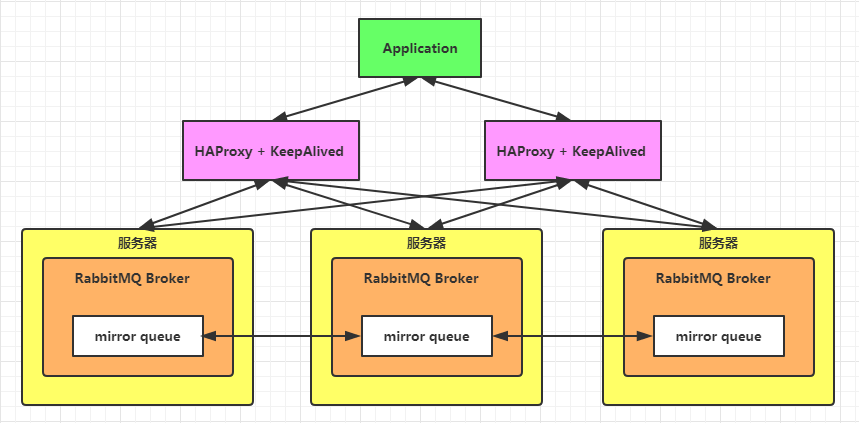
## 基于docker容器

参考：

[Docker部署rabbitmq集群](https://blog.csdn.net/xingkongtianma01/article/details/109593561)

[Docker部署rabbitmq集群【运维】](https://blog.csdn.net/xia296/article/details/108395796)

目标：



### 准备docker-compose文件

|  |
| --- |
| version: '3'  services:    mq\_node1:      image: rabbitmq:3-management-alpine      container\_name: mq\_node1      hostname: node1      ports:          - "15673:15672"          - "7563:5672"      environment:          - RABBITMQ\_ERLANG\_COOKIE=rabbitcookie      volumes:          - E:/test/rabbitmq/node1:/var/lib/rabbitmq/mnesia    mq\_node2:      image: rabbitmq:3-management-alpine      container\_name: mq\_node2      hostname: node2      ports:          - "15674:15672"          - "5674:5672"      environment:          -  RABBITMQ\_ERLANG\_COOKIE=rabbitcookie      links:          - mq\_node1:node1      volumes:          - E:/test/rabbitmq/node2:/var/lib/rabbitmq/mnesia    mq\_node3:      image: rabbitmq:3-management-alpine      container\_name: mq\_node3      hostname: node3      ports:          - "15675:15672"          - "5675:5672"      environment:          - RABBITMQ\_ERLANG\_COOKIE=rabbitcookie      links:          - mq\_node1:node1          - mq\_node2:node2      volumes:          - E:/test/rabbitmq/node3:/var/lib/rabbitmq/mnesia |

### 启动容器组

|  |
| --- |
| docker-compose up |

### 普通模式集群

一个磁盘节点【主节点】，2个内存节点【子节点】

优点：

1. 只访问主节点即可
2. 主节点数据会同步到子节点，在子节点的数据都会同步到主节点

缺点：

主节点挂了，其他节点都会挂。

|  |
| --- |
| docker exec -it mq\_node1 bash  rabbitmqctl stop\_app  rabbitmqctl reset  rabbitmqctl start\_app  exit |

|  |
| --- |
| docker exec -it mq\_node2 bash  rabbitmqctl stop\_app  rabbitmqctl reset  rabbitmqctl join\_cluster --ram rabbit@node1  rabbitmqctl start\_app  exit |

|  |
| --- |
| docker exec -it mq\_node3 bash  rabbitmqctl stop\_app  rabbitmqctl reset  rabbitmqctl join\_cluster --ram rabbit@node1  rabbitmqctl start\_app  exit |

### 镜像模式集群改造

优点：主节点挂了，其他子节点依旧存活

设置镜像队列【在任意一个节点执行】

|  |
| --- |
| #设置策略匹配所有名称的队列都进行高可用配置  rabbitmqctl set\_policy -p / ha "^" '{"ha-mode":"all","ha-sync-mode":"automatic"}'  #查询策略  rabbitmqctl list\_policies -p /  #删除策略  rabbitmqctl clear\_policy |

### HAProxy+KeepAlived【另外两台Linux机子】

[使用Docker构建RabbitMQ高可用负载均衡集群](https://www.cnblogs.com/csnjava/p/13213876.html)

[RabbitMQ 集群高可用原理及实战部署介绍](https://blog.csdn.net/javageektech/article/details/108505351)

#### HAProxy安装

|  |
| --- |
| yum install haproxy |

#### 编辑HAProxy配置文件

|  |
| --- |
| vim /etc/haproxy/haproxy.cfg |

|  |
| --- |
| #绑定RabbitMQ服务配置  listen rabbitmq\_cluster  bind 0.0.0.0:5672  #配置TCP模式  mode tcp  option tcplog  #加权轮询  balance roundrobin  #RabbitMQ集群节点配置  server node1 192.168.3.183:5673 check inter 5000 rise 2 fall 3 weight 1  server node2 192.168.3.183:5674 check inter 5000 rise 2 fall 3 weight 1  server node3 192.168.3.183:5675 check inter 5000 rise 2 fall 3 weight 1  #绑定RabbitMQ Web管理配置  listen rabbitmq\_mg\_cluster  bind 0.0.0.0:15672  #配置TCP模  mode tcp  option tcplog  #加权轮询  balance roundrobin  #RabbitMQ集群节点配置  server node1\_m 192.168.3.183:15673 check inter 5000 rise 2 fall 3 weight 1  server node2\_m 192.168.3.183:15674 check inter 5000 rise 2 fall 3 weight 1  server node3\_m 192.168.3.183:15675 check inter 5000 rise 2 fall 3 weight 1  #haproxy监控页面地址  listen monitor  bind 0.0.0.0:8100  mode http  option httplog  stats enable  stats uri /stats  stats auth admin:123456  stats refresh 5s |

启动haproxy

|  |
| --- |
| /usr/sbin/haproxy -f /etc/haproxy/haproxy.cfg |

### KeepAlived配置高可用

#### Yum安装【两台机子】

|  |
| --- |
| yum install ipvsadm -y  yum install keepalived -y |

#### 编辑配置文件

|  |
| --- |
| vim /etc/keepalived/keepalived.conf |

主节点【要特别注意监测网卡，可以先用ifconfig命令看当前机器使用哪个网卡】

|  |
| --- |
| #Keepalived配置文件  global\_defs {  router\_id NodeA #路由ID, 主备的ID不能相同  }  #自定义监控脚本  vrrp\_script chk\_haproxy {  script "/etc/keepalived/check\_haproxy.sh"  interval 5  weight 2  }  vrrp\_instance VI\_1 {  state MASTER #Keepalived的角色。Master表示主服务器，从服务器设置为BACKUP  interface wlp3s0 #指定监测网卡  virtual\_router\_id 1  priority 100 #优先级，BACKUP机器上的优先级要小于这个值  advert\_int 1 #设置主备之间的检查时间，单位为s  authentication { #定义验证类型和密码  auth\_type PASS  auth\_pass root123  }  track\_script {  chk\_haproxy  }  virtual\_ipaddress { #VIP地址，可以设置多个：  192.168.3.180  }  } |

备份节点

Backup中的配置大致和Master中的相同，不过需要修改global\_defs{}的router\_id，比如置为NodeB；其次要修改vrrp\_instance VI\_1{}中的state为BACKUP；最后要将priority设置为小于100的值。注意Master和Backup中的virtual\_router\_id要保持一致。

|  |
| --- |
| #Keepalived配置文件  global\_defs {  router\_id NodeB #路由ID, 主备的ID不能相同  }  #自定义监控脚本  vrrp\_script chk\_haproxy {  script "/etc/keepalived/check\_haproxy.sh"  interval 5  weight 2  }  vrrp\_instance VI\_1 {  state BACKUP  interface wlp3s0 #指定监测网卡  virtual\_router\_id 1  priority 80 #优先级，BACKUP机器上的优先级要小于这个值  advert\_int 1 #设置主备之间的检查时间，单位为s  authentication { #定义验证类型和密码  auth\_type PASS  auth\_pass root123  }  track\_script {  chk\_haproxy  }  virtual\_ipaddress { #VIP地址，可以设置多个：  192.168.3.180  }  } |

#### 准备监控脚本

为了防止HAProxy服务挂了，但是Keepalived却还在正常工作而没有切换到Backup上，所以这里需要编写一个脚本来检测HAProxy服务的状态。当HAProxy服务挂掉之后该脚本会自动重启HAProxy的服务，如果不成功则关闭Keepalived服务，如此便可以切换到Backup继续工作。这个脚本就对应了上面配置中vrrp\_script chk\_haproxy{}的script对应的值，/etc/keepalived/check\_haproxy.sh的内容如代码清单所示。

|  |
| --- |
| vim /etc/keepalived/check\_haproxy.sh |

|  |
| --- |
| #!/bin/bash  if [ $(ps -C haproxy --no-header | wc -l) -eq 0 ];then  systemctl start haproxy  fi  sleep 2  if [ $(ps -C haproxy --no-header | wc -l) -eq 0 ];then  systemctl stop keepalived  fi |

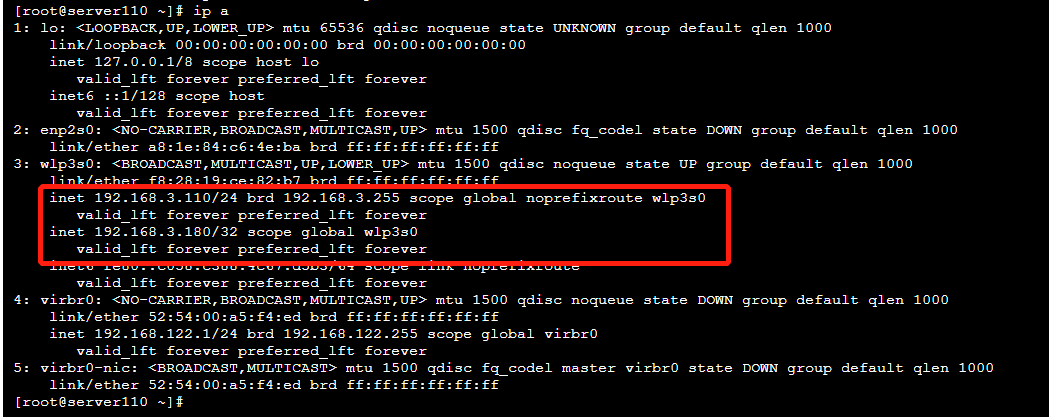
#### 启动keepalived服务

|  |
| --- |
| systemctl start keepalived |

#### 查看ip漂移

Keepalived启动后ip漂移在主节点上

|  |
| --- |
| ip a |



#### 通过virtual\_ipaddress访问RabbitMQ服务

192.168.3.180:15672