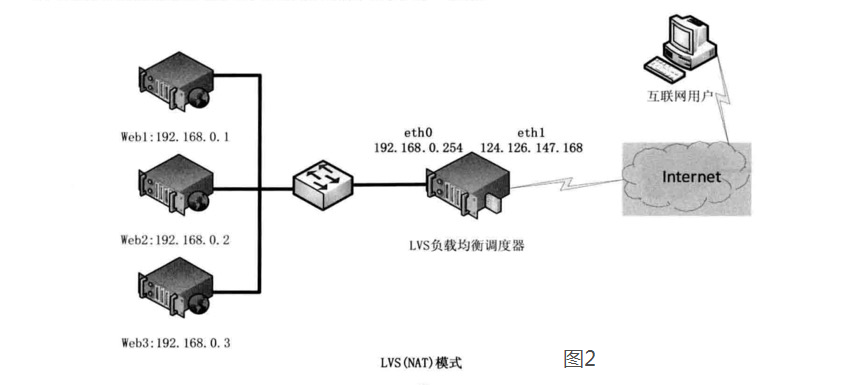
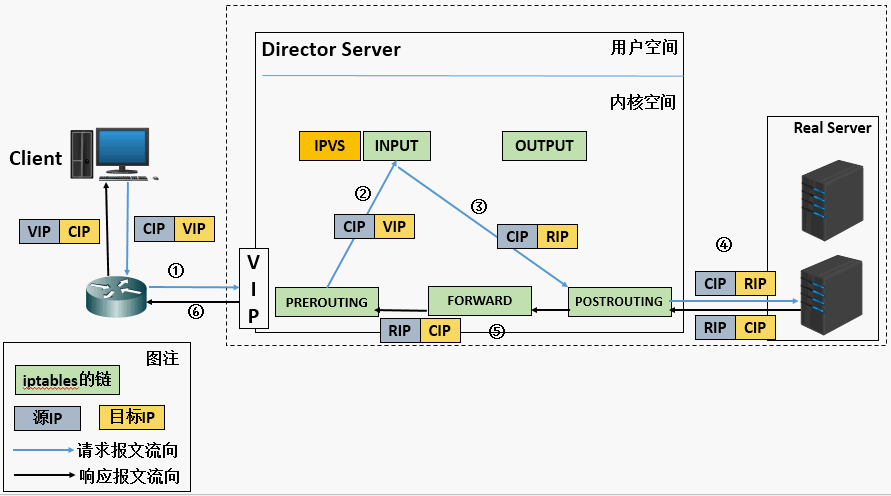
## 初遇lvs

### lvs原理介绍

#### lvs的四种调度模式

##### 基于NAT





NAT即网络地址转换，使得企业内部的私有ip地址可以访问外网，以及外部地址可以访问位于公司内部的私有ip地址。

基于nat的访问流程

请求：用户通过互联网dns服务器解析到公司负载均衡上的外网地址， director将请求的报文目的ip改为RIP，同时将目标端口也改为Realserver的相应端口，最后将报文转发给Realserver

回答：Realserver将响应报文返回给director，director将源地址和端口改为VIP及director相应端口，修改完成后将响应报文发送给用户

由于lvs调度器有一个连接hash表，该表中会记录连接请求及转发信息，当同一个连接的下一个数据包发送给调度器后，从hash表中可以直接找到以前的连接记录，并根据记录选出相同的Realserver及端口

NAT特性：

NAT模式修改的是目的ip及端口，直接走的是swith不需要修改mac地址

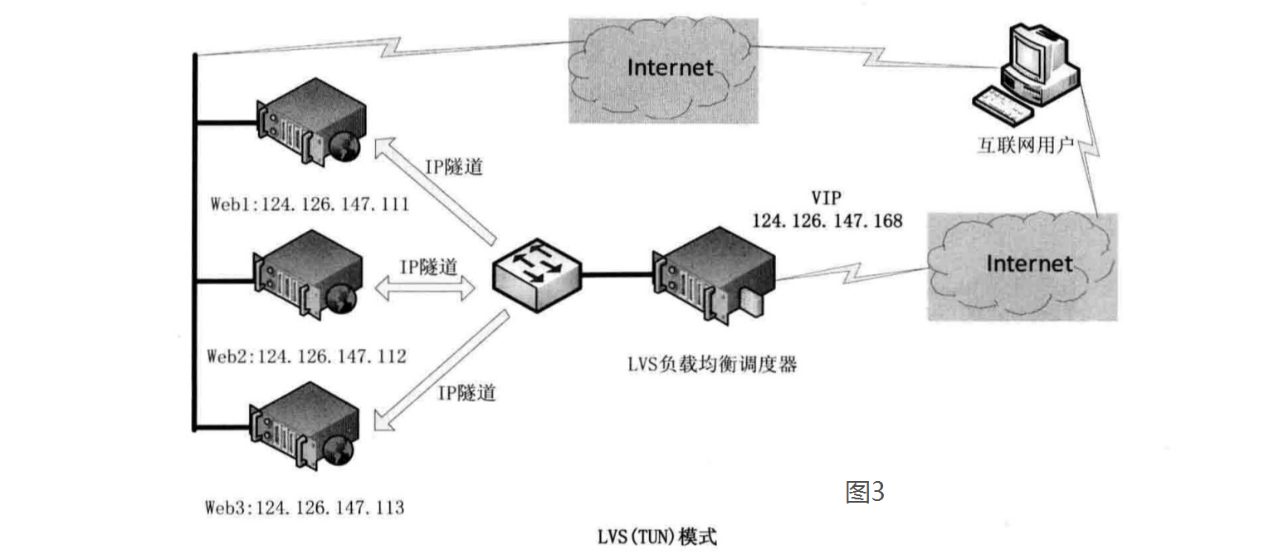
NAT模式下数据包的进出都要经过lvs调度器，所以lvs调度器会成为系统的瓶颈

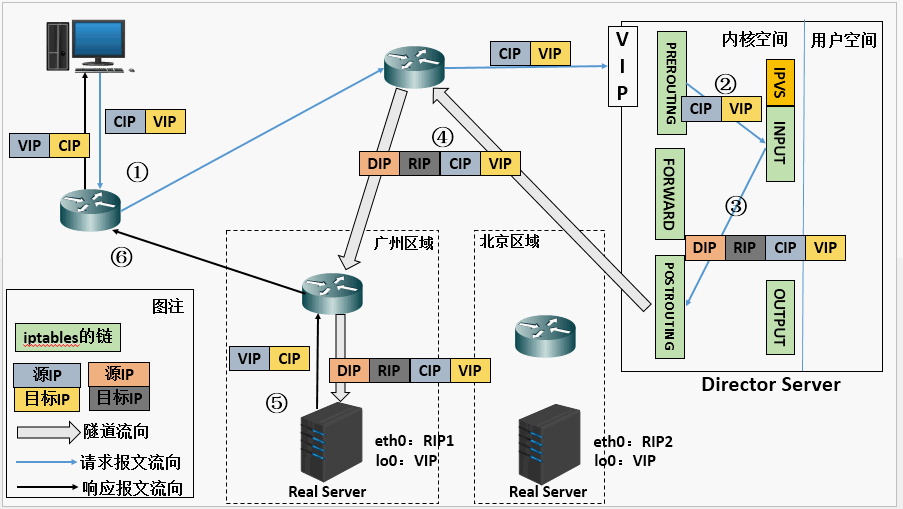
支持端口映射

节省公用ip地址

Realserver可以使用任意操作系统

##### 基于TUN





TUN（IP隧道是一种数据包封装技术，可以将原始数据包封装并添加新的包头包括新的源地址及端口、目标地址及端口）

请求：客户端发送请求给director，director通过IP-TUN加密将原始数据包封装，并在其基础上添加新的数据包头（修改目标地址为director选出来的Realserver的地址及端口）

回应：Realserver收到director的数据包后将添加的新的数据包头剥离，将响应数据包直接发送给客户端

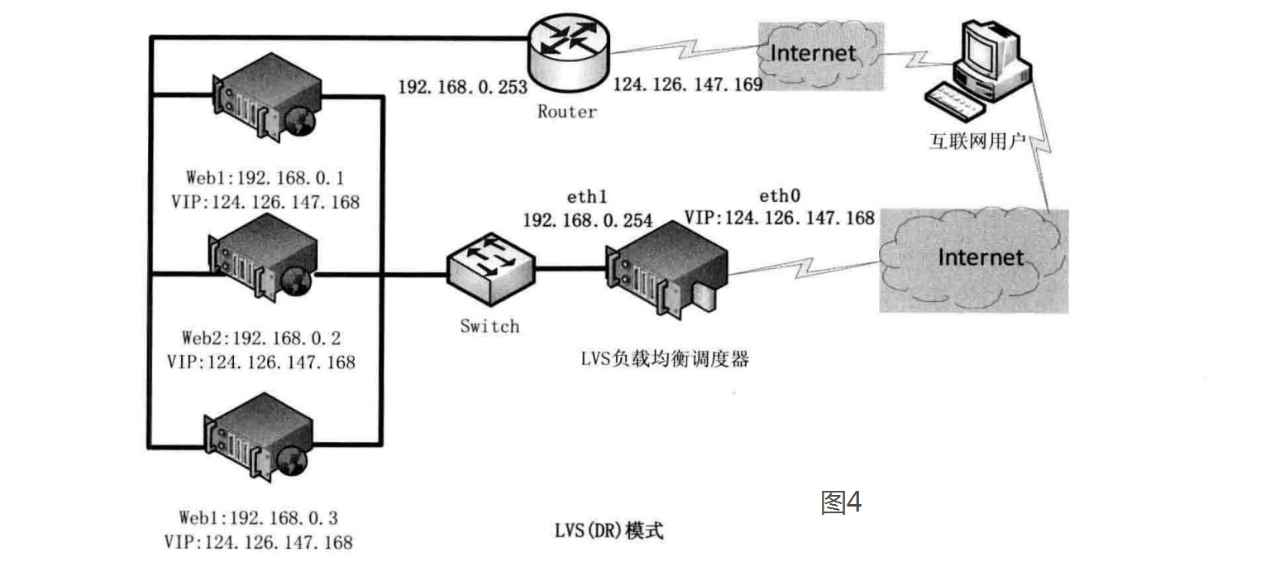
TUN特性：

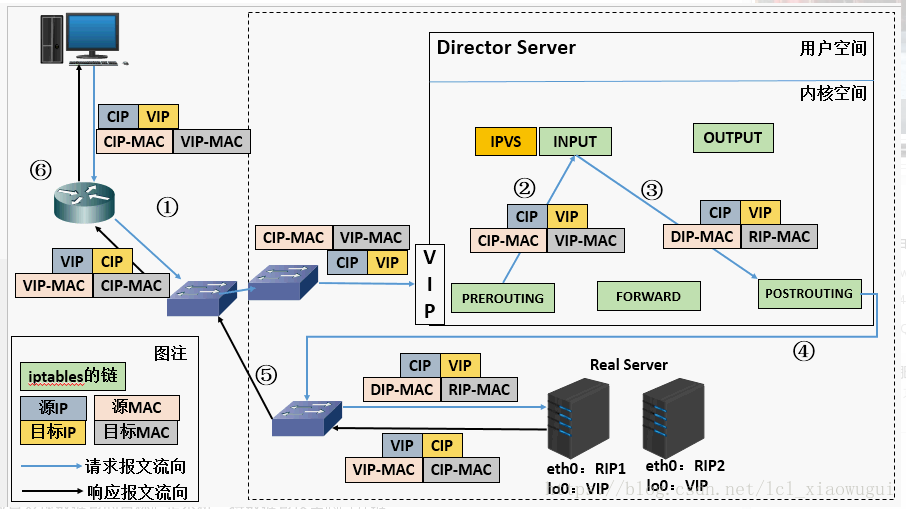
TUN必须所有Realserver绑定VIP

Realserver直接将包发给client

Realserver可以直接与外网通信

##### 基于DR





请求：用户将请求发给director，director将请求的报文目的mac地址改为后端Realserver的mac地址，然后将报文转发给Realserver

回应：Realserver的回应报文源地址为VIP，mac地址为director VIP的mac地址

DR特性：

Realserver和director必须有一个网卡在同一物理网络内，可以使用公用地址也可以使用私有地址

如果director有两个网卡，客户端只能访问一个地址，vip为局域网地址，需要打开路由转发

所有的Realserver上都有VIP地址

所有请求报文都经过director，响应报文不经过director

所有的真实服务器的VIP地址必须配置在Non-ARP的网络设备上，也就是该网络设备并不会向外广播自己的mac地址及对应ip

不支持端口映射

不能跨局域网

#### 2、lvs的调度算法

##### 2、1.轮询调度

轮询调度（Round Robin 简称'RR'）算法就是按依次循环的方式将请求调度到不同的服务器上，该算法最大的特点就是实现简单。轮询算法假设所有的服务器处理请求的能力都一样的，调度器会将所有的请求平均分配给每个真实服务器。

##### 2.2.加权轮询调度

加权轮询（Weight Round Robin 简称'WRR'）算法主要是对轮询算法的一种优化与补充，LVS会考虑每台服务器的性能，并给每台服务器添加一个权值，如果服务器A的权值为1，服务器B的权值为2，则调度器调度到服务器B的请求会是服务器A的两倍。权值越高的服务器，处理的请求越多。

##### 2.3.最小连接调度

最小连接调度（Least Connections 简称'LC'）算法是把新的连接请求分配到当前连接数最小的服务器。最小连接调度是一种动态的调度算法，它通过服务器当前活跃的连接数来估计服务器的情况。调度器需要记录各个服务器已建立连接的数目，当一个请求被调度到某台服务器，其连接数加1；当连接中断或者超时，其连接数减1。

（集群系统的真实服务器具有相近的系统性能，采用最小连接调度算法可以比较好地均衡负载。)

##### 2.4.加权最小连接调度

加权最少连接（Weight Least Connections 简称'WLC'）算法是最小连接调度的超集，各个服务器相应的权值表示其处理性能。服务器的缺省权值为1，系统管理员可以动态地设置服务器的权值。加权最小连接调度在调度新连接时尽可能使服务器的已建立连接数和其权值成比例。调度器可以自动问询真实服务器的负载情况，并动态地调整其权值。

##### 2.5.基于局部的最少连接

基于局部的最少连接调度（Locality-Based Least Connections 简称'LBLC'）算法是针对请求报文的目标IP地址的 负载均衡调度，目前主要用于Cache集群系统，因为在Cache集群客户请求报文的目标IP地址是变化的。这里假设任何后端服务器都可以处理任一请求，算法的设计目标是在服务器的负载基本平衡情况下，将相同目标IP地址的请求调度到同一台服务器，来提高各台服务器的访问局部性和Cache命中率，从而提升整个集群系统的处理能力。LBLC调度算法先根据请求的目标IP地址找出该目标IP地址最近使用的服务器，若该服务器是可用的且没有超载，将请求发送到该服务器；若服务器不存在，或者该服务器超载且有服务器处于一半的工作负载，则使用'最少连接'的原则选出一个可用的服务器，将请求发送到服务器。

##### 2.6.带复制的基于局部性的最少连接

带复制的基于局部性的最少连接（Locality-Based Least Connections with Replication 简称'LBLCR'）算法也是针对目标IP地址的负载均衡，目前主要用于Cache集群系统，它与LBLC算法不同之处是它要维护从一个目标IP地址到一组服务器的映射，而LBLC算法维护从一个目标IP地址到一台服务器的映射。按'最小连接'原则从该服务器组中选出一一台服务器，若服务器没有超载，将请求发送到该服务器；若服务器超载，则按'最小连接'原则从整个集群中选出一台服务器，将该服务器加入到这个服务器组中，将请求发送到该服务器。同时，当该服务器组有一段时间没有被修改，将最忙的服务器从服务器组中删除，以降低复制的程度。

##### 2.7.目标地址散列调度

目标地址散列调度（Destination Hashing 简称'DH'）算法先根据请求的目标IP地址，作为散列键（Hash Key）从静态分配的散列表找出对应的服务器，若该服务器是可用的且并未超载，将请求发送到该服务器，否则返回空。

不管IP，请求特定的东西，都定义到同一个RS上。

##### 2.8.源地址散列调度

源地址散列调度（Source Hashing 简称'SH'）算法先根据请求的源IP地址，作为散列键（Hash Key）从静态分配的散列表找出对应的服务器，若该服务器是可用的且并未超载，将请求发送到该服务器，否则返回空。它采用的散列函数与目标地址散列调度算法的相同，它的算法流程与目标地址散列调度算法的基本相似。

来自同一个IP地址的请求都将调度到同一个RealServer

##### 2.9.最短的期望的延迟

最短的期望的延迟调度（Shortest Expected Delay 简称'SED'）算法基于WLC算法。举个例子吧，ABC三台服务器的权重分别为1、2、3 。那么如果使用WLC算法的话一个新请求进入时它可能会分给ABC中的任意一个。使用SED算法后会进行一个运算

A：（1+1）/1=2 B：（1+2）/2=3/2 C：（1+3）/3=4/3 就把请求交给得出运算结果最小的服务器。

##### 2.10.最少队列调度

最少队列调度（Never Queue 简称'NQ'）算法，无需队列。如果有realserver的连接数等于0就直接分配过去，不需要在进行SED运算。

### lvs搭建

DR模式

Director

DIP: eth0:192.168.189.200

VIP: eth1:192.168.189.201

RIP: eth0:192.168.189.198 lo:0 192.168.189.201

eth0:192.168.189.199 lo:0 192.168.189.201

#### 1、lvsDirector安装及配置

yum install ipvsadm -y

#两个网段时打开路由转发

#只是临时修改，重启机器或重启网卡失效

# echo 1 > /proc/sys/net/ipv4/ip\_forward

**#永久生效的话**，需要修改sysctl.conf：

#net.ipv4.ip\_forward = 1

ipvsadm -A -t 192.168.189.201:80 -s rr #指定LVS调度器的VIP地址以及算法，-t代表tcp协议

ipvsadm -a -t 192.168.189.201:80 -r 192.168.189.198:80 -g #指定Real Server的地址以及工作模式，-g代表DR模式

ipvsadm -a -t 192.168.189.201:80 -r 192.168.189:199:80 -g

[root@lvsdirector ~]# ipvsadm -l

IP Virtual Server version 1.2.1 (size=4096)

Prot LocalAddress:Port Scheduler Flags

-> RemoteAddress:Port Forward Weight ActiveConn InActConn

TCP lvsdirector:http rr

-> 192.168.189.198:http Route 1 0 0

-> 192.168.189.199:http Route 1 0 0

配置完以后可以使用

ipvsadm-save /etc/sysconfig/ipvsadm 保存配置

注意：该配置文件中的主机名全部改为ip地址，否则可能因为地址解析的问题导致配置错误

设置开机自启

systemctl enable ipvsadm

#### 2、lvsRealserver配置

##### 2.1 关闭防火墙

两个节点都要执行

systemctl stop firewalld.service

systemctl disable firewalld.service

setenforce 0

sed -i 's/enforcing/disabled/g' /etc/selinux/config

###### 2.1.1 手动配置

eg:ifconfig lo:0 $VIP broadcast $VIP netmask 255.255.255.255 up

ifconfig lo:0 192.168.189.201/32 broadcast 192.168.189.201 up

route add -host 192.168.189.201 dev lo:0

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

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

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

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

在每台Real Server上做ARP抑制，其中2个参数为arp\_ignore、arp\_announce。  
arp\_ignore（接收到ARP请求后的响应级别）:  
0：本地所有网卡中，如果有对应的IP地址就会响应请求  
1：只会响应目标IP为接收网卡所配置的IP地址的请求。以此实现后端服务器回环地址上的IP不去响应广播

arp\_announce（向外回应自己网络地址）:  
0：本地所有网卡上的IP地址都可以向外响应  
1：尽量使用与本地网卡所匹配的地址向外响应  
2：只使用与本地接口匹配的网络地址向外响应，效果和arp\_ignore类似也实现了避免响应

###### 2.1.2 脚本启动

vim realserver

#!/bin/sh

VIP=192.168.189.202

. /etc/rc.d/init.d/functions

start() {

/sbin/ifconfig lo down

/sbin/ifconfig lo up

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

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

/sbin/ifconfig lo:0 $VIP netmask 255.255.255.255 broadcast $VIP up #在回环地址上绑定VIP，设定掩码，与Direct Server（自身）上的IP保持通信

/sbin/route add -host $VIP dev lo:0

echo "LVS-DR real server starts successfully.\n"

}

stop() {

/sbin/ifconfig lo:0 down

/sbin/route del $VIP >/dev/null 2>&1

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

echo "LVS-DR real server stopped.\n"

}

case "$1" in

# 禁用本地的ARP请求、绑定本地回环地址

start)

start

;;

stop)

stop

;;

status)

isLoOn=`/sbin/ifconfig lo:0 | grep "$VIP"`

isRoOn=`/bin/netstat -rn | grep "$VIP"`

if [ "$isLoON" == "" -a "$isRoOn" == "" ]; then

echo "LVS-DR real server has run yet."

else

echo "LVS-DR real server is running."

fi

exit 3

;;

\*)

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

exit 1

esac

exit 0

/etc/init.d/realserver start|stop|status

/etc/init.d/realserver satrt启动

##### 2.2 安装apache服务

yum install htttpd

echo “192.168.189.198”>/var/www/html/index.html

echo “192.168.189.199”>/var/www/html/index.html

测试即可

[root@localhost ~]# curl 192.168.189.201

192.168.189.198 lvsRealserver1

[root@localhost ~]# curl 192.168.189.201

192.168.189.199 lvsRealserver2

[root@localhost ~]# curl 192.168.189.201

192.168.189.198 lvsRealserver1

[root@localhost ~]# curl 192.168.189.201

192.168.189.199 lvsRealserver2

##### 2.3 ActiveConn InActConn详解

[root@lvsdirector ~]# ipvsadm -l

IP Virtual Server version 1.2.1 (size=4096)

Prot LocalAddress:Port Scheduler Flags

-> RemoteAddress:Port Forward Weight ActiveConn InActConn

TCP lvsdirector:http rr

-> 192.168.189.198:http Route 1 1 2

-> 192.168.189.199:http Route 1 1 2

ActiveConn是活动连接数,也就是tcp连接状态的ESTABLISHED;InActConn是指除了ESTABLISHED以外的,所有的其它状态的tcp连接.

lvs自身也有一个默认超时时间.可以用ipvsadm -L --timeout查看,默认是900 120 300,分别是TCP TCPFIN UDP的时间.也就是说一条tcp的连接经过lvs后,lvs会把这台记录保存15分钟,而不管这条连接是不是已经失效!所以如果你的服务器在15分钟以内有大量的并发请求连进来的时候,你就会看到这个数值直线上升.

[root@lvsdirector ~]# ipvsadm -l --timeout

Timeout (tcp tcpfin udp): 900 120 300

ipvsadm --set 5 10 300.设置tcp连接只保持5秒中，tcp连接

NAT模式

Director

DIP: eth0:192.168.30.100

VIP: eth1:192.168.189.201

RIP: eth0:192.168.189.198 lo:0 192.168.189.201

eth0:192.168.189.199 lo:0 192.168.189.201

### keepalived+lvs（DR模式）

lvs1 192.168.189.200

lvs2 192.168.189.201

vip 192.168.189.202

realserver1 192.168.189.198

realserver2 192.168.189.199

lvs1和lvs2节点安装httpd

添加sorry error，当realserver节点都宕机后，提示用户

编辑/var/www/html/index.html文件添加提示用户信息

lvs1

[root@lvs1 ~]# cat /etc/keepalived/keepalived.conf

! Configuration File for keepalived

global\_defs {

router\_id lvs1

# vrrp\_mcast\_group4 224.0.0.18 #vrrp的ipv4多播地址，默认为224.0.0.18，配置了VIP的网卡向这个地址广播来宣告自己的配置，需要开启VIP所在网卡的多播，ip link set multicast on dev ens33

#

}

# 定义用于实例执行的脚本内容，比如可以在线降低优先级，用于强制切换

#vrrp\_script SCRIPT\_NAME {

# script “/usr/local/keerpalived/nginx\_check.sh”

# interval 2

# weight -20

#}

vrrp\_instance VI\_1 {

state MASTER #指定keepalived的角色

interface ens33 #指定监测网络的接口

virtual\_router\_id 51 #虚拟路由标识，这个标识是一个数字，同一个vrrp\_instance下，MASTER和BACKUP必须是一致的

# use\_vmac XX:XX:XX:XX:XX #虚拟Vmac地址

priority 110 #定义优先级，数字越高优先级越高

advert\_int 1 #发送心跳间隔时长

authentication {

auth\_type PASS #设置验证类型，有PASS和AH两种

auth\_pass 1111 #设置验证密码1111或者AH，在同一个vrrp\_instance下MASTER和BACKUP密码必须保持一致

}

virtual\_ipaddress {

192.168.189.202/24 dev ens33 label ens33:0 #设置虚拟ip地址，可以设置多个地址

}

# nopreempt|preempt #抢占和非抢占模式,非抢占模式backup节点上修改即可，master不用修改

# preempt delay 300 #如果是抢占模式，设置多久再抢占，默认5分钟

# track\_script {

# chk\_nginx

# } #追踪脚本，用于执行上面的vrrp\_script定义的脚本内容

# notify\_master “” #三个指令，如果主机状态变为master|backup|fault之后会去执行的通用脚本，脚本自己编写

# notify\_backup “”

# notify\_fault “”

}

virtual\_server 192.168.189.202 80 { #设置虚拟服务器，指定ip地址及端口，之间用空格隔开

delay\_loop 6 #运行情况检查时间，单位为秒

lb\_algo rr #设置负载均衡调度算法，rr为轮询算法

lb\_kind DR #设置lvs实现负载均衡的机制

persistence\_timeout 50 #会话保持时间，单位为秒

protocol TCP #指定转发协议类型

sorry\_server 127.0.0.1 80 #如果后端服务器不可用，定向到哪个服务器上

real\_server 192.168.189.198 80 { #真实服务器，指定ip地址及端口，用空格隔开

weight 3 #服务器权重，数字越大权重越高

TCP\_CHECK {

connect\_timeout 3 #表示3秒无响应超时

delay\_defore\_retry 3 #表示重试间隔

# bindto 192.168.189.198 #检测哪个ip，不指定默认使用上面定义的ip地址

# connect\_port 80 #检查哪个端口，不指定默认检查上面定义的端口

#nb\_get\_retry 3 #表示重试次数

}

}

real\_server 192.168.189.199 80 {

weight 2

TCP\_CHECK {

connect\_timeout 3

delay\_defore\_retry 3

connect\_port 80

}

}

}

lvs2

[root@lvs2 ~]# cat /etc/keepalived/keepalived.conf

! Configuration File for keepalived

global\_defs {

router\_id lvs2

}

vrrp\_instance VI\_1 {

state BACKUP

interface ens37

virtual\_router\_id 51

priority 100

advert\_int 1

authentication {

auth\_type PASS

auth\_pass 1111

}

virtual\_ipaddress {

192.168.189.202/24 dev ens37 label ens37:0

}

}

virtual\_server 192.168.189.202 80 {

delay\_loop 6

lb\_algo rr

lb\_kind DR

persistence\_timeout 50

protocol TCP

real\_server 192.168.189.198 80 {

weight 3

TCP\_CHECK {

connect\_timeout 3

delay\_defore\_retry 3

connect\_port 80

}

}

real\_server 192.168.189.199 80 {

weight 2

TCP\_CHECK {

connect\_timeout 3

delay\_defore\_retry 3

connect\_port 80

}

}

}

systemctl start keepalived

systemctl enable keepalived

用keepalived管理lvs，lvs不用做配置

主

! Configuration File for keepalived

global\_defs {

notification\_email {

acassen@firewall.loc

failover@firewall.loc

sysadmin@firewall.loc

}

notification\_email\_from Alexandre.Cassen@firewall.loc

smtp\_server 192.168.200.1

smtp\_connect\_timeout 30

router\_id keep-lvs1

# vrrp\_skip\_check\_adv\_addr

# vrrp\_strict

# vrrp\_garp\_interval 0

# vrrp\_gna\_interval 0

}

vrrp\_script chk\_sshd { //监控kee\_lvs服务器本地的sshd状态，不是真实

script "kill -0 sshd" //提供httpd的realserver的sshd状态，如果keep\_lvs

interval 2 //的sshd断开，则keep切换到另一台

fail 2 //和下面的track\_script一起使用也可以使用脚本

rise 1

}

vrrp\_instance VI\_1 {

state MASTER

interface ens33

virtual\_router\_id 51

priority 150

advert\_int 1

authentication {

auth\_type PASS

auth\_pass 1111

}

virtual\_ipaddress {

192.168.189.100/24 dev ens33 label ens33:2

}

track\_script { //和上面的chk\_sshd一起使用，只关心chk\_sshd的返回值

chk\_sshd //是否为0，为0正常，不为0切换

}

}

#vrrp\_instance VI\_2 {

# state BACKUP

# interface ens33

# virtual\_router\_id 57

# priority 50

# advert\_int 1

# authentication {

# auth\_type PASS

# auth\_pass 1111

# }

# virtual\_ipaddress {

# 192.168.189.101/24 dev ens33 label ens33:3

#}

#}

virtual\_server 192.168.189.100 80 {

delay\_loop 3

lb\_algo wrr

lb\_kind DR

persistence\_timeout 50

protocol TCP

real\_server 192.168.189.201 80 {

weight 1

TCP\_CHECK {

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

connect\_port 80

}

}

real\_server 192.168.189.202 80 {

weight 1

TCP\_CHECK {

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

connect\_port 80

}

}

}