# 第1章 keepalived介绍

## 1.keepalived解决了什么问题

```
简单来说，keepalived解决了静态路由单点故障的问题。说人话就是给两台不同IP的服务器提供了可以自动切换的虚拟IP功能。
应用场景主要用于两台反向代理服务器之间提供可以漂移的虚拟IP，当其中一台故障时，另外一个可以继续对外提供网络服务。
```

## 2.keepalived工作原理

```
keepalived高可用功能实现的基本原理为:
两台主机同时安装好keepalived软件并启动服务,开始正常工作时
角色为Master的主机获得所有资源并对用户提供服务
角色为Backup的主机作为Master主机的热备;

当角色为Master的主机失效或出现故障时
角色为Backup的主机将自动接管Master主机的所有工作,包括接管VIP资源及相应资源服务

而当角色为Master的主机故障修复后,又会自动接管回他原来处理的工作
角色为Backup的主机则同时释放Master主机失效时他接管的工作
此时,两台主机将恢复到启动时各自的原始角色及工作状态
```

## 3.什么是VRRP协议

```
keepalived软件主要是通过VRRP协议实现高可用功能的
VRRP是Virtual Router Redundancy Protocol(虚拟路由器冗余协议)的缩写
VRRP通过竞选机制来实现虚拟路由器的功能
所有的协议报文都是通过IP多播(Multicast)包或者单播IP地址来通信
默认的多播地址224.0.0.18
```

## 4.VIP正常工作的前提条件

```
1.虚拟公网IP必须是真实可用的
2.虚拟公网IP不能重复
3.虚拟公网IP必须和相同网段的物理网卡绑定
4.组播地址必须是可以通讯的
```

## 5.面试如何说

```
keepalived高可用对之间是通过VRRP通信的,因此,我从VRRP开始给您讲起.
1.VRRP,全称Virtual Router Reduancy Protocol,中文名为虚拟路由器冗余协议,VRRP的出现是为了解决静态路由的单点故障,
2.VRRP是通过一种竞选协议来将路由任务交给某台VRRP路由器的,
3.VRRP用IP多播的方式,(默认多播地址(224.0.0.18))实现高可用对之间通信.
4.工作时主节点发包,备节点接包,当备节点接收不到主节点发的包的时候,就启动接管程序接管主节点的资源.备节点可以有多个,通过优先级竞选,但一般keepalived系统运维工作中都是一对.
5.VRRP使用了加密协议加密数据,但keepalived官方目前还是推荐用明文的方式配置认证类型和密码.
介绍完了VRRP,接下来我在介绍一下keepalived服务的工作原理;
keepalived高可用对之间是通过VRRP进行通信的,VRRP是通过竞选机制来确定主备的,主的优先级高于备,因此,工作时会优先获得所有的资源,备节点处于等待状态,当主挂了的时候,备节点就会接管主节点的资源,然后顶替主节点对外提供服务.
6.在keepalived服务对之间,只有作为主的服务器会一直发送VRRP广播包,告诉备他还活着,此时备不会抢占主,当主不可用时,即备监听不到主发送的广播包时,就会启动相关服务接管资源,保证业务的连续性,接管速度最快可以小于一秒
```

## 6.keepalived工作流程图

https://www.processon.com/diagraming/60a1078b7d9c08302440285c

# 第2章 keepalived实战

## 0.环境说明

```
lb-5    10.0.0.5    Keepalived主服务器 Nginx主负载均衡器
lb-6    10.0.0.6    Keepalived备服务器 Nginx备负载均衡器
web-7   10.0.0.7    web服务器
web-8   10.0.0.8    web服务器
```

## 1.安装keepalived

```
yum install keepalived -y
```

## 2.配置文件解释

```
global_defs {
    router_id lb-5            #设置路由ID，每个主机不一样
}

vrrp_instance VIP_1 {         #设置VRRP组名，同一组组名相同
    state MASTER              #设置角色状态，分为MASTER BACKUP
        interface eth0        #VIP绑定的网卡
        virtual_router_id 50     #虚拟路由id，同一组一样
        priority 150          #权重，权重越高，优先级越高
        advert_int 1          #发送组播间隔
        authentication {      #设置验证，密码为明文
            auth_type PASS
            auth_pass 1111
        }
        virtual_ipaddress { #设定的虚拟IP，这个虚拟IP必须是存在且合法且没有被使用的。
            10.0.0.3
        }
```

```
}
```

## 3.lb-5配置

```
[root@lb-5 ~]# cat /etc/keepalived/keepalived.conf
global_defs {
    router_id lb-5
}

vrrp_instance VIP_1 {
    state MASTER
        interface eth0
        virtual_router_id 50
        priority 150
        advert_int 1
        authentication {
            auth_type PASS
            auth_pass 1111
        }
        virtual_ipaddress {
            10.0.0.3
        }
}
```

## 4.lb-6配置

```
[root@lb-6 ~]# cat /etc/keepalived/keepalived.conf
global_defs {
    router_id lb-6
}

vrrp_instance VIP_1 {
    state BACKUP
        interface eth0
        virtual_router_id 50
        priority 100
        advert_int 1
        authentication {
            auth_type PASS
            auth_pass 1111
        }
        virtual_ipaddress {
            10.0.0.3
        }
}
```

## 5.启动服务

```
systemctl start keepalived
```

## 6.访问测试

```
关掉任意一台，观察VIP是否会漂移
恢复MASTER观察BACKUP的VIP是否会消失
```

# 第3章 keepalived脑裂现象

## 1.什么是脑裂现象

```
简单来说，就是主服务器还正常工作的情况下，备服务器收不到了主服务器发送的心跳请求包，就会以为主节点挂掉了，然后抢占VIP。
这个时候就会出现两台keepalived服务器都拥有了VIP，这时候路由器上的路由表就会混乱，导致出现莫名其妙的网络问题。
```

## 2.通过抓包查看脑裂现象

lb-5安装抓包工具

```
yum install tcpdump -y
```

lb-5执行抓包命令

```
tcpdump -nn -i any host 224.0.0.18
```

lb-6新开一个终端，然后开启防火墙

```
systemctl start firewalld.service
```

观察是否两边都有VIP

```
ip a
```

## 3.出现裂脑后的排查

```
出现上述两台服务器争抢同一IP资源问题,一般要先考虑排查两个地方:
1.主备两台服务器之间是否通讯正常,如果不正常是否有iptables防火墙阻挡?
2.主备两台服务器对应的keepalived.conf配置文件是否有错误?
例如是否同一实例的virtual_router_id配置不一样.
```

## 4.如何解决脑裂现象

第一种解决方法防：火墙添加放行规则

firewall规则：

```
firewall-cmd --direct --permanent --add-rule ipv4 filter INPUT 0 --in-interface eth0 --destination 224.0.0.18 --protocol vrrp -j ACCEPT
firewall-cmd --direct --permanent --add-rule ipv4 filter INPUT 0 --in-interface eth1 --destination 224.0.0.18 --protocol vrrp -j ACCEPT
systemctl reload firewalld
```

iptables规则：

```
iptables -I INPUT -i eth0 -d 224.0.0.0/8 -p vrrp -j ACCEPT
iptables -I OUTPUT -o eth0 -d 224.0.0.0/8 -p vrrp -j ACCEPT
```

第二种解决方法：使用单播地址而不是多播地址

主服务器配置：

```
global_defs {
    router_id lb-5
}

vrrp_instance VIP_1 {
    state MASTER
        interface eth0
        virtual_router_id 50
        priority 150
        advert_int 1
        authentication {
            auth_type PASS
            auth_pass 1111
        }
        unicast_src_ip 10.0.0.5
        unicast_peer {
            10.0.0.6
        }
        virtual_ipaddress {
            10.0.0.3
        }
}
```

备服务器配置：

```
global_defs {
    router_id lb-6
}

vrrp_instance VIP_1 {
    state BACKUP
        interface eth0
        virtual_router_id 50
        priority 100
        advert_int 1
        authentication {
            auth_type PASS
            auth_pass 1111
        }
        unicast_src_ip 10.0.0.6
        unicast_peer {
            10.0.0.5
        }
        virtual_ipaddress {
            10.0.0.3
        }
}
```

以上两种方法都只是预防，仍然不能保证一定不会出现脑裂现象，我们还可以自己编写放脑裂脚本，然后在keepalived启动的时候调用这个脚本，而脚本的内容就是备服务器定时监测主服务器和VIP的情况，当发生脑裂式，keealived备服务器自己杀死自己的进程。

# 5.防脑裂脚本

监控思路：

```
对于备服务器：
1.备份服务器定期检查主服务器上的nginx是否工作正常
2.备份服务器定期检查自己身上是否有VIP
3.如果同时满足以下条件，自己却还有VIP则认为脑裂了
  - 主服务器的NGINX工作正常
  - 主服务器有VIP
  - 备份服务器有VIP
4.如果发生了脑裂，备份服务器自己杀死自己的keepalived
5.将结果通知管理员
```

备服务器脚本编写：

```
cat > /etc/keepalived/check_vip.sh << 'EOF'
#!/bin/bash
MASTER_VIP=$(ssh 10.0.0.5 ip a|grep 10.0.0.3|wc -l)
MY_VIP=$(ip a|grep 10.0.0.3|wc -l)

if [ ${MASTER_VIP} == 1 -a ${MY_VIP} == 1 ]
then
   systemctl stop keepalived
fi
EOF
```

备服务器keepalived调用：

```
cat > /etc/keepalived/keepalived.conf << 'EOF'
global_defs {
    router_id lb-6
}

vrrp_script check_vip {
    script "/etc/keepalived/check_vip.sh"
    interval 5
}

vrrp_instance VIP_1 {
    state BACKUP
    interface eth0
    virtual_router_id 50
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        10.0.0.3
    }
```

```
    track_script {
        check_vip
    }
}
EOF
```

# 6.主服务器检查nginx状态

问题现象：

```
刚才的脚本解决了备服务器抢占VIP的问题，但是还有一种特殊情况，那就是主服务器自己的nginx已经挂了，但是keepalived还活着，此时虽然VIP存在，但是已经不能正常对外提供服务器，所以主服务器也需要编写脚本来解决这种问题：
```

解决思路：

```
对于主服务器：
1.如果自己的nginx已经挂了，但是keepalived还活着，则尝试重启2次nginx
2.如果重启2次nginx依然失败，则杀掉自己的keepalived进程，放弃主服务器角色
```

主服务器监控脚本：

```
cat > /etc/keepalived/check_web.sh << 'EOF'
#!/bin/bash
NGINX_STATUS=$(ps -ef|grep [n]ginx|wc -l)
if [ ${NGINX_STATUS} == 0 ]
then
   systemctl restart nginx
   if [ $? == 1 ]
   then
      systemctl stop keepalived
   fi
fi
EOF
```

主服务器的keepalived配置文件：

```
cat > /etc/keepalived/keepalived.conf << 'EOF'
global_defs {
    router_id lb-5
}

vrrp_script check_web {
    script "/etc/keepalived/check_web.sh"
    interval 5
}

vrrp_instance VIP_1 {
    state MASTER
        interface eth0
        virtual_router_id 50
        priority 150
        advert_int 1
        authentication {
            auth_type PASS
```

```
            auth_pass 1111
        }
        virtual_ipaddress {
            10.0.0.3
        }
        track_script {
            check_web
        }
}
EOF
```

# 第4章 keepalived双主实验

## 1.lb-5配置文件

```
global_defs {
    router_id lb-5
}

vrrp_instance VIP_1 {
    state MASTER
        interface eth0
        virtual_router_id 50
        priority 150
        advert_int 1
        authentication {
            auth_type PASS
                auth_pass 1111
        }
    virtual_ipaddress {
        10.0.0.3/24 dev eth0 label eth0:1
    }
}

vrrp_instance VIP_2 {
    state BACKUP
        interface eth0
        virtual_router_id 50
        priority 100
        advert_int 1
        authentication {
            auth_type PASS
            auth_pass 1111
        }
    virtual_ipaddress {
        10.0.0.4/24 dev eth0 label eth0:2
    }
}
```

## 2.lb-6配置文件

```
global_defs {
    router_id lb-6
}

```

```
vrrp_instance VIP_1 {
    state BACKUP
        interface eth0
        virtual_router_id 50
        priority 100
        advert_int 1
        authentication {
            auth_type PASS
            auth_pass 1111
        }
    virtual_ipaddress {
        10.0.0.3/24 dev eth0 label eth0:1
    }
}

vrrp_instance VIP_2 {
    state MASTER
        interface eth0
        virtual_router_id 50
        priority 150
        advert_int 1
        authentication {
            auth_type PASS
                auth_pass 1111
        }
    virtual_ipaddress {
        10.0.0.4/24 dev eth0 label eth0:2
    }
}
```

## 3.重启keepalived并观察现象

```
systemctl restart keepalived
```