目录

[**1. 集群架构图 2**](#_Toc534726557)

[**2. Percona XtraDB Cluster集群 2**](#_Toc534726558)

[**2.1 不同主机或网络下的PXC集群 2**](#_Toc534726559)

[**2.2 docker镜像集群 3**](#_Toc534726560)

[**3. Mycat 集群 4**](#_Toc534726561)

[**3.1 下载 4**](#_Toc534726562)

[**3.2 解压缩 5**](#_Toc534726563)

[**3.3 修改server.xml配置文件 5**](#_Toc534726564)

[**3.4 修改schema.xml配置文件 6**](#_Toc534726565)

[**3.5 启动mycat 11**](#_Toc534726566)

[**3.6 连接mycat测试 11**](#_Toc534726567)

[**3.7 启动另一个mycat 11**](#_Toc534726568)

[**4. Haproxy集群 11**](#_Toc534726569)

[**4.1 创建haproxy.cfg配置文件 11**](#_Toc534726570)

[**4.2 启动HAProxy镜像 12**](#_Toc534726571)

[**4.3 访问监控页面 13**](#_Toc534726572)

[**4.4测试 13**](#_Toc534726573)

[**5. keepalived集群 13**](#_Toc534726574)

[**5.1 安装 13**](#_Toc534726575)

[**5.2 解压 13**](#_Toc534726576)

[**5.3 安装 13**](#_Toc534726577)

[**5.4 复制或者连接文件到指定目录 14**](#_Toc534726578)

[**5.5 创建启动项 14**](#_Toc534726579)

[**5.6 修改keepalived.conf配置文件 14**](#_Toc534726580)

[**5.7 设置更新虚拟服务器（VIP）地址的arp记录到网关脚本 16**](#_Toc534726581)

[**5.8 设置HAproxy服务监控脚本 17**](#_Toc534726582)

[**5.9 内核优化 18**](#_Toc534726583)

[**5.10 启动 19**](#_Toc534726584)

[**5.11 安装配置并启动另一台机器的keepalived 19**](#_Toc534726585)

[**5.12 测试 19**](#_Toc534726586)

[**6. 问题和解决方案 19**](#_Toc534726587)

[**6.1 Percona XtraDB Cluster问题 20**](#_Toc534726588)

[**6.2 keepalived 安装问题（返回安装流程） 20**](#_Toc534726589)

[**6.3 Linux内核升级方式 21**](#_Toc534726590)

1. **集群架构图**

****

1. **Percona XtraDB Cluster集群**
   1. **不同主机或网络下的PXC集群**

要在不同的主机下运行PXC，且保证PXC集群间通信，

需要创建swarm集群和集群overlay网络。

**（1）宿主机都在同一网络(网段)**

**master机器:**

docker swarm init --advertise-addr <master内网ip>

**slave机器:**

docker swarm join –token <token> --advertise-addr <slave内网ip>

**master机器Overlay网络:**

sudo docker network create -d overlay --subnet=10.10.0.0/16 --gateway=10.10.0.254 --attachable=true pxc-overlay

然后分别在两台机器启动结点，启动命令选项的--net=pxc-overlay

**（2）宿主机不在同一网络(网段)**

**master机器:**

docker swarm init --advertise-addr <master外网ip>

**slave机器:**

docker swarm join –token <token> --advertise-addr <slave外网ip>

**master机器Overlay网络:**

sudo docker network create -d overlay --subnet=10.10.0.0/16 --gateway=10.10.0.254 --attachable=true pxc-overlay

注意：这里使用的Overlay网络需要宿主机的内核版本大于或等于3.16

然后分别在两台机器启动结点，启动命令选项的--net=pxc-overlay

* 1. **docker镜像集群**

1. 创建存放持久化数据的目录

mkdir mysql1

mkdir mysql2

chmod 777 mysql1

chmod 777 mysql2

1. 创建网络pxc-net

docker network create -d bridge --subnet=10.10.1.0/24 --gateway=10.10.1.254 --attachable=true pxc-net

1. 创建结点1

docker run -d -p 3307:3306 -e MYSQL\_ROOT\_PASSWORD=654321 -e CLUSTER\_NAME=PXC -e XTRABACKUP\_PASSWORD=123456 -v /data/mysql1/:/var/lib/mysql --privileged --name=node1 --net=pxc-net percona/percona-xtradb-cluster

注：mysql的root用户密码为654321，xtrabackup用户密码为123456，结点1的名称为node1

复制时，注意--net=pxc-net前的两个横线符号

1. 进入结点1容器执行bash命令

docker exec -it node1 bash

1. 进入mysql

mysql -uroot -p

注：在容器内进入mysql

1. 授权

grant reload,lock tables,replication client,process on \*.\* to 'xtrabackup'@'localhost' IDENTIFIED BY '123456' WITH GRANT OPTION;

flush privileges;

1. 退出mysql

exit

exit

1. 创建结点2

docker run -d -p 3308:3306 -e MYSQL\_ROOT\_PASSWORD=654321 -e CLUSTER\_NAME=PXC -e XTRABACKUP\_PASSWORD=123456 -e CLUSTER\_JOIN=node1 -v /data/mysql2/:/var/lib/mysql --privileged --name=node2 --net=pxc-net percona/percona-xtradb-cluster

注：CLUSTER\_JOIN一定要填写结点1的容器名

复制时，注意--privileged前的两个横线符号

1. 测试

往其中一个数据库插入数据，另一个自动同步。

1. **Mycat 集群**
   1. **下载**

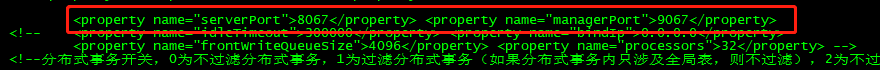
下载地址：<http://www.mycat.io/>

或者运行命令：wget http://dl.mycat.io/1.6.6.1/Mycat-server-1.6.6.1-release-20181031195535-linux.tar.gz

注：使用mycat需要先安装和配置JDK

这里启动的两个mycat分别在两台机器上，如果想要在一台机器上启动两个mycat，需要修改server.xml和wrapper.conf中的端口。

server.xml:



wrapper.conf



* 1. **解压缩**

tar –xf Mycat-server-1.6.6.1-release-20181031195535-linux.tar.gz

* 1. **修改server.xml配置文件**

vi mycat/conf/server.xml



<user name=”root”> 为用户名

<property name=”password”>为用户密码

<property name=”schemas”>为server.xml同级目录下的schema.xml中定义的<schema name="cloudsystem" checkSQLschema="true" sqlMaxLimit="100">对应，表现为mycat数据库名。

1. <!--
2. **<property** name="serverPort"**>**8066**</property>** **<property** name="managerPort"**>**9066**</property>**
3. **<property** name="idleTimeout"**>**300000**</property>** **<property** name="bindIp"**>**0.0.0.0**</property>**
4. **<property** name="frontWriteQueueSize"**>**4096**</property>** **<property** name="processors"**>**32**</property>** --**>**
5. <!--分布式事务开关，0为不过滤分布式事务，1为过滤分布式事务（如果分布式事务内只涉及全局表，则不过滤），2为不过滤分布式事务,但是记录分布式事务日志-->
6. **<property** name="handleDistributedTransactions"**>**0**</property>**

修改端口或分布式事务

**需要复制，请用word文档打开，然后复制框内的内容**

<!--

<property name="serverPort">8066</property> <property name="managerPort">9066</property>

<property name="idleTimeout">300000</property> <property name="bindIp">0.0.0.0</property>

<property name="frontWriteQueueSize">4096</property> <property name="processors">32</property> -->

<!--分布式事务开关，0为不过滤分布式事务，1为过滤分布式事务（如果分布式事务内只涉及全局表，则不过滤），2为不过滤分布 式事务,但是记录分布式事务日志-->

<property name="handleDistributedTransactions">0</property>

* 1. **修改schema.xml配置文件**

vi mycat/conf/schema.xml

1. **<?xml** version="1.0"**?>**
2. <!DOCTYPE mycat:schema SYSTEM "schema.dtd"**>**
3. **<mycat:schema** xmlns:mycat="http://io.mycat/"**>**
5. **<schema** name="cloudsystem" checkSQLschema="true" sqlMaxLimit="100"**>**
6. **<table** name="art\_article" primaryKey="ID" dataNode="dn1" **/>**
7. **<table** name="art\_article\_sku" primaryKey="ID" dataNode="dn1" **/>**
8. **<table** name="art\_article\_sku\_attament" primaryKey="ID" dataNode="dn1" **/>**
9. **<table** name="art\_article\_sku\_reject\_log" primaryKey="ID" dataNode="dn1" **/>**
10. **<table** name="art\_category" primaryKey="ID" dataNode="dn1" **/>**
11. **<table** name="art\_damage" primaryKey="ID" dataNode="dn1" **/>**
12. **<table** name="art\_damage\_attament" primaryKey="ID" dataNode="dn1" **/>**
13. **<table** name="cms\_advertisement" primaryKey="ID" dataNode="dn1" **/>**
14. **<table** name="cms\_article" primaryKey="ID" dataNode="dn1" **/>**
15. **<table** name="cms\_article\_info" primaryKey="ID" dataNode="dn1" **/>**
16. **<table** name="cms\_article\_like" primaryKey="ID" dataNode="dn1" **/>**
17. **<table** name="cms\_attament" primaryKey="ID" dataNode="dn1" **/>**
18. **<table** name="cms\_category" primaryKey="ID" dataNode="dn1" **/>**
19. **<table** name="gue\_category" primaryKey="ID" dataNode="dn1" **/>**
20. **<table** name="gue\_member\_option" primaryKey="ID" dataNode="dn1" **/>**
21. **<table** name="gue\_option" primaryKey="ID" dataNode="dn1" **/>**
22. **<table** name="gue\_question" primaryKey="ID" dataNode="dn1" **/>**
23. **<table** name="mem\_account" primaryKey="ID" dataNode="dn1" **/>**
24. **<table** name="mem\_account\_log" primaryKey="ID" dataNode="dn1" **/>**
25. **<table** name="mem\_address" primaryKey="ID" dataNode="dn1" **/>**
26. **<table** name="mem\_attament" primaryKey="ID" dataNode="dn1" **/>**
27. **<table** name="mem\_check" primaryKey="ID" dataNode="dn1" **/>**
28. **<table** name="mem\_check\_config" primaryKey="ID" dataNode="dn1" **/>**
29. **<table** name="mem\_coupon" primaryKey="ID" dataNode="dn1" **/>**
30. **<table** name="mem\_coupon\_member" primaryKey="ID" dataNode="dn1" **/>**
31. **<table** name="mem\_dept" primaryKey="ID" dataNode="dn1" **/>**
32. **<table** name="mem\_exception\_job" primaryKey="ID" dataNode="dn1" **/>**
33. **<table** name="mem\_flow" primaryKey="ID" dataNode="dn1" **/>**
34. **<table** name="mem\_goods\_member" primaryKey="ID" dataNode="dn1" **/>**
35. **<table** name="mem\_job" primaryKey="ID" dataNode="dn1" **/>**
36. **<table** name="mem\_job\_record" primaryKey="ID" dataNode="dn1" **/>**
37. **<table** name="mem\_job\_step" primaryKey="ID" dataNode="dn1" **/>**
38. **<table** name="mem\_member" primaryKey="ID" dataNode="dn1" **/>**
39. **<table** name="mem\_member\_article" primaryKey="ID" dataNode="dn1" **/>**
40. **<table** name="mem\_member\_role" primaryKey="ID" dataNode="dn1" **/>**
41. **<table** name="mem\_point" primaryKey="ID" dataNode="dn1" **/>**
42. **<table** name="mem\_point\_log" primaryKey="ID" dataNode="dn1" **/>**
43. **<table** name="mem\_role" primaryKey="ID" dataNode="dn1" **/>**
44. **<table** name="pro\_attament" primaryKey="ID" dataNode="dn1" **/>**
45. **<table** name="pro\_category" primaryKey="ID" dataNode="dn1" **/>**
46. **<table** name="pro\_goods" primaryKey="ID" dataNode="dn1" **/>**
47. **<table** name="pro\_goods\_info" primaryKey="ID" dataNode="dn1" **/>**
48. **<table** name="pro\_goods\_property" primaryKey="ID" dataNode="dn1" **/>**
49. **<table** name="pro\_goods\_sku" primaryKey="ID" dataNode="dn1" **/>**
50. **<table** name="pro\_order" primaryKey="ID" dataNode="dn1" **/>**
51. **<table** name="pro\_order\_refund" primaryKey="ID" dataNode="dn1" **/>**
52. **<table** name="pro\_property" primaryKey="ID" dataNode="dn1" **/>**
53. **<table** name="pro\_shop" primaryKey="ID" dataNode="dn1" **/>**
54. **<table** name="system\_attament" primaryKey="ID" dataNode="dn1" **/>**
55. **<table** name="system\_dept" primaryKey="ID" dataNode="dn1" **/>**
56. **<table** name="system\_log" primaryKey="ID" dataNode="dn1" **/>**
57. **<table** name="system\_resource" primaryKey="ID" dataNode="dn1" **/>**
58. **<table** name="system\_role" primaryKey="ID" dataNode="dn1" **/>**
59. **<table** name="system\_role\_resource" primaryKey="ID" dataNode="dn1" **/>**
60. **<table** name="system\_user" primaryKey="ID" dataNode="dn1" **/>**
61. **<table** name="system\_user\_role" primaryKey="ID" dataNode="dn1" **/>**
62. **</schema>**
63. **<dataNode** name="dn1" dataHost="localhost1" database="cloudsystem" **/>**
64. **<dataHost** name="localhost1" maxCon="1000" minCon="10" balance="1"
65. writeType="0" dbType="mysql" dbDriver="native" switchType="3"  slaveThreshold="100"**>**
66. **<heartbeat>**show status like 'wsrep%'**</heartbeat>**
68. **<writeHost** host="hostM1" url="123.207.77.109:3307" user="root"
69. password="654321" **/>**
70. **<writeHost** host="hostM2" url="123.207.77.109:3308" user="root"
71. password="654321" **/>**
73. **</dataHost>**
74. **</mycat:schema>**

注：

**<schema name=”DB1”>**与server.xml对应

**<table name=”test”>**为mycat的表名

**<dataNode name=”pxc”>**为数据库分片

**<dataHost name=”datahost1”>**直接定义了具体的数据库实例、读写分离配置和心跳语句，balance="1" 代表开启读写分离，默认情况第一个参与写，剩下参与读，switchType="3" 基于mysql galary cluster的切换机制，心跳语句为show status like 'wsrep%'；

数据库分片：将存放在同一个数据库中的数据分散存放到多个数据库（主机）上面，以达到分散单台设备负载的效果。

* + 1. 按照不同的表（或者Schema）来切分到不同的数据库（主机）之上，这种切可以称之为数据的垂直（纵向）切分；
    2. 根据表中的数据的逻辑关系，将同一个表中的数据按照某种条件拆分到多台数据库（主机）上面，这种切分称之为数据的水平（横向）切分。

参考：<https://blog.csdn.net/qq_35152037/article/details/79681762>

**需要复制，请用word文档打开，然后复制框内的内容**

<?xml version="1.0"?>

<!DOCTYPE mycat:schema SYSTEM "schema.dtd">

<mycat:schema xmlns:mycat="http://io.mycat/">

<schema name="cloudsystem" checkSQLschema="true" sqlMaxLimit="100">

<table name="art\_article" primaryKey="ID" dataNode="dn1" />

<table name="art\_article\_sku" primaryKey="ID" dataNode="dn1" />

<table name="art\_article\_sku\_attament" primaryKey="ID" dataNode="dn1" />

<table name="art\_article\_sku\_reject\_log" primaryKey="ID" dataNode="dn1" />

<table name="art\_category" primaryKey="ID" dataNode="dn1" />

<table name="art\_damage" primaryKey="ID" dataNode="dn1" />

<table name="art\_damage\_attament" primaryKey="ID" dataNode="dn1" />

<table name="cms\_advertisement" primaryKey="ID" dataNode="dn1" />

<table name="cms\_article" primaryKey="ID" dataNode="dn1" />

<table name="cms\_article\_info" primaryKey="ID" dataNode="dn1" />

<table name="cms\_article\_like" primaryKey="ID" dataNode="dn1" />

<table name="cms\_attament" primaryKey="ID" dataNode="dn1" />

<table name="cms\_category" primaryKey="ID" dataNode="dn1" />

<table name="gue\_category" primaryKey="ID" dataNode="dn1" />

<table name="gue\_member\_option" primaryKey="ID" dataNode="dn1" />

<table name="gue\_option" primaryKey="ID" dataNode="dn1" />

<table name="gue\_question" primaryKey="ID" dataNode="dn1" />

<table name="mem\_account" primaryKey="ID" dataNode="dn1" />

<table name="mem\_account\_log" primaryKey="ID" dataNode="dn1" />

<table name="mem\_address" primaryKey="ID" dataNode="dn1" />

<table name="mem\_attament" primaryKey="ID" dataNode="dn1" />

<table name="mem\_check" primaryKey="ID" dataNode="dn1" />

<table name="mem\_check\_config" primaryKey="ID" dataNode="dn1" />

<table name="mem\_coupon" primaryKey="ID" dataNode="dn1" />

<table name="mem\_coupon\_member" primaryKey="ID" dataNode="dn1" />

<table name="mem\_dept" primaryKey="ID" dataNode="dn1" />

<table name="mem\_exception\_job" primaryKey="ID" dataNode="dn1" />

<table name="mem\_flow" primaryKey="ID" dataNode="dn1" />

<table name="mem\_goods\_member" primaryKey="ID" dataNode="dn1" />

<table name="mem\_job" primaryKey="ID" dataNode="dn1" />

<table name="mem\_job\_record" primaryKey="ID" dataNode="dn1" />

<table name="mem\_job\_step" primaryKey="ID" dataNode="dn1" />

<table name="mem\_member" primaryKey="ID" dataNode="dn1" />

<table name="mem\_member\_article" primaryKey="ID" dataNode="dn1" />

<table name="mem\_member\_role" primaryKey="ID" dataNode="dn1" />

<table name="mem\_point" primaryKey="ID" dataNode="dn1" />

<table name="mem\_point\_log" primaryKey="ID" dataNode="dn1" />

<table name="mem\_role" primaryKey="ID" dataNode="dn1" />

<table name="pro\_attament" primaryKey="ID" dataNode="dn1" />

<table name="pro\_category" primaryKey="ID" dataNode="dn1" />

<table name="pro\_goods" primaryKey="ID" dataNode="dn1" />

<table name="pro\_goods\_info" primaryKey="ID" dataNode="dn1" />

<table name="pro\_goods\_property" primaryKey="ID" dataNode="dn1" />

<table name="pro\_goods\_sku" primaryKey="ID" dataNode="dn1" />

<table name="pro\_order" primaryKey="ID" dataNode="dn1" />

<table name="pro\_order\_refund" primaryKey="ID" dataNode="dn1" />

<table name="pro\_property" primaryKey="ID" dataNode="dn1" />

<table name="pro\_shop" primaryKey="ID" dataNode="dn1" />

<table name="system\_attament" primaryKey="ID" dataNode="dn1" />

<table name="system\_dept" primaryKey="ID" dataNode="dn1" />

<table name="system\_log" primaryKey="ID" dataNode="dn1" />

<table name="system\_resource" primaryKey="ID" dataNode="dn1" />

<table name="system\_role" primaryKey="ID" dataNode="dn1" />

<table name="system\_role\_resource" primaryKey="ID" dataNode="dn1" />

<table name="system\_user" primaryKey="ID" dataNode="dn1" />

<table name="system\_user\_role" primaryKey="ID" dataNode="dn1" />

</schema>

<dataNode name="dn1" dataHost="localhost1" database="cloudsystem" />

<dataHost name="localhost1" maxCon="1000" minCon="10" balance="1"

writeType="0" dbType="mysql" dbDriver="native" switchType="3" slaveThreshold="100">

<heartbeat>show status like 'wsrep%'</heartbeat>

<writeHost host="hostM1" url="123.207.77.109:3307" user="root"

password="654321" />

<writeHost host="hostM2" url="123.207.77.109:3308" user="root"

password="654321" />

</dataHost>

</mycat:schema>

* 1. **启动mycat**

启动脚本在mycat/bin/下

cd mycat/bin/

./mycat start 启动

./mycat stop关闭

./mycat restart重启

* 1. **连接mycat测试**

使用8066端口、root用户、123456密码连接mycat，插入数据，查看3306、3307数据库是否有数据。

* 1. **启动另一个mycat**

复制mycat到另一台机器

scp -r -p 22 mycat root@192.168.9.214:/u06/root/pkg/

启动

/u06/root/pkg/mycat start

1. **Haproxy集群**

**4.1 创建haproxy.cfg配置文件**

1. global
2. log 127.0.0.1 local0 notice
3. # user haproxy
4. # group haproxy
6. defaults
7. log global
8. retries 2
9. timeout connect 3000
10. timeout server 5000
11. timeout client 5000
13. listen mycat\_service #192.168.9.201:7066
14. mode   tcp
15. bind   \*:7066
16. # option mysql-check user haproxy\_check
17. balance roundrobin
18. server mysql01 192.168.9.220:8066 check
19. server mysql02 192.168.9.214:8066 check
21. listen stats #monitor
22. mode   http
23. bind   \*:1081
24. stats  refresh 3s
25. stats  uri /

复制haproxy.cfg到另一台机器

**需要复制，请用word文档打开，然后复制框内的内容**

global

log 127.0.0.1 local0 notice

# user haproxy

# group haproxy

defaults

log global

retries 2

timeout connect 3000

timeout server 5000

timeout client 5000

listen mycat\_service #192.168.9.201:7066

mode tcp

bind \*:7066

# option mysql-check user haproxy\_check

balance roundrobin

server mysql01 192.168.9.220:8066 check

server mysql02 192.168.9.214:8066 check

listen stats #monitor

mode http

bind \*:1081

stats refresh 3s

stats uri /

**4.2 启动HAProxy镜像**

master机器启动容器:

docker run -d --name haproxy --net=host -v /u06/root/pkg/mytest/haproxy/haproxy.cfg:/usr/local/etc/haproxy/haproxy.cfg haproxy

slave机器启动容器:

docker run -d --name haproxy --net=host -v /u06/root/pkg/mytest/haproxy/haproxy.cfg:/usr/local/etc/haproxy/haproxy.cfg haproxy

**注：需要复制使用，请用word文档打开后复制**

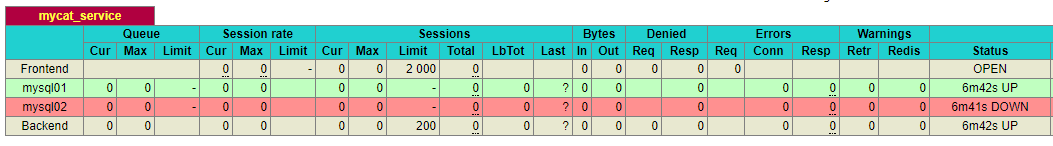
**这里使用的是HOST宿舍机网络，原因是使用其他网络（bridge、overlay）可能会导致HAProxy无法访问mycat**

## 4.3 访问监控页面

**http://ip1:1081/**

**http://ip2:1081/**

查看mycat状态是否为up



## 4.4测试

使用HAProxy端口3066连接数据库，插入数据，查看3307、3308数据库是否有数据。

1. **keepalived集群**

**5.1 安装**

官网下载：<http://www.keepalived.org/>

或执行命令wget http://www.keepalived.org/software/keepalived-2.0.10.tar.gz

**5.2 解压**

tar -xf keepalived-2.0.10.tar.gz

**5.3 安装**

cd keepalived-2.0.10

./configure -prefix=/usr/local/keepalived

make && make install

执行configure遇到的问题：

[① no acceptable C compiler found in $PATH](#configure1)

[②!!! OpenSSL is not properly installed on your system. !!!](#configure2)

执行make && make install遇到的问题：

[WARNING: 'automake-1.15' is missing on your system.](#make)

**5.4 复制或者连接文件到指定目录**

cp -p keepalived/etc/init.d/keepalived /etc/rc.d/init.d/

cp -p keepalived/etc/sysconfig/keepalived /etc/sysconfig

mkdir /etc/keepalived/

cp -p keepalived/etc/keepalived/keepalived.conf /etc/keepalived/

cp -p /usr/local/keepalived/sbin/keepalived /usr/sbin/

## 5.5 创建启动项

echo “/etc/init.d/keepalived start” >> /etc/rc.local

chmod 755 /etc/rc.d/init.d/keepalived

开机自启：

chkconfig keepalived on #开机自启

**5.6 修改keepalived.conf配置文件**

vi /etc/keepalived/keepalived.conf

注：vrrp\_script最好写在vrrp\_instance上方，否则可能会出现

(Line 22) (VI\_1) track script chk\_haproxy not found, ignoring...

1. global\_defs {
2. notification\_email {
3. root@localhost
4. }
5. notification\_email\_from keepalived@localhost
6. smtp\_server 127.0.0.1      #发送错误发送邮件的地址
7. smtp\_connect\_timeout 30
8. router\_id LVS\_master       #主从可以一样，也可以不一样
9. }
11. vrrp\_script chk\_haproxy {
12. script "/etc/keepalived/check\_haproxy.sh"   #HAProxy监控监本
13. interval 20
14. weight -5
15. }
17. vrrp\_instance VI\_1 {
18. state MASTER               #主节点上为MASTER。从节点上为BUCKUP
19. **interface** ens33            #网络接口
20. virtual\_router\_id 50       #此值主从必须一致
21. priority 100               #此值在MASTER上比BUCKUP大
22. advert\_int 1
23. mcast\_src\_ip 192.168.9.220
24. authentication {
25. auth\_type PASS
26. auth\_pass 1111
27. }
28. track\_script {
29. chk\_haproxy
30. }
31. virtual\_ipaddress {
32. 192.168.9.100 dev ens33     #Haproxy提供的虚拟ip,主从一样
33. }
34. notify\_master "/etc/keepalived/clean\_arp.sh 192.168.9.100"     #主从一样
35. }

**需要复制，请用word文档打开，然后复制框内的内容**

global\_defs {

notification\_email {

root@localhost

}

notification\_email\_from keepalived@localhost

smtp\_server 127.0.0.1 #发送错误发送邮件的地址

smtp\_connect\_timeout 30

router\_id LVS\_master #主从可以一样，也可以不一样

}

vrrp\_script chk\_haproxy {

script "/etc/keepalived/check\_haproxy.sh" #HAProxy监控监本

interval 20

weight -5

}

vrrp\_instance VI\_1 {

state MASTER #主节点上为MASTER。从节点上为BUCKUP

interface ens33 #网络接口

virtual\_router\_id 50 #此值主从必须一致

priority 100 #此值在MASTER上比BUCKUP大

advert\_int 1

mcast\_src\_ip 192.168.9.220

authentication {

auth\_type PASS

auth\_pass 1111

}

track\_script {

chk\_haproxy

}

virtual\_ipaddress {

192.168.9.100 dev ens33 #Haproxy提供的虚拟ip,主从一样

}

notify\_master "/etc/keepalived/clean\_arp.sh 192.168.9.100" #主从一样

}

**5.7 设置更新虚拟服务器（VIP）地址的arp记录到网关脚本**

vi /etc/keepalived/clean\_arp.sh

授权chmod 777 /etc/keepalived/clean\_arp.sh

1. #!/bin/bash
2. VIP=$1
3. GATEWAY=192.168.9.1                   #这个是本机的外网网卡网关地址
4. /sbin/arping -I ens33 -c 5 -s $VIP $GATEWAY &**>**/dev/null

**需要复制，请用word文档打开，然后复制框内的内容**

#!/bin/bash

VIP=$1

GATEWAY=192.168.9.1 #这个是本机的外网网卡网关地址

/sbin/arping -I ens0 -c 5 -s $VIP $GATEWAY &>/dev/null

## 5.8 设置HAproxy服务监控脚本

 vi /etc/keepalived/check\_haproxy.sh

授权chmod 777 /etc/keepalived/check\_haproxy.sh

1. #!/bin/bash
2. A=`ps -C haproxy --no-header | wc -l`
3. if [ $A -eq 0 ];then
4. docker rm -f haproxy &
5. docker run -d -p 7066:7066 -p 1081:1081 --name haproxy --net=pxc-net -v /u06/root/pkg/mytest/haproxy/haproxy.cfg:/usr/local/etc/haproxy/haproxy.cfg haproxy
6. sleep 3
7. if [ `ps -C haproxy --no-header | wc -l ` -eq 0 ];then
8. /etc/init.d/keepalived stop
9. fi
10. fi

**需要复制，请用word文档打开，然后复制框内的内容**

#!/bin/bash

A=`ps -C haproxy --no-header | wc -l`

if [ $A -eq 0 ];then

docker rm -f haproxy &

docker rm -f haproxy1 &

docker run -d -p 7066:7066 -p 1081:1081 --name haproxy --net=pxc-net -v /u06/root/pkg/mytest/haproxy/haproxy.cfg:/usr/local/etc/haproxy/haproxy.cfg haprox

y

docker run -d -p 7067:7066 -p 1082:1081 --name haproxy1 --net=pxc-net -v /u06/root/pkg/mytest/haproxy/haproxy1.cfg:/usr/local/etc/haproxy/haproxy.cfg hapr

oxy

sleep 3

if [ `ps -C haproxy --no-header | wc -l ` -eq 0 ];then

/etc/init.d/keepalived stop

fi

fi

## 5.9 内核优化

1. echo 1024 60999 > /proc/sys/net/ipv4/ip\_local\_port\_range
2. echo 30 > /proc/sys/net/ipv4/tcp\_fin\_timeout
3. echo 4096 > /proc/sys/net/ipv4/tcp\_max\_syn\_backlog
4. echo 262144 > /proc/sys/net/ipv4/tcp\_max\_tw\_buckets
5. echo 262144 > /proc/sys/net/ipv4/tcp\_max\_orphans
6. echo 300 > /proc/sys/net/ipv4/tcp\_keepalive\_time
7. echo 1 > /proc/sys/net/ipv4/tcp\_tw\_recycle
8. echo 0 > /proc/sys/net/ipv4/tcp\_timestamps
9. echo 0 > /proc/sys/net/ipv4/tcp\_ecn
10. echo 1 > /proc/sys/net/ipv4/tcp\_sack
11. echo 0 > /proc/sys/net/ipv4/tcp\_dsack

## 5.10 启动

**需要复制，请用word文档打开，然后复制框内的内容**

echo 1024 60999 > /proc/sys/net/ipv4/ip\_local\_port\_range

echo 30 > /proc/sys/net/ipv4/tcp\_fin\_timeout

echo 4096 > /proc/sys/net/ipv4/tcp\_max\_syn\_backlog

echo 262144 > /proc/sys/net/ipv4/tcp\_max\_tw\_buckets

echo 262144 > /proc/sys/net/ipv4/tcp\_max\_orphans

echo 300 > /proc/sys/net/ipv4/tcp\_keepalive\_time

echo 1 > /proc/sys/net/ipv4/tcp\_tw\_recycle

echo 0 > /proc/sys/net/ipv4/tcp\_timestamps

echo 0 > /proc/sys/net/ipv4/tcp\_ecn

echo 1 > /proc/sys/net/ipv4/tcp\_sack

echo 0 > /proc/sys/net/ipv4/tcp\_dsack

service keepalived start #启动

service keepalived stop #关闭

service keepalived restart #重启

## 5.11 安装配置并启动另一台机器的keepalived

需要修改keepalived.conf：

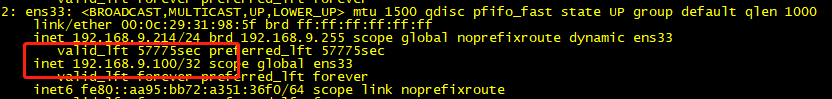
state MASTER 改为state  BACKUP

priority 100 改为 priority 99（数值小于100）

**5.12 测试**

若keepalived启动正常，master机器使用ip addr 命令可以看到虚

拟ip：



使用虚拟ip：192.168.9.100访问数据库，进行数据操作，观察数据是否正常；

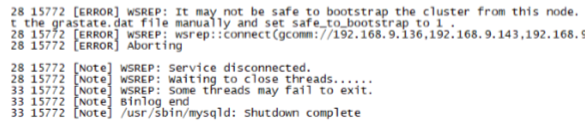
关闭master的keepalived，backup机器使用ip addr 命令可以看到虚拟ip；

继续使用虚拟ip访问数据库，进行数据操作，观察数据是否正常。

1. **问题和解决方案**

## 6.1 Percona XtraDB Cluster问题

容器删除或停止后，重新启动失败，日志显示

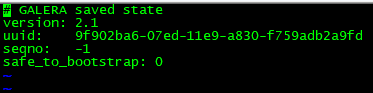


**解决方法：**

修改创建的数据目录下的grastate.dat 文件

vi /data/mysql1/grastate.dat

safe\_to\_bootstrap: 1



**6.2 keepalived 安装问题（**[**返回安装流程**](#conf1)**）**

**① 执行./configure -prefix=/usr/local/keepalived 时报错**

configure: error: in `/u06/root/pkg/mytest/keepalived/keepalived-2.0.10':

configure: error: no acceptable C compiler found in $PATH

See `config.log' for more details

**解决方法：**

**yum -y install gcc**

**②执行./configure -prefix=/usr/local/keepalived 时报错**

configure: error:

!!! OpenSSL is not properly installed on your system. !!!

!!! Can not include OpenSSL headers files. !!!

**解决方法：**

**yum -y install openssl-devel**

**③执行make && make install 时报错**

WARNING: 'automake-1.15' is missing on your system.

You should only need it if you modified 'Makefile.am' or

'configure.ac' or m4 files included by 'configure.ac'.

The 'automake' program is part of the GNU Automake package:

<http://www.gnu.org/software/automake>

It also requires GNU Autoconf, GNU m4 and Perl in order to run:

<http://www.gnu.org/software/autoconf>

<http://www.gnu.org/software/m4/>

<http://www.perl.org/>

make: \*\*\* [Makefile.in] 错误 1

**解决方法：**

下载安装包

wget https://ftp.gnu.org/gnu/automake/automake-1.15.tar.gz

解压

tar -xf automake-1.15.tar.gz

安装

cd automake-1.15

./configure

make && make install

**④安装automake时报错：**

configure.ac:20: error: Autoconf version 2.65 or higher is required

下载安装包

wget http://ftp.gnu.org/gnu/autoconf/autoconf-2.68.tar.gz

解压

tar xzf autoconf-2.68.tar.gz

安装

cd autoconf-2.68

./configure

make && make install

## 6.3 Linux内核升级方式

**（1）下载tar.gz文件**

<https://www.kernel.org/pub/linux/kernel/>

**（2）解压安装**

tar -xf linux-4.14.tar.gz

cd linux-4.14

make mrproper

make clean # 这两条用于清除之前编译生成的

# .ko和.config

make oldconfig # 使用当前的内核配置

make # 编译，半小时以上是正常的

make modules\_install # 把编译出的内核模块复制

# 到/lib/modules/${KERNEL\_VERSION}

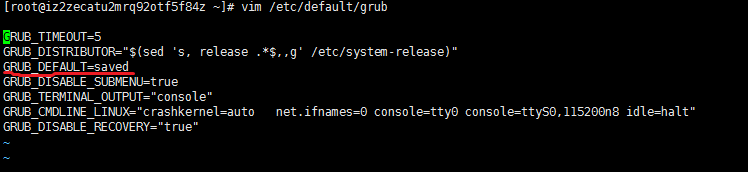
make install

**（3）查看内核版本默认启动顺序**

awk -F\' '$1=="menuentry " {print $2}' /etc/grub2.cfg

**（4）修改grub中默认版本启动顺序**

vim /etc/default/grub



如图，将GRUB\_DEFAULT=saved改为GRUB\_0=saved，保存退出vim

GRUB\_数字 表示启动的为步骤3中显示的第几条

**（5）运行grub2-mkconfig命令来重新创建内核配置：**

grub2-mkconfig -o /boot/grub2/grub.cfg

**（6）重启：reboot**