

# MySQL主从复制架构

## 一、MySQL集群概述

### ##1、集群的主要类型

- 高可用集群（High Available Cluster，HA Cluster）
- 高可用集群是指通过**特殊的软件**把独立的服务器连接起来，组成一个能够**提供故障切换**（Fail Over）功能的集群

## 2、如何衡量高可用

可用性级别(指标)	年度宕机时间	描述	叫法
99%	3.65天/年	基本可用系统	2个9
99.9%	8.76小时/年	可用系统	3个9
99.99%	52.6分钟/年	高可用系统	4个9
99.999%	5.3分钟/年	抗故障系统	5个9
99.9999%	32秒/年	容错系统	6个9

计算方法：

1年 = 365天 = 8760小时  
99% = 8760 \* 1% = 8760 \* 0.01 = 87.6小时=3.65天  
99.9 = 8760 \* 0.1% = 8760 \* 0.001 = 8.76小时  
99.99 = 8760 \* 0.0001 = 0.876小时 = 0.876 \* 60 = 52.6分钟  
99.999 = 8760 \* 0.00001 = 0.0876小时 = 0.0876 \* 60 = 5.26分钟

### ##3、常用的集群架构

- ==MySQL Replication==
- MySQL Cluster
- MySQL Group Replication（MGR） 5.7.17 多主一从
- MariaDB Galera Cluster
- ==MHA==|Keepalived|HeartBeat|Lvs，Haproxy等技术构建高可用集群

## 二、MySQL复制简介

### 1、什么是MySQL复制

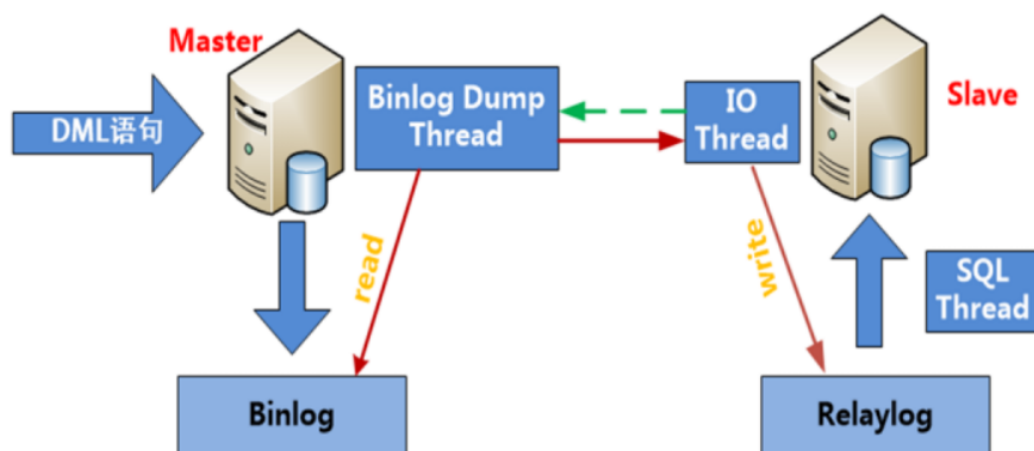
- Replication可以实现将数据从一台数据库服务器（master）复制到一台到多台数据库服务器（slave）
- 默认情况下，属于**异步**复制，所以无需维持长连接

### 2、MySQL复制原理（重点）

简单来说，master将数据库的改变写入**==二进制日志==**，slave同步这些二进制日志，并根据这些二进制日志进行**==数据重演操作==**，实现数据异步同步。

master：主

slave：从



#### 详细描述：

当主从同步配置完毕后：

1. slave端的IO线程发送请求给master端的binlog dump线程
2. master端binlog dump线程==获取二进制日志==信息(==文件名和位置信息==)发送给slave端的IO线程
3. slave端IO线程获取到的内容==依次==写到slave端relay log(中继日志)里，并把master端的bin-log文件名和位置记录到master.info里
4. slave端的SQL线程，检测到relay log中内容更新，就会解析relay log里更新的内容，并==执行这些操作==，从而达到和master数据一致

扩展：relay log中继日志

**作用：**记录从(slave)服务器接收来自主(master)服务器的二进制日志。

**场景：**用于==主从复制==

master主服务器将自己的二进制日志发送给slave从服务器，slave先保存在自己的==中继日志中==，然后再执行自己本地的relay log里的sql达到数据库更改和master保持一致。

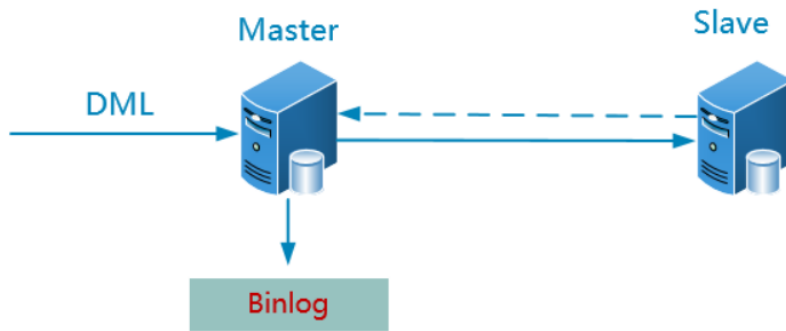
#### 如何开启？

默认中继日志==没有开启==，可以通过修改配置文件完成开启，如下：

```
# vim my.cnf
[mysqld]
#指定二进制日志存放位置及文件名
relay-log=/mysql_3306/data/relaylog
```

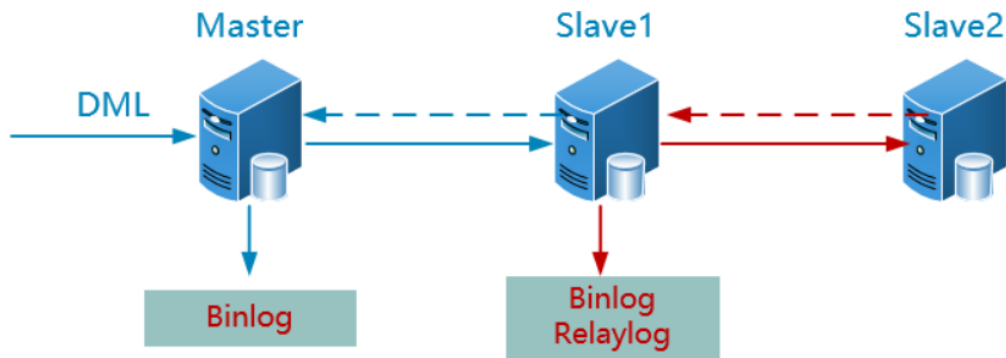
### 3、MySQL复制架构

## (一) 双机热备 (AB复制)



默认情况下，master接受读写请求，slave只接受读请求以减轻master的压力。

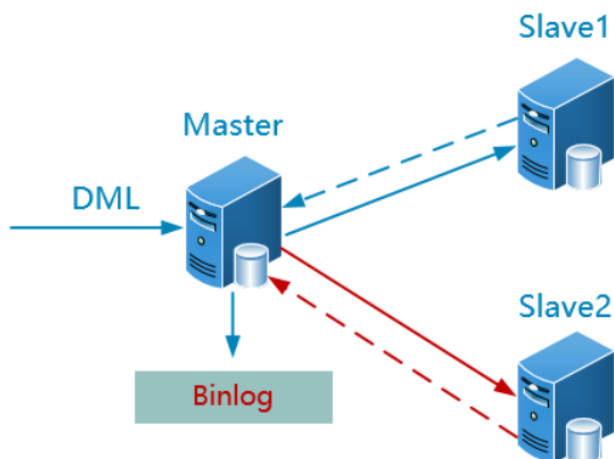
## (二) 级联复制



**优点：**进一步分担读压力

**缺点：**slave1 出现故障，后面的所有级联slave服务器都会同步失败

## (三) 并联复制 (一主多从)



**优点：**解决上面的slave1的单点故障，同时也分担读压力

**缺点：**间接增加master的压力（传输二进制日志压力）

## (四) 双主复制 (了解)



特点:

从命名来看, 两台master好像都能接受读、写请求, 但实际上, 往往运作的过程中, 同一时刻只有其中一台master会接受写请求, 另外一台接受读请求。

## 三、MySQL主从复制的搭建 (AB复制)

##1、传统AB复制架构(M-S)

说明: 在配置MySQL主从架构时, 必须保证数据库的版本高度一致, 统一版本为5.7.31

### 2、环境规划

编号	主机名称	主机IP地址	角色信息
1	master.itcast.cn	10.1.1.10	MASTER主服务器
2	slave.itcast.cn	10.1.1.100	SLAVE从服务器

### 3、安装前准备工作

第一步: 克隆两台全新的数据库服务器, MASTER/SLAVE

第二步: 首先启动MASTER, 然后启动SLAVE, 更改主机名称

Master:

```
# hostnamectl set-hostname master.itcast.cn
# su
```

Slave:

```
# hostnamectl set-hostname slave.itcast.cn
# su
```

第三步: 更改静态IP配置, 把Master和Slave都配置与规划一致, 然后关闭NetworkManager

Master:

```
# vim /etc/sysconfig/network-scripts/ifcfg-ens33
...
IPADDR=10.1.1.10
...
```

Slave:

```
# vim /etc/sysconfig/network-scripts/ifcfg-ens33
...
IPADDR=10.1.1.10
...
```

Master/Slave:

```
# systemctl stop NetworkManager
# systemctl disable NetworkManager
```

设置完成后，重启网络，然后使用MX连接Master与Slave。

第四步：由于两台机器处于集群架构，需要互相连接。绑定主机名称与IP地址到/etc/hosts

Master/Slave:

```
# vim /etc/hosts
10.1.1.10 master.itcast.cn
10.1.1.100 slave.itcast.cn
```

第五步：关闭防火墙与SELinux

```
# systemctl stop firewalld
# systemctl disable firewalld
# systemctl list-unit-files|grep firewalld
# setenforce 0
# sed -i '/^SELINUX=/cSELINUX=disabled' /etc/selinux/config
```

第六步：配置yum源（建议使用腾讯源）

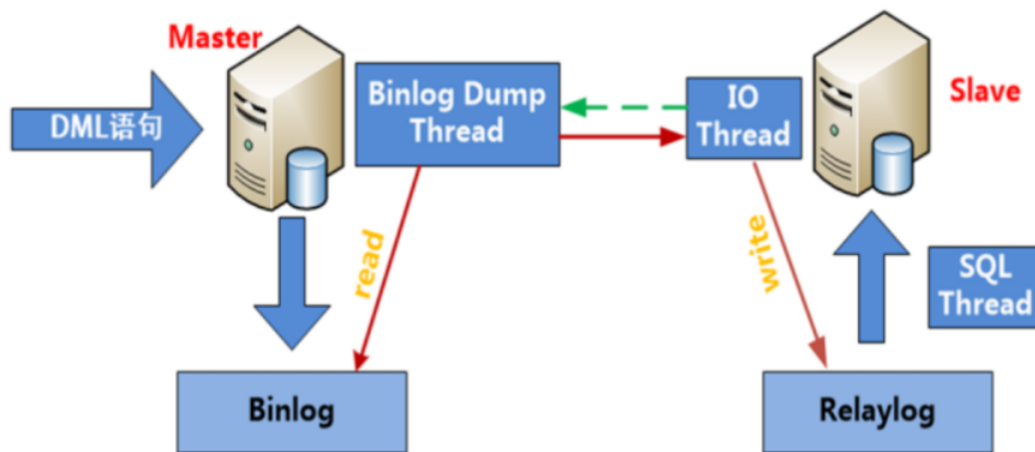
配置参考以前笔记（略）

第七步：时间同步

```
# ntpdate 182.92.12.11
```

## 4、MySQL主从复制核心思路

---



1. slave必须安装相同版本的mysql数据库软件
2. master端必须开启==二进制日志==；slave端必须开启==relay log日志==
3. master端和slave端的server-id号==不能一致==
4. slave端配置向master来同步数据
  - master端必须创建一个复制用户
  - 保证master和slave端==初始数据一致==
  - 配置主从复制（slave端）

## 5、MySQL主从复制的具体实践

### 第一步：上传MySQL软件包到Master与Slave

### 第二步：在Master端安装、初始化以及运行mysql软件

安装需求：

选项	值（自定义也可以采用默认形式）
安装路径	/usr/local/mysql
数据路径	/usr/local/mysql/data
端口号	3306

#### ① 安装MySQL软件

```
# vim mysql.sh
#!/bin/bash
yum install libaio -y
tar -xf mysql-5.7.31-linux-glibc2.12-x86_64.tar.gz
rm -rf /usr/local/mysql
mv mysql-5.7.31-linux-glibc2.12-x86_64 /usr/local/mysql
useradd -r -s /sbin/nologin mysql
rm -rf /etc/my.cnf
cd /usr/local/mysql
mkdir mysql-files
chown mysql:mysql mysql-files
chmod 750 mysql-files
bin/mysqld --initialize --user=mysql --basedir=/usr/local/mysql &>
/root/password.txt
```

```
bin/mysql_ssl_rsa_setup --datadir=/usr/local/mysql/data
cp support-files/mysql.server /etc/init.d/mysqld
service mysqld start
echo 'export PATH=$PATH:/usr/local/mysql/bin' >> /etc/profile
source /etc/profile

# source mysql.sh
```

Shell脚本其实就是命令的堆砌，把一堆Linux命令写在同一个文件中，一起执行。

## ② 安全配置

```
# mysql_secure_installation
```

## ③ 配置my.cnf（重点开启二进制日志）

```
# cd /usr/local/mysql
# vim my.cnf
[mysqld]
basedir=/usr/local/mysql
datadir=/usr/local/mysql/data
socket=/tmp/mysql.sock
port=3306
log-error=/usr/local/mysql/data/master.err
log-bin=/usr/local/mysql/data/binlog => 一定要开启二进制日志
server-id=10
character_set_server=utf8mb4 => utf8mb4相当于utf8升级版

配置完成后，重启mysqld服务
# service mysqld restart
# chkconfig --add mysqld
# chkconfig mysqld on
```

## 第三步：在Slave从服务器端安装mysql软件（不需要初始化）

### ① 安装MySQL软件

```
# vim mysql.sh
#!/bin/bash
yum install libaio -y
tar -xf mysql-5.7.31-linux-glibc2.12-x86_64.tar.gz
rm -rf /usr/local/mysql
mv mysql-5.7.31-linux-glibc2.12-x86_64 /usr/local/mysql
useradd -r -s /sbin/nologin mysql
rm -rf /etc/my.cnf
cd /usr/local/mysql
mkdir mysql-files
chown mysql:mysql mysql-files
chmod 750 mysql-files
cp support-files/mysql.server /etc/init.d/mysqld
echo 'export PATH=$PATH:/usr/local/mysql/bin' >> /etc/profile
source /etc/profile
```

相对于主服务器MySQL的安装与配置，从服务器端不需要进行初始化操作，因为其数据将来都来自于主服务器。

## ② 配置my.cnf文件

```
# cd /usr/local/mysql
# vim my.cnf
[mysqld]
basedir=/usr/local/mysql
datadir=/usr/local/mysql/data
socket=/tmp/mysql.sock
port=3310
log-error=/usr/local/mysql/data/slave.err
relay-log=/usr/local/mysql/data/relaylog => 开启中继日志
server-id=100
character_set_server=utf8mb4
```

## ③ 把master主服务器的数据目录同步到slave从服务器

a. 把MASTER服务器中的mysqld停止掉

```
# service mysqld stop
```

b. 把MASTER服务器中的/usr/local/mysql/data目录下的auto.cnf文件删除

```
# rm -rf /usr/local/mysql/data/auto.cnf
```

没安装一个mysql软件，其data数据目录都会产生一个auto.cnf文件，里面是一个唯一性的编号，相当于我们每个人的身份证号码。

c. 把MASTER服务器中/usr/local/mysql中的data目录拷贝一份到SLAVE从服务器的/usr/local/mysql目录

```
# rsync -av /usr/local/mysql/data root@10.1.1.100:/usr/local/mysql/
```

d. 同步完成后，把主服务器与从服务器中的mysqld启动

```
# service mysqld start
```

## 常见问题解决方案

☆ 常见问题1：MySQL无法启动

```
# service mysqld start
Redirecting to /bin/systemctl start mysqld.service
Failed to start mysqld.service: Unit not found.
```

出现以上问题的主要原因在于/etc/init.d目录中没有mysqld这个文件换句话说，就是你没有cp mysql.server脚本

```
# cp /usr/local/mysql/support-files/mysql.server /etc/init.d/mysqld
记住：/etc/init.d目录中的文件叫什么名字，我们service的时候就应该使用什么名字
# service mysqld start
```



☆ 常见错误2: MySQL没有安装在/usr/local/mysql目录中, service也无法启动

```
# service mysqld start
```

无法启动的原因可能是你的MySQL并没有安装在/usr/local/mysql目录中, 因为/etc/init.d/mysqld脚本中的basedir与datadir默认指向的都是/usr/local/mysql

所以如果我们更换了mysql的安装位置, 则必须更改/etc/init.d/mysqld脚本中basedir与datadir目录

```
# vim /etc/init.d/mysqld
```

basedir=你的安装路径

datadir=你的安装路径/data

☆ 常见问题3: 因为my.cnf配置文件导致mysql无法启动

```
# vim my.cnf
```

```
[mysqld]
```

```
basedir=/usr/local/mysql          => 安装路径
```

```
datadir=/usr/local/mysql/data      => 数据目录
```

```
socket=/tmp/mysql.sock            => GLIBC默认就是/tmp/mysql.sock
```

```
port=3310
```

```
log-error=/usr/local/mysql/slave.err => 错误日志到底放在哪里
```

```
relay-log=/usr/local/mysql/data/relaylog
```

```
server-id=100
```

```
character_set_server=utf8mb4
```

启动报错:

```
Starting MySQL.2020-08-31T07:17:06.554270Z mysqld_safe error: log-error set to
'/usr/local/mysql/slave.err', however file don't exists. Create writable for
user 'mysql'.
```

```
ERROR! The server quit without updating PID file
(/usr/local/mysql/data/slave.itcast.cn.pid).
```

产生以上问题的主要原因在于mysql这个用户对/usr/local/mysql文件夹没有写入权限

```
# ll -d /usr/local/mysql
```

```
drwxr-xr-x 11 7161 31415 174 Aug 31 15:16 /usr/local/mysql
```

发现文件拥有者位置与所属组位置都是两个数字, 正常应该是文件拥有者的名称与文件所属组的名称。但是由于GLIBC已经提前打包了, 我们解压后, 如果在我们系统中, 找不到原文件对应的文件拥有者与所属组, 则以两个数字代替文件拥有者与所属组的显示。

解决方案:

方案一: 建议把错误日志, 丢在数据目录中

```
# vim my.cnf
```

```
...
```

```
log-error=/usr/local/mysql/data/slave.err
```

方案二: 直接更改/usr/local/mysql目录的权限

```
# chown -R mysql:mysql /usr/local/mysql
```

#### ④ 配置MASTER-SLAVE主从同步

a. 在MASTER主服务器中创建一个账号，专门用于实现数据同步

MySQL5.7及以下版本：

```
mysql> grant replication slave on *.* to 'slave'@'10.1.1.1%' identified by '123';
```

MySQL新版本中：

```
mysql> create user 'slave'@'10.1.1.1%' identified by '123';
mysql> grant replication slave on *.* to 'slave'@'10.1.1.1%';
mysql> flush privileges;
```

b. 在MASTER中锁表，然后查看二进制文件的名称及位置

```
mysql> flush tables with read lock;
mysql> show master status;
```

File	Position	Binlog_Do_DB	Binlog_Ignore_DB	Executed_Gtid_Set
binlog.000002	597			

c. 在SLAVE从服务器中，使用change master to指定主服务器，并实现数据同步

```
mysql> change master to
master_host='10.1.1.10',master_user='slave',master_password='123',master_port=3306,master_log_file='binlog.000002',master_log_pos=597;
```

master\_host: 主机的IP地址  
master\_user: 主机的user账号  
master\_password: 主机的user账号密码  
master\_port: 主机MySQL的端口号  
master\_log\_file: 二进制日志文件名称  
master\_log\_pos: 二进制日志文件位置

技巧：主从复制的change master to语句记不住怎么办？答：求帮助，mysql> help change master to;

```
CHANGE MASTER TO
MASTER_HOST='master2.example.com',
MASTER_USER='replication',
MASTER_PASSWORD='password',
MASTER_PORT=3306,
MASTER_LOG_FILE='master2-bin.001',
```

```
MASTER_LOG_POS=4,  
MASTER_CONNECT_RETRY=10;
```

d. 启动slave数据同步

```
mysql> start slave;  
mysql> show slave status\G
```

```
mysql> show slave status\G  
***** 1. row *****  
Slave_IO_State: Waiting for master to send event  
Master_Host: 10.1.1.10  
Master_User: slave  
Master_Port: 3310  
Connect_Retry: 60  
Master_Log_File: binlog.000002  
Read_Master_Log_Pos: 1218  
Relay_Log_File: relaylog.000002  
Relay_Log_Pos: 317  
Relay_Master_Log_File: binlog.000002  
Slave_IO_Running: Yes  
Slave_SQL_Running: Yes  
Replicate_Do_DB:  
Replicate_Ignore_DB:
```

主服务器信息

IO和SQL线程都是YES说明同步配置ok

## 常见问题解决方案

在配置主从时，一般遇到错误，大部分都是change master to语句写错了（80%），解决方案：

```
mysql> stop slave;  
mysql> reset slave;  
mysql> change master to  
master_host='10.1.1.10',master_user='slave',master_password='123',master_port=33  
06,master_log_file='binlog.000002',master_log_pos=597;  
  
mysql> start slave;
```

## ⑤ 主MASTER服务器解锁

```
mysql> unlock tables;  
Query OK, 0 rows affected (0.00 sec)
```

## 总结：

1. 主从复制必须保证两台数据库实例的==server-id不一致==
2. 主服务器==必须开启二进制日志==；从服务器==必须开启中继日志==
3. 主从复制搭建==必须保证初始数据一致==
4. 主服务器必须要给从服务器==创建一个复制用户，并授予复制权限==
5. Master—>Slave架构，==主写会同步到从==；而==从写不会同步到主==

## 6、SLAVE从服务器不小心写入数据解决方案

正常情况下：

MASTER既可以读，也可以写。但是SLAVE从服务器，只能执行读取操作。一旦我们在SLAVE从服务器中写入数据，则主从架构会失败。

SLAVE:

```
# show slave status\G
```

```
Slave_IO_Running: Yes
Slave_SQL_Running: No
Replicate_Do_DB:
Replicate_Ignore_DB:
Replicate_Do_Table:
Replicate_Ignore_Table:
Replicate_Wild_Do_Table:
Replicate_Wild_Ignore_Table:
Last_Errno: 1062
Last_Error: Could not execute Write_rows event on table db_itheima.tb_student; Duplicate entry '4' for key 'PRIMARY', Error_code: 1062; handler error HA_ERR_FOUND_DUPP_KEY; the event's master log binlog.000002, end_log_pos 2358
```

遇到以上问题：如果数量较少，还可以通过跳过当前语句的方式解决。但是如果从服务器写入数据过多，则以上架构必须要重新搭建了！

解决方案：

问：

如果由于人为操作或者其他原因直接将数据更改到从服务器导致数据同步失效，怎么解决？

答：可以通过变量 `sql_slave_skip_counter` 临时跳过事务进行处理

```
SET GLOBAL sql_slave_skip_counter = N    N代表跳过N个事务
```

举例说明：

```
mysql> SET GLOBAL sql_slave_skip_counter=1;
```

```
mysql> stop slave;
```

```
mysql> start slave;
```

注意：

1. 跳过事务应该在slave上进行
2. 传统的AB复制方式可以使用变量：`sql_slave_skip_counter`，基于GTIDs的方式不支持

## 四、基于GTIDs的AB复制架构(M-S)

### 1、GTIDs概述

#### • 什么是GTIDs以及有什么特点？

1. GTIDs (Global transaction identifiers) ==全局事务标识符==，是mysql 5.6新加入的一项技术
2. 当使用GTIDs时，每一个事务都可以被==识别并且跟踪==
3. 添加新的slave或者当发生故障需要将master身份或者角色迁移到slave上时，都无需考虑是哪一个二进制日志以及哪个position值，==极大简化了相关操作==
4. GTIDs是完全基于事务的，因此==不支持MYISAM存储引擎==
5. ==GTID由source\_id和transaction\_id组成==：
  - 1) source\_id来自于server\_uuid,可以在auto.cnf中看到
  - 2) transaction\_id是一个序列数字，自动生成.

#### • 使用GTIDs的限制条件有哪些？

1. 不支持非事务引擎 (MyISAM)，因为可能会导致多个gtid分配给同一个事务
2. create table ... select 语句不支持 (主库语法报错)
3. create/drop **temporary table** 语句不支持
4. 必须使用enforce-gtid-consistency参数
5. sql-slave-skip-counter不支持(传统的跳过错误方式)

6. GTID复制环境中必须要求统一开启和GTID或者关闭GTID
7. 在mysql 5.6.7之前, 使用==mysql\_upgrade==命令会出现问题

## 2、基于GTIDs的主从复制

在生产环境中, 大多数情况下使用的MySQL5.6基本上都是从5.5或者更低的版本升级而来, 这就意味着之前的mysql replication方案是基于传统的方式部署, 并且已经在运行, 因此, 接下来我们就利用已有的环境升级至基于GTIDs的Replication

### ○ 思路

1. 修改配置文件支持GTIDs (主+从)
2. 重启数据库 (主+从)
3. 为了保证数据一致性, master和slave设置为只读模式 (主+从)
4. ==从服务器上重新配置同步== (从)

## 3、基于GTIDs的主从复制实践

### ① 修改配置文件支持GTIDs

MASTER => my.cnf

```
# vim my.cnf
...
gtid-mode=on
log-slave-updates=1
enforce-gtid-consistency
```

SLAVE => my.cnf

```
# rm -rf data/binlog.*
# vim my.cnf
...
log-bin=/usr/local/mysql/data/binlog    => 必须要开启二进制
gtid-mode=on
log-slave-updates=1
enforce-gtid-consistency
skip-slave-start    => 当MASTER主服务器GTIDS没有启动时, 跳过SLAVE服务器的启动
```

说明:

- 1) 开启GTIDs需要在master和slave上都配置gtid-mode, log-bin, log-slave-updates, enforce-gtid-consistency (该参数在5.6.9之前是--disable-gtid-unsafe-statement)
- 2) 其次, slave还需要增加skip-slave-start参数, 目的是启动的时候, 先不要把slave起来, 需要做一些配置
- 3) 基于GTIDs复制从服务器必须开启二进制日志!

### ② 重新启动mysqld服务

```
# service mysqld restart
```

### ③ 主从配置只读模式

```
mysql> set @@global.read_only=ON;
```

### ④ SLAVE重新配置change master to

```
mysql> stop slave;
mysql> reset slave;
mysql> change master to
master_host='10.1.1.10',master_user='slave',master_password='123',master_port=3306,master_auto_position=1;
```

注意:

1. 确保有复制用户
2. 主要区别于传统复制的参数是: master\_auto\_position=1

```
mysql> start slave;
mysql> show slave status\G
```

### ⑤ 关闭主从服务器的只读模式

```
mysql> set @@global.read_only=OFF;
```

测试验证 (往主服务器中写入部分数据, 验证一下)

## 4、SLAVE从服务器不小心写入数据解决方案

方法一: 跳过事务

指定需要跳过的GTIDs编号

```
SET GTID_NEXT='aaa-bbb-ccc-ddd:N';
```

开始一个空事务

```
BEGIN;
```

```
COMMIT;
```

使用下一个自动生成的全局事务ID。

```
SET GTID_NEXT='AUTOMATIC';
```

举例说明:

```
mysql> stop slave;
mysql> SET @@SESSION.GTID_NEXT= '13e36f00-eb70-11ea-91a8-000c29d1f40a:1'/*!*/;
mysql> BEGIN;
mysql> commit;
mysql> SET @@SESSION.GTID_NEXT= 'AUTOMATIC';
mysql> start slave;
mysql> show slave status\G
```

说明: 需要跳过哪个事务, 需要手动查看relaylog文件得到

```
[root@slave1 data]# ../bin/mysqlbinlog relay.000003|less
```

```
.....
```

```
# at 756
```

```
#181015 12:04:45 server id 10 end_log_pos 817 CRC32 0x5374f49e GTID
```

```
[commit=yes]
```

```
SET @@SESSION.GTID_NEXT= '044e6392-cf9b-11e8-a748-000c294ca304:3'/*!*/;
```

方法二：重新同步data目录，重新change master to...