实战MYSQL 8.0 .* 主主复制配置过程

> 搭建环境:

Server name	IP
mysql1	192.168.200.1
mysql2	192.168.200.2

虚拟机,centos7*,mysql8.0.9

推荐配置文件内容

```
[mysqld]
datadir=/usr/local/mysql/data
socket=/tmp/mysql.sock
server-id = 1
log-bin=mysql-bin
relay-log = mysql-relay-bin
replicate-wild-ignore-table=mysql.%
replicate-wild-ignore-table=test.%
replicate-wild-ignore-table=information_schema.%
symbolic-links=0
log-error=/usr/local/mysql/data/mysql1.err
pid-file=/usr/local/mysql/data/mysql1.pid
```

socket变量是用来指定socket文件的路径，方便客户端连接，当在本机登录mysql时，就是使用下面指定的路径以文件socket连接的

是否支持符号链接，即数据库或表可以存储在my.cnf中指定datadir之外的分区或目录，0不开

讲程id文件

进入mysql1的数据库

```
mysql -uroot -p #输入数据库的密码
mysql> flush table with read lock;
```

锁住数据库的写操作后，不要退出这个终端，复制一个终端，将/usr/local/mysql/data 压缩打包，然后再scp到mysql2上。

```
tar zcvf data.tar.gz data
scp data.tar.gz root@mysql2:/usr/local/mysql/
```

然后再mysql2上解压data.tar.gz， 并删除 auto.cnf文件

```
tar zxvf data.tar.gz
rm -rf auto.cnf
```

都重启mysqld服务

```
service mysqld restart
```

mysql1： MASTER； mysql2：SLAVE

在mysql1上创建复制用户(repl_user)，并授权

```
grant replication slave on . to 'repl_user'@'mysql2' identified by 'REPL_PASSWORD';
show master status;
```

并记录master的输出：File和Position的值

登录到mysql2上，设置mysql1为自己的主服务器

```
change master to
master_host='mysql1',master_user='repl_user',master_password='REPL_PASSWORD',master_log_fi
le='mysql1上File的值',master_log_pos=mysql1上的position值;
start slave;
show slave status \G;
```

mysql2： MASTER； mysql1：SLAVE

在mysql2上创建复制用户(repl_user)，并授权

```
grant replication slave on . to 'repl_user'@'mysql1' identified by 'REPL_PASSWORD';
show master status;
```

并记录master的输出：File和Position的值

登录到mysql1上，设置mysql2为自己的主服务器

```
change master to
master_host='mysql2',master_user='repl_user',master_password='REPL_PASSWORD',master_log_fi
le='mysql2上File的值',master_log_pos=mysql2上的position值;
start slave;
show slave status \G;
```

注意事项：

1. 两个数据库的数据在初始状态必须保持一致， 这样才能保证 show master status；的时候输出的结果才一致；

2. show slave status \G; 的时候。 Slave_IO_Running 和 Slave_SQL_Running 的结果为YES；
3. 数据库的配置文件：/etc/my.cnf 中的server-id的值必须不一样；
4. 两台服务器中的数据库版本必须一样；

实现原理:

当从库IO线程接受到主库传递来的二进制日志(Binlog)并将之保存为从库的中继日志(relay log),然后从库SQL线程将中继日志(relay log)的事件重做到从库上，实现主从数据同步。
如果SQL线程发现该事件的server_id与当前从库的server_id相同，则会丢弃该事件，因此如果两台MySQL如何互为主从，不会导致相同的事件被重复执行。

设计目的:

在一套MySQL复制群集中，通过双主或多主架构，解决一主多从架构的单点故障，减少主从切换的故障处理时间，增加MySQL群集的高可用性。

实现方案:

1、主备模式，两台MySQL互为主从，其中一台作为主节点对外提供服务，另外一台作为备机节点(standby)，当提供服务的主节点发生故障后，将服务请求快速切换到备用节点，原主节点故障恢复后转换为备用节点(standby)。

2、主主模式，两台MySQL互为主从，且两台MySQL均作为主节点对外提供服务，当其中一台MySQL发生故障后，将指向该故障节点的请求快速切换到另外一台MySQL，原来指向非故障节点的请求不受影响。

在主主模式下，两个主库都提供读写服务，如果应用通过两个主库操作相同数据，则会发生冲突导致数据覆盖(使用语句模式复制)或复制异常(使用行模式复制)，因此需要对读写服务进行控制：
1、基于自主主键控制，通过设置自增属性auto_increment_offset和auto_increment_increment来控制每个主节点生产不同的自增值，并根据不同自增值访问不同主节点。
2、基于库级别或表级别控制，如应用APP1访问节点node1上的DB1库，而应用APP2访问节点node2上的DB2库，两个主节点间不会操作相同表的数据，因此不会存在事务冲突。

设置auto_increment_increment和auto_increment_offset

```
show variables like '%auto_inc%';
show session variables like '%auto_inc%';  --  //session会话变量
show global variables like '%auto_inc%';  --  //全局变量
    SET @auto_increment_increment = 3 ;
    SET session auto_invrement_increment=2;
   SET global auto_increment_increment=1;
```

mysql中有自增长字段，在做数据库的主主同步时需要设置自增长的两个相关配置：auto_increment_offset和auto_increment_increment。

auto_increment_offset表示自增长字段从那个数开始，他的取值范围是1 .. 65535
auto_increment_increment表示自增长字段每次递增的量，其默认值是1，取值范围是1 .. 65535

在主主同步配置时，需要将两台服务器的auto_increment_increment增长量都配置为2，而要把auto_increment_offset分别配置为1和2.

优点/缺点

优点:

1、主主模式能将读写请求分摊到两个主节点，有效提升服务器使用率。

2、主节点发生故障后，能快速进行主从切换。

3、当故障节点恢复后，故障节点能通过复制进行数据恢复(应用其他节点数据)和数据同步(将未同步数据发生给其他节点)。

缺点:

1、当主节点上MySQL实例发生故障后，可能会存在部分数据(Binlog)未同步到另外的主节点，导致数据丢失(直到故障节点恢复)。

2、主主模式下，很容易因数据访问控制不当导致数据冲突。

3、为提高系统高可用性，双主架构会被扩展成双主多从结构，同样存在主节点发生故障后多个从库选主和恢复复制的问题。