**大型网站高并发解决方案LVS（Linux virtual server，也叫集群代理服务器）**

**集群功能分类**

**LB：load balancing，负载均衡，有一定高可用能力，但不是高可用集群，是以提高服务的并发处理能力为根本。**

**软件负载均衡设备**

**LVS：四层路由设备，根据用户请求的IP与端口号，实现将用户的请求分发至不同的主机。**

**HAproxy**

**硬件负载均衡设备：F5:BIG IP Citrix, Netscaler A10 深信服**

**HA：High Availability，高可用集群（增加服务可用性）**

**高可用集群是以提升服务的始终在线能力为着眼点，不会因为宕机而导致服务不能用。衡量可用性：在线时间/（在线时间+故障处理时间）**

**99%=一年有三天不在线 99.9%=一年有0.3天不在线**

**keepalived技术**

**负载均衡实现方式**

**http重定向：下载网站使用较多，工作在应用层的业务代码中。原理是根据用户的http请求计算出一个真实的web服务器地址，并将该web服务器地址写入http重定向响应中返回给客户浏览器，由浏览器重新进行访问。**

**优点：比较简单**

**缺点：客户浏览器需要多次请求服务器才能完成一次访问，性能较差。**

**DNS负载均衡：DNS负载提供域名到IP解析的过程，我们实例查看百度的域名解析其实是一对多的。这时，DNS服务器就充当了负载均衡。原理：在DNS服务器上配置多个域名对应IP的记录。例如一个域名www.baidu.com对应一组web服务器IP地址，域名解析时经过DNS服务器的算法将一个域名请求分配到适合的真实服务器上。**

**反向代理负载均衡：nginx**

**IP网络层负载均衡：在网络层修改IP地址进行负载均衡**

**数据链路层负载均衡：在数据链路层修改Mac地址进行负载均衡**

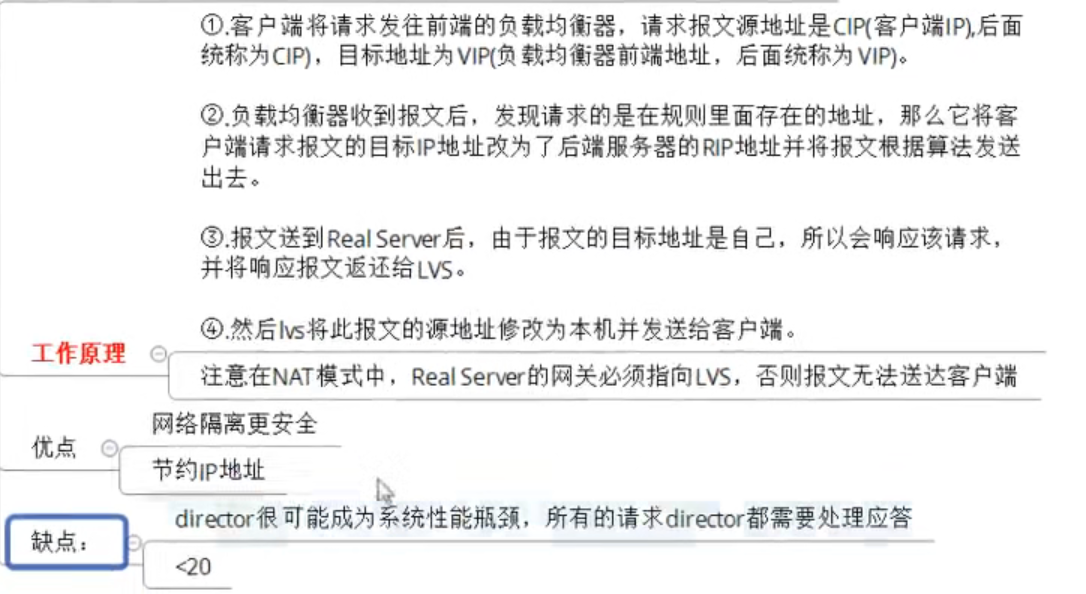
**四层负载和七层负载：所谓四层就是基于ip+端口的负载均衡，主要代表有lvs。**

**七层负载也称内容交换，就是基于URL等应用层信息的负载均衡，主要代表有nginx。**

**LVS：LVS工作在一台server上提供Directory（负载均衡器）的功能，本身并不提供服务，只是把特定的请求转发给对应的real server（真正提供服务的主机），从而实现集群环境中的负载均衡。**

**LVS工作模式**

**LVS-NAT转发模式：Network Address Translation。NAT就是一种更换IP地址的方案，在整个集群中只有NAT服务器自己连接了外网，在外部通过NAT服务器的IP访问到NAT服务器，然后NAT服务器会在这个请求上封装一层内网提供web服务主机的IP去找该主机，此时集群内部是内网通信，然后web主机反馈之后NAT服务器会拆掉封装，以自己的外网IP返回给用户。**

****

**准备NAT实验环境：**

**打开vmware，点击编辑，选择虚拟网络编辑器，点击添加网络添加两个新的网络。**

**准备四台机器，其中两两分开，每两台连接一个网络。点击虚拟机的网络适配器，选择自定义，更改网络。**

**四台机器就形成了两个不同的网段。**

**地址分配：客户和LVS接在VMNET0，另外两个作为服务器接在另一个网络。LVS需要增加一张网卡（在vm中设置），一个用来接外网，一个用来接服务器内网。在部署LVS的虚拟机上选择设置，添加一个网络适配器，将网络选择服务器所在的网络。**

**在服务器主机上执行：route add -net 192.168.0.0/24 gw 192.168.152.130 ##192.168.0.0是外网的网段，192.168.152.130是LVS的内网IP。**

**LVS上：echo 1 > /proc/sys/net/ipv4/ip\_forward ##开启路由功能**

**yum install -y ipvsadm**

**ipvsadm -A -t 192.168.3.36:80 -s rr ##此ip为LVS的外网IP**

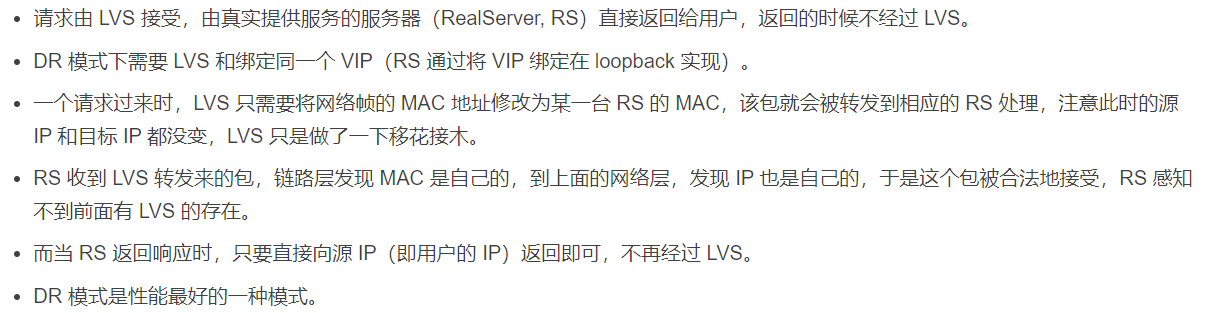
**ipvsadm -a -t 192.168.3.36:80 -r 192.168.152.128:80 -m**

**ipvsadm -a -t 192.168.3.36:80 -r 192.168.152.129:80 -m**

**##去网站服务器上改网卡配置！！！！把网卡配置改成手动，然后网关设置为LVS的内网IP！！！关闭桥接网卡！！！！（如果网卡配置是dhcp好像就不会有这种问题）**

**DR（direct routing）直接路由模式**

**客户端将请求发往前段的负载均衡器，请求报文源地址时CIP，目标地址为VIP。LVS受到报文后，发现请求的是在规则里面存在的地址，那么它将客户端请求报文的源MAC地址改为自己DIP（即与服务器通信的内网IP）的MAC，并将此包发送给服务器。服务器发现请求报文中的目的MAC是自己，就会将次报文接受，处理完请求后将响应报文通过IO接口送给网卡直接发送给客户端。**

****

**DR实验**

**准备四台同网段的机器。给LVS和两台服务器配置一个共有的虚拟IP，这个IP是提供给外网访问的。**

**ifconfig ens33:0 192.168.108.123 broadcast 192.168.108.255 netmask 255.255.255.0 up**

**##ens33是自己的网卡名字，0代表了接口，就是给ens33的0号接口配置上192.168.108.123这个IP，这个IP不会影响本来的真实IP。broadcast是指广播域。**

**route add -host 192.168.108.123 dev ens33:0**

**##将192.168.108.123添加到主机路由，指定这个路由使用ens33:0通信。**

****

**ipvsadm -C ##清除之前的所有配置**

**ipvsadm -A -t 192.168.108.123:80 -s rr**

**ipvsadm -a -t 192.168.108.123:80 -r 192.168.108.135:80 -g**

**ipvsadm -a -t 192.168.108.123:80 -r 192.168.108.128:80 -g**

**在两个服务器上部署web服务**

**给服务器配置虚拟ip：ifconfig lo:0 192.168.108.123/32**

**##lo为虚拟网卡，32是为了创造出特有的子网掩码，防止ip地址重复造成冲突。**

**给两个服务器设置内核参数**

**echo 1 > /proc/sys/net/ipv4/conf/all/arp\_ignore**

**##忽略arp响应，不允许收，就是外部有人通过vip来访问时，两个服务器会忽略掉，而只有LVS会去响应**

**echo 2 > /proc/sys/net/ipv4/conf/all/arp\_announce**

**##为了让vip发包出去，允许发**

****

**传统的NAT模式，DIR和RS必须在同一个VLAN下，否则 DIR无法作为RS的网关。**

**这引发的两个问题是：**

**1 同一个 VLAN的限制导致[运维](https://cloud.tencent.com/solution/operation?from=10680" \t "https://cloud.tencent.com/developer/article/_blank)不方便，跨VLAN的RS无法接入。**

**2 当RS横向扩容时，总有一天其上的单点DIR会成为瓶颈。**

**Full-NAT解决的是DIR和RS跨VLAN的问题，而跨VLAN问题解决后，DIR和RS不再存在VLAN上的从属关系，可以做到多个DIR对应多个RS，解决水平扩容的问题。**

**Full-NAT相比NAT的主要改进是，在SNAT/DNAT的基础上，加上另一种转换，转换过程如下：**

**IMG_256**

**在包从DIR转到 RS 的过程中，源地址从客户端IP被替换成了LVS 的内网IP。内网IP之间可以通过多个交换机跨VLAN通信。**

**当RS处理完接受到的包，返回时，会将这个包返回给DIR的内网IP，这一步也不受限于 VLAN。**

**DIR收到包后，在NAT模式修改源地址的基础上，再把RS发来的包中的目标地址从DIR内网IP改为客户端的 IP。**

**Full-NAT主要的思想是把网关和其下机器的通信，改为了普通的网络通信，从而解决了跨VLAN 的问题。采用这种方式，DIR和RS的部署在VLAN上将不再有任何限制，大大提高了运维部署的便利性。**

**HA：High Availability，高可用集群（增加服务可用性）**

**一般会采用一主一从，主服务器与从服务器通过一根网线连接，互相监听对方的状态，如果主服务器一旦发生故障，从服务器会立刻取代主服务器的位置。**

**脑裂：主从服务器之间的心跳线断开，双方互相断开联系，默认判定对方发生故障，于是便会造成资源抢夺，数据损坏等。**

**解决脑裂的方案：添加更多的心跳线。磁盘锁，正在服务一方锁住共享磁盘。设置仲裁机制。对脑裂进行监控报警。**

**keepalived：集群管理中保证集群高可用的一个服务软件，用来防止单点故障。**

**keepalived是以vrrp协议为基础实现的，vrrp全称virtual router redundancy protocol，即虚拟路由冗余协议。**

**工作原理：将N台提供相同功能的服务器组成一个服务器组，这个组里面有一个master和多个backup，master上面有一个对外提供服务的vip（该服务器所在局域网内其他机器的默认路由为该vip），master会发组播，当backup收不到vrrp包时就认为master宕机了，这时就需要根据vrrp的优先级来选举一个backup当master。**

**keepalived的三个主要模块**

**core：keepalived的核心，负责主进程的启动、维护以及全局配置文件的加载和解析。**

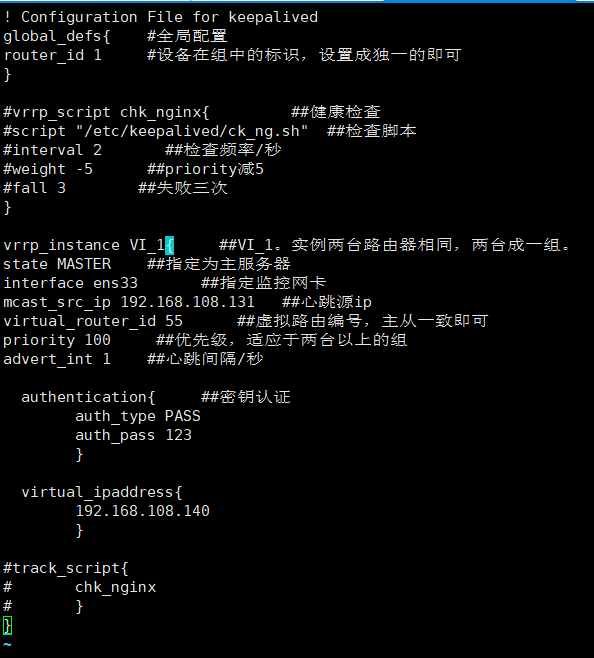
**check：负责健康检查，包括常见的各种检查方式。**

**vrrp：实现VRRP协议。**

**组建HA集群**

**在服务器上安装keepalived**

**编辑配置文件vim /etc/keepalived/keepalived.conf**

****

**记得每个{前面要空格！图片上没有空格，会报错！**

**注意两个机器配置需要改动router\_id和mcast\_src\_id**

**配置nginx**

**启动nginx**

**启动keepalived**

**访问vip**

**断开MASTER的网络连接或者stop掉keepalived**

**页面转移到BACKUP机器上了**

**当我们使用systemctl stop network/nginx这样的方式来测试时，我们会发现VIP并没有转移，网站无法访问。因为keepalived只关心自己和IP，并不关心某个服务是否启动，只有自己停止或者服务器IP丢失的时候才会进行转移VIP。**

**解决keepalived不感应nginx的方法**

**（想到的一个方法就是编写一个脚本监测nginx，大概思路是if test一下nginx是否在运行中，如果nginx处于关闭状态，那么我们执行systemctl stop keepalived就能实现VIP转移。）**

**正确方案：添加nginx监控脚本 （跟我的想法一样）**

**vim /etc/keepalived/ck\_ng.sh**

**#!/bin/bash**

**counter=$(ps -C nginx --no-heading | wc -l)**

**if test ${counter}==0**

**then**

**systemctl restart nginx**

**sleep5**

**counter=$(`ps -C nginx --no-heading | wc -l`)**

**if test ${counter}==0**

**then**

**systemctl stop keepalived**

**fi**

**fi**

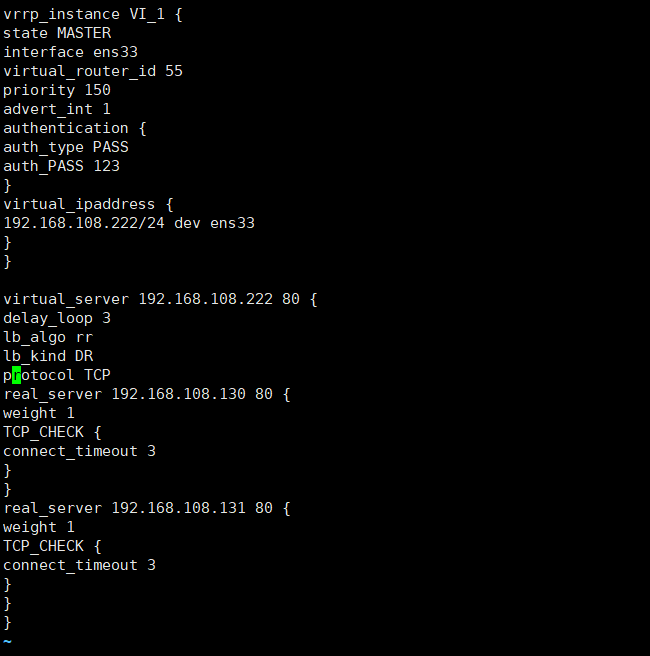
**chmod +x /etc/keepalived/ck\_ng.sh**

**将之前配置文件的注释取消掉**

**keepalived+LVS**

**准备四台同一网段的机器，其中两台做LVS，两台做Web服务器**

**安装ipvsadm时不用启动它**

****

**在RS上配置VIP的脚本**

#!/bin/bash

SNS\_VIP=172.17.13.252

case "$1" in

start)

ifconfig lo:0 $SNS\_VIP netmask 255.255.255.255 broadcast $SNS\_VIP

/sbin/route add -host $SNS\_VIP dev lo:0

echo "1" >/proc/sys/net/ipv4/conf/lo/arp\_ignore

echo "2" >/proc/sys/net/ipv4/conf/lo/arp\_announce

echo "1" >/proc/sys/net/ipv4/conf/all/arp\_ignore

echo "2" >/proc/sys/net/ipv4/conf/all/arp\_announce

sysctl -p >/dev/null 2>&1

echo "RealServer Start OK" ;;

stop)

ifconfig lo:0 down

route del $SNS\_VIP >/dev/null 2>&1

echo "0" >/proc/sys/net/ipv4/conf/lo/arp\_ignore

echo "0" >/proc/sys/net/ipv4/conf/lo/arp\_announce

echo "0" >/proc/sys/net/ipv4/conf/all/arp\_ignore

echo "0" >/proc/sys/net/ipv4/conf/all/arp\_announce

echo "RealServer Stoped" ;;

\*)

echo "Usage: $0 {start|stop}"

exit 1

esac

exit 0

**第二台LVS记得改配置**

**web服务器安装服务，配置虚拟地址**

**cp -R /etc/sysconfig/network-scripts /tmp ##备份**

**vim /etc/sysconfig/network-scripts/ifcfg-lo:0**

**DEVICE=lo:0**

**IPADDR=192.168.108.222**

**NETMASK=255.255.255.0**

**ONBOOT=yes**

**其他行注释掉**

**配置路由vim /etc/rc.local**

**/sbin/route add -host 192.168.108.222 dev lo:0**

**配置ARP vim /etc/sysctl.conf**

**net.ipv4.conf.all.arp\_ignore = 1**

**net.ipv4.conf.add.arp\_announce = 2**

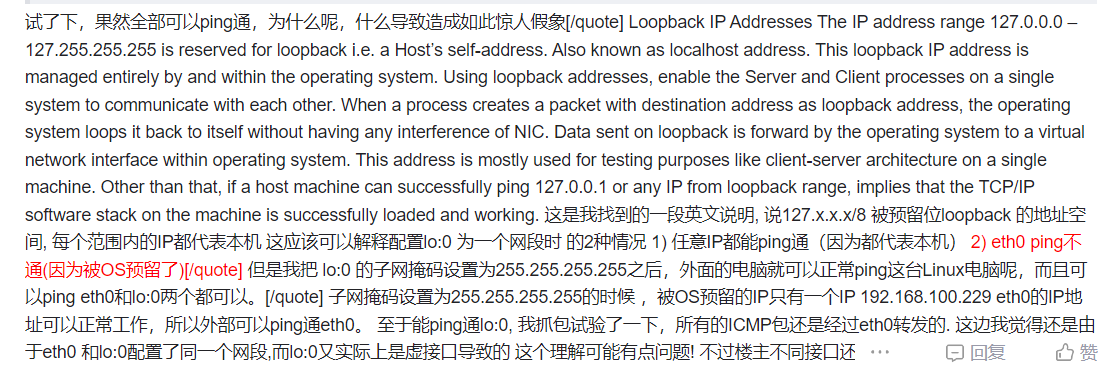
**net.ipv4.conf.default.arp\_ignore = 1**

**net.ipv4.conf.default.arp\_announce = 2**

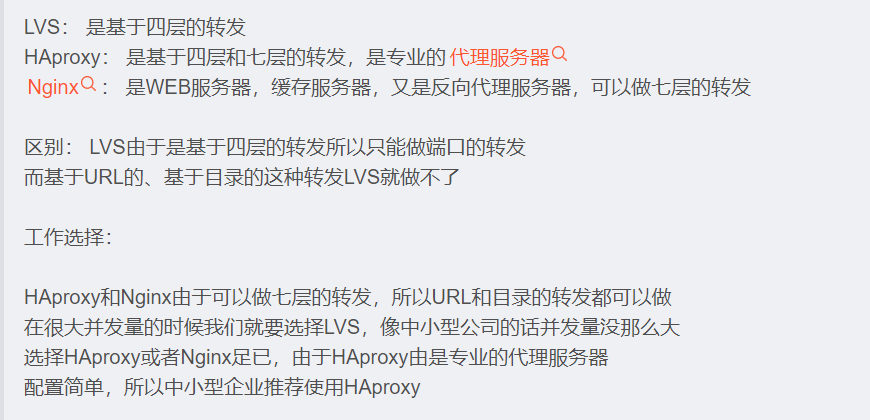
**net.ipv4.conf.lo.arp\_ignore = 1**

**net.ipv4.conf.lo.arp\_announce = 2**

**请将设置里的192.168.108.222/24改为/32，同时rs添加lo:0时也将子网掩码设置为255.255.255.255，否则实验不成功！！！**

****

**7层负载均衡：根据用户请求的内容将请求转发到不同的后端服务器，用于实现动静分离。之前的4层负载均衡是按照IP地址来划分，7层负载均衡是按照URL来划分。**

****

**HAproxy：一款高性能的负载均衡软件，专注负载均衡技术，比nginx更专业。**

**实验环境：四台机器，一台用户，一台负均衡，两台服务器。**

**服务器关闭防火墙，部署好网站。**

**安装HAproxy：yum -y install epel-release haproxy**

**配置HAproxy：vim /etc/haproxy/haproxy.cfg**

**HAproxy配置主要有以下五部分：**

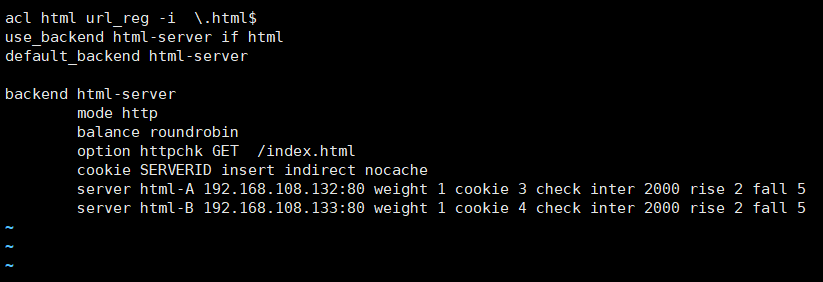
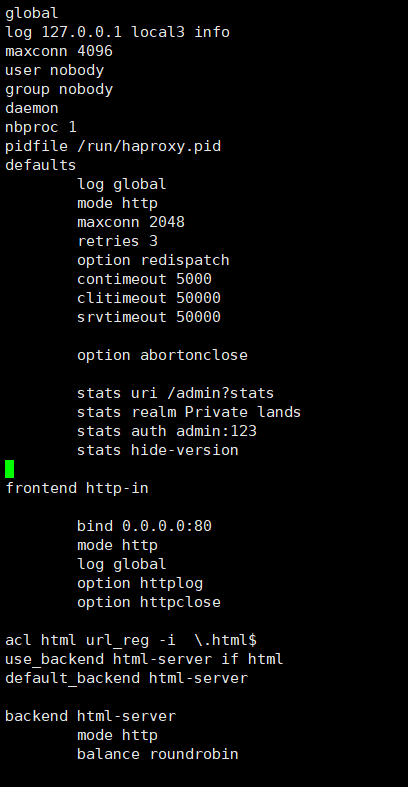
**global:设置全局配置参数，属于进程的配置，通常是和操作系统相关。**

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

**frontend:接收请求的前段虚拟节点，frontend可以更加规则直接指定具体使用后端的backend**

**backend：后端服务器集群的配置，是真实服务器，一个backend对应一个或多个实体服务器**

**Listen：frontend和backend的组合体。**

****

[学习haproxy基础配置，这一篇就够了 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/356921154)

**192.168.108.131/admin?stats ##访问haproxy自带的监控网页**

**haproxy+keepalived实现动静分离**

**132,133,137,138为服务器，其中133和138是php，137有jpg和png。131和136是haproxy服务器。139和140是LVS服务器并且互相之间做了keepalived。**

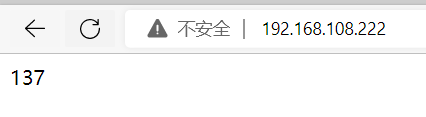
**实验成功**

**实验效果：访问192.168.108.222这个VIP时，多次刷新能看到后端四台服务器轮流响应。**

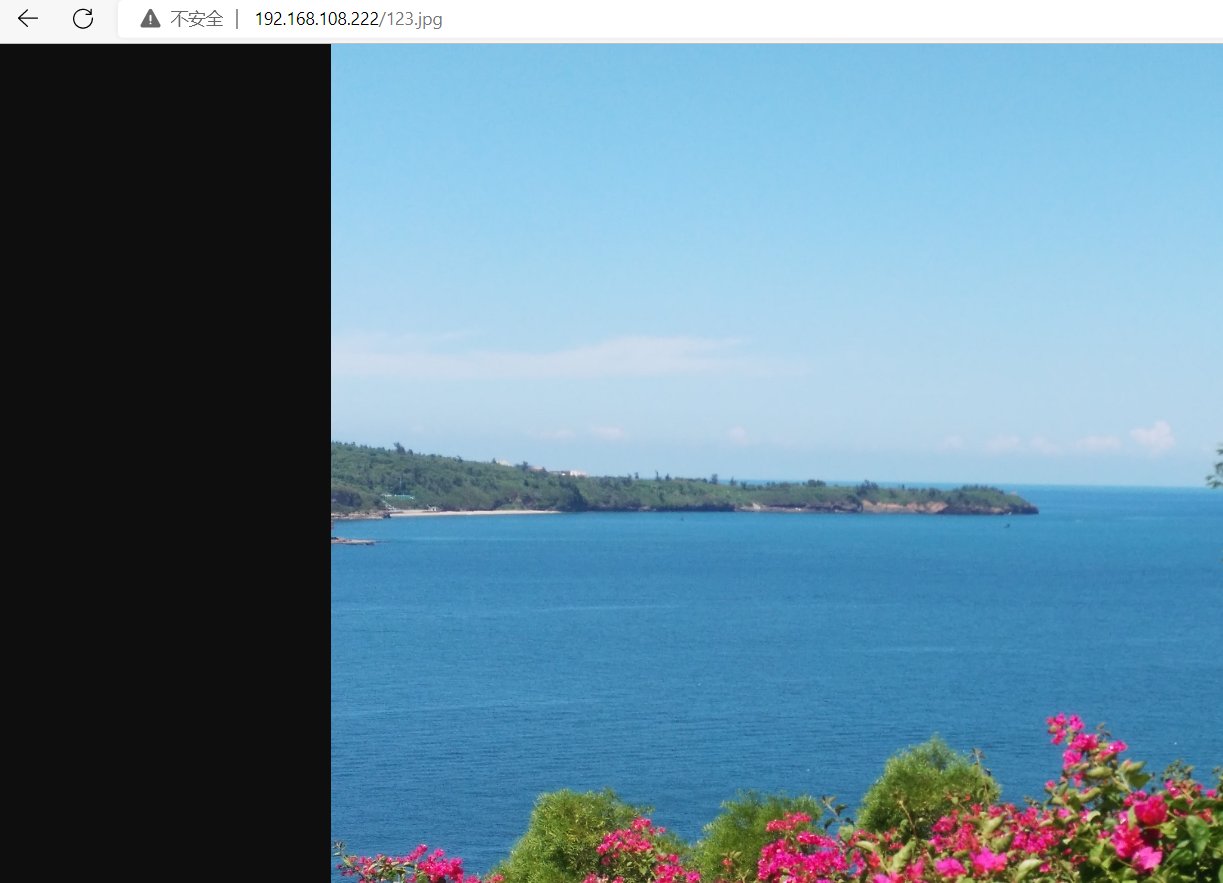
****

****

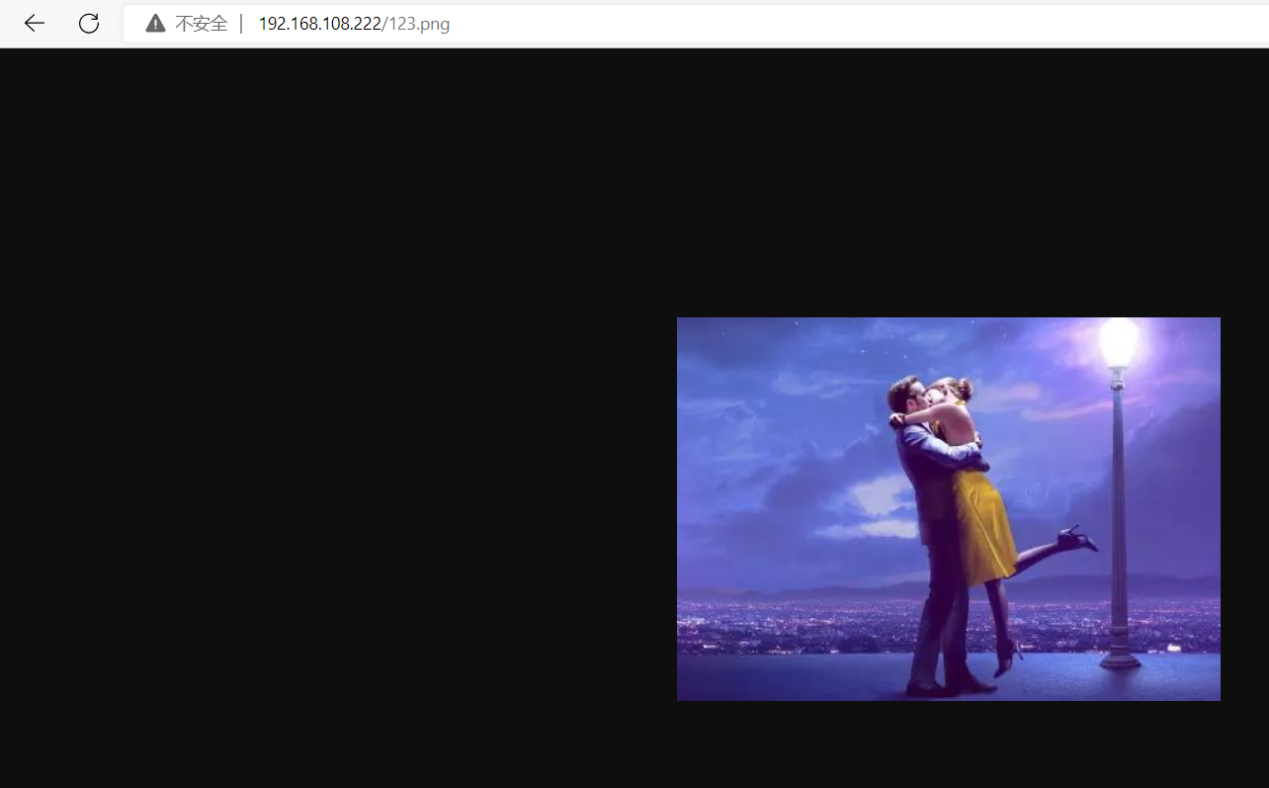
****

****

**访问192.168.108.222/123.jpg**

****

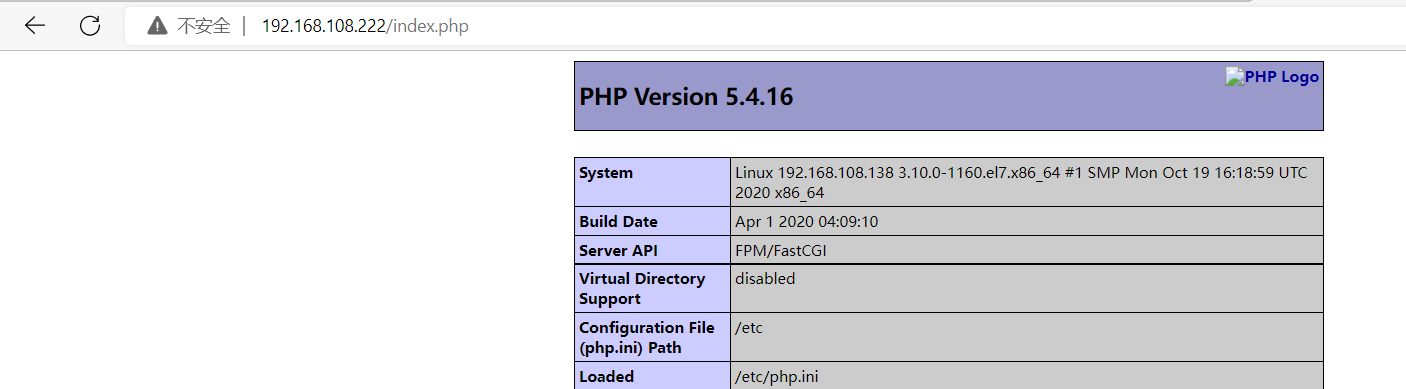
**访问192.168.108.222/123.png**

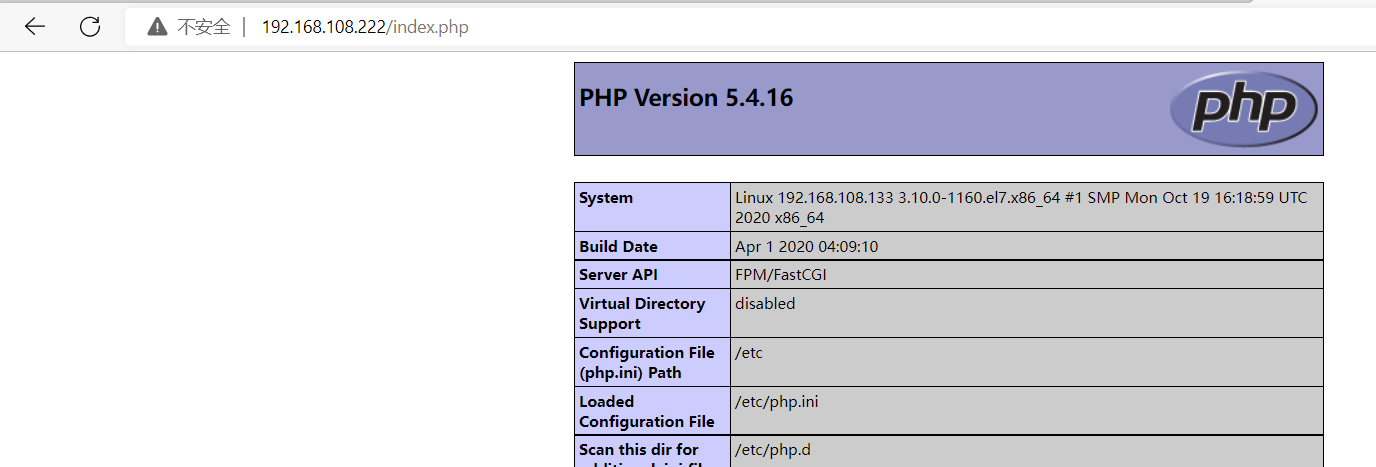
****

**可以看到存放在137上的图片**

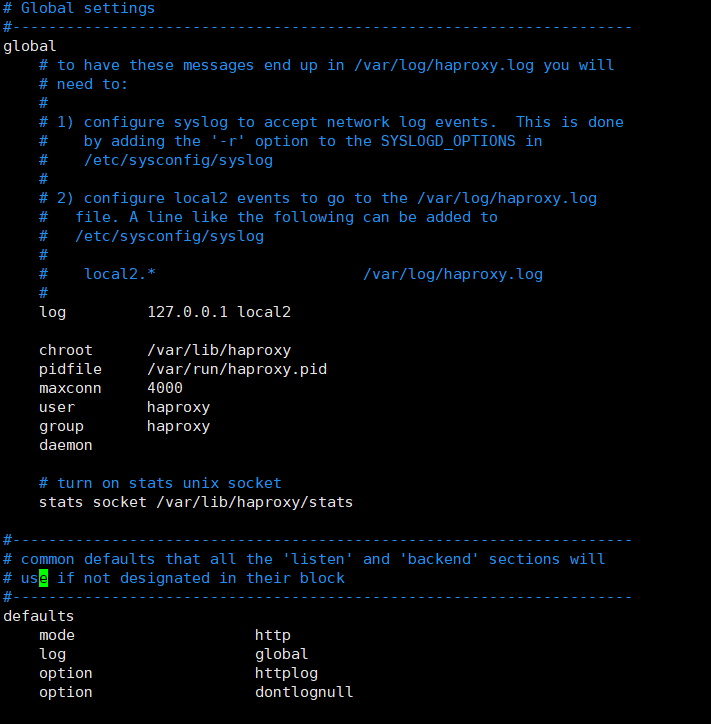
**访问192.168.108.222/index.php**

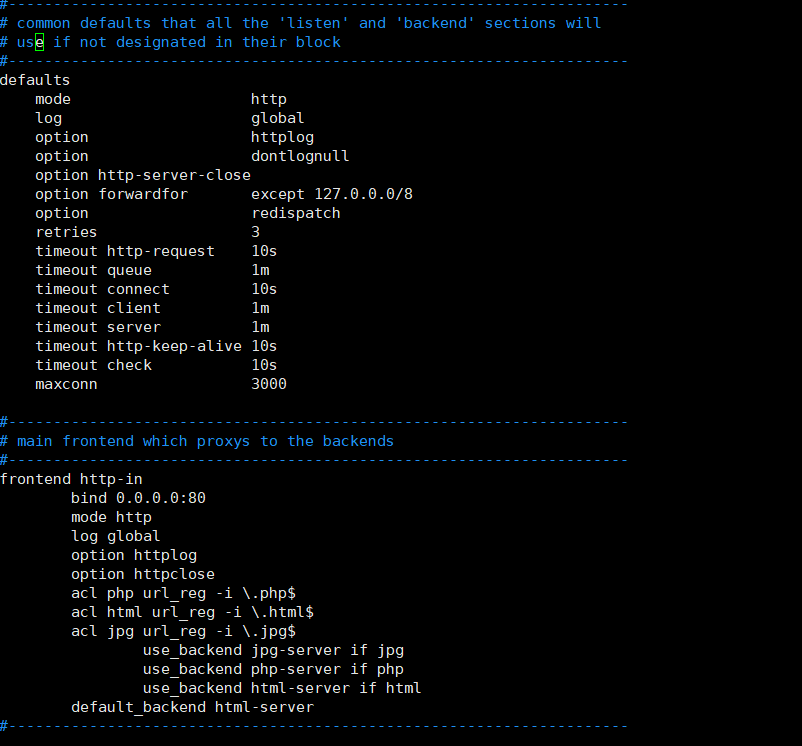
**刷新可以看到133和138上的php**

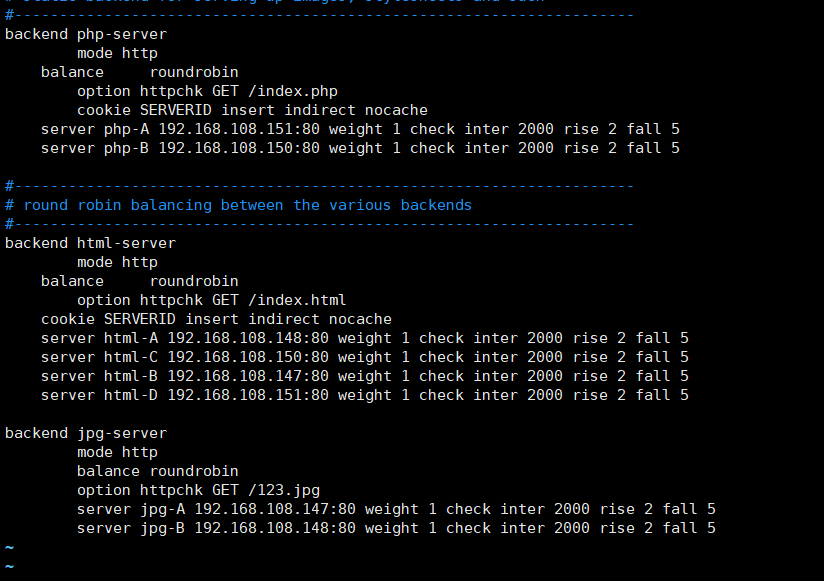
****

****

**第二次实验时成功的配置**

****

****

****

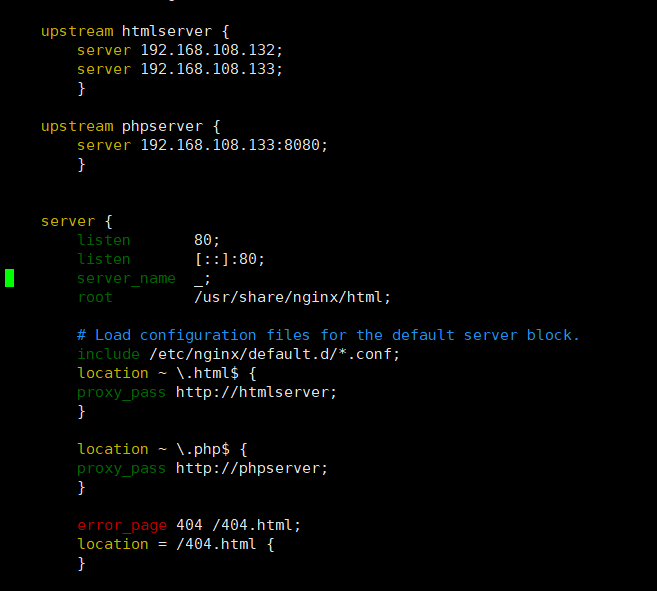
**其实只有frontend和backend部分是自己编写的，其他都是保持官方的设定没有动。如果在server的每行加上cookie，那么真机访问html静态页面时不会出现转跳。**

**如果在配置里没有设置有关jpg的配置，（我只在147和148上面放了各一张不同的图片）在刷新时会出现两次访问失败，因为haproxy默认会遍历四个服务器，所以手动设定jpg访问规则后就不会出现这样的情况。**

**第二次用了147,148,150,151,152,153,154,155一共8台设备，其中147,148存放了图片，150和151配置了php，154和155是haproxy，实现147,148,150,151的负载均衡，152与153做了keepalived，152,153,154,155之间做了LVS，152为154,155进行代理并且实现154和155的负载均衡。**

**nginx7层负载均衡**

**实验成功，在上面实验的基础上，将HAproxy的两台机器改为nginx代理即可，需要修改nginx.conf里的配置。**

****

**注意要配置好对应服务器的8080端口。**