**[Nginx+keepalived双机热备（主从模式）](http://www.cnblogs.com/kevingrace/p/6138185.html)**

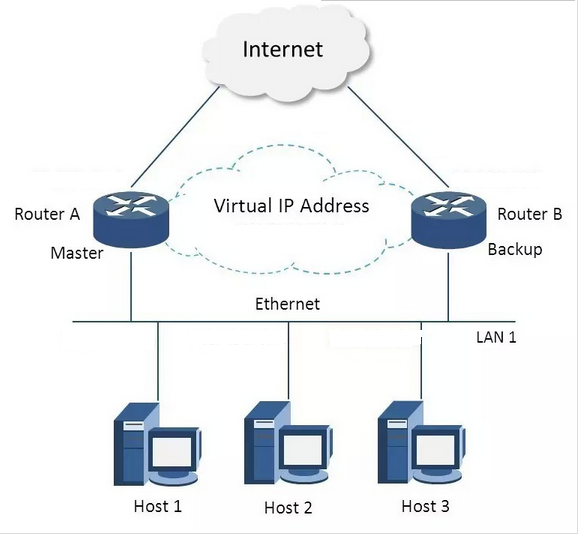
负载均衡技术对于一个网站尤其是大型网站的web服务器集群来说是至关重要的！做好负载均衡架构，可以实现故障转移和高可用环境，避免单点故障，保证网站健康持续运行。  
关于负载均衡介绍，可以参考：[linux负载均衡总结性说明（四层负载/七层负载）](http://www.cnblogs.com/kevingrace/p/6137881.html)

由于业务扩展，网站的访问量不断加大，负载越来越高。现需要在web前端放置nginx负载均衡,同时结合keepalived对前端nginx实现HA高可用。  
1）nginx进程基于Master+Slave(worker)多进程模型，自身具有非常稳定的子进程管理功能。在Master进程分配模式下，Master进程永远不进行业务处理，只是进行任务分发，从而达到Master进程的存活高可靠性，Slave(worker)进程所有的业务信号都 由主进程发出，Slave(worker)进程所有的超时任务都会被Master中止，属于非阻塞式任务模型。  
2）Keepalived是Linux下面实现VRRP备份路由的高可靠性运行件。基于Keepalived设计的服务模式能够真正做到主服务器和备份服务器故障时IP瞬间无缝交接。二者结合，可以构架出比较稳定的软件LB方案。

Keepalived介绍：  
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。但是它一般不会单独出现，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。

VRRP协议：  
VRRP全称 Virtual Router Redundancy Protocol，即 虚拟路由冗余协议。可以认为它是实现路由器高可用的容错协议，即将N台提供相同功能的路由器组成一个路由器组(Router Group)，这个组里面有一个master和多个backup，但在外界看来就像一台一样，构成虚拟路由器，拥有一个虚拟IP（vip，也就是路由器所在局域网内其他机器的默认路由），占有这个IP的master实际负责ARP相应和转发IP数据包，组中的其它路由器作为备份的角色处于待命状态。master会发组播消息，当backup在超时时间内收不到vrrp包时就认为master宕掉了，这时就需要根据VRRP的优先级来选举一个backup当master，保证路由器的高可用。

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。

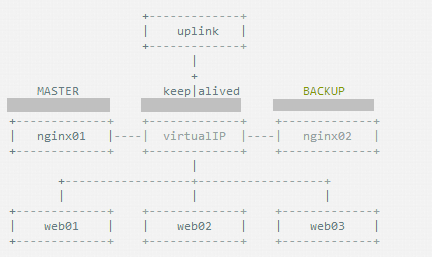


keepalived与heartbeat/corosync等比较：  
Heartbeat、Corosync、Keepalived这三个集群组件我们到底选哪个好呢？  
首先要说明的是，Heartbeat、Corosync是属于同一类型，Keepalived与Heartbeat、Corosync，根本不是同一类型的。  
Keepalived使用的vrrp协议方式，虚拟路由冗余协议 (Virtual Router Redundancy Protocol，简称VRRP)；  
Heartbeat或Corosync是基于主机或网络服务的高可用方式；  
简单的说就是，Keepalived的目的是模拟路由器的高可用，Heartbeat或Corosync的目的是实现Service的高可用。  
所以一般Keepalived是实现前端高可用，常用的前端高可用的组合有，就是我们常见的LVS+Keepalived、Nginx+Keepalived、HAproxy+Keepalived。而Heartbeat或Corosync是实现服务的高可用，常见的组合有Heartbeat v3(Corosync)+Pacemaker+NFS+Httpd 实现Web服务器的高可用、Heartbeat v3(Corosync)+Pacemaker+NFS+MySQL 实现MySQL服务器的高可用。总结一下，Keepalived中实现轻量级的高可用，一般用于前端高可用，且不需要共享存储，一般常用于两个节点的高可用。而Heartbeat(或Corosync)一般用于服务的高可用，且需要共享存储，一般用于多节点的高可用。这个问题我们说明白了。

那heartbaet与corosync又应该选择哪个好？  
一般用corosync，因为corosync的运行机制更优于heartbeat，就连从heartbeat分离出来的pacemaker都说在以后的开发当中更倾向于corosync，所以现在corosync+pacemaker是最佳组合。

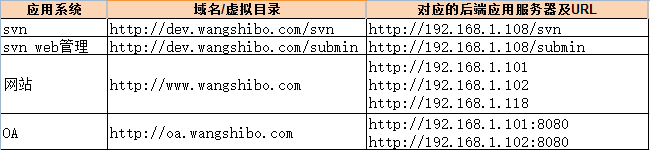
双机高可用一般是通过虚拟IP（飘移IP）方法来实现的，基于Linux/Unix的IP别名技术。  
双机高可用方法目前分为两种：  
1）双机主从模式：即前端使用两台服务器，一台主服务器和一台热备服务器，正常情况下，主服务器绑定一个公网虚拟IP，提供负载均衡服务，热备服务器处于空闲状态；当主服务器发生故障时，热备服务器接管主服务器的公网虚拟IP，提供负载均衡服务；但是热备服务器在主机器不出现故障的时候，永远处于浪费状态，对于服务器不多的网站，该方案不经济实惠。  
2）双机主主模式：即前端使用两台负载均衡服务器，互为主备，且都处于活动状态，同时各自绑定一个公网虚拟IP，提供负载均衡服务；当其中一台发生故障时，另一台接管发生故障服务器的公网虚拟IP（这时由非故障机器一台负担所有的请求）。这种方案，经济实惠，非常适合于当前架构环境。

**今天在此分享下Nginx+keepalived实现高可用负载均衡的主从模式的操作记录：**

keepalived可以认为是VRRP协议在Linux上的实现，主要有三个模块，分别是core、check和vrrp。  
core模块为keepalived的核心，负责主进程的启动、维护以及全局配置文件的加载和解析。  
check负责健康检查，包括常见的各种检查方式。  
vrrp模块是来实现VRRP协议的。  


一、环境说明：  
操作系统：centos6.8，64位  
master机器（master-node）：103.110.98.14/192.168.1.14  
slave机器（slave-node）：103.110.98.24/192.168.1.24  
公用的虚拟IP（VIP）：103.110.98.20      //负载均衡器上配置的域名都解析到这个VIP上

应用环境如下：



二、环境安装  
安装nginx和keepalive服务（master-node和slave-node两台服务器上的安装操作完全一样）。  
安装依赖  
[root@master-node ~]# yum -y install gcc pcre-devel zlib-devel openssl-devel  
下载（百度云盘下载地址：[https://pan.baidu.com/s/1ckTOKI](https://pan.baidu.com/s/1ckTOKI" \t "_blank)     提取秘钥：gi8i）  
[root@master-node ~]# cd /usr/local/src/  
[root@master-node src]# wget <http://nginx.org/download/nginx-1.9.7.tar.gz>  
[root@master-node src]# wget <http://www.keepalived.org/software/keepalived-1.3.2.tar.gz>  
安装nginx  
[root@master-node src]# tar -zvxf nginx-1.9.7.tar.gz   
[root@master-node src]# cd nginx-1.9.7  
添加www用户，其中-M参数表示不添加用户家目录，-s参数表示指定shell类型  
[root@master-node nginx-1.9.7]# useradd www -M -s /sbin/nologin   
[root@master-node nginx-1.9.7]# vim auto/cc/gcc   
#将这句注释掉 取消Debug编译模式 大概在179行  
#CFLAGS="$CFLAGS -g"  
[root@master-node nginx-1.9.7]# ./configure --prefix=/usr/local/nginx --user=www --group=www --with-http\_ssl\_module --with-http\_flv\_module --with-http\_stub\_status\_module --with-http\_gzip\_static\_module --with-pcre  
[root@master-node nginx-1.9.7]# make && make install  
安装keepalived  
[root@master-node src]# tar -zvxf keepalived-1.3.2.tar.gz   
[root@master-node src]# cd keepalived-1.3.2  
[root@master-node keepalived-1.3.2]# ./configure   
[root@master-node keepalived-1.3.2]# make && make install  
[root@master-node keepalived-1.3.2]# cp /usr/local/src/keepalived-1.3.2/keepalived/etc/init.d/keepalived /etc/rc.d/init.d/  
[root@master-node keepalived-1.3.2]# cp /usr/local/etc/sysconfig/keepalived /etc/sysconfig/  
[root@master-node keepalived-1.3.2]# mkdir /etc/keepalived  
[root@master-node keepalived-1.3.2]# cp /usr/local/etc/keepalived/keepalived.conf /etc/keepalived/  
[root@master-node keepalived-1.3.2]# cp /usr/local/sbin/keepalived /usr/sbin/  
将nginx和keepalive服务加入开机启动服务  
[root@master-node keepalived-1.3.2]# echo "/usr/local/nginx/sbin/nginx" >> /etc/rc.local  
[root@master-node keepalived-1.3.2]# echo "/etc/init.d/keepalived start" >> /etc/rc.local

三、配置服务

先关闭SElinux、配置防火墙 （master和slave两台负载均衡机都要做）  
[root@master-node ~]# vim /etc/sysconfig/selinux  
#SELINUX=enforcing                      #注释掉  
#SELINUXTYPE=targeted                #注释掉  
SELINUX=disabled                           #增加  
[root@master-node ~]# setenforce 0                               #使配置立即生效

[root@master-node ~]# vim /etc/sysconfig/iptables  
.......  
-A INPUT -s 103.110.98.0/24 -d 224.0.0.18 -j ACCEPT                        #允许组播地址通信  
-A INPUT -s 192.168.1.0/24 -d 224.0.0.18 -j ACCEPT  
-A INPUT -s 103.110.98.0/24 -p vrrp -j ACCEPT                                  #允许 VRRP（虚拟路由器冗余协）通信  
-A INPUT -s 192.168.1.0/24 -p vrrp -j ACCEPT  
-A INPUT -p tcp -m state --state NEW -m tcp --dport 80 -j ACCEPT      #开通80端口访问

[root@master-node ~]# /etc/init.d/iptables restart                               #重启防火墙使配置生效

1.配置nginx  
master-node和slave-node两台服务器的nginx的配置完全一样,主要是配置/usr/local/nginx/conf/nginx.conf的http，当然也可以配置vhost虚拟主机目录，然后配置vhost下的比如LB.conf文件。  
其中:  
多域名指向是通过虚拟主机（配置http下面的server）实现;  
同一域名的不同虚拟目录通过每个server下面的不同location实现;  
到后端的服务器在vhost/LB.conf下面配置upstream,然后在server或location中通过proxy\_pass引用。  
要实现前面规划的接入方式，LB.conf的配置如下（添加proxy\_cache\_path和proxy\_temp\_path这两行，表示打开nginx的缓存功能）：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65 | [root@master-node ~]# vim /usr/local/nginx/conf/nginx.conf  user  www;  worker\_processes  8;    #error\_log  logs/error.log;  #error\_log  logs/error.log  notice;  #error\_log  logs/error.log  info;    #pid        logs/nginx.pid;      events {      worker\_connections  65535;  }      http {      include       mime.types;      default\_type  application/octet-stream;      charset utf-8;        ######      ## set access log format      ######      log\_format  main  '$http\_x\_forwarded\_for $remote\_addr $remote\_user [$time\_local] "$request" '                        '$status $body\_bytes\_sent "$http\_referer" '                        '"$http\_user\_agent" "$http\_cookie" $host $request\_time';        #######      ## http setting      #######      sendfile       on;      tcp\_nopush     on;      tcp\_nodelay    on;      keepalive\_timeout  65;      proxy\_cache\_path /var/www/cache levels=1:2 keys\_zone=mycache:20m max\_size=2048m inactive=60m;      proxy\_temp\_path /var/www/cache/tmp;        fastcgi\_connect\_timeout 3000;      fastcgi\_send\_timeout 3000;      fastcgi\_read\_timeout 3000;      fastcgi\_buffer\_size 256k;      fastcgi\_buffers 8 256k;      fastcgi\_busy\_buffers\_size 256k;      fastcgi\_temp\_file\_write\_size 256k;      fastcgi\_intercept\_errors on;        #      client\_header\_timeout 600s;      client\_body\_timeout 600s;     # client\_max\_body\_size 50m;      client\_max\_body\_size 100m;               #允许客户端请求的最大单个文件字节数      client\_body\_buffer\_size 256k;            #缓冲区代理缓冲请求的最大字节数，可以理解为先保存到本地再传给用户        gzip  on;      gzip\_min\_length  1k;      gzip\_buffers     4 16k;      gzip\_http\_version 1.1;      gzip\_comp\_level 9;      gzip\_types       text/plain application/x-javascript text/css application/xml text/javascript application/x-httpd-php;      gzip\_vary on;        ## includes vhosts      include vhosts/\*.conf;  } |

[root@master-node ~]# mkdir /usr/local/nginx/conf/vhosts  
[root@master-node ~]# mkdir /var/www/cache  
[root@master-node ~]# ulimit 65535

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121 | [root@master-node ~]# vim /usr/local/nginx/conf/vhosts/LB.conf  upstream LB-WWW {        ip\_hash;        server 192.168.1.101:80 max\_fails=3 fail\_timeout=30s;     #max\_fails = 3 为允许失败的次数，默认值为1        server 192.168.1.102:80 max\_fails=3 fail\_timeout=30s;     #fail\_timeout = 30s 当max\_fails次失败后，暂停将请求分发到该后端服务器的时间        server 192.168.1.118:80 max\_fails=3 fail\_timeout=30s;      }    upstream LB-OA {        ip\_hash;        server 192.168.1.101:8080 max\_fails=3 fail\_timeout=30s;        server 192.168.1.102:8080 max\_fails=3 fail\_timeout=30s;  }      server {        listen      80;        server\_name dev.wangshibo.com;          access\_log  /usr/local/nginx/logs/dev-access.log main;        error\_log  /usr/local/nginx/logs/dev-error.log;          location /svn {           proxy\_pass http://192.168.1.108/svn/;           proxy\_redirect off ;           proxy\_set\_header Host $host;           proxy\_set\_header X-Real-IP $remote\_addr;           proxy\_set\_header REMOTE-HOST $remote\_addr;           proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;           proxy\_connect\_timeout 300;             #跟后端服务器连接超时时间，发起握手等候响应时间           proxy\_send\_timeout 300;                #后端服务器回传时间，就是在规定时间内后端服务器必须传完所有数据           proxy\_read\_timeout 600;                #连接成功后等待后端服务器的响应时间，已经进入后端的排队之中等候处理           proxy\_buffer\_size 256k;                #代理请求缓冲区,会保存用户的头信息以供nginx进行处理           proxy\_buffers 4 256k;                  #同上，告诉nginx保存单个用几个buffer最大用多少空间           proxy\_busy\_buffers\_size 256k;          #如果系统很忙时候可以申请最大的proxy\_buffers           proxy\_temp\_file\_write\_size 256k;       #proxy缓存临时文件的大小           proxy\_next\_upstream error timeout invalid\_header http\_500 http\_503 http\_404;           proxy\_max\_temp\_file\_size 128m;           proxy\_cache mycache;           proxy\_cache\_valid 200 302 60m;           proxy\_cache\_valid 404 1m;         }          location /submin {           proxy\_pass http://192.168.1.108/submin/;           proxy\_redirect off ;           proxy\_set\_header Host $host;           proxy\_set\_header X-Real-IP $remote\_addr;           proxy\_set\_header REMOTE-HOST $remote\_addr;           proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;           proxy\_connect\_timeout 300;           proxy\_send\_timeout 300;           proxy\_read\_timeout 600;           proxy\_buffer\_size 256k;           proxy\_buffers 4 256k;           proxy\_busy\_buffers\_size 256k;           proxy\_temp\_file\_write\_size 256k;           proxy\_next\_upstream error timeout invalid\_header http\_500 http\_503 http\_404;           proxy\_max\_temp\_file\_size 128m;           proxy\_cache mycache;           proxy\_cache\_valid 200 302 60m;           proxy\_cache\_valid 404 1m;          }      }    server {       listen       80;       server\_name  www.wangshibo.com;          access\_log  /usr/local/nginx/logs/www-access.log main;        error\_log  /usr/local/nginx/logs/www-error.log;         location / {           proxy\_pass http://LB-WWW;           proxy\_redirect off ;           proxy\_set\_header Host $host;           proxy\_set\_header X-Real-IP $remote\_addr;           proxy\_set\_header REMOTE-HOST $remote\_addr;           proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;           proxy\_connect\_timeout 300;           proxy\_send\_timeout 300;           proxy\_read\_timeout 600;           proxy\_buffer\_size 256k;           proxy\_buffers 4 256k;           proxy\_busy\_buffers\_size 256k;           proxy\_temp\_file\_write\_size 256k;           proxy\_next\_upstream error timeout invalid\_header http\_500 http\_503 http\_404;           proxy\_max\_temp\_file\_size 128m;           proxy\_cache mycache;           proxy\_cache\_valid 200 302 60m;           proxy\_cache\_valid 404 1m;          }  }     server {         listen       80;         server\_name  oa.wangshibo.com;          access\_log  /usr/local/nginx/logs/oa-access.log main;        error\_log  /usr/local/nginx/logs/oa-error.log;           location / {           proxy\_pass http://LB-OA;           proxy\_redirect off ;           proxy\_set\_header Host $host;           proxy\_set\_header X-Real-IP $remote\_addr;           proxy\_set\_header REMOTE-HOST $remote\_addr;           proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;           proxy\_connect\_timeout 300;           proxy\_send\_timeout 300;           proxy\_read\_timeout 600;           proxy\_buffer\_size 256k;           proxy\_buffers 4 256k;           proxy\_busy\_buffers\_size 256k;           proxy\_temp\_file\_write\_size 256k;           proxy\_next\_upstream error timeout invalid\_header http\_500 http\_503 http\_404;           proxy\_max\_temp\_file\_size 128m;           proxy\_cache mycache;           proxy\_cache\_valid 200 302 60m;           proxy\_cache\_valid 404 1m;          }  } |

验证方法（保证从负载均衡器本机到后端真实服务器之间能正常通信）：  
1）首先在本机用IP访问上面LB.cong中配置的各个后端真实服务器的url  
2）然后在本机用域名和路径访问上面LB.cong中配置的各个后端真实服务器的域名/虚拟路径

----------------------------------------------------------------------------------------------------------------------------  
后端应用服务器的nginx配置，这里选择192.168.1.108作为例子进行说明  
由于这里的192.168.1.108机器是openstack的虚拟机，没有外网ip，不能解析域名。  
所以在server\_name处也将ip加上，使得用ip也可以访问。  
[root@108-server ~]# cat /usr/local/nginx/conf/vhosts/svn.conf   
server {  
listen 80;  
#server\_name dev.wangshibo.com;  
server\_name dev.wangshibo.com 192.168.1.108;

access\_log /usr/local/nginx/logs/dev.wangshibo-access.log main;  
error\_log /usr/local/nginx/logs/dev.wangshibo-error.log;

location / {  
root /var/www/html;  
index index.html index.php index.htm;  
}  
}

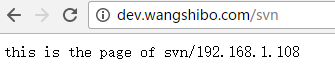
[root@108-server ~]# ll /var/www/html/  
drwxr-xr-x. 2 www www 4096 Dec 7 01:46 submin  
drwxr-xr-x. 2 www www 4096 Dec 7 01:45 svn  
[root@108-server ~]# cat /var/www/html/svn/index.html  
this is the page of svn/192.168.1.108  
[root@108-server ~]# cat /var/www/html/submin/index.html  
this is the page of submin/192.168.1.108

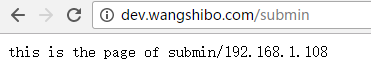
[root@108-server ~]# cat /etc/hosts  
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4  
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6  
192.168.1.108 dev.wangshibo.com

[root@108-server ~]# curl http://dev.wangshibo.com       //由于是内网机器不能联网，亦不能解析域名。所以用域名访问没有反应。只能用ip访问  
[root@ops-server4 vhosts]# curl http://192.168.1.108  
this is 192.168.1.108 page!!!  
[root@ops-server4 vhosts]# curl http://192.168.1.108/svn/           //最后一个/符号要加上，否则访问不了。  
this is the page of svn/192.168.1.108  
[root@ops-server4 vhosts]# curl http://192.168.1.108/submin/  
this is the page of submin/192.168.1.108  
----------------------------------------------------------------------------------------------------------------------------

然后在master-node和slave-node两台负载机器上进行测试（iptables防火墙要开通80端口）：  
[root@master-node ~]# curl http://192.168.1.108/svn/  
this is the page of svn/192.168.1.108  
[root@master-node ~]# curl http://192.168.1.108/submin/  
this is the page of submin/192.168.1.108

浏览器访问：  
在本机host绑定dev.wangshibo.com，如下，即绑定到master和slave机器的公网ip上测试是否能正常访问（nginx+keepalive环境正式完成后，域名解析到的真正地址是VIP地址）  
103.110.98.14 dev.wangshibo.com  
103.110.98.24 dev.wangshibo.com





2.keepalived配置  
1）master-node负载机上的keepalived配置（sendmail部署可以参考：[linux下sendmail邮件系统安装操作记录](http://www.cnblogs.com/kevingrace/p/6143977.html)）  
[root@master-node ~]# cp /etc/keepalived/keepalived.conf /etc/keepalived/keepalived.conf.bak  
[root@master-node ~]# vim /etc/keepalived/keepalived.conf

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41 | ! Configuration File for keepalived     #全局定义    global\_defs {  notification\_email {     #指定keepalived在发生事件时(比如切换)发送通知邮件的邮箱  ops@wangshibo.cn   #设置报警邮件地址，可以设置多个，每行一个。 需开启本机的sendmail服务  tech@wangshibo.cn  }    notification\_email\_from ops@wangshibo.cn   #keepalived在发生诸如切换操作时需要发送email通知地址  smtp\_server 127.0.0.1      #指定发送email的smtp服务器  smtp\_connect\_timeout 30    #设置连接smtp server的超时时间  router\_id master-node     #运行keepalived的机器的一个标识，通常可设为hostname。故障发生时，发邮件时显示在邮件主题中的信息。  }    vrrp\_script chk\_http\_port {      #检测nginx服务是否在运行。有很多方式，比如进程，用脚本检测等等      script "/opt/chk\_nginx.sh"  #这里通过脚本监测      interval 2           #脚本执行间隔，每2s检测一次      weight -5           #脚本结果导致的优先级变更，检测失败（脚本返回非0）则优先级priority -5      fall 2             #检测连续2次失败才算确定是真失败。会用weight减少优先级（1-255之间）      rise 1             #检测1次成功就算成功。但不修改优先级  }    vrrp\_instance VI\_1 {    #keepalived在同一virtual\_router\_id中priority（0-255）最大的会成为master，也就是接管VIP，当priority最大的主机发生故障后次priority将会接管      state MASTER    #指定keepalived的角色，MASTER表示此主机是主服务器，BACKUP表示此主机是备用服务器。注意这里的state指定instance(Initial)的初始状态，就是说在配置好后，这台服务器的初始状态就是这里指定的，但这里指定的不算，还是得要通过竞选通过优先级来确定。如果这里设置为MASTER，但如若他的优先级不及另外一台，那么这台在发送通告时，会发送自己的优先级，另外一台发现优先级不如自己的高，那么他会就回抢占为MASTER      interface em1          #指定HA监测网络的接口。实例绑定的网卡，因为在配置虚拟IP的时候必须是在已有的网卡上添加的      mcast\_src\_ip 103.110.98.14  # 发送多播数据包时的源IP地址，这里注意了，这里实际上就是在哪个地址上发送VRRP通告，这个非常重要，一定要选择稳定的网卡端口来发送，这里相当于heartbeat的心跳端口，如果没有设置那么就用默认的绑定的网卡的IP，也就是interface指定的IP地址      virtual\_router\_id 51         #虚拟路由标识，这个标识是一个数字，同一个vrrp实例使用唯一的标识。即同一vrrp\_instance下，MASTER和BACKUP必须是一致的      priority 101                 #定义优先级，数字越大，优先级越高，在同一个vrrp\_instance下，MASTER的优先级必须大于BACKUP的优先级      advert\_int 1                 #设定MASTER与BACKUP负载均衡器之间同步检查的时间间隔，单位是秒      authentication {             #设置验证类型和密码。主从必须一样          auth\_type PASS           #设置vrrp验证类型，主要有PASS和AH两种          auth\_pass 1111           #设置vrrp验证密码，在同一个vrrp\_instance下，MASTER与BACKUP必须使用相同的密码才能正常通信      }      virtual\_ipaddress {          #VRRP HA 虚拟地址 如果有多个VIP，继续换行填写          103.110.98.20      }    track\_script {                      #执行监控的服务。注意这个设置不能紧挨着写在vrrp\_script配置块的后面（实验中碰过的坑），否则nginx监控失效！！     chk\_http\_port                    #引用VRRP脚本，即在 vrrp\_script 部分指定的名字。定期运行它们来改变优先级，并最终引发主备切换。  }  } |

2）slave-node负载机上的keepalived配置  
[root@slave-node ~]# cp /etc/keepalived/keepalived.conf /etc/keepalived/keepalived.conf.bak  
[root@slave-node ~]# vim /etc/keepalived/keepalived.conf

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42 | ! Configuration File for keepalived    global\_defs {  notification\_email {  ops@wangshibo.cn  tech@wangshibo.cn  }    notification\_email\_from ops@wangshibo.cn  smtp\_server 127.0.0.1  smtp\_connect\_timeout 30  router\_id slave-node  }    vrrp\_script chk\_http\_port {      script "/opt/chk\_nginx.sh"      interval 2      weight -5      fall 2      rise 1  }    vrrp\_instance VI\_1 {      state BACKUP      interface em1      mcast\_src\_ip 103.110.98.24      virtual\_router\_id 51      priority 99      advert\_int 1      authentication {          auth\_type PASS          auth\_pass 1111      }      virtual\_ipaddress {          103.110.98.20      }    track\_script {     chk\_http\_port  }    } |

让keepalived监控NginX的状态：  
1）经过前面的配置，如果master主服务器的keepalived停止服务，slave从服务器会自动接管VIP对外服务；  
一旦主服务器的keepalived恢复，会重新接管VIP。 但这并不是我们需要的，我们需要的是当NginX停止服务的时候能够自动切换。  
2）keepalived支持配置监控脚本，我们可以通过脚本监控NginX的状态，如果状态不正常则进行一系列的操作，最终仍不能恢复NginX则杀掉keepalived，使得从服务器能够接管服务。

如何监控NginX的状态  
最简单的做法是监控NginX进程，更靠谱的做法是检查NginX端口，最靠谱的做法是检查多个url能否获取到页面。

注意：这里要提示一下keepalived.conf中vrrp\_script配置区的script一般有2种写法：  
1）通过脚本执行的返回结果，改变优先级，keepalived继续发送通告消息，backup比较优先级再决定。这是直接监控Nginx进程的方式。  
2）脚本里面检测到异常，直接关闭keepalived进程，backup机器接收不到advertisement会抢占IP。这是检查NginX端口的方式。  
上文script配置部分，"killall -0 nginx"属于第1种情况，"/opt/chk\_nginx.sh" 属于第2种情况。个人更倾向于通过shell脚本判断，但有异常时exit 1，正常退出exit 0，然后keepalived根据动态调整的 vrrp\_instance 优先级选举决定是否抢占VIP：  
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，原本配置的优先级不变，即配置文件中priority对应的值。

**提示：**  
优先级不会不断的提高或者降低  
可以编写多个检测脚本并为每个检测脚本设置不同的weight（在配置中列出就行）  
不管提高优先级还是降低优先级，最终优先级的范围是在[1,254]，不会出现优先级小于等于0或者优先级大于等于255的情况  
在MASTER节点的 vrrp\_instance 中 配置 **nopreempt** ，当它异常恢复后，即使它 prio 更高也不会抢占，这样可以避免正常情况下做无谓的切换  
以上可以做到利用脚本检测业务进程的状态，并动态调整优先级从而实现主备切换。

另外：在默认的keepalive.conf里面还有 virtual\_server,real\_server 这样的配置，我们这用不到，它是为lvs准备的。

如何尝试恢复服务  
由于keepalived只检测本机和他机keepalived是否正常并实现VIP的漂移，而如果本机nginx出现故障则不会漂移VIP。  
所以编写脚本来判断本机nginx是否正常，如果发现NginX不正常，重启之。等待3秒再次校验，仍然失败则不再尝试，关闭keepalived，其他主机此时会接管VIP；

根据上述策略很容易写出监控脚本。此脚本必须在keepalived服务运行的前提下才有效！如果在keepalived服务先关闭的情况下，那么nginx服务关闭后就不能实现自启动了。  
该脚本检测ngnix的运行状态，并在nginx进程不存在时尝试重新启动ngnix，如果启动失败则停止keepalived，准备让其它机器接管。  
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

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | #!/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 |

[root@master-node ~]# chmod 755 /opt/chk\_nginx.sh  
[root@master-node ~]# sh /opt/chk\_nginx.sh  
80/tcp open http

此架构需考虑的问题  
1）master没挂，则master占有vip且nginx运行在master上  
2）master挂了，则slave抢占vip且在slave上运行nginx服务  
3）如果master上的nginx服务挂了，则nginx会自动重启，重启失败后会自动关闭keepalived，这样vip资源也会转移到slave上。  
4）检测后端服务器的健康状态  
5）master和slave两边都开启nginx服务，无论master还是slave，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。（nginx服务宕停后会自动启动，启动失败后会强制关闭keepalived，从而致使vip资源漂移到另一台机器上）

最后验证（将配置的后端应用域名都解析到VIP地址上）：关闭主服务器上的keepalived或nginx，vip都会自动飘到从服务器上。  
**验证keepalived服务故障情况：**  
1）先后在master、slave服务器上启动nginx和keepalived，保证这两个服务都正常开启:  
[root@master-node ~]# /usr/local/nginx/sbin/nginx   
[root@master-node ~]# /etc/init.d/keepalived start  
[root@slave-node ~]# /usr/local/nginx/sbin/nginx   
[root@slave-node ~]# /etc/init.d/keepalived start  
2）在主服务器上查看是否已经绑定了虚拟IP：   
[root@master-node ~]# ip addr  
.......  
2: em1: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc mq state UP qlen 1000  
link/ether 44:a8:42:17:3d:dd brd ff:ff:ff:ff:ff:ff  
inet 103.110.98.14/26 brd 103.10.86.63 scope global em1  
valid\_lft forever preferred\_lft forever  
inet 103.110.98.20/32 scope global em1  
valid\_lft forever preferred\_lft forever  
inet 103.110.98.20/26 brd 103.10.86.63 scope global secondary em1:0  
valid\_lft forever preferred\_lft forever  
inet6 fe80::46a8:42ff:fe17:3ddd/64 scope link   
valid\_lft forever preferred\_lft forever  
......  
3）停止主服务器上的keepalived:   
[root@master-node ~]# /etc/init.d/keepalived stop   
Stopping keepalived (via systemctl): [ OK ]  
[root@master-node ~]# /etc/init.d/keepalived status  
[root@master-node ~]# ps -ef|grep keepalived  
root 26952 24348 0 17:49 pts/0 00:00:00 grep --color=auto keepalived  
[root@master-node ~]#   
4）然后在从服务器上查看，发现已经接管了VIP：  
[root@slave-node ~]# ip addr  
.......  
2: em1: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc mq state UP qlen 1000  
link/ether 44:a8:42:17:3c:a5 brd ff:ff:ff:ff:ff:ff  
inet 103.110.98.24/26 brd 103.10.86.63 scope global em1  
inet 103.110.98.20/32 scope global em1  
inet6 fe80::46a8:42ff:fe17:3ca5/64 scope link   
valid\_lft forever preferred\_lft forever  
.......  
发现master的keepalived服务挂了后，vip资源自动漂移到slave上，并且网站正常访问，丝毫没有受到影响！  
5）重新启动主服务器上的keepalived，发现主服务器又重新接管了VIP，此时slave机器上的VIP已经不在了。  
[root@master-node ~]# /etc/init.d/keepalived start  
Starting keepalived (via systemctl): [ OK ]  
[root@master-node ~]# ip addr  
.......  
2: em1: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc mq state UP qlen 1000  
link/ether 44:a8:42:17:3d:dd brd ff:ff:ff:ff:ff:ff  
inet 103.110.98.14/26 brd 103.10.86.63 scope global em1  
valid\_lft forever preferred\_lft forever  
inet 103.110.98.20/32 scope global em1  
valid\_lft forever preferred\_lft forever  
inet 103.110.98.20/26 brd 103.10.86.63 scope global secondary em1:0  
valid\_lft forever preferred\_lft forever  
inet6 fe80::46a8:42ff:fe17:3ddd/64 scope link   
valid\_lft forever preferred\_lft forever  
......

[root@slave-node ~]# ip addr  
.......  
2: em1: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc mq state UP qlen 1000  
link/ether 44:a8:42:17:3c:a5 brd ff:ff:ff:ff:ff:ff  
inet 103.110.98.24/26 brd 103.10.86.63 scope global em1  
inet6 fe80::46a8:42ff:fe17:3ca5/64 scope link   
valid\_lft forever preferred\_lft forever

**接着验证下nginx服务故障，看看keepalived监控nginx状态的脚本是否正常？**  
如下：手动关闭master机器上的nginx服务，最多2秒钟后就会自动起来（因为keepalive监控nginx状态的脚本执行间隔时间为2秒）。域名访问几乎不受影响！  
[root@master-node ~]# /usr/local/nginx/sbin/nginx -s stop  
[root@master-node ~]# ps -ef|grep nginx  
root 28401 24826 0 19:43 pts/1 00:00:00 grep --color=auto nginx  
[root@master-node ~]# ps -ef|grep nginx  
root 28871 28870 0 19:47 ? 00:00:00 /bin/sh /opt/chk\_nginx.sh  
root 28875 24826 0 19:47 pts/1 00:00:00 grep --color=auto nginx  
[root@master-node ~]# ps -ef|grep nginx  
root 28408 1 0 19:43 ? 00:00:00 nginx: master process /usr/local/nginx/sbin/nginx  
www 28410 28408 0 19:43 ? 00:00:00 nginx: worker process  
www 28411 28408 0 19:43 ? 00:00:00 nginx: worker process  
www 28412 28408 0 19:43 ? 00:00:00 nginx: worker process  
www 28413 28408 0 19:43 ? 00:00:00 nginx: worker process

最后可以查看两台服务器上的/var/log/messages，观察VRRP日志信息的vip漂移情况~~~~

**-------------------------------------------------------------------------------------------------------------------------------------------------**  
可能出现的问题：  
**1）VIP绑定失败**  
原因可能有：  
-> iptables开启后，没有开放允许VRRP协议通信的策略（也有可能导致脑裂）；可以选择关闭iptables  
-> keepalived.conf文件配置有误导致，比如interface绑定的设备错误

**2）VIP绑定后，外部ping不通**  
可能的原因是：  
-> 网络故障，可以检查下网关是否正常；  
-> 网关的arp缓存导致，可以进行arp更新，命令是"arping -I 网卡名 -c 5 -s VIP 网关"