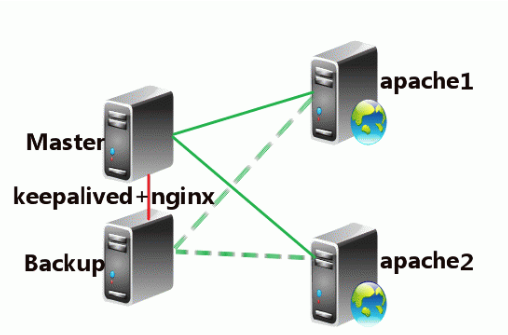
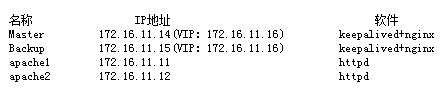
一　Keepalived+Nginx 实现双机热备

１，拓扑图

２，拓扑图的规划



３，此架构考虑的问题

1）、Master没挂，则Master占有vip且nginx运行在Master上

2）、Master挂了，则Backup抢占vip且在Backup上运行nginx服务

3）、如果Master服务器上的nginx服务挂了，则vip资源转移到Backup服务器上

4）、检测后端服务器的健康状态

Master和Backup两边都开启nginx服务，无论Master还是Backup，当其中的一个keepalived服务停止后，vip都会漂移到keepalived服务还在的节点上。

如果要想使nginx服务挂了，vip也漂移到另一个节点，则必须用脚本或者在配置文件里面用shell命令来控制。

首先必须明确后端服务器（apache）的健康状态检测keepalived在这种架构上是无法检测的，后端服务器的健康状态检测是有nginx来判断的，但是nginx 的检测机制有一定的缺陷，后端服务器某一个宕机之后，nginx还是会分发请求给它，在一定的时间内后端服务响应不了，nginx则会发给另外一个服务 器，然后当客户的请求来了，nginx会一段时间内不会把请求分发给已经宕机的服务器，但是过一段时间后，nginx还是会把分发请求发给宕机的服务器 上。

４，实验环境

Master：

OS ：CentOS 7

RIP ：172.16.11.14

Software ：

keepalived-1.3.5

nginx-1.15.0

Backup：

OS ：CentOS 7

RIP ：172.16.11.15

Software ：

keepalived-1.3.5

nginx-1.8.0

VIP ：172.16.11.16

５，配置文件

Master 的 nginx.conf 部分：

＃vim /usr/local/nginx/conf/nginx.conf

..........

upstream localhost {

server 172.16.11.11:80 weight=1 max\_fails=1 fail\_timeout=30;

server 172.16.11.12:80 weight=2 max\_fails=2 fail\_timeout=30;

}

server {

listen 8080;

server\_name 172.16.11.14;

#charset koi8-r;

#access\_log logs/host.access.log main;

location / {

proxy\_set\_header Host $host:8080;

proxy\_connect\_timeout 3;

proxy\_send\_timeout 30;

proxy\_read\_timeout 30;

proxy\_pass http://localhost;

}

}

...........

Backup 的 nginx.conf 部分：

＃vim /usr/local/nginx/conf/nginx.conf

........

upstream localhost {

server 172.16.11.11:80;

server 172.16.11.12:80;

}

server {

listen 8080;

server\_name 172.16.11.15;

#charset koi8-r;

#access\_log logs/host.access.log main;

location / {

proxy\_set\_header Host $host:8080;

proxy\_connect\_timeout 3;

proxy\_send\_timeout 30;

proxy\_read\_timeout 30;

proxy\_pass http://localhost;

}

}

........

Master 的 keepalived.conf：

# vim /etc/keepalived/keepalived.conf

! Configuration File for keepalived

global\_defs {

notification\_email {

root@localhost

}

notification\_email\_from keepalived@localhost

smtp\_server 127.0.0.1

smtp\_connect\_timeout 30

router\_id LVS\_DEVEL\_1

}

vrrp\_script chk\_http\_port {

script "</dev/tcp/127.0.0.1/8080" #用shell命令检查nginx服务是否运行

＃上一行也可换成：script "/opt/chk\_nginx.sh" 用shell脚本检查nginx服务是否运行

interval 1 #时间间隔为1秒检测一次

weight -2 #当nginx的服务不存在了，就把当前的权重-2

fall 2 #测试失败的次数

rise 1 #测试成功的次数

}

vrrp\_instance VI\_1 {

state MASTER #只有 MASTER 和 BACKUP 两种状态

interface eth0 #进行通信的端口

virtual\_router\_id 51 #同一个vrrp实例的唯一标识。即同一个vrrp\_stance,MASTER和BACKUP的virtual\_router\_id是一致的。在整个vrrp内也唯一。

priority 101 #权重，数值越大，权重越大。MASTER大于SLAVE

advert\_int 1 #MASTER和SLAVE负载均衡器之间同步检查的时间间隔。单位是：秒

authentication {

auth\_type PASS

auth\_pass 1111

}

virtual\_ipaddress {

172.16.11.16

}

track\_script {

chk\_http\_port #引用上面的vrrp\_script定义的脚本名称

}

}

virtual\_server 172.16.11.16 8080 {

delay\_loop 6

lb\_algo wlc #负载均衡调度算法rr|wrr|lc|wlc|sh|dh|lblc

lb\_kind DR #负载均衡转发规则NAT|DR|TUN

nat\_mask 255.255.255.0

#persistence\_timeout 60

protocol TCP

sorry\_server 127.0.0.1 80

real\_server 172.16.11.14 8080 {

weight 1 #权重，数值越大，权重越高。分发的可能越大

HTTP\_GET {

url {

path /

status\_code 200

}

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

}

Backup 的 keepalived.conf：

# vim /etc/keepalived/keepalived.conf

Backup 的 keepalived.conf：

! Configuration File for keepalived

global\_defs {

notification\_email {

root@localhost

}

notification\_email\_from keepalived@localhost

smtp\_server 127.0.0.1

smtp\_connect\_timeout 30

router\_id LVS\_DEVEL\_2

}

vrrp\_script chk\_http\_port {

script "</dev/tcp/127.0.0.1/8080" 　＃这一行也可换成：script "/opt/chk\_nginx.sh" 用shell脚本检查nginx服务是否运行

interval 1

weight -2

fall 2

rise 1

}

vrrp\_instance VI\_1 {

state MASTER

interface eth0

virtual\_router\_id 51

priority 100

advert\_int 1

authentication {

auth\_type PASS

auth\_pass 1111

}

virtual\_ipaddress {

172.16.11.16

}

track\_script {

chk\_http\_port

}

}

virtual\_server 172.16.11.16 8080 {

delay\_loop 6

lb\_algo wlc

lb\_kind DR

nat\_mask 255.255.255.0

#persistence\_timeout 50

protocol TCP

sorry\_server 127.0.0.1 80

real\_server 172.16.11.15 8080 {

weight 1

HTTP\_GET {

url {

path /

status\_code 200

}

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

}

}

}

注意：

state ： 都是MASTER（不是一个MASTER，一个BACKUP）

priority ： Master 的 > Backup 的

virtual\_router\_id： 参数值要一样

为什么主备的参数state都是MASTER，state都设置成MASTER后，会根据priority的值大小竞争来决定谁是真正的MASTER，脚本检测也是在失败的时候会把权重减去相应的值（-2），比如原来 Master(14) 的priority=101，如果脚本检测到端口8080无法连接，就会 priority-2=99，小于 Backup(15) 的priority（100），此时 Backup 将竞争成为MASTER，这样就实现了Nginx应用的热备

6，测试场景

# ip addr 　　　　查看网卡虚拟的ip

1.服务器层的双机热备

1）其中一个服务器宕机

2）其中一个服务器上的Keepalived宕掉

3）其中一个服务器网络不通

2.应用层的双机热备

1）其中一个服务器上的Nginx进程被意外kill

2) 其中一个服务器上的应用端口（8080)不通

脚本：

监控脚本如下（master和slave都要有这个监控脚本）：

[root@master-node ~]# vim /opt/chk\_nginx.sh

#!/bin/bash

counter=$( ss -antlp | grep -v grep | grep nginx | wc -l)

if [ $counter -eq 0 ];then

/usr/local/nginx/sbin/nginx

sleep 2

counter=$( ps aux | grep -v grep | grep nginx | wc -l)

if [ $counter -eq 0 ];then

systemctl stop keepalived

fi

fi

#!/bin/bash

counter=$(ps -C nginx --no-heading|wc -l)

if [ "${counter}" = "0" ]; then

/usr/local/nginx/sbin/nginx

sleep 2

counter=$(ps -C nginx --no-heading|wc -l)

if [ "${counter}" = "0" ]; then

/etc/init.d/keepalived stop

fi

fi