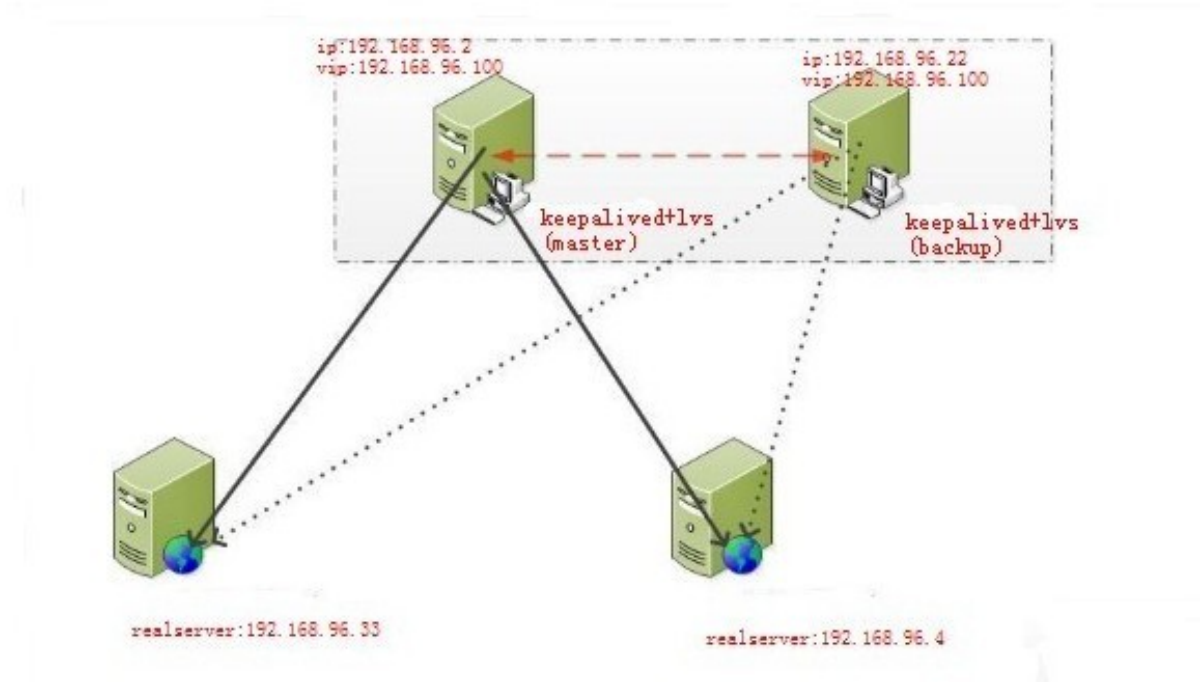
keepalived lvs dr

首先，需要把下面几个概念解释清楚。

lvs：实现负载均衡的一个解决方案（名称而已） ipvsadmin：这个解决方案的一个具体软件、程序 ip_vs：实现lvs负载均衡所需要的模块（lsmod | grep ip_vs），至于虚地址是否有该模块来实现，暂时不确定 keepalived：另一款用于实现lvs负载均衡的软件，用的也是ip_vs模块；还带有健康检测和主备切换功能
lvs：实现负载均衡，load-balance keepalived：实现高可用，high-availability，具有 health-check、failover等功能 在这套架构中，lvs就是用于实现lb的，而keepalived仅仅是实现health-check以及failover的功能。

本文涉及到的架构如图



图片来自网络，注释非本文所有，请忽略。但是整个架构就是这样的。
前端两台负载均衡器拥有一个vip，做主备；
后端两台real_server，提供了nginx服务，通过前端的那个vip来做负载均衡。

安装keepalived

###主备负载均衡器上安装 见【keepalived 安装文档】

安装ipvsadm

###在主备负载均衡器上安装

```
cd /usr/local/src
上传ipvsadm-1.24.tar.gz
tar -xzf ipvsadm-1.24.tar.gz
cd ipvsadm-1.24
ln -s /usr/src/kernels/2.6.32-71.el6.i686/ /usr/src/linux       ---不能少
make
make install
```

各台服务器所需要做的配置

```
###主负载均衡器
global_defs {
 router_id test_node1
}

vrrp_instance Nginx {
 state BACKUP
 interface eth0
 virtual_router_id 100
 priority 200
 advert_int 1
 nopreempt
 track_interface {
  eth0
 }
 authentication {
  auth_type PASS
  auth_pass 123456
 }
 virtual_ipaddress {
  192.168.223.200
 }
}

virtual_server 192.168.223.200 80 {
 delay_loop 6
 lb_algo wlc
 lb_kind DR
 persistence_timeout 1
 protocol TCP
 real_server 192.168.223.102 80 {
  weight 3
  TCP_CHECK {
   nb_get_retry 3
   delay_before_retry 3
   connect_port 80
   connect_timeout 3
  }
 }
 real_server 192.168.223.103 80 {
  weight 3
  TCP_CHECK {
   nb_get_retry 3
   delay_before_retry 3
```

```
###备负载均衡器
global_defs {
 router_id test_node2
}

vrrp_instance Nginx {
 state BACKUP
 interface eth0
 virtual_router_id 100
 priority 190
 advert_int 1
 track_interface {
  eth0
 }
 authentication {
  auth_type PASS
  auth_pass 123456
 }
 virtual_ipaddress {
  192.168.223.200
 }
}

virtual_server 192.168.223.200 80 {
 delay_loop 6
 lb_algo wlc
 lb_kind DR
 persistence_timeout 1
 protocol TCP

 real_server 192.168.223.102 80 {
  weight 3
  TCP_CHECK {
   nb_get_retry 3
   delay_before_retry 3
   connect_port 80
   connect_timeout 3
  }
 }
 real_server 192.168.223.103 80 {
  weight 3
  TCP_CHECK {
   nb_get_retry 3
   delay_before_retry 3
```

```
  connect_port 80
  connect_timeout 3
 }
}
}
```

```
  connect_port 80
  connect_timeout 3
 }
}
}
```

real_server上都需要加上该脚本

第一行的虚地址根据实际情况做修改，其余不变

vi real_server.sh

```
SNS_VIP=192.168.223.200
. /etc/rc.d/init.d/functions
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 Stopped"
        ;;
*)
        echo "Usage: $0 {start|stop}"
        exit 1
esac
exit 0
```

chmod a+x real_server.sh ---为脚本加上X权限

./real_server.sh start ---开启real_server

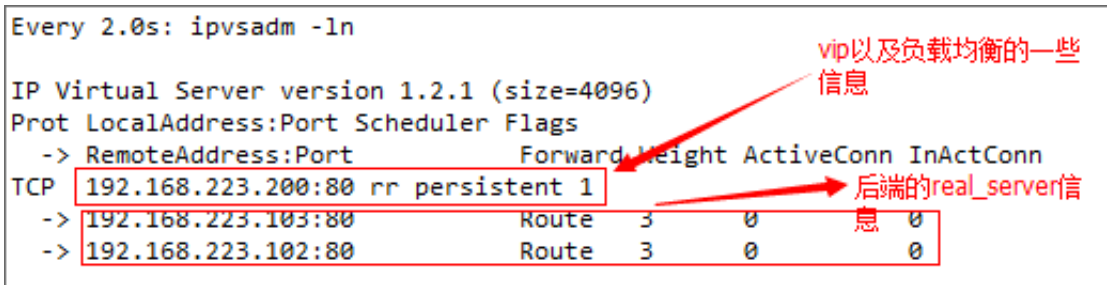
./real_server.sh stop ---停止real_server

实验现象

①主备负载均衡器开启keepalived服务；

②real_server上./real_server.sh start，再开启我们所需要进行负载的业务，本实验中为nginx服务；

在主负载均衡器上会出现192.168.223.200的vip，通过watch ipvsadm -ln命令可以看到已经对real_server进行了负载均衡（只有在拥有vip的负载均衡器上才可以看到ipvsadm的负载均衡信息！）



测试结果

在本地浏览器中，可以通过192.168.223.200访问到后端的两台real_server中的一台，并且关闭任意一台，依旧可以正常访问另外一台，服务不中断！

上面是操作文档，下面对实验中的一些现象以及配置进行说明。

【DR模式需要lb和所有rs均处于同一网段】

①上面文档中的负载均衡器的keepalived虚地址配置项方面，没有涉及到track script、vrrp group以及多个vrrp instance，如果在生产环境中需要配置这些参数，那么请参照【keepalived 单个实例 单个VIP】和【keepalived 多个实例 多个VIP】

②上面文档中的负载均衡器的keepalived virtual_server的配置只是一般配置而已，具体配置请根据实际情况进行修改。 可能涉及到的重要修改项有：lb_algo、persistence_timeout、TCP_CHECK（可能需要其他类型的检查方式）等

③上面文档中我们在两台负载均衡器上均安装了ipvsadm，这里安装ipvsadm仅仅是为了用watch ipvsadm -ln命令来对负载情况进行监控而已，即使不安装ipvsadm依旧是可以通过keepalived来实现负载均衡的

④由于在服务器上访问该台服务器ip列表中的ip地址，请求是不会被转发该服务器外部的。因此，我们在拥有vip的负载均衡器上curl 192.168.223.200，是不会成功的，因为该负载均衡器上80端口是没起服务的；并且我们在各台real_server上curl 192.168.223.200，永远只会访问本地的80端口，因为vip 192.168.223.200也在real_server的ip地址列表中；

⑤virtual_server与real_server的端口是否需要一致的问题。 经过简单的不完全测试，virtual_server 8080端口、real_server 80端口，这样的配置不可以通过virtual_server 8080端口访问到real_server 80端口的。 因此，暂时先认为两者的端口需要保持一致吧。