

Docker 应用部署-MySQL

一、安装MySQL

1 拉取mysql镜像

```
1 docker pull mysql:8.0.20
```

2 创建容器

通过下面的命令，创建容器并设置端口映射、目录映射

```
1 # 在用户名目录下创建mysql目录用于存储mysql数据信息
2 mkdir /home/mysql
3 cd /home/mysql
4 # 创建docker容器
5 docker run -id \
6 -p 3306:3306 \
7 --name mysql8 \
8 --restart always \
9 -v $PWD/conf:/etc/mysql/conf.d \
10 -v $PWD/log:/var/log/mysql \
11 -v $PWD/data:/var/lib/mysql \
12 -e MYSQL_ROOT_PASSWORD=123456 \
13 mysql:8.0.20 \
14 --lower_case_table_names=1
```

参数说明：

- **-p 3306:3306**：将容器的3306端口映射到宿主机的3306端口。
- **-v \$PWD/conf:/etc/mysql/conf.d**：将主机当前目录下的conf/目录挂载到容器的 /etc/mysql/conf.d目录。
- **-v \$PWD/log:/var/log/mysql**：将主机当前目录下的log目录挂载到容器的 /var/log/mysql日志目录
- **-v \$PWD/data:/var/lib/mysql**：将主机当前目录下的data目录挂载到容器的/var/lib/mysql数据目录
- **-e MYSQL_ROOT_PASSWORD=123456**：初始化root用户的密码。
- **--lower_case_table_names**：用于控制表名和数据库名在比较时是否应该忽略大小写。它的值可以是0、1或2，分别表示：
 - 0：表名存储为给定的大小和比较是区分大小写的。
 - 1：表名存储在磁盘是小写的，但是比较的时候是不区分大小写。
 - 2：表名存储为给定的大小写但是比较的时候是小写的。

注意：在云服务器上部署的时候需要把密码复杂度设置高一点，避免被无差别扫描攻击，然后在做端口映射的时候宿主机端口就不要使用3306了

```
1 # 查看当前启动的容器
2 docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
2da4390a9919	mysql:8.0.20	"docker-entrypoint.s..."	6 seconds ago	Up 6 seconds	0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp	mysql8

看到STATUS为up状态表示创建容完成。

3 配置防火墙

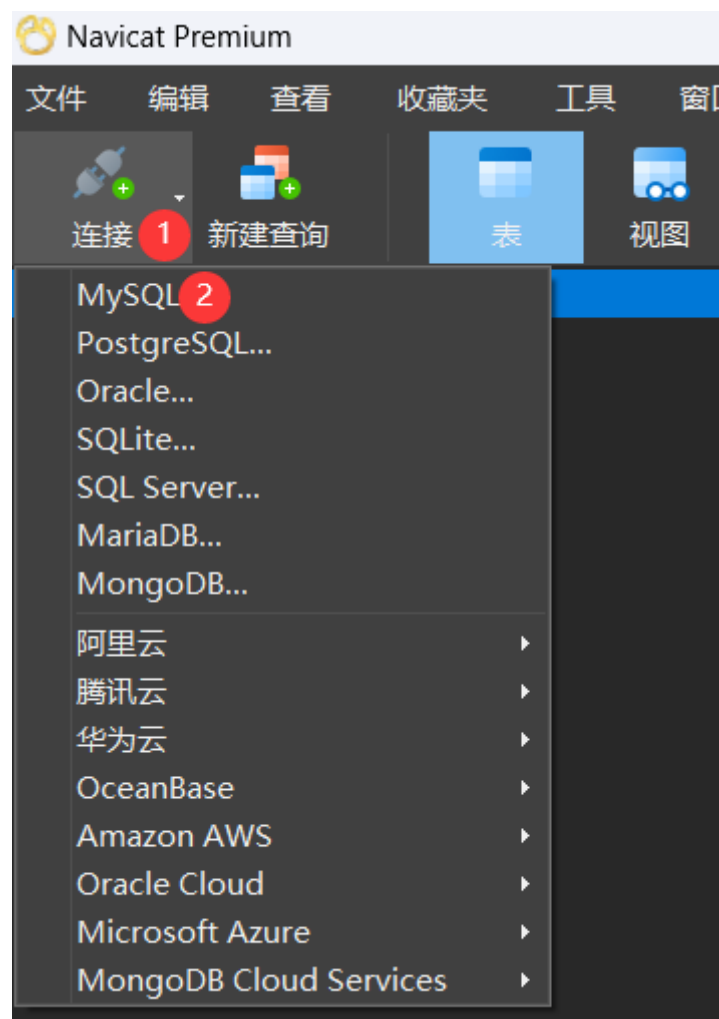
想要外部访问到你的数据库需要开放刚才映射的宿主机端口，我这里映射的是3306端口，所以开放3306端口。

注意：如果是云服务器，不需要通过下面命令开放端口，需要登录云控制台，配置防火墙入方向规则

```
1 #查看是否已经开放3306端口
2 firewall-cmd --list-port
3 #没有开放使用下面命令开放
4 firewall-cmd --add-port 3306/tcp --permanent
5 #重新加载防火墙
6 firewall-cmd --reload
```

4 远程连接测试

打开Navicat，然后新建一个MySQL连接。



填写连接参数信息，测试并保存连接参数信息。



5 mysql参数调整

```
1 #进入目录修改配置
2 cd /home/mysql/conf
3 #创建配置文件
4 vi my.cnf
```

配置文件中写入以下内容

注意：写入配置的时候将注释内容也去掉，避免中文影响，还每条配置占用一行，不要一条配置写成多行。

```
1 [mysqld]
2 # 修改sql_mode内容，目的在于去掉ONLY_FULL_GROUP_BY
3 sql_mode=STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION
4 # 调整mysql最大连接数
5 max_connections=1000
```

修改完配置后，重启容器，连接mysql，然后执行show variables like 'sql_mode';命令来查看对应变量值是否正确，示例如下图所示



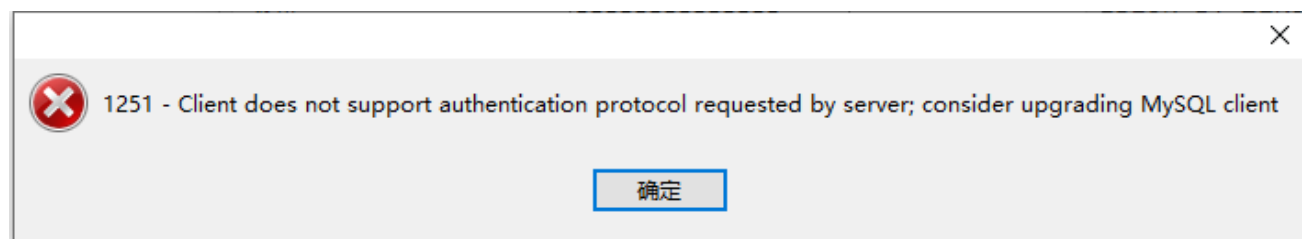
The screenshot shows a SQL client interface with a toolbar at the top containing icons for saving, creating queries, beautifying SQL, code snippets, text, exporting results, and creating charts. Below the toolbar is a connection bar with the IP address 192.168.220.128 and buttons for running, stopping, and explaining queries. The main area displays the SQL command 'show variables like 'sql_mode';' with a red circle and the number 1 next to it, and a tooltip that says '执行SQL语句'. Below the command is a table with two columns: 'Variable_name' and 'Value'. The table has one row with 'sql_mode' as the variable name and 'STRICT_TRANS_TABLES,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION' as the value. A red circle and the number 2 are next to the table, with a tooltip that says '检查sql_mode对应值，是否与配置文件中的值一致'.

Variable_name	Value
sql_mode	STRICT_TRANS_TABLES,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION

二、常见问题的解决方案

1 低版本navicat连接不了mysql8

连接时候出现类似下图的错误



出现这个问题是由于mysql8之前的版本中加密规则是mysql_native_password,而在mysql8之后,加密规则是caching_sha2_password

处理方式有两种：

- 升级Navicat（推荐）
- 新增一个使用mysql_native_password加密方式的用户
 - 进入容器，并连接mysql

```
1 docker exec -it mysql8 /bin/bash
2 mysql -uroot -p
```

- 然后输入命令

```
1 # 进入MySQL数据库
2 use mysql;
3 # 创建一个新的登录用户
4 create user 'root1'@'%' identified by '123456';
5 # 授予访问权限
6 grant all on *.* to 'root1'@'%';
7 # 修改登录账号密码, 使用mysql_native_password加密方式
8 ALTER USER 'root1'@'%' IDENTIFIED WITH mysql_native_password BY '123456';
9 # 刷新权限
10 FLUSH PRIVILEGES;
```

2 忘记密码怎么修改

在my.cnf中加入

```
1 #免密码登录
2 skip-grant-tables
```

然后重启MySQL容器, 然后进入容器

```
1 # 进入容器
2 docker exec -it mysql8 /bin/bash
3 # 登录数据库
4 mysql -uroot -p
5 # 提示输入密码的时候, 回车即可, 因为已开启免密登录, 然后输入下列指令将root密码之置空
6 UPDATE mysql.user SET authentication_string='' WHERE user='root' and host='%';
```

退出容器后, 将配置文件中免密登录修改回来, 重启MySQL容器, 然后在进入容器, 来修改root的密码

```
1 # 进入容器
2 docker exec -it mysql8 /bin/bash
3 # 登录数据库
4 mysql -uroot -p
5 # 提示输入密码的时候, 回车即可, 因为密码已经置空了, 然后输入下列指令修改root密码
6 ALTER USER 'root'@'%' IDENTIFIED WITH mysql_native_password BY '123456';
7 # ALTER USER 'root'@'%' IDENTIFIED WITH caching_sha2_password BY '123456';
8 # 刷新权限
9 FLUSH PRIVILEGES;
```

3 数据库时间与主机不同步

使用docker安装mysql, 时间要比主机晚8个小时, 此时需要将本地时间文件复制到MySQL容器中

```
1 docker cp /usr/share/zoneinfo/Asia/Shanghai 容器名称:/etc/localtime
```

```
[root@aubinll ~]# docker cp /usr/share/zoneinfo/Asia/Shanghai mysql8:/etc/localtime
```

然后进入容器再看看时间是否正常

```
[root@aubinll ~]# date Thu Dec 23 11:18:15 CST 2021
[root@aubinll ~]# docker exec -it mysql8 /bin/bash
root@a0e4adfee2e:/# date Thu Dec 23 11:18:25 CST 2021
root@7a0e4adfee2e:/#
```

外部linux服务器

mysql容器

三、主从复制搭建[可选]

主从环境一般在测试环境和生产环境中使用，开发环境一般都是单机模式。

注意：本教程搭建主从复制都是在同一台物理机器上演示，在生产环境下，一般是在多台电脑上搭建的，搭建思路和过程是一样的。

1 一主一从

准备目录

准备一个目录存储主从复制相关的配置和数据

```
1 # 创建目录
2 mkdir -p /home/mysql-ms/msql-master/conf/ /home/mysql-ms/msql-slave/conf/
3 # 进入目录
4 cd /home/mysql-ms
```

服务编排

创建docker-compose.yaml文件，并写入如下内容

```
1 version: "3.8"
2 services:
3   mysql-master:
4     container_name: mysql-master
5     image: mysql:8.0.20
6     restart: always
7     ports:
8       - 3340:3306
9     privileged: true
10    volumes:
11      - $PWD/msql-master/log:/var/log/mysql
12      - $PWD/msql-master/conf/my.cnf:/etc/mysql/my.cnf
13      - $PWD/msql-master/data:/var/lib/mysql
14      - $PWD/msql-master/mysql-files:/var/lib/mysql-files
15    environment:
16      MYSQL_ROOT_PASSWORD: "123456"
17    command: [
18      '--character-set-server=utf8mb4',
19      '--collation-server=utf8mb4_general_ci',
20      '--max_connections=3000'
21    ]
22    networks:
```

```

23     - mysql-ms
24 mysql-slave:
25     container_name: mysql-slave
26     image: mysql:8.0.20
27     restart: always
28     ports:
29     - 3341:3306
30     privileged: true
31     volumes:
32     - $PWD/msql-slave/log:/var/log/mysql
33     - $PWD/msql-slave/conf/my.cnf:/etc/mysql/my.cnf
34     - $PWD/msql-slave/data:/var/lib/mysql
35     - $PWD/msql-slave/mysql-files:/var/lib/mysql-files
36     environment:
37     MYSQL_ROOT_PASSWORD: "123456"
38     command: [
39     '--character-set-server=utf8mb4',
40     '--collation-server=utf8mb4_general_ci',
41     '--max_connections=3000'
42     ]
43     networks:
44     - mysql-ms
45 networks:
46     mysql-ms:
47     driver: bridge

```

编写配置

编写主服务配置，在msql-master/conf/目录下新建my.cnf配置文件并写一下内容

```

1  [mysqld]
2  secure-file-priv=NULL
3  # 【必须】 服务器唯一ID，默认是1，一般取IP最后一段
4  server-id=1
5
6  # 【必须】启用二进制日志
7  log-bin=zos-mysql-bin
8
9  # 【可选】 设置需要忽略同步的数据库
10 # 需要忽略多个数据库，配置多行即可，比如：
11 # binlog-ignore-db=db1
12 # binlog-ignore-db=db2
13 # binlog-ignore-db=db3
14 # MySQL5.7或更低版本这样配置：binlog-ignore-db=db1,db2,db3
15 # 这里配置忽略mysql库的同步
16 binlog-ignore-db=mysql
17
18 # 【可选】 设置需要同步的数据库
19 # 需要同步多个数据库，配置多行即可，比如：
20 # binlog_do_db=db1
21 # binlog_do_db=db2
22 # binlog_do_db=db3
23 # MySQL5.7或更低版本这样配置：binlog_do_db=db1,db2,db3
24 # 如果不设置此项配置，表示同步除了binlog-ignore-db设置的忽略的库外所有库
25
26 # 【可选】 设置二进制日志使用内存大小（事务）

```

```

27 binlog_cache_size=1M
28
29 # 【可选】 设置使用的二进制日志格式 (mixed,statement,row)
30 binlog_format=mixed
31
32 # 【可选】 二进制日志过期清理时间,这里设置为7天(86400 * 7)
33 binlog_expire_logs_seconds=604800
34
35 # 【可选】 跳过主从复制中遇到的所有错误或指定类型的错误,避免slave端复制中断。
36 # 如: 1062错误是指一些主键重复, 1032错误是因为主从数据库数据不一致
37 slave_skip_errors=1062

```

编写从服务配置, 在mysql-slave/conf/目录下面新建my.cnf配置文件并写一下内容

```

1 [mysqld]
2 secure-file-priv=NULL
3 # 【必须】 服务器唯一ID, 默认是1, 一般取IP最后一段
4 server-id=2
5
6 # 【必须】 开启二进制日志功能, 以备Slave作为其它Slave的Master时使用
7 log-bin=zos-mysql-bin
8
9 # 【可选】 设置需要忽略同步的数据库
10 # 需要忽略多个数据库, 配置多行即可, 比如:
11 # replicate-ignore-db=db1
12 # replicate-ignore-db=db2
13 # replicate-ignore-db=db3
14 # MySQL5.7或更低版本这样配置: replicate-ignore-db=db1,db2,db3
15 # 这里配置忽略mysql库的同步
16 replicate-ignore-db=mysql
17
18 # 【可选】 设置需要同步的数据库
19 # 需要同步多个数据库, 配置多行即可, 比如:
20 # replicate-do-db=db1
21 # replicate-do-db=db2
22 # replicate-do-db=db3
23 # MySQL5.7或更低版本这样配置: replicate-do-db=db1,db2,db3
24 # 如果不设置此项配置, 表示同步除了replicate-ignore-db设置的忽略的库外所有库
25
26 # 【可选】 设置二进制日志使用内存大小 (事务)
27 binlog_cache_size=1M
28
29 # 【可选】 设置使用的二进制日志格式 (mixed,statement,row)
30 binlog_format=mixed
31
32 # 【可选】 二进制日志过期清理时间,这里设置为7天(86400 * 7)
33 binlog_expire_logs_seconds=604800
34
35 # 【可选】 跳过主从复制中遇到的所有错误或指定类型的错误,避免slave端复制中断。
36 # 如: 1062错误是指一些主键重复, 1032错误是因为主从数据库数据不一致
37 slave_skip_errors=1062
38
39 # slave设置为只读 (具有super权限的用户除外)
40 read_only=1

```

相关配置可以参考资源目录下面的mysql-ms提供的配置文件。

启动服务

执行`docker-compose up -d`命令启动服务，启动成功后，使用`docker ps`命令查看，效果如下图所示

```
[root@localhost mysql-ms]# docker ps | grep mysql-
bd8ef1bf9a58   mysql:8.0.20   "docker-entrypoint.s..."   5 seconds ago   Up 3 seconds   33060/tcp, 0.0.0.0:3341->3306/tcp, :::3341->3306/tcp   mysql-slave
c30e5f00c25a   mysql:8.0.20   "docker-entrypoint.s..."   5 seconds ago   Up 3 seconds   33060/tcp, 0.0.0.0:3340->3306/tcp, :::3340->3306/tcp   mysql-master
```

设置主服务器

进入主服务器容器

```
1 # 进入容器
2 docker exec -it mysql-master bash
3 # 登录mysql
4 mysql -uroot -p123456
```

依次执行下列SQL命令

```
1 # 查看server_id是否生效
2 show variables like '%server_id%';
3 # 创建数据同步的用户
4 CREATE USER 'slave'@'%' IDENTIFIED WITH 'mysql_native_password' BY '123456';
5 # 对用户进行授权操作
6 GRANT REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO 'slave'@'%';
7 # 刷新权限
8 flush privileges;
9 # 看master信息File和Position从服务上要用
10 show master status;
```

执行过程如下图所示

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show variables like '%server_id%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| server_id     | 1     |
| server_id_bits | 32    |
+-----+-----+
2 rows in set (0.00 sec)

mysql> CREATE USER 'slave'@'%' IDENTIFIED WITH 'mysql_native_password' BY '123456';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO 'slave'@'%';
Query OK, 0 rows affected (0.00 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)

mysql> show master status;
+-----+-----+-----+-----+-----+
| File           | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
+-----+-----+-----+-----+-----+
| zos-mysql-bin.000003 |      848 |              | mysql              |                    |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

设置从服务器

进入从服务器容器

```
1 # 进入容器
2 docker exec -it mysql-slave bash
3 # 登录mysql
4 mysql -uroot -p123456
```

依次执行下列SQL命令

```
1 # 查看server_id是否生效
2 show variables like '%server_id%';
3 # 连接主服务，其中master_log_file和master_log_pos的值要填写master里查出来的值
4 change master to
5     master_host='mysql-master',
6     master_user='slave',
7     master_password='123456',
8     master_port=3306,
9     master_log_file='zos-mysql-bin.000003',
10    master_log_pos=848,
11    master_connect_retry=30;
12 # 启动slave
13 start slave;
```

连接主服务器参数说明：

master_host：Master的地址，由于在同一个物理机器和网络下，我这里直接使用容器名称代替IP地址

master_port：Master的端口，由于在同一个物理机器和网络下，我这里直接使用容器的端口而不是映射的宿主机端口

master_user：用于同步数据的用户

master_password：用于同步数据的用户的密码

master_log_file：指定Slave从哪个日志文件开始复制数据，即前面提到的File字段的值

master_log_pos：从哪个Position开始读，即前面提到的Position字段的值

master_connect_retry：如果连接失败，重试的时间间隔，单位是秒，默认是60秒

执行过程如下图所示

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show variables like '%server_id%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| server_id     | 2     |
| server_id_bits | 32    |
+-----+-----+
2 rows in set (0.01 sec)

mysql> change master to
-> master_host='mysql-master',
-> master_user='slave',
-> master_password='123456',
-> master_port=3306,
-> master_log_file='zos-mysql-bin.000003',
-> master_log_pos=848,
-> master_connect_retry=30;
Query OK, 0 rows affected, 2 warnings (0.01 sec)

mysql> start slave;
Query OK, 0 rows affected (0.01 sec)
```

查看主从状态

```
1 # 查看状态
2 show slave status \G
```

执行结果如下图

```
mysql> start slave;
Query OK, 0 rows affected (0.01 sec)

mysql> show slave status \G
***** 1. row *****
      Slave_IO_State: Waiting for master to send event
      Master_Host: mysql-master
      Master_User: slave
      Master_Port: 3306
      Connect_Retry: 30
      Master_Log_File: zos-mysql-bin.000003
      Read_Master_Log_Pos: 848
      Relay_Log_File: c37140a79246-relay-bin.000002
      Relay_Log_Pos: 328
      Relay_Master_Log_File: zos-mysql-bin.000003
      Slave_IO_Running: Yes
      Slave_SQL_Running: Yes
      Replicate_Do_DB:
      Replicate_Ignore_DB: mysql
      Replicate_Do_Table:
```

如果出现两个Yes，说明主从复制成功。

主从同步问题参考：<http://www.cnblogs.com/kevingrace/p/6261111.html>

2 一主多从

一主多从和一主一从类似，其操作步骤如下所示：

1. 在服务器上创建对应的目录，存放配置文件和数据文件
2. 将一主一从的所有配置文件复制一份，然后修改复制过来的从配置文件为mysql-slave1
3. 复制mysql-slave1服务配置，你要创建多少个从服务器就复制多少份，复制完成后注意修改server-id的值
4. 修改服务编排文件，创建对应个数的容器即可，以及修改容器名称、端口、数据卷目录
5. 通过服务器编排文件启动所有数据库容器
6. 配置主服务
7. 配置各个从服务器

相关配置可以参考资源目录下面的mysql-mss提供的配置文件。

配置完成后你可以通过下列命令查看从服务器列表。

```
1 # 进入master容器
2 docker exec -it mysql-master bash
3 # 登录数据库
4 mysql -uroot -p123456
```

执行查看从服务器列表

```
1 # 显示从服务器列表信息
2 SHOW SLAVE HOSTS;
```

下面是示例效果

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> SHOW SLAVE HOSTS;
+-----+-----+-----+-----+-----+
| Server_id | Host | Port | Master_id | Slave_UUID |
+-----+-----+-----+-----+-----+
| 3 | | 3306 | 1 | 0688944f-da08-11ed-9217-0242c0a81003 |
| 2 | | 3306 | 1 | 0663743e-da08-11ed-877e-0242c0a81004 |
+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

提示：从节点建议不超过5台。从节点越多，同步延迟越久，而且写入数据的时候占用带宽越大（因为要将数据要拷贝多份到多台服务器上）。

3 级联主从

该模式数据只能从第一台主节点写入。级联的层数不宜超过3层，否则会造成较大的延迟。

级联主从和一主多从类似，只是在配置第二个slave的时候的master需要指向第一个slave，其操作步骤如下所示：

1. 在服务器上创建对应的目录，存放配置文件和数据文件
2. 将一主多从配置文件复制一份，并在slave1的配置加入log_slave_updates=1

3. 通过服务器编排文件启动所有数据库容器
4. 配置主服务
5. 配置从服务器1（设置主服务器为master，添加复制账号）

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

```
mysql> change master to
-> master_host='mysql-master',
-> master_user='slave',
-> master_password='123456',
-> master_port=3306,
-> master_log_file='zos-mysql-bin.000003',
-> master_log_pos=848,
-> master_connect_retry=30;
Query OK, 0 rows affected, 2 warnings (0.02 sec)
```

```
mysql> start slave;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> show slave status \G
***** 1. row *****
      Slave_IO_State: Waiting for master to send event
        Master_Host: mysql-master
        Master_User: slave
        Master_Port: 3306
        Connect_Retry: 30
        Master_Log_File: zos-mysql-bin.000003
        Read_Master_Log_Pos: 848
        Relay_Log_File: 17976911f64b-relay-bin.000002
        Relay_Log_Pos: 328
        Relay_Master_Log_File: zos-mysql-bin.000003
        Slave_IO_Running: Yes
        Slave_SQL_Running: Yes
        Get_master_public_key: 0
        Network_Namespace:
1 row in set (0.01 sec)
```

```
mysql> CREATE USER 'slave'@'%' IDENTIFIED WITH 'mysql_native_password' BY '123456';
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> GRANT REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO 'slave'@'%';
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> show master status;
+-----+-----+-----+-----+-----+
| File           | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
+-----+-----+-----+-----+-----+
| zos-mysql-bin.000003 |      848 |              |                  |                  |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

6. 配置从服务器2（设置主服务器为slave1）

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show variables like '%server_id%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| server_id     | 3     |
| server_id_bits | 32    |
+-----+-----+
2 rows in set (0.00 sec)

mysql> change master to
-> master_host='mysql-slave1',
-> master_user='slave',
-> master_password='123456',
-> master_port=3306,
-> master_log_file='zos-mysql-bin.000003',
-> master_log_pos=848,
-> master_connect_retry=30;
Query OK, 0 rows affected, 2 warnings (0.01 sec)

mysql> start slave;
Query OK, 0 rows affected (0.01 sec)

mysql> show slave status \G
***** 1. row *****
      Slave_IO_State: Waiting for master to send event
      Master_Host: mysql-slave1
      Master_User: slave
      Master_Port: 3306
      Connect_Retry: 30
      Master_Log_File: zos-mysql-bin.000003
      Read_Master_Log_Pos: 848
      Relay_Log_File: ccec7c6c0fe8-relay-bin.000002
      Relay_Log_Pos: 328
      Relay_Master_Log_File: zos-mysql-bin.000003
      Slave_IO_Running: Yes
      Slave_SQL_Running: Yes
```

相关配置可以参考资源目录下面的mysql-mmss提供的配置文件。

分别进入master和slave1查看从服务器列表，如下图所示

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> SHOW SLAVE HOSTS;

Server_id	Host	Port	Master_id	Slave_UUID
2		3306	1	b85cad9b-da75-11ed-9412-0242ac180004

1 row in set (0.00 sec)

mysql> exit

Bye

root@88ebcb500853:/# exit

exit

[root@localhost mysql-mmss]# docker exec -it mysql-slave1 bash

root@17976911f64b:/# mysql -uroot -p123456

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> SHOW SLAVE HOSTS;

Server_id	Host	Port	Master_id	Slave_UUID
3		3306	2	b85951c2-da75-11ed-842b-0242ac180003

1 row in set (0.00 sec)