**目录**

[一主一从](https://www.cnblogs.com/keerya/p/7873502.html#_label0)

[1.1 环境准备：](https://www.cnblogs.com/keerya/p/7873502.html#_label0_0)

[1.2 准备步骤：](https://www.cnblogs.com/keerya/p/7873502.html#_label0_1)

[1.3 实现步骤:](https://www.cnblogs.com/keerya/p/7873502.html#_label0_2)

[1.3.1 配置master主服务器](https://www.cnblogs.com/keerya/p/7873502.html#_label0_2_0)

[1.3.2 创建复制帐号](https://www.cnblogs.com/keerya/p/7873502.html#_label0_2_1)

[1.3.3 查看主服务器状态](https://www.cnblogs.com/keerya/p/7873502.html#_label0_2_2)

[1.3.4 配置slave从服务器](https://www.cnblogs.com/keerya/p/7873502.html#_label0_2_3)

[1.3.5 启动从服务器复制线程](https://www.cnblogs.com/keerya/p/7873502.html#_label0_2_4)

[1.3.6 查看从服务器状态](https://www.cnblogs.com/keerya/p/7873502.html#_label0_2_5)

[1.3.7 测试](https://www.cnblogs.com/keerya/p/7873502.html#_label0_2_6)

[1.4 扩展——实现一主多从](https://www.cnblogs.com/keerya/p/7873502.html#_label0_3)

[1.4.1 需求分析](https://www.cnblogs.com/keerya/p/7873502.html#_label0_3_0)

[1.4.2 具体实现过程](https://www.cnblogs.com/keerya/p/7873502.html#_label0_3_1)

[1.5 扩展——实现半同步复制](https://www.cnblogs.com/keerya/p/7873502.html#_label0_4)

[1.5.1 三种复制方式](https://www.cnblogs.com/keerya/p/7873502.html#_label0_4_0)

[1.5.2 实现半同步复制](https://www.cnblogs.com/keerya/p/7873502.html#_label0_4_1)

[互为主从（双主）](https://www.cnblogs.com/keerya/p/7873502.html#_label1)

[2.1 环境准备：](https://www.cnblogs.com/keerya/p/7873502.html#_label1_0)

[2.2 准备步骤：](https://www.cnblogs.com/keerya/p/7873502.html#_label1_1)

[2.3 实现步骤：](https://www.cnblogs.com/keerya/p/7873502.html#_label1_2)

[2.3.1 配置第一台 master 主服务器](https://www.cnblogs.com/keerya/p/7873502.html#_label1_2_0)

[2.3.2 配置第二台 master 主服务器](https://www.cnblogs.com/keerya/p/7873502.html#_label1_2_1)

[2.3.3 创建复制账号](https://www.cnblogs.com/keerya/p/7873502.html#_label1_2_2)

[2.3.4 查看服务器状态](https://www.cnblogs.com/keerya/p/7873502.html#_label1_2_3)

[2.3.5 启动复制线程](https://www.cnblogs.com/keerya/p/7873502.html#_label1_2_4)

[2.3.5 测试](https://www.cnblogs.com/keerya/p/7873502.html#_label1_2_5)

[高可用架构MHA的实现](https://www.cnblogs.com/keerya/p/7873502.html#_label2)

**正文**

[回到顶部](https://www.cnblogs.com/keerya/p/7873502.html#_labelTop)

**一主一从**

**1.1 环境准备：**

　　centos系统服务器2台、 一台用户做Mysql主服务器， 一台用于做Mysql从服务器， 配置好yum源、 防火墙关闭、 各节点时钟服务同步、 各节点之间可以通过主机名互相通信

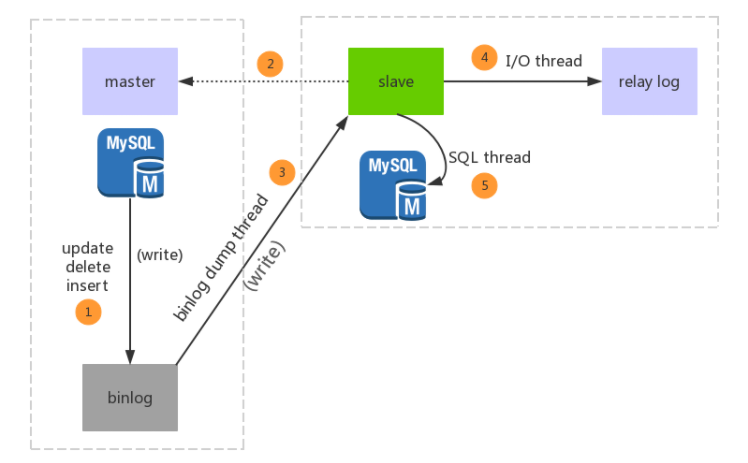
**1.2 准备步骤：**

1）iptables -F && setenforce 清空防火墙策略，关闭selinux

2）拿两台服务器都使用yum方式安装Mysql服务， 要求版本一致

3）分别启动两台服务器mysql服务， 确保服务正常

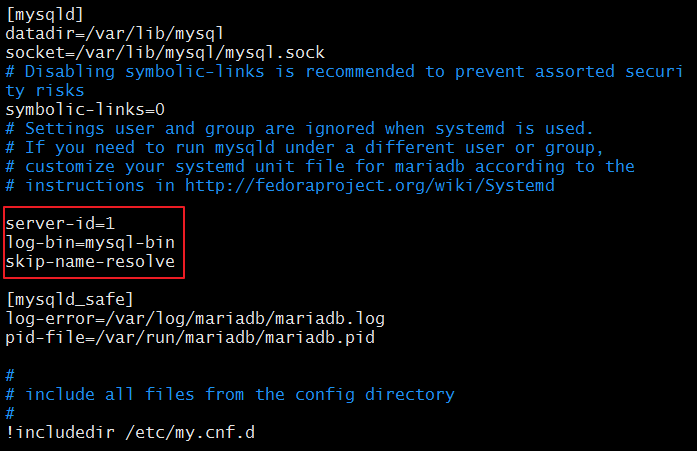
　　架构图:



**1.3 实现步骤:**

**1.3.1 配置master主服务器**

对master进行配置，包括打开二进制日志，指定唯一的servr ID。例如，在配置文件加入如下值　　
[mysqld]
#打开Mysql日志，日志格式为二进制
log-bin=/usr/local/mysql/data/binlog/mysql-bin
#[必须]服务器唯一ID，默认是1
server-id=1
#每次执行写入就与硬盘同步
sync-binlog=1
#需要同步的二进制数据库名
binlog-do-db=tuling
#只保留7天的二进制日志，以防磁盘被日志占满
expire-logs-days = 7
#不备份的数据库
binlog-ignore-db=information\_schema
binlog-ignore-db=performation\_schema
binlog-ignore-db=sys
skip-name-resolve　　//关闭名称解析，（非必须）



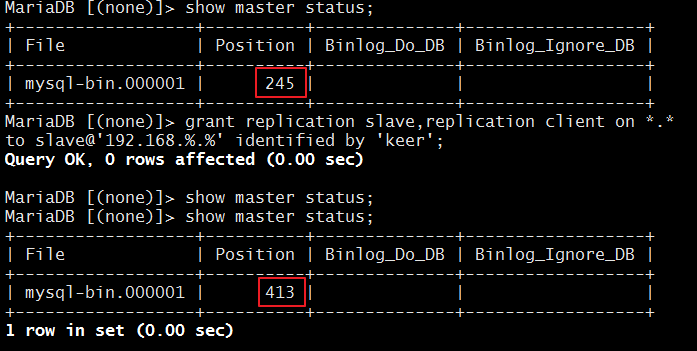
**1.3.2 创建复制帐号**

　　在 Master 的数据库中建立一个**备份帐户(user = tuling,pwd=root)**：每个 slave 使用标准的 MySQL 用户名和密码连接 master 。进行复制操作的用户会授予 REPLICATION SLAVE 权限。

#创建用户
MariaDB [(none)]> CREATE USER 'tuling'@'192.168.%.%' IDENTIFIED BY 'root';
#授权
MariaDB [(none)]> grant replication slave,replication client on \*.\* to tuling@'192.168.%.%' identified by 'root';
#刷新权限
MariaDB [(none)]> FLUSH PRIVILEGES;

**1.3.3 查看主服务器状态**

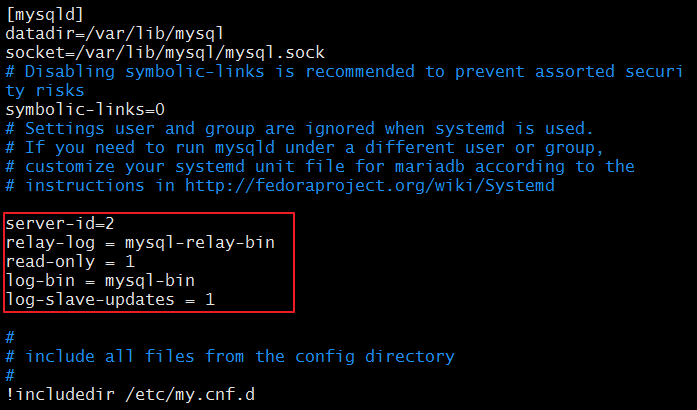
　　在 Master 的数据库执行 **show master status**，查看主服务器二进制日志状态及位置号。



**1.3.4 配置slave从服务器**

　　对 slave 进行配置，打开中继日志，指定唯一的 servr ID，设置只读权限。在配置文件加入如下值：

server-id=2 //配置server-id，让从服务器有唯一ID号
relay\_log = mysql-relay-bin //打开Mysql中继日志，日志格式为二进制
read\_only = 1 //设置只读权限
log\_bin = mysql-bin //开启从服务器二进制日志
log\_slave\_updates = 1 //使得更新的数据写进二进制日志中
#如果salve库名称与master库名相同，使用本配置
replicate-do-db = tuling
#如果master库名[tuling]与salve库名[tuling01]不同，使用以下配置[需要做映射]
replicate-rewrite-db = tuling[主库名] -> tuling01[从库名]
#如果不是要全部同步[默认全部同步]，则指定需要同步的表
replicate-wild-do-table=tuling01.t\_order
replicate-wild-do-table=tuling01.t\_order\_item



**1.3.5 启动从服务器复制线程**

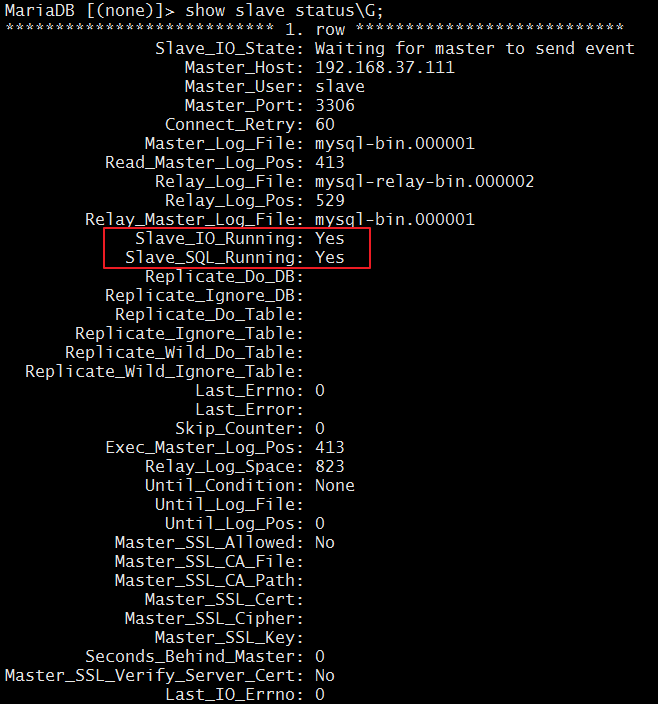
　　让 slave 连接 master ，并开始重做 master 二进制日志中的事件。

MariaDB [(none)]> change master to master\_host='192.168.241.198', [你的master-ip]
>master\_user='tuling',
>master\_password='root',
>master\_log\_file='mysql-bin.000007', //master对应的bin-log文件
>master\_log\_pos=413; //master对应的position
>start slave; //启动复制线程。
>SHOW SLAVE STATUS\G; //查看从节点状态
#启动可能报错：
ERROR 3021 (HY000): This operation cannot be performed with a running slave io thread;run STOP SLAVE IO\_THREAD FOR CHANNEL '' first.
#解决办法
//在设置同步关系时，没有先停止slaves线程。解决办法为在设置同步关系之前，
//先执行下“stop slave;”，以停止slave线程。然后启动线程"start slave;"

**1.3.6 查看从服务器状态**

　　可使用**SHOW SLAVE STATUS\G;**查看从服务器状态，如下所示，也可用**show processlist \G;**查看当前复制状态：

**Slave\_IO\_Running**: Yes //IO线程正常运行
**Slave\_SQL\_Running**: Yes //SQL线程正常运行



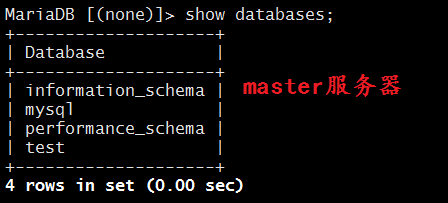
**1.3.7 测试**

**1）测试主从同步**

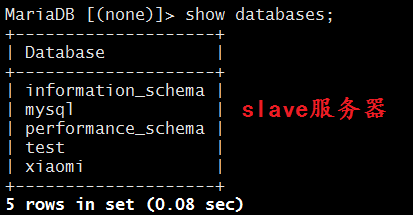
　　我们在 master 服务器上创建一个数据库，再使用该数据库创建一个表，添加一条记录，来看一看 slave 服务器有没有同步成功。

　　首先，我们先来查看一下两个服务器上有什么数据库：

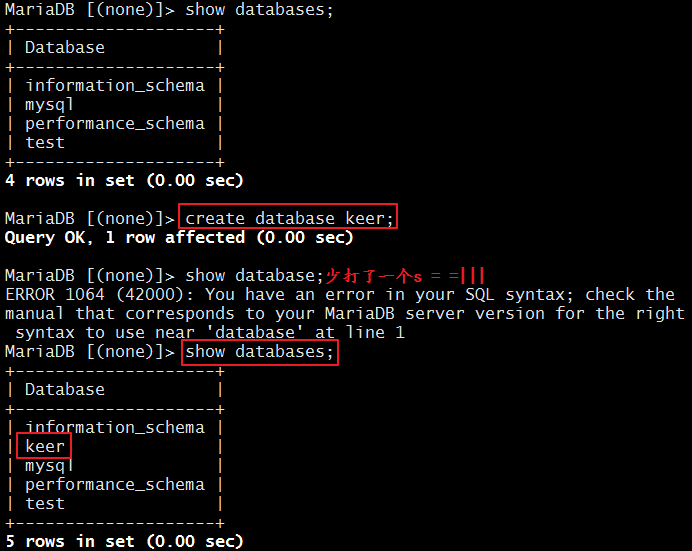
　　master：



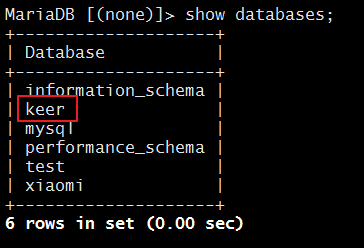
　　slave：



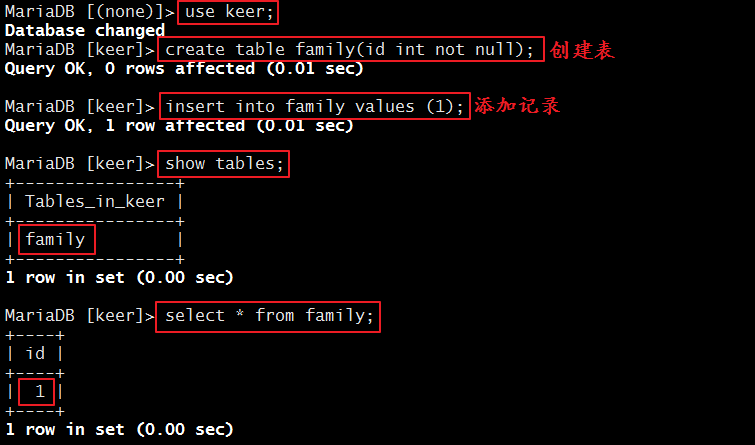
　　现在，我们在主服务器上创建一个数据库“keer”：



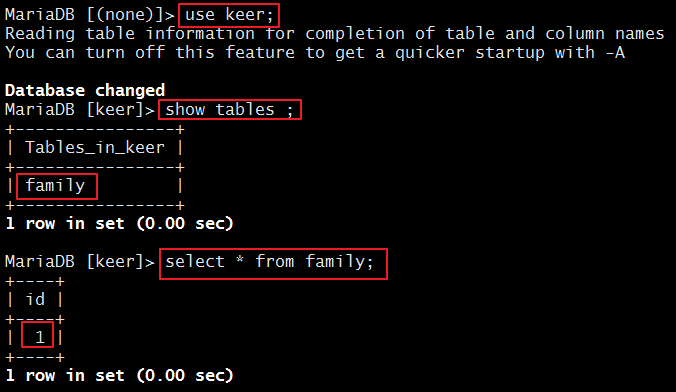
　　然后，我们来看看从服务器是否已经更新：



　　我们可以看出，数据库已经同步，接下来，我们就来创建表，并在表中插入一条记录：

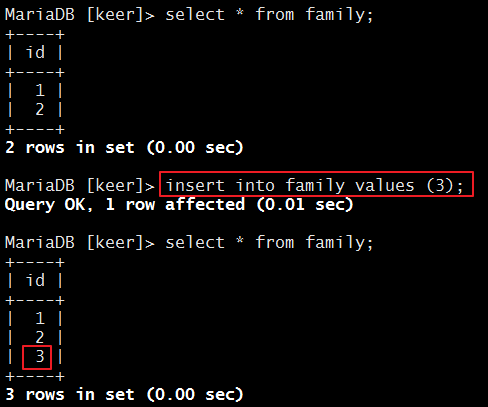


　　然后，我们来看看从服务器是否更新：



**2）测试只读属性**

　　我们在从服务器上设置了只读，所以我们来试试看能不能插入数据：



　　这就很尴尬了，为什么我们设置只读还是可以插入数据呢？这要如何解决呢？

　　因为我们登录的时候使用的是root超级用户，这个大变态是不受只读设置影响的，所以，我们还是可以往里面去插入数据，当然，如果我们换成其他的普通用户就不会出现这样的问题了。

　　解决方式也很简单，我们只需要把表锁起来就可以了：

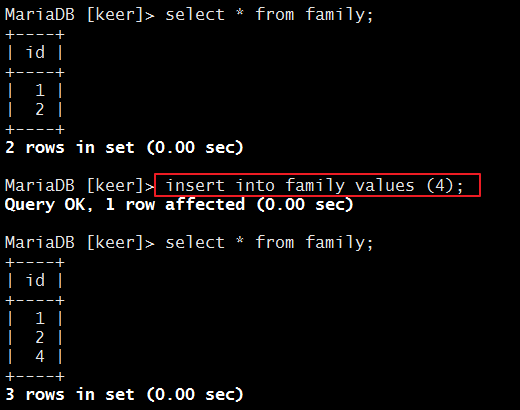
MariaDB [keer]> flush tables with read lock;

　　进行锁表操作以后，我们再来插入一条数据：

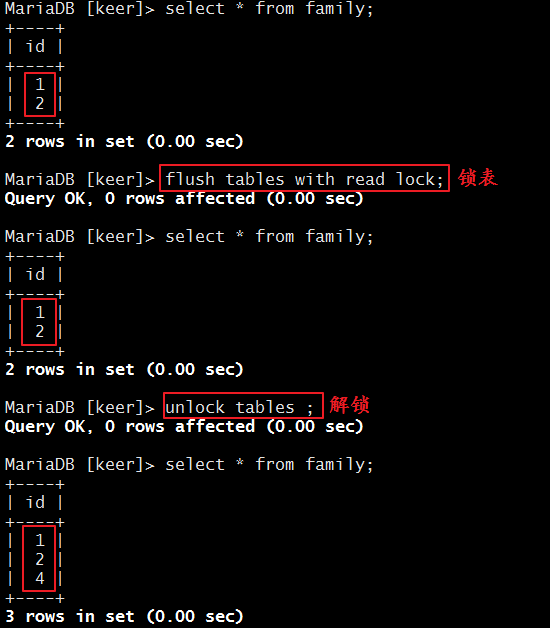
2-1247112071.png

　　这样的话，即使我们是超级用户登录进来的，也不能再插入数据了。这样会安全很多。但是，有利就有弊，这样的话，我们的主从就不能实时同步了，因为我们的主从同步是通过把主上的 sql 语句放在从上再执行一遍来实现的，可是我们锁表的话，sql 语句就不能执行了。想要同步方法也很简单，直接把表解锁就可以了。

　　我们在主上插入一条数据：



　　此时，我们的从上就要解表以后才能实现同步：



**1.4 扩展——实现一主多从**

**1.4.1 需求分析**

　　当我们的服务器运行一段时间后，流量变得越来越多，这时，一主一从能够实现的高可用性和负载均衡不能满足我们的需求，我们就要选择再添加一台从服务器。

　　可是现在我们的 master 已经运行很久了，我们也需要对新安装的 slave 进行数据同步，甚至它没有 master 的数据。

　　此时，有几种方法可以使 slave 从另一个服务开始，例如，从 master 拷贝数据，从另一个 slave 克隆，从最近的备份开始一个 slave 。为了加快 slave 与 master 同步，可用以下方式先进行数据同步：

（1）master的某个时刻的数据快照；

（2）数据库的备份数据；

（3）master的二进制日志文件。

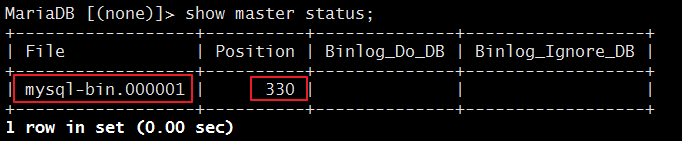
　　接下来，我们就来实现一下这个过程：

**1.4.2 具体实现过程**

1）对 master 进行完全备份

[root@master ~]# mkdir /backup
[root@master ~]# mysqldump --all-database > /backup/mysql-backup-`date +%F-%T`-all.sql

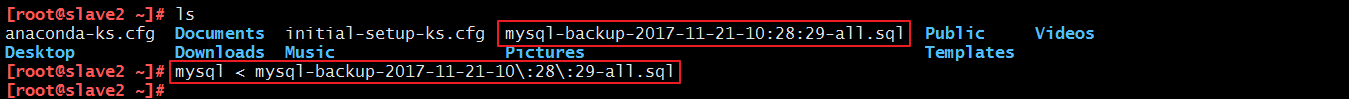
照例查看一下主的二进制日志状态及位置号。



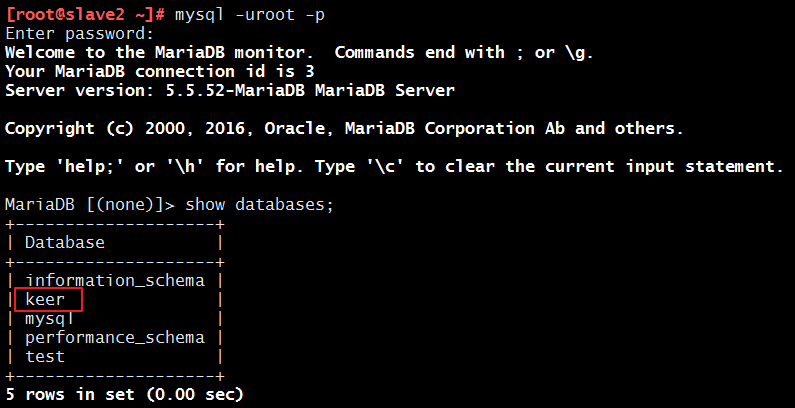
2）把完全备份的数据发到新增的从上

[root@master ~]# scp /backup/mysql-backup-2017-11-21-10\:28\:29-all.sql @192.168.37.133:~
root@192.168.37.133's password:
mysql-backup-2017-11-21-10:28:29-all.sql 100% 504KB 503.9KB/s 00:00

3）把这个完全备份导入从的数据库：

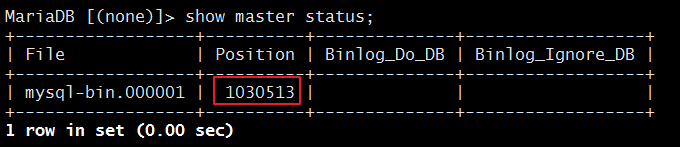


4）查看从是否恢复成功：





　　我们来查看一下从的二进制日志状态及位置号：



　　我们的数据已经恢复成功

5）设置主从

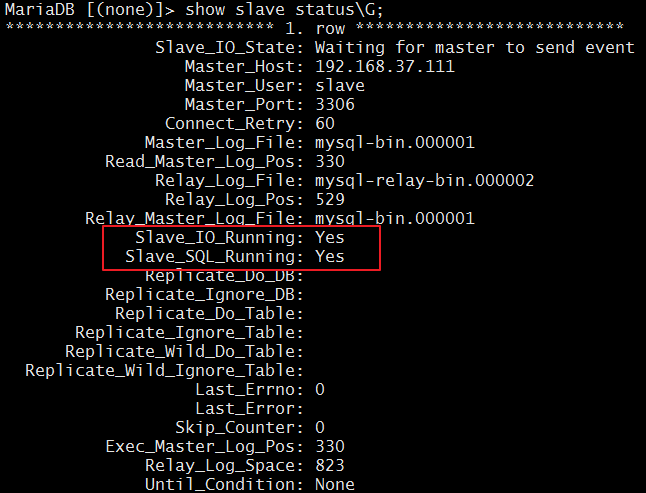
接下来就是按照上述步骤，对该从服务器进行设置就好：

[root@slave2 ~]# vim /etc/my.cnf
[mysqld]
server-id = 3
relay-log = mysql-relay-bin
read-only = 1
log-bin = mysql-bin
log-slave-updates = 1

　　然后，我们就来设置从服务器：

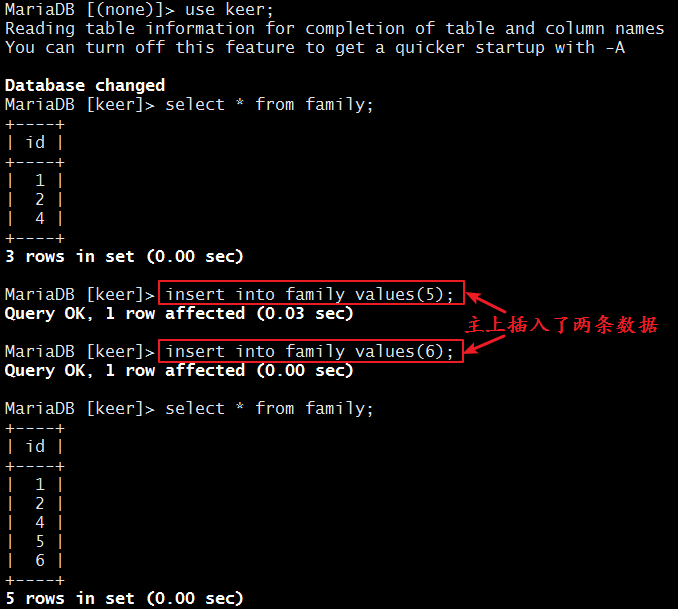
[root@slave2 ~]# mysql -uroot -p
Enter password:
MariaDB [(none)]> change master to master\_host='192.168.37.111',
>master\_user='slave',
>master\_password='keer',
>master\_log\_file='mysql-bin.000001',
>master\_log\_pos=330;
MariaDB [(none)]> start slave;

　　然后我们来查看一下从服务器的状态：



6）测试

　　设置完毕，我们来测试一下，当主上有操作时，从上是否更新：





　　至此，我们就完成了一主两从。如果有需要，我们还可以继续添加从服务器。

**1.5 扩展——实现半同步复制**

**1.5.1 三种复制方式**

**1）同步复制**

　　指当主库执行完一个事务，**所有的从库都执行了该事务**才返回给客户端。因为需要等待所有从库执行完该事务才能返回，所以全同步复制的**性能必然会收到严重的影响**。需要有超时时间。

**2）异步复制**

　　**MySQL默认的复制**即是异步的，**主库在执行完客户端提交的事务后会立即将结果返回给客户端，并不关心从库是否已经接收并处理，**这样就会有一个问题，主如果crash掉了，此时主上已经提交的事务可能并没有传到从上，如果此时，强行将从提升为主，**可能导致新主上的数据不完整。**

**3）半同步复制**

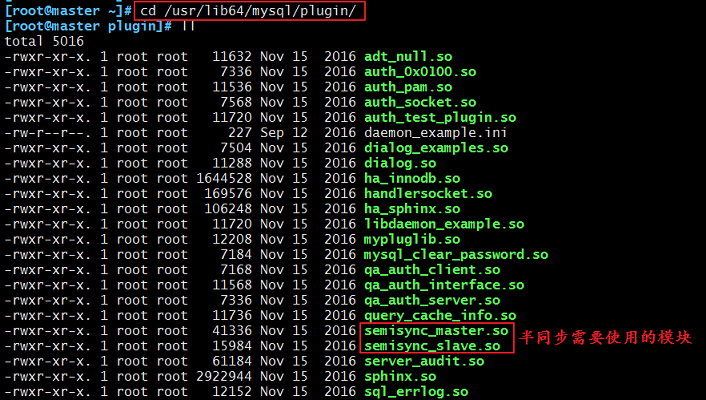
　　介于异步复制和全同步复制之间，**主库在执行完客户端提交的事务后不是立刻返回给客户端，而是等待至少一个从库接收到并写到relay log中才返回给客户端。**相对于异步复制，半同步复制**提高了数据的安全性，同时它也造成了一定程度的延迟，**这个延迟最少是一个TCP/IP往返的时间。所以，半同步复制最好在低延时的网络中使用。

**1.5.2 实现半同步复制**

　　半同步复制是基于特定的模块来实现。不过非常 nice 的一点就是，在我们的 mariadb5.5 以上的版本默认是带有这个模块的。

　　如果我们的 centos7 上安装了 mariadb ，我们就可以进入目录去查看一下：

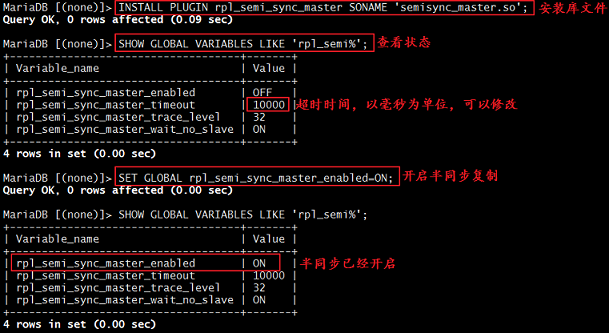
[root@master ~]# cd /usr/lib64/mysql/plugin/



　　找到我们的模块后，我们就可以通过命令来安装，然后进行简单配置即可使用了。下面就是具体方法：

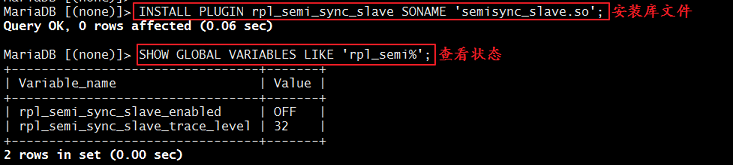
master 服务器上：

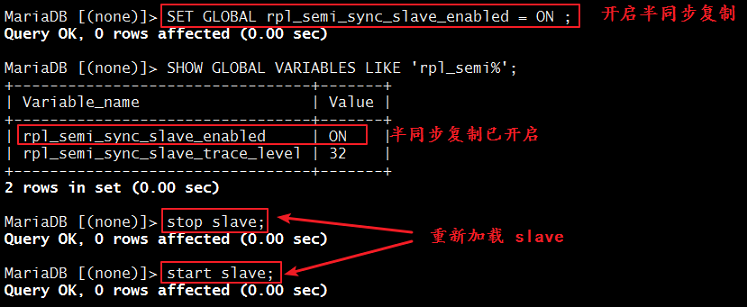
MariaDB [(none)]> INSTALL PLUGIN rpl\_semi\_sync\_master SONAME 'semisync\_master.so';
MariaDB [(none)]> SHOW GLOBAL VARIABLES LIKE 'rpl\_semi%';
MariaDB [(none)]> SET GLOBAL rpl\_semi\_sync\_master\_enabled=ON;



slave 服务器上：

MariaDB [(none)]> INSTALL PLUGIN rpl\_semi\_sync\_slave SONAME 'semisync\_slave.so';
MariaDB [(none)]> SHOW GLOBAL VARIABLES LIKE 'rpl\_semi%';
MariaDB [(none)]> SET GLOBAL rpl\_semi\_sync\_slave\_enabled = ON ;
MariaDB [(none)]> stop slave;
MariaDB [(none)]> start slave;

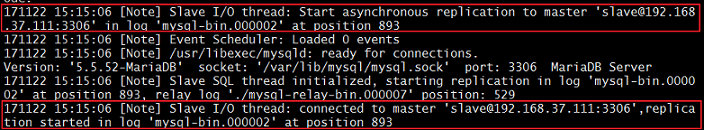




　　现在，我们的半同步复制就开启了，我们可以来查看一下日志：

[root@slave ~]# tail -200 /var/log/mariadb/mariadb.log

　　从日志里也可以看出我们的半同步复制已经开启。



　　以上，实验一完成。

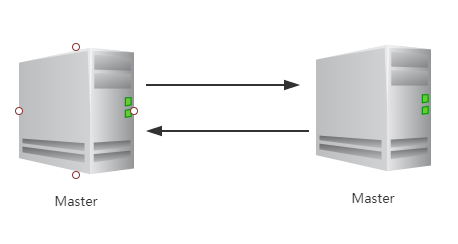
[回到顶部](https://www.cnblogs.com/keerya/p/7873502.html#_labelTop)

**互为主从（双主）**

　　还有一种数据库的架构为双主模式，也就是互为主从，这种设定在企业中虽不常用，但是我们还是需要掌握的。

　　Master-Master复制的两台服务器，既是master，又是另一台服务器的slave。这样，任何一方所做的变更，都会通过复制应用到另外一方的数据库中。

　　架构图如下：



**2.1 环境准备：**

　　centos系统服务器2台、 两台用户实现Mysql互为主从， 配置好yum源、 防火墙关闭、 各节点时钟服务同步、 各节点之间可以通过主机名互相通信

**2.2 准备步骤：**

1）iptables -F && setenforce 清空防火墙策略，关闭selinux

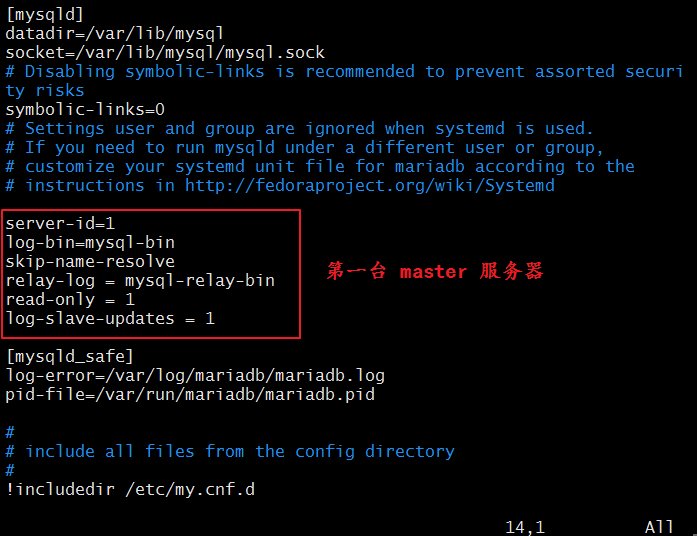
2）拿两台服务器都使用yum方式安装Mysql服务， 要求版本一致

3）分别启动两台服务器mysql服务， 确保服务正常

**2.3 实现步骤：**

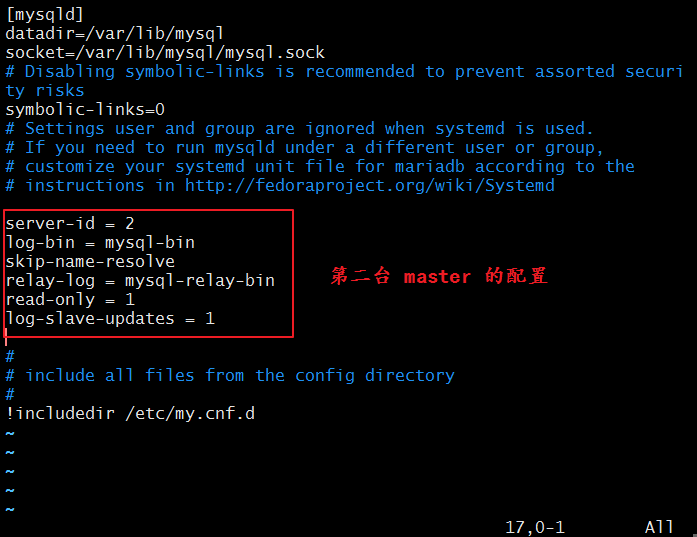
**2.3.1 配置第一台 master 主服务器**

server-id = 1　　　　　 //配置server-id，让主服务器有唯一ID号
log-bin = mysql-bin　　//打开Mysql日志，日志格式为二进制
skip-name-resolve 　　//关闭名称解析，（非必须）
relay\_log = mysql-relay-bin //打开Mysql日志，日志格式为二进制
read\_only = 1 //设置只读权限
log\_slave\_updates = 1 //使得更新的数据写进二进制日志中
auto\_increment\_offset = 1　　　 //表示自增长字段从那个数开始
auto\_increment\_increment = 2　　//表示自增长字段每次递增的量



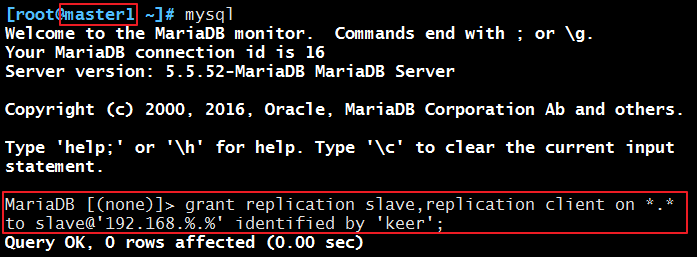
**2.3.2 配置第二台 master 主服务器**

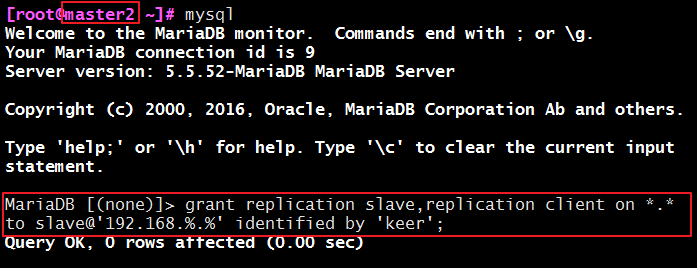
server-id = 2　　　　　 //配置server-id，让主服务器有唯一ID号
log-bin = mysql-bin　　//打开Mysql日志，日志格式为二进制
skip-name-resolve　 　//关闭名称解析，（非必须）
relay\_log = mysql-relay-bin //打开Mysql日志，日志格式为二进制
read\_only = 1 //设置只读权限
log\_slave\_updates = 1 //使得更新的数据写进二进制日志中
auto\_increment\_offset = 2　　　 //表示自增长字段从那个数开始
auto\_increment\_increment = 2　　//表示自增长字段每次递增的量



**2.3.3 创建复制账号**

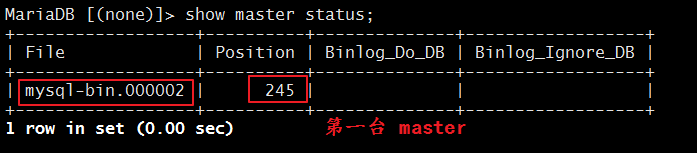
　　因为我们实现的是双主模式，所以我们每一台机器上都要创建复制账号：

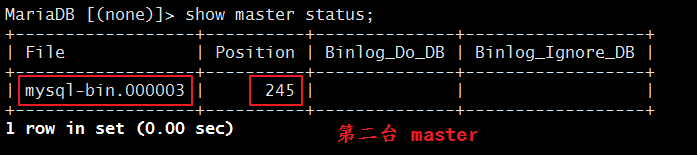




**2.3.4 查看服务器状态**

　　我们分别查看两台服务器的状态，从而为我们下一步的设置奠下基础：





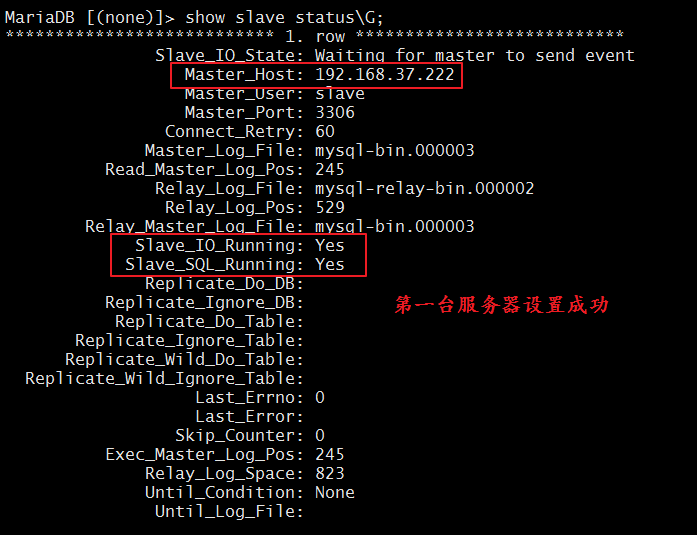
**2.3.5 启动复制线程**

　　我们来分别对两台机器进行的设置：

　　首先，先对第一台 master 机器进行设置：

MariaDB [(none)]> change master to master\_host='192.168.37.222',master\_user='slave',master\_password='keer',master\_log\_file='mysql-bin.000003',master\_log\_pos=245;
MariaDB [(none)]> start slave;

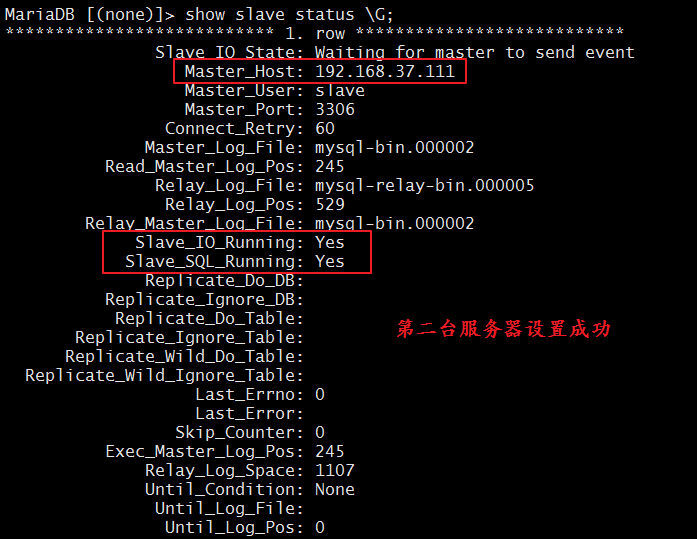
　　接着，我们来查看一下他的状态：



　　现在，我们来对第二台 master 进行设置：

MariaDB [(none)]> change master to master\_host='192.168.37.111',master\_user='slave',master\_password='keer',master\_log\_file='mysql-bin.000002',master\_log\_pos=245;
MariaDB [(none)]> start slave;

　　我们来查看一下他的状态：



　　可以看出，我们的两个机器都设置完毕，接下来，我们就可以来测试一下了。

**2.3.5 测试**

　　在我们的配置文件中，我们设置了自增长字段的起始值和增长值，现在我们去创建一个自增长的表：

MariaDB [keer]> create table test(id int PRIMARY KEY AUTO\_INCREMENT,name char(20) not null);

　　因为我们设置了主从，所以在任一机器使用该命令即可。

　　创建好了表以后，我们就可以插入数据进行测试了，我们一个插入多条数据来看看效果：

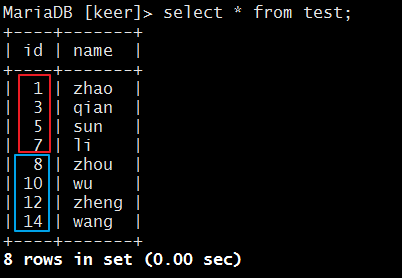
第一个服务器：

MariaDB [keer]> insert into test(name) value('zhao'),('qian'),('sun'),('li');
Query OK, 4 rows affected (0.00 sec)
Records: 4 Duplicates: 0 Warnings: 0

第二个服务器：

MariaDB [keer]> insert into test(name) value('zhou'),('wu'),('zheng'),('wang');
Query OK, 4 rows affected (0.00 sec)
Records: 4 Duplicates: 0 Warnings: 0

　　数据我们都插入了，我们来看看效果如何：



　　可以看出，两个表插入的数据是不会重复的。

　　但是现在也不是我们想象的连续的 id ，这是因为，我们两台机器都是同时插入4条数据，而且不是同步插入的，这样就会有一个先后顺序，第一台机器上的 sql 语句执行结束以后才执行第二台机器上的。所以会出现这样的情况。

　　但是，这样已经不会插入重复的数据了。我们的目的也是达到了。