**Keepalived—高可用**

**一、keepalived简述**

**1、VRRP(Virtual Router Redundancy Protocol:** **虚拟路由冗余协议)**

**(1)VRRP(虚拟路由冗余协议)定义：**

Virtual Router Redundancy Protocol(虚拟路由冗余协议)，是一种容错协议，通过把几台路由设备虚拟成一台对外服务的网关路由，内部的路由器组之间就是采用VRRP协议进行通信，MASTER节点固定频率地通过BROADCAST向BACKUP节点advert(通告)自己的HEARTBEAT(心跳信息)和PRIORITY(优先级)等信息。一旦主节点故障，备节点就会从BACKUP状态转为MASTER状态，显然，如果主节点降低优先级，备节点通过抢占(preempt)即可转为MASTER状态，也就是VIP的转移。因此，新MASTER需发起ARP广播以告知自己和其他主机VMAC(虚拟主机地址)地址发生了变化。 以此，两台冗余的路由器提供了高可用的服务。

**(2)VRRP(虚拟路由冗余协议)工作过程**

**1）**虚拟路由器中的路由器根据优先级选举出Master。Master路由器通过发送免费ARP（Address Resolution Protocol：地址解析协议）报文，将自己的虚拟MAC地址通知给与它连接的设备或者主机，从而承担报文转发任务；   
 **2）**Master路由器周期性发送VRRP报文，以公布其配置信息（优先级等）和工作状况；   
 **3）** 如果Master路由器出现故障，虚拟路由器中的Backup路由器将根据优先级重新选举新的Master；   
 **4）**虚拟路由器状态切换时，Master路由器由一台设备切换为另外一台设备，新的Master路由器只是简单地发送一个携带虚拟路由器的MAC地址和虚拟IP地址信息的免费ARP报文，这样就可以更新与它连接的主机或设备中的ARP相关信息。网络中的主机感知不到Master路由器已经切换为另外一台设备。   
 **5）**Backup路由器的优先级高于Master路由器时，由Backup路由器的工作方式（抢占方式和非抢占方式）决定是否重新选举Master。 由此可见，为了保证Master路由器和Backup路由器能够协调工作，VRRP需要实现以下功能：

Master路由器的选举；

Master路由器健康状态报告；

为了安全性，VRRP提供了认证机制。

在VRRP协议实现里，虚拟路由器使用 00-00-5E-00-01-XX 作为虚拟MAC地址，XX就是唯一的 VRID （Virtual Router IDentifier），这个地址同一时间只有一个物理路由器占用。在虚拟路由器里面的物理路由器组里面通过多播IP地址 224.0.0.18 来定时发送通告消息。每个Router都有一个 1-255 之间的优先级别，级别最高的（highest priority）将成为主控（master）路由器。通过降低master的优先权可以让处于backup状态的路由器抢占（pro-empt）主路由器的状态，两个backup优先级相同的IP地址较大者为master，接管虚拟IP。

**2、keepalived简介**

**(1)keepalived引入**

keepalived最初的设计目标是为了实现lvs(Linux Virtual Server)设备的高可用，keepalived能够根据配置文件中的定义生成ipvs规则，并能够对各个real server的健康状态进行检测(LVS实际上按照高可用的角度来讲，只有两个资源，一个是VIP，一个是内核上的ipvs规则)。Keepalived是一个基于VRRP协议来实现的服务高可用方案，可以利用其来避免IP单点故障，类似的工具还有heartbeat、corosync、pacemaker(corosync+pacemaker是最佳组合，但与keepalived是不同工作机制的高可用方案)。但是keepalived一般不会单独出现，而是与其它负载均衡技术（如lvs、haproxy、nginx）一起工作来达到集群的高可用。

**(2)keepalived介绍**

keepalive启动后，是由主进程(Control Plane) 读取和分析配置文件，并根据配置文件，指挥两个子进程(Checkers & VRRP Stack)完成相关工作。

*VRRP Stack :* 是整个keepalive功能实现的核心子进程 ;

*Checkers :* 主要用于检测real server的健康状态 , 可以基于TCP Check , http\_get , https\_get , misc\_get等多种方法;

主进程还能利用内核提供的watchdog模块，实现对两个子进程的健康性检查，当发现某个子进程故障后，主进程会kill掉这个子进程，然后再从新启一个子进程，而且watchdog能够让两个子进程每隔一段时间，向主进程的socket套接字上发送信息，当某一时刻，主进程无法收到子进程的信息，就判断子进程故障，然后就杀掉，重启子进程

**(3)keepalived主备选择**

两个节点之间哪个为主，哪个为备用节点，是由优先级来决定的，当初始时，优先级高的为主节点，当其中某一个节点故障，无法启动，则另一个节点会替换上来，作为主的，向外提供服务。另一种情况，当节点本身正常，但是节点上的某服务，也就是某资源不正常时，这时keepalived的子进程checkers就会将当前节点的优先级降级，从而实现了另一个备用节点的优先级要高于该节点，当下次节点间相互通告信息时，备用节点就会发现自己的优先级比当前活动节点的优先级要高，然后就提升为主节点，启动资源，对外提供服务

优先级的数字是0-255，值越大，优先级越高。其中0和255被系统自身占用，不能作为可用的优先级的数字进行调整。

**二、keepalived的安装配置**

**1、keepalived安装前提：**

    (1) 各节点时间必须同步(ntpdate(chrony)、crond服务)。

    (2) 确保iptables及selinux不会成为阻碍。

(3) 各节点之间可通过主机名互相通信；建议使用/etc/hosts文件实现；

(4) 各节点之间的root用户可以基于密钥认证的ssh服务完成互相通信。

**[root@web ~]# ssh-keygen -f /root/.ssh/id\_rsa -P "" ##建立ssh主机秘钥**

**[root@web ~]#ssh-copy-id -I /root/.ssh/id\_rsa.pub root@node1 ##传播ssh主机公钥至个主机**

(5) 网卡的MULTICAST(多播)功能应打开：[root@ha ~]#ip link set DEVICE multicast on

**2、keepalived的安装配置**

**(1)安装keepalived软件：**

CentOS 6.4以后的版本以及收录到base repository中，直接yum安装即可。也可在其官方网站：http://www.keepalived.org/download.html，下载最新版本进行编译安装

**[root@web ~]#wget http://www.keepalived.org/software/keepalived-1.2.15.tar.gz**

**[root@web ~]#tar zxvf keepalived-1.2.15.tar.gz**

**[root@web ~]#cd keepalived-1.2.15**

**[root@web ~]#./configure --prefix=/usr/local/keepalived**

**[root@web ~]#make && make install**

**[root@web ~]#echo “export PATH=$PATH:/usr/local/keepalived/bin” >/etc/profile.d/ka.sh**

**[root@web ~]#source /etc/profile.d/ka.sh**

或者配置本地yum源，yum安装：

**[root@web ~]#yum install keepalived -y**

**(2)keepalived相关文件介绍**

配置文件：/etc/keepalived/keepalived.conf

Unit File：/etc/rc.d/init.d/keepalived

主程序：/usr/sbin/keepalived

配置文件示例：/usr/share/doc/keepalived-1.2.13/samples/\*

**配置文件说明**

*keepalived的主配置(/etc/keepalived.conf)文件分为3段，分别为全局的配置段、VRRP实例段(可包含多个实例)、VS实例段(可包含多个实例)：*

***1)全局配置段：***设置象征性的通知邮箱、路由标识、vrrp多播地址(D类)；

**1.global\_defs {  
2.    notification\_email {  
3.        root@localhost ##通知邮件  
4.    }  
5.    notification\_email\_from keepaliced@localhost  
6.    smtp\_server 127.0.0.1 ##邮件服务器  
7.    smtp\_connect\_timeout 30 ##设置连接邮件服务器超时时间  
8.    router\_id LVS\_node1 ##设置连接##邮件服务器的路由  
9.    vrrp\_mcast\_group4 224.0.22.22 # optional, default 224.0.0.18 ~ 239，选择虚拟路由  
10.}**

***(2)VRRP实例段：***初始状态、物理接口、唯一编号、优先级、vrrp认证、VIP、检测的网卡、抢占模式、通知脚本等；

**1.vrrp\_instance <STRING> {  
2.    state MASTER**

**#MASTER|BACKUP，当前节点在此虚拟路由器上的初始状态，只能有一个MASTER，多个BACKUP  
3.    interface eth0 # 绑定当前虚拟路由器的物理接口  
4.    virtual\_router\_id 12 # 当前虚拟路由器的唯一标识，0 ~ 255  
5.    priority 100 # 当前主机在此虚拟路由器的优先级，1 ~ 254  
6.    advert\_int 1 # vrrp通告的时间间隔  
7.    authentication {  
8.        auth\_type PASS # PASS||AH，Simple Passwd (suggested)认证方式  
9.        auth\_pass 1234 # Only the first eight (8) characters认证密码，明文(8位)  
10.    }  
11.    virtual\_ipaddress { ##虚拟IP地址的设置  
12.        #<IPADDR>/<MASK>  brd  <IPADDR> dev <STRING> scope <SCOPE> label <LABEL>  
13.        192.168.200.17/24 dev eth0               # 不用别名也可有多个ip地址  
14.        #192.168.200.18/24 dev eth2 label eth2:1 # 以网卡别名形式配置多个ip地址  
15.    }  
16.    track\_interface {  
17.        eth0  
18.        eth1 # 监听的接口，一旦网卡故障则转为fault状态，通常用于监控内网接口的网卡  
19.        ...  
20.    }  
21.    #nopreempt # 非抢占模式，默认为preempt  
22.    preempt\_delay 300     # 抢占模式下，节点上线后触发新选举的延迟时间  
23.    notify\_master <STRING>|<QUOTED-STRING> # 节点成为主节点触发的脚本  
24.    notify\_backup <STRING>|<QUOTED-STRING> # 节点转为备节点时触发的脚本  
25.    notify\_fault <STRING>|<QUOTED-STRING> # 当前节点转为失败状态时触发的脚本  
26.    #notify <STRING>|<QUOTED-STRING> #一个脚本可完成以上三种状态的转换时的通知；  
27.}**

***(3)VS示例段：***VIP及端口、检测间隔、调度算法、集群类型、持久连接时长、SorryServer、RealServer{权重、通知脚本、健康检测{HTTP/SSL/TCP/SMTP}、超时、重试、延时}等。以下以lvs的高可用为例：

**1.virtual\_server IP port {  
3.    delay\_loop <INT>                   # 对RS健康检测的间隔  
4.    lb\_algo rr|wrr|lc|wlc|lblc|sh|dh   # LB调度算法   
5.    lb\_kind NAT|DR|TUN                 # LVS类型  
6.    persistence\_timeout <INT>           # 持久时间  
7.    protocol TCP                       #协议  
8.    sorry\_server <IPADDR> <PORT>       #本机作为RS不可用的sorry主机及端口，备用页面  
9.    real\_server <IPADDR> <PORT>{ #后端实际服务器设置  
10.        weight <INT> #权重  
11.        notify\_up <STRING>|<QUOTED-STRING>  
12.        notify\_down <STRING>|<QUOTED-STRING>  
13.        HTTP\_GET|SSL\_GET|TCP\_CHECK|SMTP\_CHECK|MISC\_CHECK { #可使用不同的检测方法定义主机的健康状态，较常用为HTTP\_GET和TCP\_CHECK  
14.            url {  
15.                path <URL\_PATH> # 要监控的URL  
16.                status\_code <INT> # 健康状态的响应码（200正常）  
17.                digest <STRING> # 健康状态响应内容的摘要  
18.            }  
19.            nb\_get\_retry <INT> # 重试次数  
20.            delay\_before\_retry <INT> # 向当前RS的哪个IP地址发起监控状态请求  
21.            connect\_port <PORT> # 向RS的哪个port发起健康监测  
22.            bindto <IP ADDRESS> # 发出健康状态检测请求的原IP  
23.            connect\_timeout <INTEGER> # 连接请求的超时时长  
24.        }         
25.    }  
26.}**

**注：使用man keepalived.conf查看更多的配置文件的 设置**

**三、keepalived的模型和配置**

**1、单主(*一个虚拟路由实例*)模型虚拟路由的配置文件：**



前提：VS1的初始状态为MASTER，VS2的初始状态为BACKUP；且BACKUP节点的优先级应小于MASTER节点。*红色标注的为从机上必须修改的地方*。(经验：对于keepalived的运用，主要在于脚本的编写和配置文件的设计)

**[root@web ~]# vim /etc/keepalived.conf**

**2. global\_defs {**

**3. notification\_email {**

**4. root@localhost**

**5. }**

**6. notification\_email\_from keepalived@jasonmc.com**

**7. smtp\_server localhost**

**8. smtp\_connect\_timeout 30**

**9. router\_id node1**

**10. vrrp\_mcast\_group4 224.22.29.1**

**11. }**

**12**

**13. vrrp\_instance VI\_1 {**

**14. state MASTER #在从机上应为BACKUP**

**15. interface eth0**

**16. virtual\_router\_id 10**

**17. priority 100 #从机上应低于此值**

**18. advert\_int 10**

**19. authentication {**

**20. auth\_type PASS**

**21. auth\_pass d351ac09**

**22. }**

**23. virtual\_ipaddress {**

**24. 10.1.253.11 dev eth0**

**25. }**

**26 }**

**2、双主模型虚拟路由(*两个虚拟路由实例*)的配置文件：**

前提：两台主机都需要安装keepalived，在VS1中VIP1为MASTER，VIP2为BACKUP；在VS2中VIP1为BACKUP，VIP2为MASTER。且要保证对应VIP的优先级(priority)不同，即从VIP的优先级值小于主VIP的优先级，两台主机应保持使用相同名称的网卡 ，相同的VIP使用相同的虚拟路由ID(virtual\_router\_id)，配置文件中红线部分为VS2上必须修改项。



**[root@node1 ~]# vim /etc/keepalived.conf**

**2.global\_defs {**

**3. notification\_email {**

**4. root@localhost**

**5. }**

**6. notification\_email\_from keepalived@jasonmc.com**

**7. smtp\_server localhost**

**8. smtp\_connect\_timeout 30**

**9. router\_id node1**

**10. vrrp\_mcast\_group4 224.22.29.1 #224~239**

**11.}**

**13.vrrp\_instance VI\_1 {**

**14. state MASTER #VS2应修改为BACKUO**

**15. interface eth0**

**16. virtual\_router\_id 1**

**17. priority 100 #VS2应小于该值**

**18. advert\_int 1 #vrrp通告时长**

**19. authentication {**

**20. auth\_type PASS**

**21. auth\_pass cc5042a6**

**#openssl rand -hex 4**

**22. }**

**23. virtual\_ipaddress {**

**24. 10.1.253.11 dev eth0**

**25. }**

**29.}**

**31.vrrp\_instance VI\_2 { ##备用VIP**

**32. state BACKUP #VS2应为MASTER**

**33. interface eth0**

**34. virtual\_router\_id 2 #虚拟路由器的唯一标识，0 ~ 255**

**35. priority 98 #VS2应大于该值**

**36. advert\_int 1**

**37. authentication {**

**38. auth\_type PASS**

**39. auth\_pass ac5342a5**

**40. }**

**41. virtual\_ipaddress {**

**42. 10.1.253.12 dev eth0**

**43. }**

**44. notify\_master "/etc/keepalived/script/notify.sh master" #状态转化master触发脚本**

**45. notify\_backup "/etc/keepalived/script/notify.sh backup" #状态转化slave触发脚本**

**46. notify\_fault "/etc/keepalived/script/notify.sh fault" #状态转化fault触发脚本**

**47.}**

**3、通知脚本示例：**

当以上各节点状态发生转化时会自动触发以下脚本：

**[root@web ~]#vim /etc/keepalived/notify.sh**

**2.#!/bin/bash  
4.receiver='root@localhost'  
5.notify() {  
6.    mailsubject="$(hostname) to $1,vip floating."  
7.    content="$(date + '%F %T') vrrp state transion, $(hostname) changed to be $1"  
8.    echo "$content" | mail -s "$mailsubject" $receiver  
9.}  
10.case $1 in  
11.master)  
12.    notify master  
13.    ;;  
14.backup)  
15.    notify backup  
16.    ;;  
17.fault)  
18.    notify fault  
19.    ;;  
20.\*)  
21.    echo "Usage $(basename $0) {master|backup|fault}"  
22.    exit 1  
23.    ;;**

**24.esac**

**4、使用检测脚本的过程：**

**定义一个脚本  
vrrp\_script <NAME>｛   
 script “string or /path”   
 interval INT  ##间隔时间  
 weight -INT  ##权重分配  
}**

**##可添加多了检测脚本**

**##定义不同的脚本名称**

**在vrrp实例中调用vrrp\_script**

**vrrp\_instance VI\_1 {**

**......  
track\_script {   
 NAME  
 }**

**......**

**}**

注：keepalived使服务拥有高可用性的特点，可与redis、memcache、mysql、nginx、httpd、lvs等各种以tcp等协议连接为基础的服务相结合，实现各个对应服务的高可用性，在实际业务中有广泛的运用