候，因为客户端请求服务端是一种无状态的连接,那么服务器端怎么知道是谁，以及加入到谁的购物车。  解决的方式就是request到达服务器端的时候，服务器给在response中加一个“小饼干”,这个“小饼干”中就包含用户登录时候的一些基本的信息。当第二次客户端发起请求的时候，服务端检查到这个“小饼干”,据此识别出客户端，并进行相应的操作。而这个所谓的“小饼干”就是Cookie,持有这个小饼干的是客户端-即客户端保存中数据。    **Session：**  Cookie方法的是把数据保存在客户端，而Session是把数据保存在服务器，而客户端具有一个唯一的ID.  小例子与上述一样，只是在两个类的处理方法的时候用到的是Session.  **总结；**  **Cookie的优缺点；**  1.cookie数据存放在客户的浏览器上；  2.cookie不是很安全，别人可以分析存放在本地的COOKIE并进行COOKIE欺骗；  3.单个cookie保存的数据不能超过4K，很多浏览器都限制一个站点最多保存20个cookie；  **Session的优缺点；**  1.考虑到安全应当使用session。  2.session会在一定时间内保存在服务器上。当访问增多，会比较占用你服务器的性能    考虑到减轻服务器性能方面，应当使用COOKIE。 |

**案例步骤：**

* 部署两台nginx节点（两台节点配置相同，在此列举一台的配置）；
* 部署两台lamp节点（两台节点配置相同，在此列举一台的配置）；
* 安装配置后端存储节点的nfs服务并设置共享的数据；
* 安装配置后端存储节点的数据库服务，并配置项目中的数据库；
* 两台nginx节点挂载后端存储（两台节点配置相同，在此列举一台的配置）；
* 两台lamp节点挂载后端存储（两台节点配置相同，在此列举一台的配置）；
* 安装两台调度器节点的haproxy服务（两台节点配置相同，在此列举一台的配置）；
* 配置两台调度器节点的haproxy服务（两台节点配置相同，在此列举一台的配置）；
* 安装两台调度器节点的keepalived服务（两台节点配置相同，在此列举一台的配置）；
* 配置master主调度器并启动服务；
* 配置backup从调度器并启动服务；
* 客户端访问集群的静态网站；
* 客户端访问集群vip地址并安装bbs论坛项目；
* 模拟nginx1节点down，测试集群可用性；
* 模拟lamp1节点down，测试集群可用性；
* 模拟ld1主调度器节点down，测试集群可用性；

**案例实施：**

* **部署两台nginx节点（两台节点配置相同，在此列举一台的配置）；**

[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 "192.168.100.103" >/usr/local/nginx/html/index.html

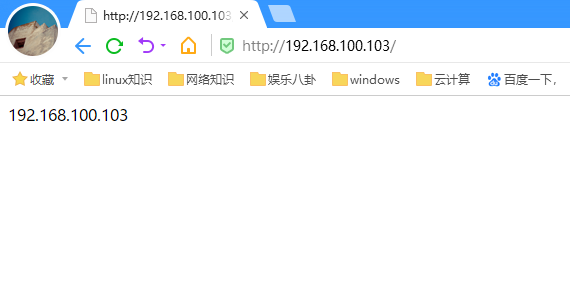
[root@ng1 ~]# systemctl start nginx

[root@ng1 ~]# systemctl enable nginx

Created symlink from /etc/systemd/system/multi-user.target.wants/nginx.service to /usr/lib/systemd/system/nginx.service.

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

tcp 0 0 0.0.0.0:80 0.0.0.0:\* LISTEN 3506/nginx: master



* **部署两台lamp节点（两台节点配置相同，在此列举一台的配置）；**

[root@lamp1 ~]# yum -y install httpd mariadb-server mysql php-mysql php

[root@lamp1 ~]# sed -i '/short\_open/s/Off/On/g' /etc/php.ini

[root@lamp1 ~]# systemctl start httpd mariadb

[root@lamp1 ~]# systemctl enable httpd mariadb

Created symlink from /etc/systemd/system/multi-user.target.wants/httpd.service to /usr/lib/systemd/system/httpd.service.

Created symlink from /etc/systemd/system/multi-user.target.wants/mariadb.service to /usr/lib/systemd/system/mariadb.service.

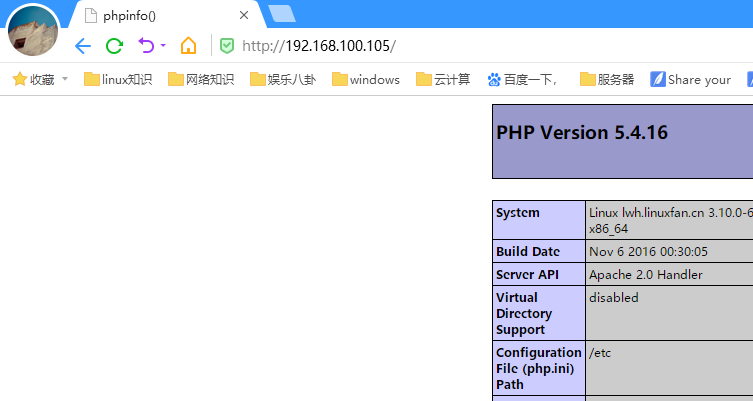
[root@lamp1 ~]# cat <<END >>/var/www/html/index.php

<?php

phpinfo();

?>

END



* **安装配置后端存储节点的nfs服务并设置共享的数据；**

[root@st ~]# yum -y install rpcbind nfs-utils

[root@st ~]# for i in rpcbind nfs;do systemctl enable $i; done

Created symlink from /etc/systemd/system/multi-user.target.wants/nfs-server.service to /usr/lib/systemd/system/nfs-server.service.

[root@st ~]# vi /etc/exports

/opt/nginx 192.168.100.0/24(rw,sync,no\_root\_squash)

/opt/lamp 192.168.100.0/24(rw,sync,no\_root\_squash)

[root@st ~]# mkdir /opt/nginx

[root@st ~]# mkdir /opt/lamp

[root@st ~]# chmod 777 /opt/nginx/

[root@st ~]# echo -e "this is a beautiful page\!\!\!" >>/opt/nginx/index.html

[root@st ~]# chmod 777 /opt/lamp/

[root@st ~]# unzip discuz\_7.2\_full\_sc\_utf8.zip

[root@st ~]# cp -rf upload/ /opt/lamp/bbs

[root@st ~]# ls /opt/nginx/

index.html

[root@st ~]# ls /opt/lamp/

bbs

[root@st ~]# systemctl start rpcbind

[root@st ~]# systemctl start nfs

Job for nfs-server.service failed because the control process exited with error code. See "systemctl status nfs-server.service" and "journalctl -xe" for details.

[root@st ~]# kill -HUP `cat /run/gssproxy.pid`

[root@st ~]# systemctl start nfs

[root@st ~]# showmount -e 192.168.100.107

Export list for 192.168.100.107:

/opt/lamp 192.168.100.0/24

/opt/nginx 192.168.100.0/24

[root@st ~]# cd /opt/lamp/bbs

[root@st bbs]# chmod 777 forumdata/ attachments/ uc\_client/data/cache/ templates/ config.inc.php -R

[root@st bbs]# cd

* **安装配置后端存储节点的数据库服务，并配置项目中的数据库；**

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

[root@st ~]# systemctl start mariadb

[root@st ~]# systemctl enable mariadb

Created symlink from /etc/systemd/system/multi-user.target.wants/mariadb.service to /usr/lib/systemd/system/mariadb.service.

[root@st ~]# mysqladmin -uroot password

123123

123123

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

MariaDB [(none)]> create database bbsdb;

MariaDB [(none)]> grant all on bbsdb.\* to 'linuxfan'@'192.168.100.%' identified by '123123';

MariaDB [(none)]> flush privileges;

MariaDB [(none)]> exit

* **两台nginx节点挂载后端存储（两台节点配置相同，在此列举一台的配置）；**

[root@ng1 ~]# yum -y install nfs-utils rpcbind

[root@ng1 ~]# echo "192.168.100.107:/opt/nginx /usr/local/nginx/html/ nfs defaults,\_netdev 0 0" >>/etc/fstab

[root@ng1 ~]# mount -a

[root@ng1 ~]# ls /usr/local/nginx/html/

**index.html**

* **两台lamp节点挂载后端存储（两台节点配置相同，在此列举一台的配置）；**

[root@lamp1 ~]# yum -y install nfs-utils rpcbind

[root@lamp1 ~]# echo "192.168.100.107:/opt/lamp /var/www/html/ nfs defaults,\_netdev 0 0" >>/etc/fstab

[root@lamp1 ~]# mount -a

[root@lamp1 ~]# ls /var/www/html/

bbs

* **安装两台调度器节点的haproxy服务（两台节点配置相同，在此列举一台的配置）；**

[root@ld1 ~]# yum -y install pcre-devel bzip2-devel

[root@ld1 ~]# tar zxvf haproxy-1.4.24.tar.gz -C /usr/src/

[root@ld1 ~]# cd /usr/src/haproxy-1.4.24/

[root@ld1 haproxy-1.4.24]# make TARGET=linux310

[root@ld1 haproxy-1.4.24]# make install

[root@ld1 haproxy-1.4.24]# cd

[root@ld1 ~]# mkdir /etc/haproxy

[root@ld1 ~]# cp /usr/src/haproxy-1.4.24/examples/haproxy.init /etc/init.d/haproxy

[root@ld1 ~]# chmod +x /etc/init.d/haproxy

[root@ld1 ~]# ln -s /usr/local/sbin/haproxy /usr/sbin/

* **配置两台调度器节点的haproxy服务（两台节点配置相同，在此列举一台的配置）；**

[root@ld1 ~]# vi /etc/haproxy/haproxy.cfg

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，不能注释，注释掉后则无法切换nginx与lamp

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 www.linuxfan.cn

acl linuxfan2 hdr\_end(host) -i 192.168.100.88

acl linuxfan4 hdr\_reg -i .\*\.(php|css|jsp|js|do|png|jpg|jpeg)$

acl linuxfan5 path\_beg -i /wordpress

acl linuxfan6 path\_beg -i /bbs

use\_backend dongtai if linuxfan5 or linuxfan6

use\_backend dongtai if linuxfan1 linuxfan5 linuxfan4

use\_backend dongtai if linuxfan2 linuxfan5 linuxfan4

use\_backend dongtai if linuxfan1 linuxfan4

use\_backend dongtai if linuxfan2 linuxfan4

default\_backend jingtai

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 192.168.100.103:80 check inter 2000 fall 3 ##定义节点

server jingtai02 192.168.100.104:80 check inter 2000 fall 3

backend dongtai

mode http

balance roundrobin

server dongtai01 192.168.100.105:80 check inter 2000 fall 3

server dongtai02 192.168.100.106:80 check inter 2000 fall 3

:wq

[root@ld1 ~]# /etc/init.d/haproxy start

[root@ld1 ~]# netstat -utpln |grep 80

tcp 0 0 0.0.0.0:80 0.0.0.0:\* LISTEN 28122/haproxy

* **安装两台调度器节点的keepalived服务（两台节点配置相同，在此列举一台的配置）；**

[root@ld1 ~]# yum -y install kernel-devel openssl-devel popt-devel

[root@ld1 ~]# tar -zxvf keepalived-1.2.13.tar.gz -C /usr/src/

[root@ld1 ~]# cd /usr/src/keepalived-1.2.13/

[root@ld1 keepalived-1.2.13]# ./configure --prefix=/usr/local/keepalived

[root@ld1 keepalived-1.2.13]# make &&make install

[root@ld1 keepalived-1.2.13]# cd

[root@ld1 ~]# mkdir -p /etc/keepalived

[root@ld1 ~]# cp /usr/local/keepalived/etc/keepalived/keepalived.conf /etc/keepalived/

[root@ld1 ~]# cp /usr/local/keepalived/etc/sysconfig/keepalived /etc/sysconfig/

[root@ld1 ~]# cp /usr/local/keepalived/etc/rc.d/init.d/keepalived /etc/init.d/

[root@ld1 ~]# cp /usr/local/keepalived/sbin/keepalived /usr/sbin/

[root@ld1 ~]# chmod 755 /etc/init.d/keepalived

* **配置master主调度器并启动服务；**

[root@ld1 ~]# vi /etc/keepalived/keepalived.conf

global\_defs {

router\_id HA\_TEST\_R1 ##本服务器的名称，若环境中有多个keepalived时，此名称不能一致

}

vrrp\_instance VI\_1 { ##定义VRRP热备实例，每一个keep组都不同

state MASTER ##MASTER表示主服务器

interface eth0 ##承载VIP地址的物理接口

virtual\_router\_id 1 ##虚拟路由器的ID号，每一个keep组都不同

priority 100 ##优先级，数值越大优先级越高

advert\_int 1 ##通告检查间隔秒数（心跳频率）

authentication { ##认证信息

auth\_type PASS ##认证类型

auth\_pass 123456 ##密码字串

}

virtual\_ipaddress {

192.168.100.88 ##指定漂移地址（VIP）

}

}

virtual\_server 192.168.100.88 80 { #vip配置

persistence\_timeout 60 ##会话保持时间

protocol TCP ##选择协议

real\_server 192.168.100.101 80 { ##本机地址

weight :3 ##服务器的权重

notify\_down /etc/keepalived/check.sh ##real server检测失败后执行的脚本

TCP\_CHECK {

connect\_timeout 10 ##连接超时时间

nb\_get\_retry 3 ##重连次数

delay\_before\_retry 3 ##重连间隔时间

connect\_port 80 ##健康检查端口

}

}

}

[root@ld1 ~]# vi /etc/keepalived/check.sh

#!/bin/bash

/etc/init.d/keepalived stop

:wq

[root@ld1 ~]# chmod +x /etc/keepalived/check.sh

[root@ld1 ~]# /etc/init.d/keepalived start

[root@ld1 ~]# ip a|grep 192.168.100.88

inet 192.168.100.88/32 scope global eth0

* **配置backup从调度器并启动服务；**

[root@ld2 ~]# vi /etc/keepalived/keepalived.conf

global\_defs {

router\_id HA\_TEST\_R2 ##本服务器的名称

}

vrrp\_instance VI\_1 {

state BACKUP ##BACKUP表示从服务器

interface eth0

virtual\_router\_id 1

priority 99 ##优先级，低于主服务器

advert\_int 1

authentication {

auth\_type PASS

auth\_pass 123456

}

virtual\_ipaddress {

192.168.100.88

}

}

virtual\_server 192.168.100.88 80 { ##vip配置

persistence\_timeout 60 ##会话保持时间

protocol TCP

real\_server 192.168.100.102 80 { ##本机地址

weight :3

notify\_down /etc/keepalived/check.sh ##real server检测失败后执行的脚本

TCP\_CHECK {

connect\_timeout 10 ##连接超时时间

nb\_get\_retry 3 ##重连次数

delay\_before\_retry 3 ##重连间隔时间

connect\_port 80 ##健康检查端口

}

}

}

[root@ld2~]# vi /etc/keepalived/check.sh

#!/bin/bash

/etc/init.d/keepalived stop

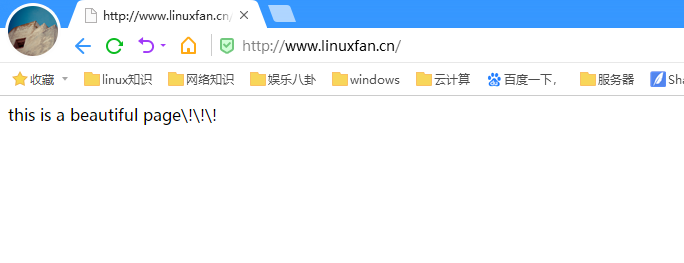
:wq

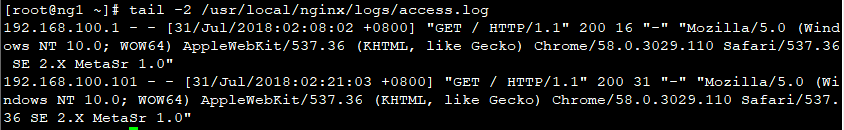
[root@ld2~]# chmod +x /etc/keepalived/check.sh

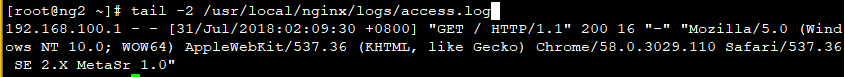
[root@ld2 ~]# /etc/init.d/keepalived start

[root@ld2 ~]# ip a|grep 192.168.100.88

* **客户端访问集群的静态网站；**







* **客户端访问集群vip地址并安装bbs论坛项目；**







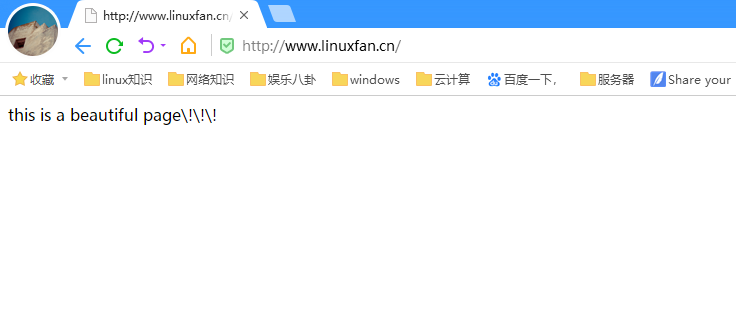




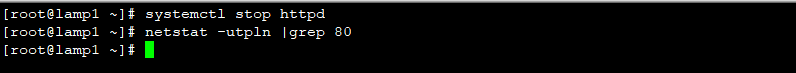


* **模拟nginx1节点down，测试集群可用性；**





* **模拟lamp1节点down，测试集群可用性；**





* **模拟ld1主调度器节点down，测试集群可用性；**

[root@ld1 ~]# ip a|grep 192.168.100.88

inet 192.168.100.88/32 scope global eth0

[root@ld1 ~]# /etc/init.d/haproxy stop

Stopping haproxy (via systemctl): [ 确定 ]

[root@ld1 ~]# ip a|grep 192.168.100.88

[root@ld1 ~]# ip a|grep 192.168.100.88

[root@ld2 ~]# ip a|grep 192.168.100.88

inet 192.168.100.88/32 scope global eth0

