

[MySQL]

[集群搭建指南]

陈芝奔 骆飞

2019年02月

**深圳麦科田生物医疗技术有限公司**

目录

[1、简述 3](#_Toc1456693012)

[2、 集群的搭建 3](#_Toc1325876884)

[2.1 etcd集群 3](#_Toc1722468116)

[2.2虚拟入口 4](#_Toc1442064052)

[2.3 mysql集群 4](#_Toc270081922)

[3、 集群的异常处理 7](#_Toc1633916943)

[3.1未全部宕机 7](#_Toc1368666812)

[3.2已全部宕机 7](#_Toc1485766267)

[4、 数据热备份 7](#_Toc333802153)

[4.1简介 7](#_Toc985499507)

[4.2配置 8](#_Toc1896328485)

**1、简述**

我使用的mysql集群是pxc集群，全称是*Percona XtraDB Cluster*。pxc原理请查看[此处](https://www.cnblogs.com/zengkefu/p/5678279.html)，这在培训的ppt中也有详细描述。为了保证数据的强一致性，选择了pxc集群，有兴趣地请翻阅ppt（*\\192.168.81.248\share\培训资料\mysql集群\mysql集群.pptx*）。

本文档会主要介绍mysql的部署，和关于某些配置的修改。

1. **集群的搭建**

**2.1 etcd集群**

我使用的是docker部署mysql服务，并使用etcd来注册和发现mysql服务。首先，需要先搭建一个etcd的集群。

etcd使用2379端口作为服务端口，集群的通信端口为2380。我使用*192.168.81.11*和*192.168.81.21*两台虚拟机搭建etcd集群。集群的启动脚本分别如下：

31：

*#!/usr/bin/env bash*

NODE1\_IP\_ADDR=${1:-"http://192.168.81.11"}

NODE2\_IP\_ADDR=${2:-"http://192.168.81.21"}

docker stop etcd

docker rm etcd

docker volume rm etcd-data

docker run -d --name etcd \

-p 2379:2379 \

-p 2380:2380 \

--restart=always \

-v /root/sql/localtime:/etcd/localtime \

--volume=etcd-data:/etcd-data \

quay.io/coreos/etcd \

/usr/local/bin/etcd \

--data-dir=/etcd-data --name node1 \

--initial-advertise-peer-urls ${NODE1\_IP\_ADDR}:2380 \

--listen-peer-urls http://0.0.0.0:2380 \

--advertise-client-urls ${NODE1\_IP\_ADDR}:2379 \

--listen-client-urls http://0.0.0.0:2379 \

--initial-cluster-state new \

--initial-cluster-token docker-etcd \

--initial-cluster node1=${NODE1\_IP\_ADDR}:2380,node2=${NODE2\_IP\_ADDR}:2380

exit 0

32：

*#!/usr/bin/env bash*

NODE1\_IP\_ADDR=${1:-"http://192.168.81.11"}

NODE2\_IP\_ADDR=${2:-"http://192.168.81.21"}

docker stop etcd

docker rm etcd

docker volume rm etcd-data

docker run -d --name etcd \

-p 2379:2379 \

-p 2380:2380 \

--restart=always \

-v /root/sql/localtime:/etcd/localtime \

--volume=etcd-data:/etcd-data \

quay.io/coreos/etcd \

/usr/local/bin/etcd \

--data-dir=/etcd-data --name node1 \

--initial-advertise-peer-urls ${NODE2\_IP\_ADDR}:2380 \

--listen-peer-urls http://0.0.0.0:2380 \

--advertise-client-urls ${NODE2\_IP\_ADDR}:2379 \

--listen-client-urls http://0.0.0.0:2379 \

--initial-cluster-state new \

--initial-cluster-token docker-etcd \

--initial-cluster node1=${NODE1\_IP\_ADDR}:2380,node2=${NODE2\_IP\_ADDR}:2380

exit 0

注意：

1. *localtime*，使用的是现成的docker，存在时区问题，把你要的localtime替换docker缘由的localtime
2. *--name*的节点名和ip要与下方的集群描述一致。
3. 若集群方案是其他机器，只需要在脚本后面跟上相关集群机器的ip地址即可。
4. 所有脚本中，倒数第二行内容必须一致，即：*node1*在前，*node2*在后，以此类推。

**2.2虚拟入口**

在做好etcd集群之后，使用N+K搭建的nginx主从集群来负载均衡，并设置vip为*192.168.81.51*。

etcd的使用，可以查看[此处](http://www.itshouce.com.cn/other/etcd-command.html)。

**2.3 mysql集群**

这里我们会使用etcd集群来注册mysql服务，并使之构成集群。

这里是我使用的mysql的容器构建脚本（ip地址写etcd集群中的任何一个即可）：

*#!/usr/bin/env bash*

ENTRANCE\_IP\_ADDR=${1:-"192.168.81.51"}

CLUSTER\_NAME=cluster\_mect

HOST\_PORT=${ENTRANCE\_IP\_ADDR}:2379

NETWORK\_NAME=host

PASSWORD=MECT888!

*#docker network create -d bridge $NETWORK\_NAME*

*#docker stop $(docker ps -aq |grep percona-xtradb-cluster)*

*#docker rm $(docker ps -aq |grep percona-xtradb-cluster)*

*#docker volume rm mysql\_conf mysql\_backup mysql\_node slave\_node1\_conf slave\_node1\_data slave\_node2\_conf slave\_node2\_data*

echo "Starting new node..."

docker run -d --net host \

     -e MYSQL\_ROOT\_PASSWORD=$PASSWORD \

     -e DISCOVERY\_SERVICE=$HOST\_PORT \

     -e CLUSTER\_NAME=$CLUSTER\_NAME \

     -e XTRABACKUP\_PASSWORD=$PASSWORD \

     -v mysql\_node:/var/lib/mysql \

     -v mysql\_conf:/etc/mysql \

     -v mysql\_backup:/data \

     -v /root/sql/localtime:/etc/localtime \

     --restart=always \

     percona/percona-xtradb-cluster

*#--general-log=1 --general\_log\_file=/var/lib/mysql/general.log*

echo "Started $(docker ps -l -q)"

*# --wsrep\_cluster\_address="gcomm://$QCOMM"*

exit 0

这里容器使用etcd注册节点信息只有ip（四端口集群，端口已经固定好了）

因此一台虚拟机最好只使用一个mysql节点，因为服务端口号，和集群的通信端口号是固定好的。（redis集群的通信端口是默认和服务端口相差固定值10000，所以端口号并不影响）

所以这里使用的docker网络是host网络，保证docker使用的ip处于虚拟机的网段，更改hostname显示为本机ip即可正确注册。

1. *mysql\_node*:存储节点数据的地方。
2. *mysql\_conf*:存储配置文件的地方
3. *mysql\_backup*:数据备份的地方

在其他几台机上部署完毕后，访问相应目录，即可看到容器发送的心跳包，如下图所示。  
<http://192.168.81.51:2379/v2/keys/pxc-cluster/cluster_mect>



使用etcd的好处在于方便查看和管理，以及构成动态集群。如果不使用etcd，只能选中一节点为主节点，而docker的启动特性会让容器在每次启动时都连接此节点，这就代表主节点不能宕机。下面我们来看一下这种集群的启动方式：

*docker run -d -p 3306:3306 -e MYSQL\_ROOT\_PASSWORD=abc123456 -e CLUSTER\_NAME=PXC -e XTRABACKUP\_PASSWORD=abc123456 -v v1:/var/lib/mysql -v backup:/data --privileged --name=node1 --net=net1 --ip 172.18.0.2 pxc*

其中net1是创建的*172.18.0.0/24*的docker网络，然后所有的节点链接node1。

这种方法不但对于节点的启动具有局限性（node1需要优先其他节点启动，同时若node1宕机，想要重新连上集群只能删除数据，参照[此处](http://www.imooc.com/qadetail/263927)），同时也对容器处于同一个网段具有硬性要求，难以用端口映射来解决这个问题，若想在多台机子上部署pxc集群，必须使用docker的*swarm*使创建的docker网络享有共同网络。

etcd也有缺点，mysql的运行会依赖etcd的运行，而etcd需要集群，也就需要nginx来负载，那么mysql的运行就需要nginx和etcd具有高可用性，但相比之下，这样的要求比起另一种搭建pxc集群的方式的缺陷来说，是完全可以接受的。

1. **集群的异常处理**

**3.1未全部宕机**

在没有修改过用户名密码且集群节点未全部宕机的情况下，使用docker对容器的设置*--restart=always*可以让节点正常恢复。

但是如果在链接数据库后修改过用户名密码，由于docker在启动时设置了*root*和*root\_password*所以其会启动失败（这种无法启动的情况一般会出现在节点全部宕机的情况下）。

这种情况的处理方式有两种避免：

①.root作为管理员，密码不变，重新创建新的用户来使用数据库

②.（建议）后续使用中间件，可以防止修改数据结构，也可以扩展分库分表操作，登录中间件的用户名密码和mysql无关，不会影响mysql管理

简单来说，就是集群创建完成后，不要再修改root的密码，影响后续的集群管理操作。

如果一不小心修改了密码，而集群又宕机了以后，就需要删除容器，开始重新搭建集群了。或者在配置中加入跳过密码，进入恢复管理员root的密码。

**3.2已全部宕机**

集群节点全部宕机后(停止所有容器)，启动会失败，因为新的容器会默认到某个节点里去同步数据，而这样的节点已经不存在了。

这时候选中一个节点作为启动节点，修改*mysql\_node*数据卷下的文件名为*grastate.dat*  
vim 进这个文件，修改*safe\_to\_bootstrap: 1*，然后到etcd中删除原有的集群（这里是*cluster\_mect*）就可以启动成功了。

这时其它自动重启的节点会自动找到这个节点去同步数据，那么集群就重启成功了。

步骤：

1. docker stop 所有容器（mysql的，找镜像名为*percona/percona-xtradb-cluster*的对应容器，停止即可）
2. 执行*curl -s 'http://${etcd\_ip}:${etcd\_port}(默认2379)/v2/keys/pxc-cluster/${cluster\_name}?recursive=true' -XDELETE* 执行过后，去浏览器刷新*http://${etcd\_ip}:${etcd\_port}/v2/keys/pxc-cluster/${cluster\_name}*，当出现*key not found*的时候说明集群删除成功（未全部宕机可以忽略此步骤）
3. 修改*mysql\_node*数据卷下(*/var/lib/docker/volumes/mysql\_node/\_data*)的文件名为*grastate.dat*  vim 进这个文件，修改*safe\_to\_bootstrap: 1*（让其成为主节点，当其他节点上线后，自动同步它的数据），然后到etcd中删除原有的集群（这里是*cluster\_mect*）就可以启动成功了。
4. 重新启动容器
5. **数据热备份**

**4.1简介**

MySQL的数据备份方式有很多种，如自带的MySQLDUMP命令，或者直接拷贝数据文件等等方式，但是这些方式都是系统的冷备份，即需要停机服务进行备份，这种方式我没有采用。

本次备份方案是热备份，MySQL的热备份的方式有两种：1、LVM 2、xtrabackup

*lvm*是Linux自带的一种，通过对某一个分区创建快照进行备份，所以可以备份多种数据库。但是缺点是会对数据库加锁，使得数据库在备份区间只读不可写，生产环境这是不允许的。

*xtrabackup*是percona公司的开源免费的MySQL备份工具，在热备份上不需要锁表进行备份，所以一般MySQL上备份采用*xtrabackup*，它还有个优势是备份数据是被压缩过的，会减小磁盘压力。

备份方式两种：全量、增量，一般一周一次全量备份，一天一次增量备份。

由于PXC安装在docker上的，备份方式和单机备份有一些差异：

差异详情参看[此处](https://blog.csdn.net/qq_21108311/article/details/82995615)。

**4.2配置**

如果要对原始配置文件进行修改，如加入*federated* （引擎）或*skip-name-resolve*（跳过dns解析，加快mysql效率）则推荐使用脚本方式进行修改，如（当节点信息都刷新出来的时候再执行这个命令，执行完毕重启容器即可）

*sed -i "/3306/s/$/ \nfederated \nskip-name-resolve/" /var/lib/docker/volumes/mysql\_conf/\_data/my.cnf*

脚本在sql里写有案例*config.sh*。

然后用客户端连接mysql，之后新建查询，使用“*show engines*”查询命令查询引擎，当*FEDERATED*显示*YES*则配置文件修改成功，如下图所示：

