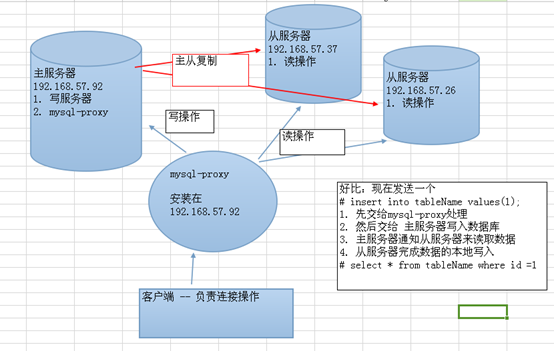
# Docker部署MySQL

## 安装mysql



下载mysql镜像

|  |
| --- |
| #docker pull mysql:5.7 |

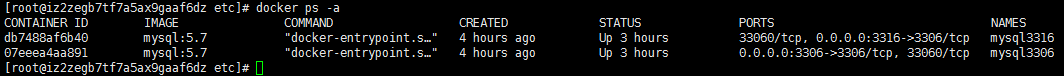
启动两个mysql镜像，分别映射3306和3316端口  
主库：3306端口

|  |
| --- |
| #docker run -p 3306:3306 --name mysql3306 -v /opt/mysql/data/data3306:/var/lib/mysql  -v /opt/mysql/logs/logs3306:/logs -e MYSQL\_ROOT\_PASSWORD=123456 -d mysql:5.7 |

备库：3316端口

|  |
| --- |
| #docker run -p 3316:3306 --name mysql3316 -v /opt/mysql/data/data3316:/var/lib/mysql  -v /opt/mysql/logs/logs3316:/logs -e MYSQL\_ROOT\_PASSWORD=123456 -d mysql:5.7 |

检查是否启动



## 二.主从复制

### 方法一

#### 1.主库配置

把主库的docker容器中把默认的配置文件拷贝出来

目录为（/opt/mysql/conf/conf3306）

|  |
| --- |
| #mkdir -p /opt/mysql/conf/conf3306 |

|  |
| --- |
| #docker cp 37735c072370:/etc/mysql/mysql.conf.d/mysqld.cnf /opt/mysql/conf/conf3306  备注：37735c072370为主库 |

修改配置文件增加两行

在目录为（/opt/mysql/conf/conf3306）修改mysqld.cnf

|  |
| --- |
| #vim /opt/mysql/conf/conf3306/mysqld.cnf |

添加内容为

|  |
| --- |
| [mysqld]  #开启二进制日志 log-bin=mysql-bin  #设置server-id server-id=1 |

将修改后的配置文件拷回容器里，并重启容器

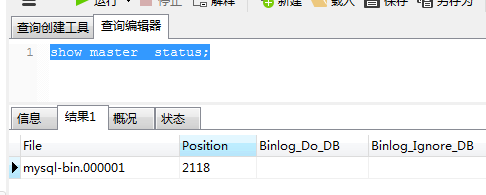
|  |
| --- |
| # docker cp /opt/mysql/conf/conf3306/mysqld.cnf 37735c072370:/etc/mysql/mysql.conf.d/ # docker restart 37735c072370 |

创建主从复制需要的mysql账户,并分配权限

|  |
| --- |
| mysql>CREATE USER 'repl'@'192.168.12.10' IDENTIFIED BY 'repl123'; mysql>GRANT REPLICATION SLAVE ON \*.\* TO ['repl'@'192.168.12.10';](mailto:'repl'@'xxx.xxx.xxx.xxx';)  mysql>FLUSH PRIVILEGES; |

查看主库状态，记录下File和Position

mysql>SHOW MASTER STATUS;



#### 2.从库配置

#vim mysqld.cnf   
[mysqld]  
server-id=2 #设置server-id

同样拷进从库镜像，并重启容器

**重复以上主库操作**

把从库的docker容器中把默认的配置文件拷贝出来

目录为（/opt/mysql/conf/conf3316）

|  |
| --- |
| #mkdir -p /opt/mysql/conf/conf3316 |

|  |
| --- |
| docker cp 37735c072370:/etc/mysql/mysql.conf.d/mysqld.cnf /opt/mysql/conf/conf3316 |

修改配置文件增加两行

在目录为（/opt/mysql/conf/conf3316）修改mysqld.cnf

|  |
| --- |
| #vim /opt/mysql/conf/conf3316/mysqld.cnf |

添加内容为

|  |
| --- |
| [mysqld]  #开启二进制日志 log-bin=mysql-bin  #设置server-id server-id=2 |

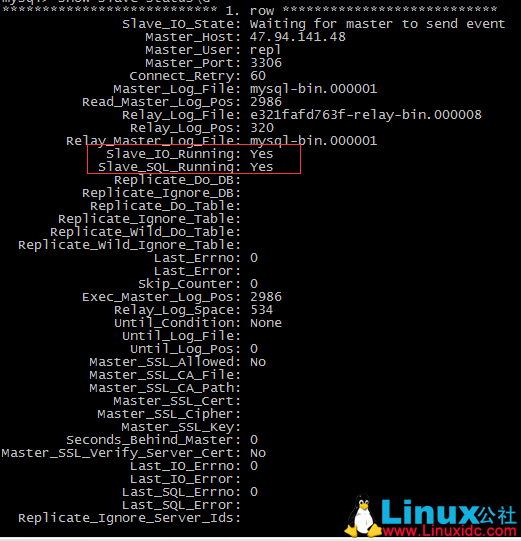
将修改后的配置文件拷回容器里，并重启容器

|  |
| --- |
| # docker cp /opt/mysql/conf/conf3316/mysqld.cnf 37735c072370:/etc/mysql/mysql.conf.d/ # docker restart 37735c072370 |

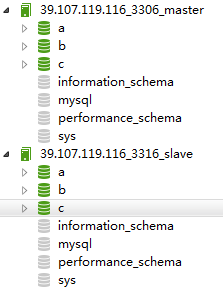
mysql中执行配置主库的语句

|  |
| --- |
| mysql>change master to master\_host='xxx.xxx.xxx.xxx', #Master 服务器Ip master\_port=3306, master\_user='repl', master\_password='repl123',  master\_log\_file='master-bin.000001',#Master服务器产生的日志 master\_log\_pos=2986;#Master服务器产生的position  mysql>start slave;  mysql>show slave status \G; |

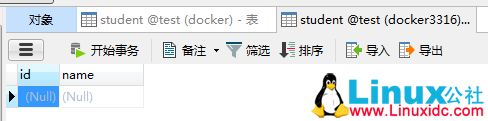
如果slave\_io\_running和slave\_sql\_running都为yes，表示启动同步成功

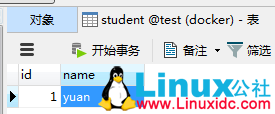
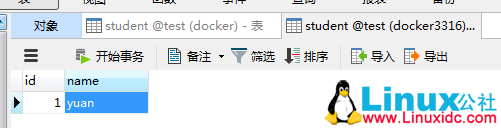


建立两个客户端连接并创建测试库



主库建立student表并插入一条记录，可以看到，从库自动同步了数据，配置成功

### 方法二

启动镜像实例

Master(主)：

|  |
| --- |
| #docker run -p 3306:3306 --name mymysql-master -e MYSQL\_ROOT\_PASSWORD=123456 -d mysql:5.7 |

Slave(从)：

|  |
| --- |
| #docker run -p 3307:3306 --name mymysql-slave -e MYSQL\_ROOT\_PASSWORD=123456 -d mysql:5.7 |

查看是否启动

|  |
| --- |
| #docker ps -a |

#### 配置Master(主)

|  |
| --- |
| #docker exec -it mymysql-master /bin/bash  进入  #cd /etc/mysql  #ls  #vi my.cnf  报错：bash: vi: command not found  安装vim  #apt-get update  #apt-get install vim |

最后面加上

|  |
| --- |
| [mysqld]  ## 同一局域网内注意要唯一  server-id=100  ## 开启二进制日志功能，可以随便取（关键）  log-bin=mysql-bin |

在mysql进程容器里面执行

|  |
| --- |
| service mysql restart (执行完就退出容器了)  docker start mymysql-master |

在mysql命令中执行

|  |
| --- |
| Mysql>CREATE USER 'slave'@'%' IDENTIFIED BY '123456';  Mysql>GRANT REPLICATION SLAVE, REPLICATION CLIENT ON \*.\* TO 'slave'@'%'; |

#### 配置Slave(从)

（多个从节点的写法，注意是 改下 server-id=101 以此递增，其他不改动）

和配置Master(主)一样，在Slave配置文件my.cnf中添加如下配置：

配置slave(从)

|  |
| --- |
| #docker exec -it mymysql-slave /bin/bash  进入  #cd /etc/mysql  #ls  #vi my.cnf  报错：bash: vi: command not found  安装vim  #apt-get update  #apt-get install vim |

最后面加上

|  |
| --- |
| [mysqld]  ## 设置server\_id,注意要唯一  server-id=101  ## 开启二进制日志功能，以备Slave作为其它Slave的Master时使用  log-bin=mysql-slave-bin |

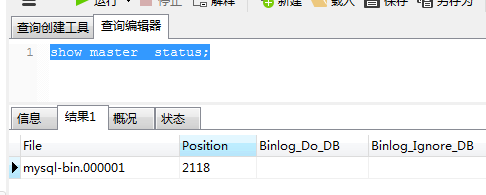
在mysql进程容器里面执行

|  |
| --- |
| service mysql restart (执行完就退出容器了)  docker start mymysql-slave |

#### 最后：

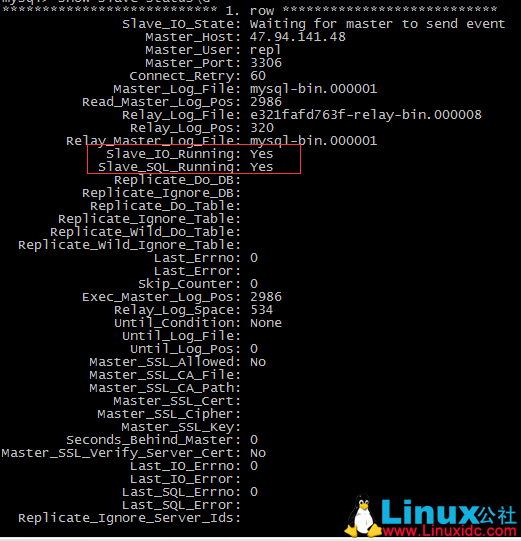
主库中执行命令：

|  |
| --- |
| Mysql>show master status; |



从库中执行命令：

|  |
| --- |
| Mysql>change master to master\_host='172.17.0.2', master\_user='slave', master\_password='123456', master\_port=3306, master\_log\_file='mysql-bin.000001', master\_log\_pos= 2830, master\_connect\_retry=30;  Mysql>start slave  Mysql>show slave status \G; |



如果slave\_io\_running和slave\_sql\_running都为yes，表示启动同步成功

参考资料为:

<https://www.cnblogs.com/songwenjie/p/9371422.html>

## 三.实现读写分离（mysql-proxy）

原理:让主数据库（master）处理[事务](https://baike.baidu.com/item/%E4%BA%8B%E5%8A%A1" \t "https://baike.baidu.com/item/%E8%AF%BB%E5%86%99%E5%88%86%E7%A6%BB/_blank)性增、改、删操作（INSERT、UPDATE、DELETE），而从数据库（slave）处理SELECT查询操作

检查lua是否安装

|  |
| --- |
| # rpm -q lua |

下载安装包



解压

|  |
| --- |
| #tar xf mysql-proxy-0.8.5-linux-glibc2.3-x86-64bit.tar.gz -C /usr/local/ |

进入：

|  |
| --- |
| #cd /usr/local/ |

关联成 mysql-proxy

|  |
| --- |
| #ln -sv mysql-proxy-0.8.5-linux-glibc2.3-x86-64bit/ mysql-proxy |

建立启动文件

|  |
| --- |
| #vim /etc/profile.d/mysql-proxy.sh |

文件内容为：

export PATH=$PATH:/usr/local/mysql-proxy/bin

进入挂载点

|  |
| --- |
| # cd /usr/local/mysql-proxy  # . /etc/profile.d/mysql-proxy.sh |

执行 读写分离 启动命令

|  |
| --- |
| #mysql-proxy --daemon --log-level=debug --log-file=/var/log/mysql-proxy.log --plugins="proxy" --proxy-backend-addresses="192.168.12.11:3339" --proxy-read-only-backend-addresses="192.168.12.11:3340" --proxy-lua-script="/usr/local/mysql-proxy/share/doc/mysql-proxy/rw-splitting.lua" |

备注：（一主多从的写法）

mysql-proxy --daemon --log-level=debug --log-file=/var/log/mysql-proxy.log --plugins="proxy"

写服务器

--proxy-backend-addresses="192.168.12.11:3339"

读服务器1

--proxy-read-only-backend-addresses="192.168.12.11:3340"

读服务器2

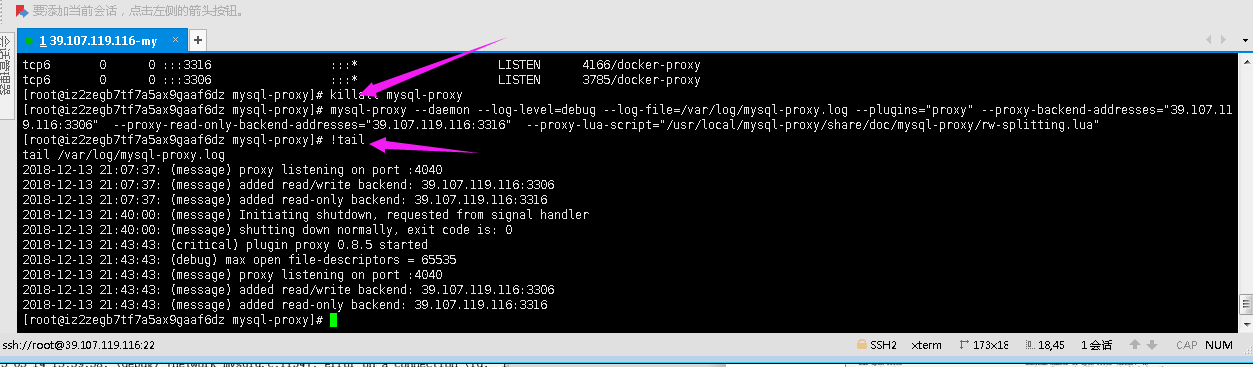
--proxy-read-only-backend-addresses="47.112.27.86:3308"

读写分离脚本

--proxy-lua-script="/usr/local/mysql-proxy/share/doc/mysql-proxy/rw-splitting.lua"

注意查看日志,端口为4040

|  |
| --- |
| #tail /var/log/mysql-proxy.log |



杀死进程命令

|  |
| --- |
| #killall mysql-proxy |

建立代理用户

|  |
| --- |
| mysql> grant all on \*.\* to root@'%' identified by '666666';  mysql>flush privileges; |

验证读写分离

只修改从库的数据，然后通过mysql-proxy查询，可以看到读取的数据是从库修改过的数据，读分离验证成功。

停止从库的docker容器，通过mysql-proxy插入一条数据，主库中刷新也有新增的数据，写分离验证成功。

## 实现读写分离（mycat）

原理:让主数据库（master）处理[事务](https://baike.baidu.com/item/%E4%BA%8B%E5%8A%A1" \t "https://baike.baidu.com/item/%E8%AF%BB%E5%86%99%E5%88%86%E7%A6%BB/_blank)性增、改、删操作（INSERT、UPDATE、DELETE），而从数据库（slave）处理SELECT查询操作

下载mycat安装包

如：

Mycat-server-1.6.5-release-20180122220033-linux.tar.gz

解压

|  |
| --- |
| #tar -zxvf Mycat-server-1.6.5-release-20180122220033-linux.tar.gz -C /home/ |

配置环境变量

|  |
| --- |
| #vi /etc/profile |

|  |
| --- |
| #mycat配置  export MYCAT\_HOME=/home/mycat  export PATH=$PATH:$MYCAT\_HOME/bin |

刷新环境变量

|  |
| --- |
| #source /etc/profile |

修改2个重要的文件

schema.xml

server.xml

编辑

|  |
| --- |
| #vi /home/mycat/conf/schema.xml |

内容为：

|  |
| --- |
| <?xml version="1.0"?>  <!DOCTYPE mycat:schema SYSTEM "schema.dtd">  <mycat:schema xmlns:mycat="http://io.mycat/">  <schema name="rc\_schema2" checkSQLschema="false" sqlMaxLimit="100" dataNode="rc\_dn2"></schema>  <schema name="pay\_schema2" checkSQLschema="false" sqlMaxLimit="100" dataNode="pay\_dn2"></schema>    <dataNode name="rc\_dn2" dataHost="dtHost2" database="roncoo" />  <dataNode name="pay\_dn2" dataHost="dtHost2" database="edu\_simple\_pay" />    <dataHost name="dtHost2" maxCon="1000" minCon="10" balance="1"  writeType="0" dbType="mysql" dbDriver="native" switchType="2" slaveThreshold="100">  <heartbeat>show slave status</heartbeat>  <writeHost host="hostM1" url="39.107.119.116:3306" user="root" password="123456"/>  <writeHost host="hostS2" url="39.107.119.116:3307" user="root" password="123456" />  </dataHost>  </mycat:schema> |

编辑

|  |
| --- |
| #vi /home/mycat/conf/server.xml |

内容为：

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE mycat:server SYSTEM "server.dtd">  <mycat:server xmlns:mycat="http://io.mycat/">  <system>  <property name="defaultSqlParser">druidparser</property>  <property name="charset">utf8mb4</property>  </system>  <user name="user2">  <property name="password">user2</property>  <property name="schemas">rc\_schema2,pay\_schema2</property>  </user>  <user name="user3">  <property name="password">user3</property>  <property name="schemas">rc\_schema2,pay\_schema2</property>  <property name="readOnly">true</property>  </user>  </mycat:server> |

相关命令

|  |
| --- |
| 启动mycat  mycat start  停止  mycat stop  查看状态  mycat status |

访问数据库默认端口为：8066

还有一个端口为：9066