链滴

后端 | 基于 Docker 部署 MySQL8 集群（一主二从）

作者：z875479694h

原文链接：https://ld246.com/article/1626629488358

来源网站：链滴

一.CentOS7.9安装Docker20

1. 安装yum-utils工具

```
yum install -y yum-utils
```

2. 设置docker的依赖源

```
yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

注释：CentOS直接使用yum命令安装的Docker版本为1.13.1属于旧版docker的最后一个版本，所需要配置一个repo，才能安装新版的Docker-CE（社区版）。Docker-EE（企业版）需收费读者自了解即可，这里使用CE社区版

3. 安装docker

```
yum -y install docker-ce
```

4. 查看安装的版本

```
docker -v
docker version
```

```
[root@localhost ~]# docker -v
Docker version 20.10.7, build f0df350
[root@localhost ~]# docker version
Client: Docker Engine - Community
 Version:           20.10.7
 API version:       1.41
 Go version:        go1.13.15
 Git commit:        f0df350
 Built:             Wed Jun  2 11:58:10 2021
 OS/Arch:           linux/amd64
 Context:           default
 Experimental:      true

Server: Docker Engine - Community
 Engine:
  Version:          20.10.7
  API version:      1.41 (minimum version 1.12)
  Go version:       go1.13.15
  Git commit:       b0f5bc3
  Built:            Wed Jun  2 11:56:35 2021
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.4.6
  GitCommit:        d71fcd7d8303cbf684402823e425e9dd2e99285d
 runc:
  Version:          1.0.0-rc95
  GitCommit:        b9ee9c6314599f1b4a7f497e1f1f856fe433d3b7
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
```

5. 查看配套设置的版本

```
yum list installed | grep docker
```

```
[root@localhost ~]# yum list installed | grep docker
containerd.io.x86_64                    1.4.6-3.1.el7                   @docker-ce-stable
docker-ce.x86_64                        3:20.10.7-3.el7                 @docker-ce-stable
docker-ce-cli.x86_64                    1:20.10.7-3.el7                 @docker-ce-stable
docker-ce-rootless-extras.x86_64        20.10.7-3.el7                   @docker-ce-stable
docker-scan-plugin.x86_64               0.8.0-3.el7                     @docker-ce-stable
```

6.拉取MySQL8镜像

```
docker pull mysql:8
```

注解：mysql:5.7代表mysql版本为5.7

7. 查看docker镜像

```
docker images
```

```
[root@localhost ~]# docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
mysql        8         5c62e459e087   3 weeks ago   556MB
```

二.部署MySQL集群（一主二从）

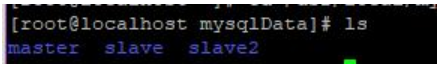
1.创建主从MySQL的配置及数据文件的存储目录

```
# 创建主服务的配置目录和数据目录
mkdir -p /usr/local/mysqlData/master/cnf
mkdir -p /usr/local/mysqlData/master/data

# 创建1号从服务器的配置目录和数据目录
mkdir -p /usr/local/mysqlData/slave/cnf
mkdir -p /usr/local/mysqlData/slave/data
```

```
# 创建2号从服务器的配置目录和数据目录
mkdir -p /usr/local/mysqlData/slave2/cnf
mkdir -p /usr/local/mysqlData/slave2/data
```

创建两个从服务器的配置是因为MySQL配置的server-id不能重复



2.配置主服务器的配置文件

```
vim /usr/local/mysqlData/master/cnf/mysql.cnf
```

配置文件如下

```
[mysqld]
## 设置server_id,注意要唯一
server-id=1
## 开启binlog
log-bin=mysql-bin
## binlog缓存
binlog_cache_size=1M
## binlog格式(mixed、statement、row,默认格式是statement)
binlog_format=mixed
##设置字符编码为utf8mb4
character-set-server = utf8mb4
collation-server = utf8mb4_unicode_ci
init_connect='SET NAMES utf8mb4'
[client]
default-character-set = utf8mb4
[mysql]
default-character-set = utf8mb4
```

3.配置从服务器的配置文件

```
# 1号从服务器
vim /usr/local/mysqlData/slave/cnf/mysql.cnf
# 2号从服务器
vim /usr/local/mysqlData/slave2/cnf/mysql.cnf
```

配置文件如下（1号的server-id设置为2，2号的server-id设置为3，不重复即可）

```
[mysqld]
## 设置server_id,注意要唯一
server-id=2
## 开启binlog
log-bin=mysql-slave-bin
## relay_log配置中继日志
relay_log=edu-mysql-relay-bin
## 如果需要同步函数或者存储过程
log_bin_trust_function_creators=true
## binlog缓存
binlog_cache_size=1M
## binlog格式(mixed、statement、row,默认格式是statement)
binlog_format=mixed
```

```
##设置字符编码为utf8mb4
character-set-server = utf8mb4
collation-server = utf8mb4_unicode_ci
init_connect='SET NAMES utf8mb4'
slave_skip_errors=1062
[client]
default-character-set = utf8mb4
[mysql]
default-character-set = utf8mb4
```

4.创建主从MySQL镜像

```
# 主服务器实例化
docker run -itd -p 3307:3306 --name master -v /usr/local/mysqlData/master/cnf:/etc/mysql/cnf.d -v /usr/local/mysqlData/master/data:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=12346 mysql:8

# 1号从服务器实例化
docker run -itd -p 3308:3306 --name slaver -v /usr/local/mysqlData/slave/cnf:/etc/mysql/conf.d -v /usr/local/mysqlData/slave/data:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=123456 mysql:8

# 2号从服务器实例化
docker run -itd -p 3309:3306 --name slaver2 -v /usr/local/mysqlData/slave2/cnf:/etc/mysql/cnf.d -v /usr/local/mysqlData/slave2/data:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=12345  mysql:8
```

参数解释

-p 指定容器暴露的端口,宿主机（物理机）端口: docker实例端口

-p 3307:3306 把物理机的3307端口给实例的端口3306端口进行映射

-v 给容器挂载存储卷，挂载到容器的某个目录

-v /usr/local/mysqlData/master/cnf:/etc/mysql/conf.d 把刚创建的配置文件夹映射成实例的/etc/ysql/conf.d

-v /usr/local/mysqlData/master/data:/var/lib/mysql 数据文件夹的映射

-e 指定环境变量，容器中可以使用该环境变量

-e MYSQL_ROOT_PASSWORD=123456 设置MySQL的root账号密码为123456

5. 查看已创建的实例

```
docker ps -a
```

```
[root@localhost mysqlData]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED       STATUS       PORTS                                                  NAMES
63589895fc60   mysql:8   "docker-entrypoint.s…"   2 hours ago   Up 2 hours   33060/tcp, 0.0.0.0:3309->3306/tcp, :::3309->3306/tcp   slaver2
45d3c9a2dd4f   mysql:8   "docker-entrypoint.s…"   2 hours ago   Up 2 hours   33060/tcp, 0.0.0.0:3308->3306/tcp, :::3308->3306/tcp   slaver
09a45836ccf6   mysql:8   "docker-entrypoint.s…"   3 hours ago   Up 3 hours   33060/tcp, 0.0.0.0:3307->3306/tcp, :::3307->3306/tcp   master
[root@localhost mysqlData]# docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED       STATUS       PORTS                                                  NAMES
63589895fc60   mysql:8   "docker-entrypoint.s…"   2 hours ago   Up 2 hours   33060/tcp, 0.0.0.0:3309->3306/tcp, :::3309->3306/tcp   slaver2
45d3c9a2dd4f   mysql:8   "docker-entrypoint.s…"   2 hours ago   Up 2 hours   33060/tcp, 0.0.0.0:3308->3306/tcp, :::3308->3306/tcp   slaver
09a45836ccf6   mysql:8   "docker-entrypoint.s…"   3 hours ago   Up 3 hours   33060/tcp, 0.0.0.0:3307->3306/tcp, :::3307->3306/tcp   master
```

6. 创建mysql连接用户

```
# 创建用户 reader设置密码为reader
```

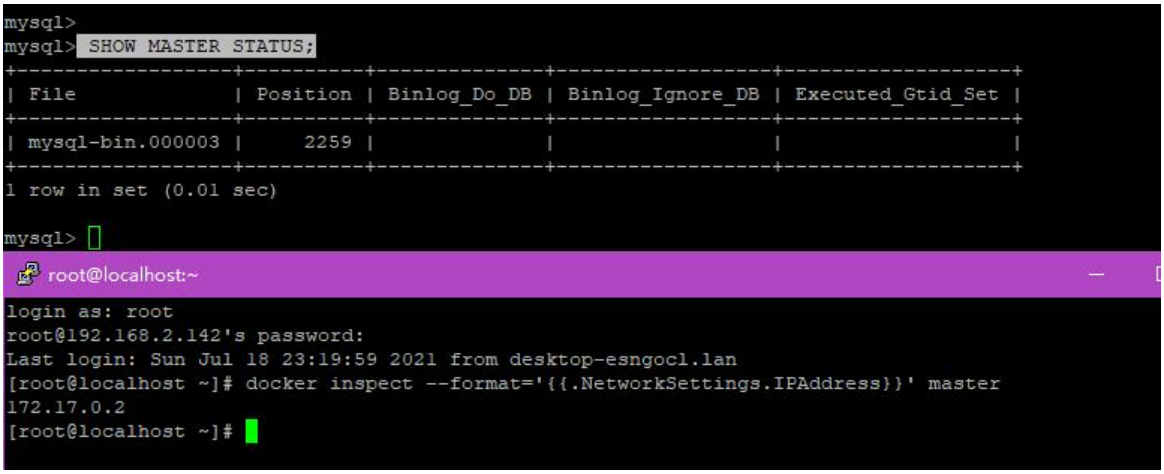
```
CREATE USER reader IDENTIFIED BY 'reader';
# 给予reader同步权限
GRANT REPLICATION SLAVE ON *.* to 'reader'@'%';
FLUSH PRIVILEGES;
```

注解：其余的用户，远程连接的自行设置

7. 获取主服务器的连接信息

```
# MySQL的连接信息
SHOW MASTER STATUS;

#新开连接 获取master实例的在docker的地址
docker inspect --format='{{.NetworkSettings.IPAddress}}' master
```



8. 从服务器连接主服务器（两台从服务器均是以下操作）

```
# 配置连接的参数
change master to master_host='172.17.0.2',master_user='reader',master_password='reader',
aster_log_file='mysql-bin.000003',master_log_pos=2259;
# 启动同步
start slave;
# 查看是否成功
show slave status\G

# 两项都为Yes时代表成功。
# Slave_IO_Running: Yes
# Slave_SQL_Running: Yes

# 失败需要使用停止连接后检查其他账号密码，地址，pos等参数

# 停止连接，如果一次成功无需使用该命令
stop slave;
```

```
mysql> change master to master_host='172.17.0.2',master_user='reader',master_password='reader',master_log_file='mysql-bin.000003',master_log_pos=2259;
Query OK, 0 rows affected, 8 warnings (0.27 sec)

mysql> start slave;
Query OK, 0 rows affected, 1 warning (0.08 sec)

mysql> show slave status\G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 172.17.0.2
                  Master_User: reader
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: mysql-bin.000003
          Read_Master_Log_Pos: 2259
               Relay_Log_File: edu-mysql-relay-bin.000002
                Relay_Log_Pos: 324
        Relay_Master_Log_File: mysql-bin.000003
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
```

三.结果

主服务器执行命令

SHOW SLAVE HOSTS;

```
mysql> SHOW SLAVE HOSTS;
+-----------+------+------+-----------+--------------------------------------+
| Server_id | Host | Port | Master_id | Slave_UUID                           |
+-----------+------+------+-----------+--------------------------------------+
|         3 |      | 3306 |         1 | 3ffbda88-e7d3-11eb-bdeb-0242ac110004 |
|         2 |      | 3306 |         1 | 7d48bb18-e7cf-11eb-a371-0242ac110003 |
+-----------+------+------+-----------+--------------------------------------+
2 rows in set, 1 warning (0.01 sec)
```

能从主服务器查询到两台从服务器的ID以及端口。完成MySQL部署。