Docker 应用部署-MySQL

一、安装MySQL

1 搜索mysql镜像

```
docker search mysql
```

2 拉取mysql镜像

```
docker pull mysql:8.0.20
```

3 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

```
# 在用户名目录下创建mysql目录用于存储mysql数据信息
mkdir ~/mysql
cd ~/mysql
# 创建docker容器
docker run -id \
-p 3306:3306 \
--name mysql8 \
--restart always \
-v $PWD/conf:/etc/mysql/conf.d \
-v $PWD/log:/var/log/mysql \
-v $PWD/data:/var/lib/mysql \
-e MYSQL_ROOT_PASSWORD=123456 \
mysql:8.0.20
```

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。

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

```
# 查看当前启动的容器
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。

4 配置防火墙

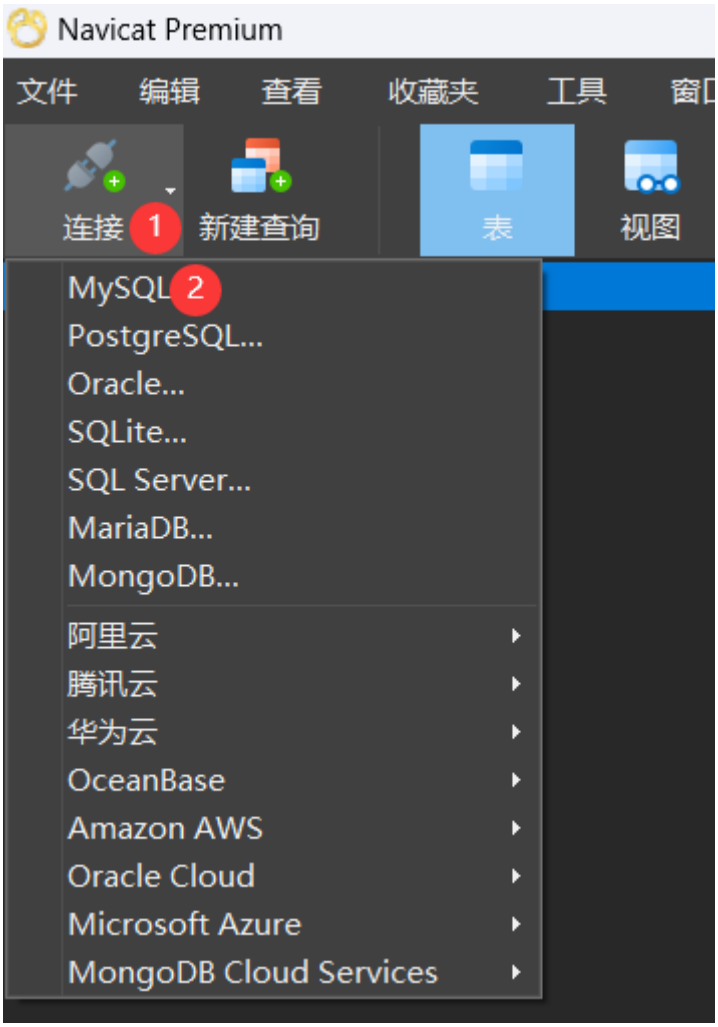
想要外部访问到你的数据库需要开放刚才映射的宿主机端口，我这里映射的是3306端口，所以开放3306端口。

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

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

5 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。

6 mysql参数调整

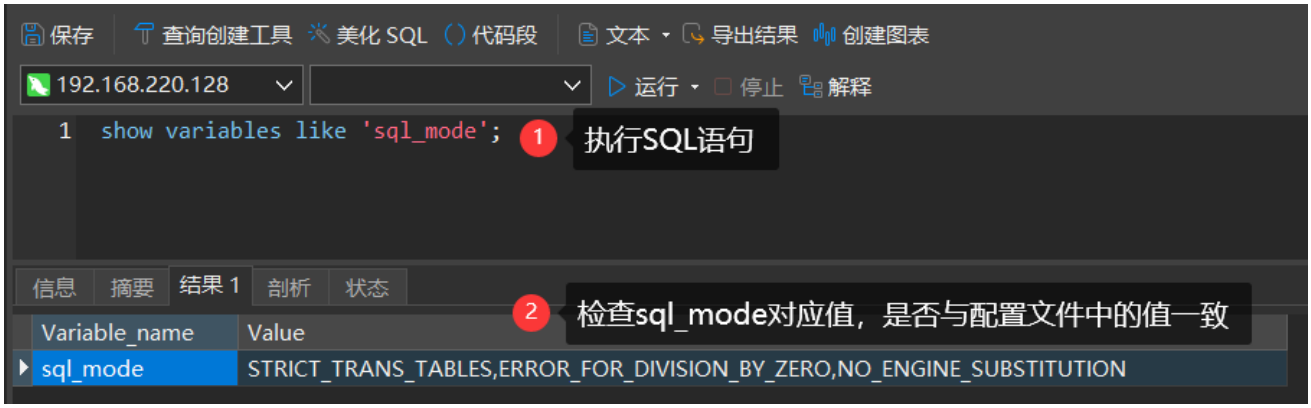
```
#进入目录修改配置
cd ~/mysql/conf
#创建配置文件
vi my.cnf
```

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。

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

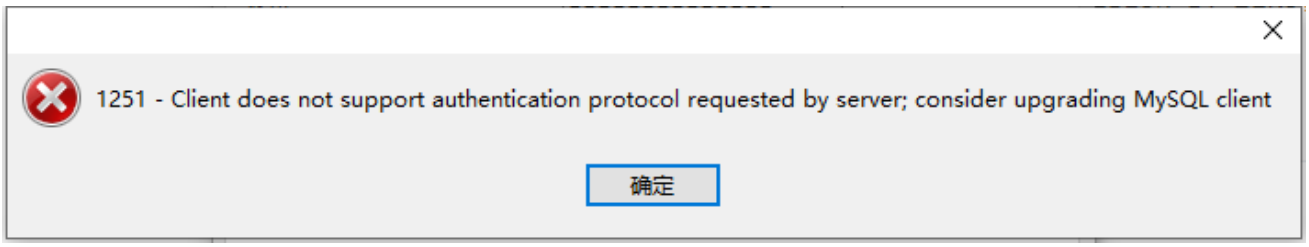
修改完配置后，重启容器，连接mysql，然后执行`show variables like 'sql_mode'`;命令来查看对应变量值是否正确，示例如如下图所示



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
    - ```
      docker exec -it mysql8 /bin/bash
      mysql -uroot -p
      ```
    - 然后输入命令
      ```

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

2 忘记密码怎么修改

在my.cnf中加入

```
#免密码登录
skip-grant-tables
```

然后重启MySQL容器，然后进入容器

```
# 进入容器
docker exec -it mysql8 /bin/bash
# 登录数据库
mysql -uroot -p
# 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
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

```
# 进入容器
docker exec -it mysql8 /bin/bash
# 登录数据库
mysql -uroot -p
# 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
ALTER USER 'root'@'%' IDENTIFIED WITH mysql_native_password BY '123456';
# ALTER USER 'root'@'%' IDENTIFIED WITH caching_sha2_password BY '123456';
# 刷新权限
FLUSH PRIVILEGES;
```

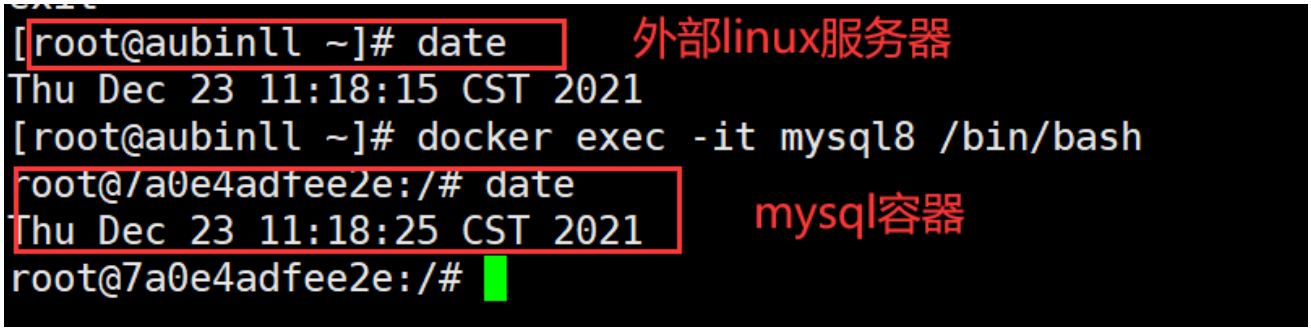
3 数据库时间与主机不同步

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

```
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



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

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

服务编排

创建docker-compose.yaml文件，并写入如下内容

```
version: "3.8"
services:
  mysql-master:
    container_name: mysql-master
    image: mysql:8.0.20
    restart: always
    ports:
      - 3340:3306
    privileged: true
    volumes:
      - $PWD/msql-master/log:/var/log/mysql
      - $PWD/msql-master/conf/my.cnf:/etc/mysql/my.cnf
      - $PWD/msql-master/data:/var/lib/mysql
      - $PWD/msql-master/mysql-files:/var/lib/mysql-files
    environment:
      MYSQL_ROOT_PASSWORD: "123456"
    command: [
        '--character-set-server=utf8mb4',
        '--collation-server=utf8mb4_general_ci',
        '--max_connections=3000'
    ]
    networks:
```

```
      - mysql-ms
  mysql-slave:
    container_name: mysql-slave
    image: mysql:8.0.20
    restart: always
    ports:
      - 3341:3306
    privileged: true
    volumes:
      - $PWD/msql-slave/log:/var/log/mysql
      - $PWD/msql-slave/conf/my.cnf:/etc/mysql/my.cnf
      - $PWD/msql-slave/data:/var/lib/mysql
      - $PWD/msql-slave/mysql-files:/var/lib/mysql-files
    environment:
      MYSQL_ROOT_PASSWORD: "123456"
    command: [
        '--character-set-server=utf8mb4',
        '--collation-server=utf8mb4_general_ci',
        '--max_connections=3000'
    ]
    networks:
      - mysql-ms
networks:
  mysql-ms:
    driver: bridge
```

编写配置

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

```
[mysqld]
secure-file-priv=NULL
# [必须] 服务器唯一ID，默认是1，一般取IP最后一段
server-id=1

# [必须]启用二进制日志
log-bin=zos-mysql-bin

# [可选] 设置需要忽略同步的数据库
# 需要忽略多个数据库，配置多行即可，比如：
# binlog-ignore-db=db1
# binlog-ignore-db=db2
# binlog-ignore-db=db3
# MySQL5.7或更低版本这样配置：binlog-ignore-db=db1,db2,db3
# 这里配置忽略mysql库的同步
binlog-ignore-db=mysql

# [可选] 设置需要同步的数据库
# 需要同步多个数据库，配置多行即可，比如：
# binlog_do_db=db1
# binlog_do_db=db2
# binlog_do_db=db3
# MySQL5.7或更低版本这样配置：binlog_do_db=db1,db2,db3
# 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

# [可选] 设置二进制日志使用内存大小（事务）
```

```
binlog_cache_size=1M

# [可选] 设置使用的二进制日志格式（mixed,statement,row）
binlog_format=mixed

# [可选] 二进制日志过期清理时间,这里设置为7天(86400 * 7)
binlog_expire_logs_seconds=604800

# [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。
# 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
slave_skip_errors=1062
```

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

```
[mysqld]
secure-file-priv=NULL
# [必须] 服务器唯一ID，默认是1，一般取IP最后一段
server-id=2

# [必须] 开启二进制日志功能，以备Slave作为其它Slave的Master时使用
log-bin=zos-mysql-bin

# [可选] 设置需要忽略同步的数据库
# 需要忽略多个数据库，配置多行即可，比如：
# replicate-ignore-db=db1
# replicate-ignore-db=db2
# replicate-ignore-db=db3
# MySQL5.7或更低版本这样配置：replicate-ignore-db=db1,db2,db3
# 这里配置忽略mysql库的同步
replicate-ignore-db=mysql

# [可选] 设置需要同步的数据库
# 需要同步多个数据库，配置多行即可，比如：
# replicate-do-db=db1
# replicate-do-db=db2
# replicate-do-db=db3
# MySQL5.7或更低版本这样配置：replicate-do-db=db1,db2,db3
# 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

# [可选] 设置二进制日志使用内存大小（事务）
binlog_cache_size=1M

# [可选] 设置使用的二进制日志格式（mixed,statement,row）
binlog_format=mixed

# [可选] 二进制日志过期清理时间,这里设置为7天(86400 * 7)
binlog_expire_logs_seconds=604800

# [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。
# 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
slave_skip_errors=1062

# slave设置为只读（具有super权限的用户除外）
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

```
# 进入容器
docker exec -it mysql-master bash
# 登录mysql
mysql -uroot -p123456
```

依次执行下列SQL命令

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

```
# 进入容器
docker exec -it mysql-slave bash
# 登录mysql
mysql -uroot -p123456
```

依次执行下列SQL命令

```
# 查看server_id是否生效
show variables like '%server_id%';
# 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
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;
# 启动slave
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

```
# 查看状态
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夹为msql-slave1
3. 复制msql-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。

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

执行查看从服务器列表

```
# 显示从服务器列表信息
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)
```