# Msql主从环境搭建

在2017/11/05 19:00上被[亮 刘](http://er.msqltec.com:8080/xwiki/bin/view/XWiki/liuliang)修改

准备搭建一套完整体系的开发框架，使用docker将环境全部搭建好，使用jenkins进行一建部署分布式所有环境及项目

使用到的技术：Radis集群、MYSQL、服务集群、dubbo+ zookeeper(微服务）、分布式

今天首先进行MYSQL读写分离数据库环境，下面来进行实操

网络上有人已经做到MYSQL的docker镜像直接下载就好，docker安装不在介绍请去百度。

mysql镜像下载地址：https://c.163.com/hub#/m/repository/?repoId=2955

使用命令下载：docker pull hub.c.163.com/library/tomcat:latest

下载成功之后，有很多版本，有5.5、5.6、5.7 我使用的是5.7，直接启动即可

docker run -v /opt/mysql/data:/var/lib/mysql -p 3307:3306 --name zhumysql -e MYSQL\_ROOT\_PASSWORD=12345678 -d mysql:5.7

注：多容器不能公用一个mysql数据地址

命令进行了端口映射，由3307映射到docker的3306，MYSQL\_ROOT\_PASSWORD是数据库链接密码，

会安装一段时间。成功之后使用mysql客户端进行链接即可。在运行上面命令启动从数据库，将端口映射到3308.

两个数据库弄好之后，我们为进行主从配置。

使用命令进入主库容器：docker exec -it 5706b3e20817 /bin/sh

安装vim:apt-get install vim

修改my.cnf配置文件

## master配置

在主服务器的my.cnf配置文件中，增加

[mysqld]

log-bin=mysql-bin

server-id=1

说明:   
1. log-bin:开启二进制日志，该日志是在事务提交时写日志文件的。默认大小是1G，后面加001,002这样的后缀顺加。   
2. server-id，唯一标识主机，mysql主从每个mysql实例配置都不一样就行。这个值默认是0，如果是0，主服务器拒绝任何从服务器的连接。 

## slave配置

在从服务器的my.cnf配置文件中，增加

[mysqld]log-bin=mysql-bin

server-id=2

说明:   
server-id唯一就行。如果默认为0，则拒绝连接主服务器。

## master创建复制用户

在master上，root登录mysql，执行

mysql -u root -p

GRANT REPLICATION SLAVE ON \*.\* to '用户名'@'%' identified by '密码';

"\*.\*"表示对所有库的所有操作，“%”表示所有客户端都可能连，也可用具体客户端IP代替，如192.168.145.226，加强安全

说明：   
简单的创建用户，授予REPLICATION SLAVE权限。访问限制，密码，用户名等，根据实际情况自行设定，后面注意保持一致。

 查看master状态  
登陆主服务器mysql命令行  
mysql>show master status;  
+-------------------------+----------+  
| File | Position |  
+-------------------------+----------+  
| mysql-bin.000002 | 1308 |  
+-------------------------+----------+  
mysql-bin.000004 是用于主从复制的文件名  
1308 是日志文件内的最新位置  
5. slave指向master  
登陆从服务器mysql命令行，使用之前创建的用户和master的日志文件及其位置  
mysql>change master to master\_host='master ip',master\_user='用户名',master\_port=端口号,master\_password='密码',  
master\_log\_file='mysql-bin.000002',master\_log\_pos=1308; //注意不要断开，“1308”无单引号。  
6. 启动slave  
mysql>start slave;  
7. 查看slave状态  
mysql> show slave status\G;  
结果中有两个重要数据项：  
常见的问题是SQL线程没有正常工作 Slave\_SQL\_Running: No  
通常是两边的数据库不是完全对应的，需要确保master上的库及到目前为止的最新记录都复制到slave上了  
8. 测试  
当IO线程和SQL线程都正常后，到master中随意测试下插入、修改、删除操作，同时到slave中检查

1) Slave\_IO\_Running: Yes

IO线程状态，必须YES

2) Slave\_SQL\_Running: Yes

SQL线程状态，必须YES

----------------------------------------------------------------

如果改到主从的IP 进行以下操作

1.进入从库docker容器

docker exec -it f8535b41adee /bin/bash

2.进入mysql执行命令

change master to master\_host='IP',master\_port=3307,master\_user='zcroot',master\_password='12345678',master\_log\_file='mysql-bin.000006',master\_log\_pos=1174;

注：从镜像重启容器后，数据库的master\_log\_file,master\_log\_pos两项会变，需要到容器里mysql中进行修改才能有效同步。

如果slave在运行，先停止在执行上面语句

stop slave;

start slave;

-----------------------------------------------

show slave status\G;

主从配置即可完成。

--------------------------------------------------------------

## Mysql 5.7 主从复制的多线程复制配置方式

数据库复制的主要性能问题就是数据延时

为了优化复制性能，Mysql 5.6 引入了 “多线程复制” 这个新功能

但 5.6 中的每个线程只能处理一个数据库，所以如果只有一个数据库，或者绝大多数写操作都是集中在某一个数据库的，那么这个“多线程复制”就不能充分发挥作用了

Mysql 5.7 对 “多线程复制” 进行了改善，可以按照逻辑时钟的方式来分配线程，大大提高了复制性能

下面看一下在5.7中如何配置 “多线程复制”

1、对两个 mysql 实例配置好主从复制

配置成功后，在从库上使用 show processlist 查看现在的状态

2、在从库上停止复制

mysql> stop slave;

3、设置并发同步类型为逻辑时钟方式

先看下现在 slave 的并发类型，通过变量 slave\_parallel\_type 的值来获得，这个变量用来决定如何使用多线程复制

mysql> show variables like 'slave\_parallel\_type';

默认是datebase，每个线程只能处理一个数据库

配置成基于逻辑时钟的方式

mysql> set global slave\_parallel\_type='logical\_clock';

4、设置复制线程的数量

先看下当前的并发数量，通过变量 slave\_parallel\_workers 的值来获得，这个变量用来决定并发处理的线程数

mysql> show variables like 'slave\_parallel\_workers';

现在是 0，我们把他改成 4

mysql> set global slave\_parallel\_workers=4;

5、启动复制

mysql> start slave;

6、验证配置结果

mysql> show processlist;

可以看到已经有4个IO线程了，配置完成

--------------------------------------------------------------------------------------

未完，之后会写mysql负载