keepalived 单个实例 单个VIP

以下配置文件主要在"keepalived 基础配置文档"的基础上，通过vrrp_script检测生产业务状态（nginx）来动态切换vip。

主：

```
global_defs {
 router_id test_node1 ###本机标识符，不重复即可
}
vrrp_script check_shell {
 script "/opt/check_shell/check_shell.sh"  ###必须要有X权限，否则主备不能正常切换!!!
 interval 5  ###脚本检测间隔
 weight -50 ###脚本检测结果影响的weight值
 fall 3        ###连续3次脚本返回非0，才会认为是failed
}
vrrp_instance TEST_CLUSTER { ###没做过测试是否需要一致，但尽量一致吧
 stateBACKUP ###该状态为初始化状态，并不是最终状态，没多大意义
 interface eth0 ###vip起在哪个网卡上
 virtual_router_id 100###一组keepalived设备必须拥有相同的vrid!!!这个很重要!!!
 priority 200 ###该节点的优先级（0-255），决定master/backup的最主要的因素
 advert_int 1 ###vrrp组播通告的时间间隔
 garp_master_delay 10 ###在切换到master状态后，延迟进行免费的ARP(gratuitous ARP)请求（不知道有什么用）
 nopreempt ###不主动抢占master
 track_interface {
  eth0  ###健康检查的网卡，出现问题时keepalived会进入fault状态，可以理解为keepalived挂了
 }
 authentication { ###同一个keepalived集群的写一样吧
  auth_type PASS
  auth_pass 123456
 }
 virtual_ipaddress {
  192.168.223.200 ###这就是我们所虚拟出来的vip（也可以是多个，暂未研究）
 }

 track_script {
  check_shell ###使用上面定义的监控脚本
 }
 notify_master "/bin/bash /opt/check_shell/notify.sh Master"
 notify_backup "/bin/bash /opt/check_shell/notify.sh Backup"
 notify_fault "/bin/bash /opt/check_shell/notify.sh Fault"
 notify_stop "/bin/bash /opt/check_shell/notify.sh Stop"
}
```

备：

```
global_defs {
 router_id test_node2
}
vrrp_script check_shell {
```

```
 script "/opt/check_shell/check_shell.sh"
 interval 5
 weight -50
 fall 3
}
vrrp_instance TEST_CLUSTER {
 state BACKUP
 interface eth0
 virtual_router_id 100
 priority 190
 advert_int 1
 garp_master_delay 10
 track_interface {
  eth0
 }
 authentication {
  auth_type PASS
  auth_pass 123456
 }
 virtual_ipaddress {
  192.168.223.200
 }
 track_script {
  check_shell
 }
 notify_master "/bin/bash /opt/check_shell/notify.sh Master"
 notify_backup "/bin/bash /opt/check_shell/notify.sh Backup"
 notify_fault "/bin/bash /opt/check_shell/notify.sh Fault"
 notify_stop "/bin/bash /opt/check_shell/notify.sh Stop"
}
```

script "/opt/check_shell/check_shell.sh"内容：

```
#!/bin/bash
_get_nginx_status() {    ###通过wget本地nginx某个端口的某个页面来获取nginx状态
 wget 127.0.0.1/index.html -O /dev/null -q
}
_get_nginx_status
[ $? -eq 0 ] && {
 exit 0
} || {
 exit -1
}
```

notify_master "/bin/bash /opt/check_shell/notify.sh Master"内容：

```
echo $(date +%F-%T)" Change to $1" >> /opt/check_shell/log
```

关键配置详细解释

vrrp_script中的weight -50

首先明确vrrp_script中脚本的返回值只有两种： 0 和 非0

如果脚本执行结果为0，并且weight配置的值大于0，则优先级相应的增加

如果脚本执行结果非0，并且weight配置的值小于0，则优先级相应的减少

其他情况，维持原本配置的优先级，即配置文件中priority对应的值。

第一种情况，
正常时 p1 + w1 > p2 + w2
异常时 p1 < p2 + w2

第二种情况，
正常时 p1 > p2
异常时 p1 - w1 < p2

from 范兆明

在一般的生产环境中，我们碰到的场景一般用第二条就可以满足了，即

weight -50 配置负值，当脚本检测异常返回非0时，该节点的优先级就会相应降低50；

fall 3 在keepalived认为failded之前，需要3次脚本返回非0（若不配置默认为1次，即只要脚本检测返回非0一次，keepalived就会认为failed）；

vrrp_script failed只会出现一次日志，优先级也只会降低一次（在keepalived认为failed之后，打印一次日志，随后不会在打印failed的日志，也不会再对优先级进行扣减） --- 因此，不要指望连续5次检测失败返回非0，然后会连续5次对优先级进行扣减的情况出现！！！

keepalived failed之后再重新successed，优先级会重置，如初始优先级为100，failed后扣减为80，那么重新successed之后，会变回100，而不是停留在80!

vrrp_script中的fall 3

即在keepalived认为failed之前，需要经历3次该检测脚本返回非0；
优先级会在3次连续失败之后，扣减一次；
只有连续3次的非0，才会认为是失败，而不是累计3次；
配合上interval 5的话，会在nginx宕机5*3=15秒之后，keepalived才会认为是failed，然后进行优先级扣减，最后进行vip切换。

相应的，还有一个rise 3，这个是在认为successed之前，需要经历3次脚本返回0

state BACKUP

state指定的状态是该节点启动keepalived时候的初始状态
最终的MASTER/BACKUP状态需要由优先级 以及 【启动顺序+是否主动抢占】来共同决定。

nopreempt

该配置为【不主动抢占】，默认不配置为【主动抢占】。
【主动抢占】谁的优先级高，就会成为MASTER。造成的结果为一个组内的MASTER永远是当前优先级最高的那个节点；
【不主动抢占】处于BACKUP的节点当发现自己的优先级处于组内最高的时候，也不会主动的去成为MASTER；只有在其余节点的keepalived处于fault状态（挂了），才会被动的去成为MASTER。

notify_master 节点进入MASTER状态时，会执行后面双引号中包含的内容
notify_backup 节点进入MASTER状态时，会执行后面双引号中包含的内容
notify_fault 在track_interface指定的接口down的时候，节点进入fault状态，此时同样会执行后面的内

容

个人认为较为合理的一个keepalived方案

主：state BACKUP ＋ priority (X+Y) ＋ nopreempt
备：state BACKUP ＋ priority X

①保证主的初始优先级大于备的初始优先级
②主的vrrp_script weight值需要为负，并且在failed的时候，能够使得主的优先级扣减后小于备的优先级
③主开启【不主动抢占】

那么，针对这个方案，一套正常的启动+切换流程如下：
①主机先起keepalived，由于当前组内只有该节点，因此主机成为MASTER获得vip
②备机后起keepalived(最好确保在主机获得vip之后再启动)，由于备机优先级低于主机，因此成为BACKUP
③发生异常时（如nginx宕了等等情况），vrrp_script脚本返回非0，keepalived认为该节点failed，对主机优先级进行扣减，
然后主机优先级低于了备机，备机成功成为MASTER获得vip，主机进入BACKUP
④人工对主机上的异常进行恢复，使得vrrp_script返回0，keepalived重新认为是successed，就对优先级进行了重置；
重置优先级过后，主机的优先级已经高于备机的优先级了，但是由于主机配置了【不主动抢占】，因此就算主机优先级高，也不会成为MASTER
⑤人工对备机上的keepalived进行重启（最好是先stop再start），备机keepalived stop的时候会发出fault信号，此时整个组内就只有主机的keepalived处于正常状态，
那么主机又会重新成为MASTER获得vip；随后备机keepalived启动成功，主备机又回到了正常MASTER/BACKUP状态。