26.MySQL主从复制–原理及传统主从搭建√

1.主从复制–“复制”介绍

Source–Replication

> 复制是指：将主库的DDL、DML等操作通过binlog日志，传输到复制服务器（副本），副本进行回放这些日志，从而使得从库和主库数据保持<近似>同步的工作模式。
> 复制架构:
> （1）传统（异步复制，基于网络）：1主1从、1主多从、级联主从、双主（早期使用较多，因为没有成熟的高可用软件）
> （2）演变：(增强)半同步、过滤、延时、GTID、MTS(多SQL线程并发回放relaylog)
> （3）新型：MSR多源复制（5.7+支持）
> （4）MGR：组复制 5.7.17+支持、8.0增强（WS:WriteSets）
> 复制主要应用场景：
> 1. 备份
> 2. 高可用
> 3. 读写分离
> 4. 分布式架构 （mycat）
> 5. 迁移升级

2.主从复制前提（传统环境搭建）

> a.准备2台以上的MySQL实例(同版本、同平台),分别作为主从节点。时间同步
> b.主从节点具备不同的server_id,server_uuid
> server_id和server_uuid是在主从关系中区分不同节点的身份。server_id手动配置默认等于1 ，server_uuid会自动生成。
> c.主库打开二进制日志（binlog）
> d.主库中创建专用复制用户，用于连接从库(拥有专业复制权限 replication slave ）
> e.初始化从库数据（将主库已有数据备份到从库中进行恢复，驱使主从数据大体一致）
> 可以使用mysqldump ,pxb,clone
> f.告知从库复制起点信息等。change master to user,password, ip ,port, filename、pos (gtid)
> g.启动专用复制线程，进行复制操作 start slave;
> h.查看复制状态

3.传统主从结构搭建（5.7版本之前）

通过clone–plugin备份方式搭建传统主从结构

0.准备两台节点，分别作为主从

1.配置主从节点 server_id和server_uuid不同

Bash | Copy

```
保证server_id 不同，使用配置文件修改的方式
主节点配置文件      |        从节点配置文件
【mysqld】          |        【mysqld】
server_id=51        |       server_id=52

保证server_uuid不同
因为我们db02节点是克隆db01机器，uuid号是db01的
所以删除db02数据目录下的auto.cnf 重启自动生成属于db02的uuid号
]# rm -rf /data/3306/data/auto.cnf
]# /etc/init.d/mysqld restart

查看server_id
主节点
mysql> select @@server_id;
+-------------+
| @@server_id |
+-------------+
|           1 |
+-------------+
从节点
mysql> select @@server_id;
+-------------+
| @@server_id |
+-------------+
|           2 |
+-------------+

查看server_uuid
主节点
mysql> select @@server_uuid;
+--------------------------------------+
| @@server_uuid                        |
+--------------------------------------+
| 7a46afe9-ae1b-11eb-a707-000c29ef43a9 |
+--------------------------------------+

从节点
mysql> select @@server_uuid;
+--------------------------------------+
| @@server_uuid                        |
+--------------------------------------+
| f611ee43-afa5-11eb-ad6a-000c2906fbc7 |
+--------------------------------------+

```

2.主库打开binlog日志

```
编辑主节点配置文件
【mysqld】
server_id=51
log_bin=/data/3306/binlog/mysql-bin

重启数据库生效
/etc/init.d/mysqld restart

清空binlog日志
mysql> reset master;
Query OK, 0 rows affected (0.01 sec)

mysql> show master status;
+------------------+----------+--------------+------------------+-------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
+------------------+----------+--------------+------------------+-------------------+
| mysql-bin.000001 |      156 |              |                  |                   |
+------------------+----------+--------------+------------------+-------------------+

```

3.主库创建专业复制用户repl

```
mysql> create user repl@'10.0.0.%' identified with mysql_native_password by '123';

mysql> grant replication slave on *.* to repl@'10.0.0.%';
```

4.clone-plugin备份主库已有数据到从库恢复

4.1 各个节点加载clone-plugin插件

```
mysql> INSTALL PLUGIN clone SONAME 'mysql_clone.so';
或
[mysqld]
plugin-load-add=mysql_clone.so
clone=FORCE_PLUS_PERMANENT

查看加载的clone插件
mysql>  SELECT PLUGIN_NAME, PLUGIN_STATUS FROM INFORMATION_SCHEMA.PLUGINS WHERE PLUGIN_NAME LIKE 'clone';
+-------------+---------------+
| PLUGIN_NAME | PLUGIN_STATUS |
+-------------+---------------+
| clone       | ACTIVE        |
+-------------+---------------+
```

4.2 创建远程clone用户

```
主库：捐赠者（db01）创建克隆用户
create user test1@'%' identified with mysql_native_password by '123';
grant backup_admin on *.* to test1@'%';
从库：接收者（db02）创建克隆用户
create user test2@'%' identified with mysql_native_password by '123';
grant clone_admin on *.* to test2@'%';
```

4.3 从库设置白名单（登陆root用户）

```
登陆从库root用户
mysql> SET GLOBAL clone_valid_donor_list='10.0.0.51:3306';
```

4.4 从库进行克隆操作（登陆克隆接收者用户）

```
登陆从库克隆接收者用户（test2）
]# mysql -utest2 -p123
mysql> CLONE INSTANCE FROM test1@'10.0.0.51':3306 IDENTIFIED BY '123';
指定克隆捐赠者用户test1
```

4.5 告诉从库复制起点信息

使用mysql命令 change master to

我们可以 `mysql> help change master to` 查看使用模板

```
CHANGE MASTER TO
  MASTER_HOST='master2.example.com',      主库主机地址
  MASTER_USER='replication',              复制用户
  MASTER_PASSWORD='password',             复制用户密码
  MASTER_PORT=3306,                       端口
  MASTER_LOG_FILE='master2-bin.001',      克隆完成后的日志文件
  MASTER_LOG_POS=4,                       克隆完成的位置点
  MASTER_CONNECT_RETRY=10;                主从网络连接失败后，进行连接尝试的频率（每隔10秒连接一次）

  查看克隆完成后的日志文件和位置点
  mysql>  select * from performance_schema.clone_status\G
*************************** 1. row ***************************
             ID: 1
            PID: 0
          STATE: Completed
     BEGIN_TIME: 2021-05-08 18:46:25.762
       END_TIME: 2021-05-08 18:46:33.946
         SOURCE: 10.0.0.51:3306
    DESTINATION: LOCAL INSTANCE
       ERROR_NO: 0
  ERROR_MESSAGE:
    BINLOG_FILE: mysql-bin.000001
BINLOG_POSITION: 889
  GTID_EXECUTED:
```

将模板改我们搭建环境的信息，在从库中进行执行(登陆root用户)

```
CHANGE MASTER TO
  MASTER_HOST='10.0.0.51',
  MASTER_USER='repl',
  MASTER_PASSWORD='123',
  MASTER_PORT=3306,
  MASTER_LOG_FILE='mysql-bin.000001',
  MASTER_LOG_POS=889,
  MASTER_CONNECT_RETRY=10;
```

5.从库启动专用复制线程

```
mysql> start slave;
```

6.查看从库状态

Bash | Copy

```
mysql> show slave status \G;
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 10.0.0.51
                  Master_User: repl
                  Master_Port: 3306
                Connect_Retry: 10
              Master_Log_File: mysql-bin.000001
          Read_Master_Log_Pos: 889
               Relay_Log_File: db02-relay-bin.000002
                Relay_Log_Pos: 324
        Relay_Master_Log_File: mysql-bin.000001
             Slave_IO_Running: Yes                重点关注
            Slave_SQL_Running: Yes                重点关注 两个yes就是主从环境搭建成功！！！！
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Master_Log_Pos: 889
              Relay_Log_Space: 532
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Master_SSL_Allowed: No
           Master_SSL_CA_File:
           Master_SSL_CA_Path:
              Master_SSL_Cert:
            Master_SSL_Cipher:
               Master_SSL_Key:
        Seconds_Behind_Master: 0
Master_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 0
                Last_IO_Error:
               Last_SQL_Errno: 0
               Last_SQL_Error:
  Replicate_Ignore_Server_Ids:
             Master_Server_Id: 1
                  Master_UUID: 7a46afe9-ae1b-11eb-a707-000c29ef43a9
             Master_Info_File: mysql.slave_master_info
                    SQL_Delay: 0
          SQL_Remaining_Delay: NULL
      Slave_SQL_Running_State: Slave has read all relay log; waiting for more updates
           Master_Retry_Count: 86400
                  Master_Bind:
      Last_IO_Error_Timestamp:
```

```
      Last_SQL_Error_Timestamp:
                Master_SSL_Crl:
            Master_SSL_Crlpath:
            Retrieved_Gtid_Set:
             Executed_Gtid_Set:
                 Auto_Position: 0
          Replicate_Rewrite_DB:
                  Channel_Name:
            Master_TLS_Version:
        Master_public_key_path:
         Get_master_public_key: 0
             Network_Namespace:

```

3.主从复制原理

1.主从复制架构中涉及到的线程

主库

Bash | Copy

```
Binlog_Dump_Thread :负责接收从库请求和返回结果（binlog）
查看
mysql> show processlist;
+----+-----------------+-----------------+------+-------------+------+-----------------------------------------------------------------
| Id | User            | Host            | db   | Command     | Time | State
+----+-----------------+-----------------+------+-------------+------+-----------------------------------------------------------------
|  5 | event_scheduler | localhost       | NULL | Daemon      |  627 | Waiting on empty queue
|  8 | repl            | 10.0.0.52:17674 | NULL | Binlog Dump |  627 | Master has sent all binlog to slave; wa
|  9 | root            | localhost       | NULL | Query       |    0 | starting
+----+-----------------+-----------------+------+-------------+------+-----------------------------------------------------------------
```

从库

Bash | Copy

```
1.Slave_IO_Running
2.Slave_SQL_Running
查看
[root@db02 ~]# mysql -uroot -p123 -e "show slave status \G" |grep Running:
mysql: [Warning] Using a password on the command line interface can be insecure.
             Slave_IO_Running: Yes
             Slave_SQL_Running: Yes

```

2.主从复制架构中涉及到的文件或表

主库

设置的binlog文件

从库

1.relay-log中继日志：暂时存储从库接收主库的binlog日志，使用完成会自动清空。

2.master-info:记录主库信息的文件（user ip port password binlog_file binlog_pos）

记录方式有两种

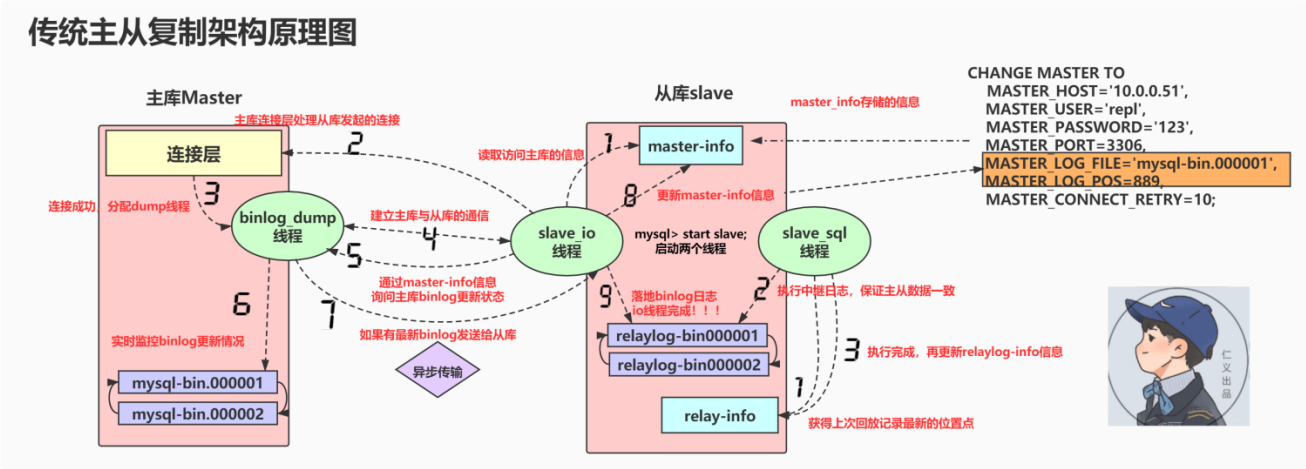
- 文件方式：master.info （mysql8.0版本已经不存在，默认存放在表中）
- 表方式: mysql.slave_master_info

3.relay-info:存储中继日志被sql线程回放的记录(断点续传，断点回放）

记录方式有两种

- 文件方式：relay.info (mysql8.0版本已经不存在，默认存放在表中)
- 表方式: mysql.slave_relay_log_info

3.传统主从复制架构原理图



4.传统主从复制线程工作原理图

5.传统主从复制原理文字说明

Bash | Copy

```
a. 从库: change master to ....,主库相关信息记录到master_info(MI)中.
b. 从库: start slave , 启动 IO SQL线程
c. 从库: 根据MI的信息,连接主库.
d. 主库: 生成一个DUMP线程和IO通信
e. 从库: 根据MI的信息(binlog pos),向主库请求新的binlog.
f. 主库: DUMP截取新的binlog,发送给从库IO
g. 从库: IO接收binlog,更新MI信息,存储binlog到relaylog中
h. 从库: SQL 读取Relay-info(RI)信息,获取上次会放到的位置点,回放最新的relaylog.
i. 从库: 从库SQL回放完成后,更新RI信息.
补充:
  主库DUMP实时监控binlog的变化,通知给从库IO.
  从库relaylog,会自动被清理
```

Replication%20%E5%A4%8D%E5%88%B6%E6%98%AF%E6%8C%87%EF%BC%9A%E5%B0%86%E4%B8%BB%E5%BA%93%E7%9A%84DDL%E3%80%81DML%E7%AD%89%E6%93%8D%E4%BD%9C%E9%