**MySQL配置手册**

目录

[1 MySQL复制 4](#_Toc480143745)

[1.1 概述 4](#_Toc480143746)

[1.1.1 基本介绍 4](#_Toc480143747)

[1.1.2 原理 4](#_Toc480143748)

[1.2 实现 6](#_Toc480143749)

[1.2.1 linux服务器 6](#_Toc480143750)

[1.2.2 Windows服务器 75](#_Toc480143751)

[2 MySQL Cluster 75](#_Toc480143752)

[2.1 概述 75](#_Toc480143753)

[3 MySQL监控 75](#_Toc480143754)

# MySQL复制

## 概述

### 基本介绍

目前MySQL数据库已经占去数据库市场上很大的份额，其一是由于MySQL数据的开源性和高性能，当然还有重要的一条就是免费，不过不知道还能免费多久，不容乐观的未来，我们需要熟练掌握MySQL数据的架构和安全备份等功能。

**MySQL数据库支持同步复制、单向复制、异步复制**，在复制的过程中一个服务器充当主服务，而一个或多个服务器充当从服务器。主服务器将更新写入二进制日志文件，并维护文件的一个索引以跟踪日志循环。这些日志可以记录发送到从服务器的更新。当一个从服务器连接主服务器时，它通知主服务器从服务器在日志中读取的最后一次成功更新的位置。从服务器接收从那时起发生的任何更新，然后封锁并等待主服务器通知新的更新。

**请注意当你进行复制时，所有对复制中的表的更新必须在主服务器上进行。否则，你必须要小心，以避免用户对主服务器上的表进行的更新与对从服务器上的表所进行的更新之间的冲突**。

单向复制有利于健壮性、速度和系统管理：

1.健壮性：主服务器/从服务器设置增加了健壮性。主服务器出现问题时，你可以切换到从服务器作为备份。

2.速度快：通过在主服务器和从服务器之间切分处理客户查询的负荷，可以得到更好的客户响应时间。SELECT查询可以发送到从服务器以降低主服务器的查询处理负荷。但修改数据的语句仍然应发送到主服务器，以便主服务器和从服务器保持同步。如果非更新查询为主，该负载均衡策略很有效，但一般是更新查询。

3.系统管理：使用复制的另一个好处是可以使用一个从服务器执行备份，而不会干扰主服务器。在备份过程中主服务器可以继续处理更新。

### 原理

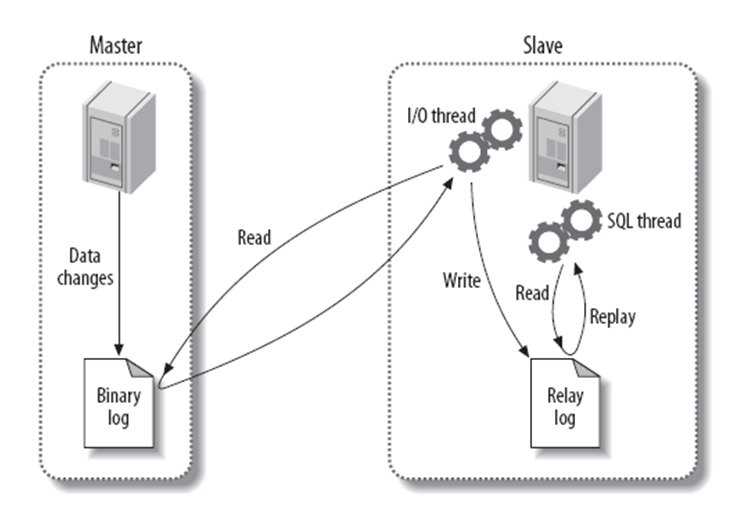
M**ySQL复制基于主服务器在二进制日志中跟踪所有对数据库的更改(更新、删除等等)。因此，要进行复制，必须在主服务器上启用二进制日志**。

每个从服务器从主服务器接收主服务器已经记录到其二进制日志的保存的更新，以便从服务器可以对其数据拷贝执行相同的更新。

认识到二进制日志只是一个从启用二进制日志的固定时间点开始的记录非常重要。任何设置的从服务器需要主服务器上的在主服务器上启用二进制日志时的数据库拷贝。如果启动从服务器时，其数据库与主服务器上的启动二进制日志时的状态不相同，从服务器很可能失败。

将主服务器的数据拷贝到从服务器的一个途径是使用LOAD DATA FROM MASTER语句。请注意LOAD DATA FROM MASTER目前只在所有表使用MyISAM存储引擎的主服务器上工作。并且，该语句将获得全局读锁定，因此当表正复制到从服务器上时，不可能在主服务器上进行更新。当我们执行表的无锁热备份时，则不再需要全局读锁定。

MySQL数据复制的原理图大致如下：



从上图我们可以看出MySQL数据库的复制需要启动三个线程来实现：

其中1个在主服务器上，另两个在从服务器上。当发出START SLAVE时，从服务器创建一个I/O线程，以连接主服务器并让它发送记录在其二进制日志中的语句。主服务器创建一个线程将二进制日志中的内容发送到从服务器。该线程可以识别为主服务器上SHOW PROCESSLIST的输出中的Binlog Dump线程。从服务器I/O线程读取主服务器Binlog Dump线程发送的内容并将该数据拷贝到从服务器数据目录中的本地文件中，即中继日志。第3个线程是SQL线程，是从服务器创建用于读取中继日志并执行日志中包含的更新。

在前面的描述中，每个从服务器有3个线程。有多个从服务器的主服务器创建为每个当前连接的从服务器创建一个线程；每个从服务器有自己的I/O和SQL线程。

这样读取和执行语句被分成两个独立的任务。如果语句执行较慢则语句读取任务没有慢下来。例如，如果从服务器有一段时间没有运行了，当从服务器启动时，其I/O线程可以很快地从主服务器索取所有二进制日志内容，即使SQL线程远远滞后。如果从服务器在SQL线程执行完所有索取的语句前停止，I/O 线程至少已经索取了所有内容，以便语句的安全拷贝保存到本地从服务器的中继日志中，供从服务器下次启动时执行。这样允许清空主服务器上的二进制日志，因为不再需要等候从服务器来索取其内容。

## 实现

### linux服务器

#### 环境准备

本文Linux安装环境为Centos7.2 64位系统。

本文适用与Centos7以上64位系统，不适用于32位。

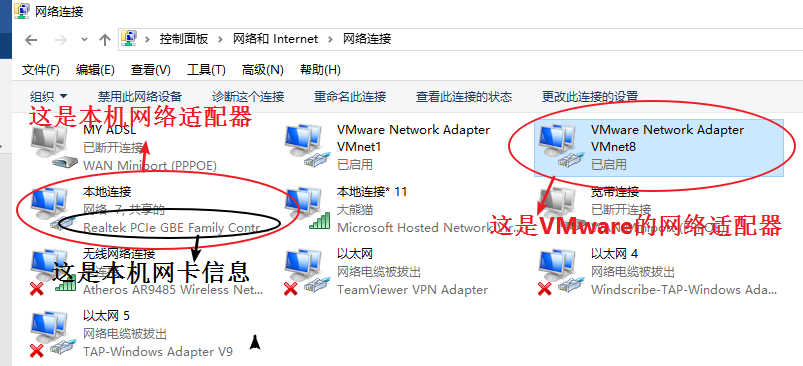
MySQL版本：mysql Ver 14.14 Distrib 5.7.17, for Linux (x86\_64)

MySQL安装步骤见MySQL安装手册.docx

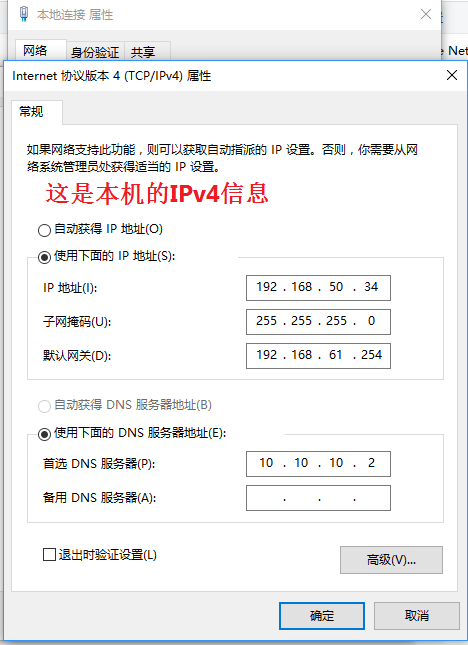
以下步骤为配置VMware虚拟机配置静态IP，若是物理机配置MySQL主从同步可以跳过这步

##### Windows

打开控制面板-网络和Internet-网络连接找到

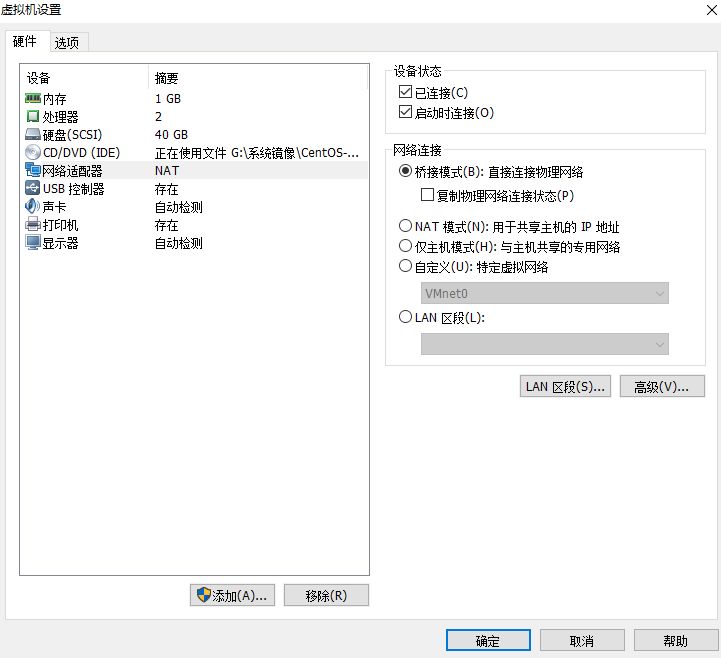


选择本机网络适配器，查看IPv4信息，后面配置需要。

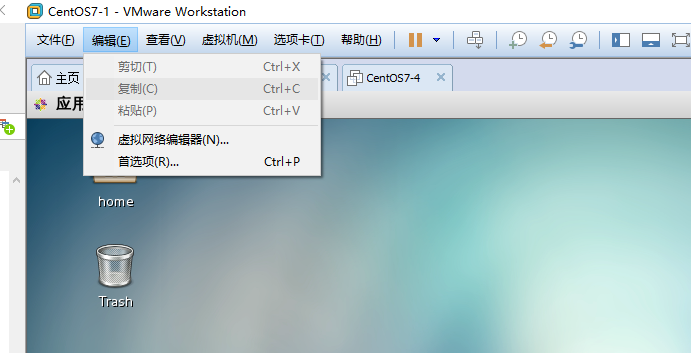


##### Vmware

在虚拟机设置里面设置网络适配器为桥接模式



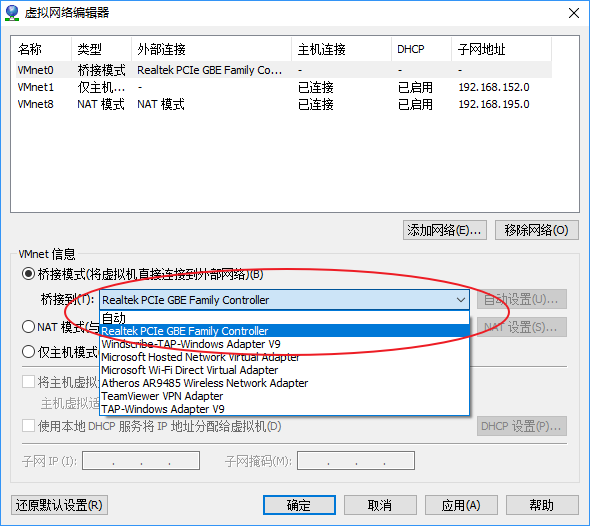
在Vmware上面菜单栏中找到编辑-虚拟网络编辑器，打开配置





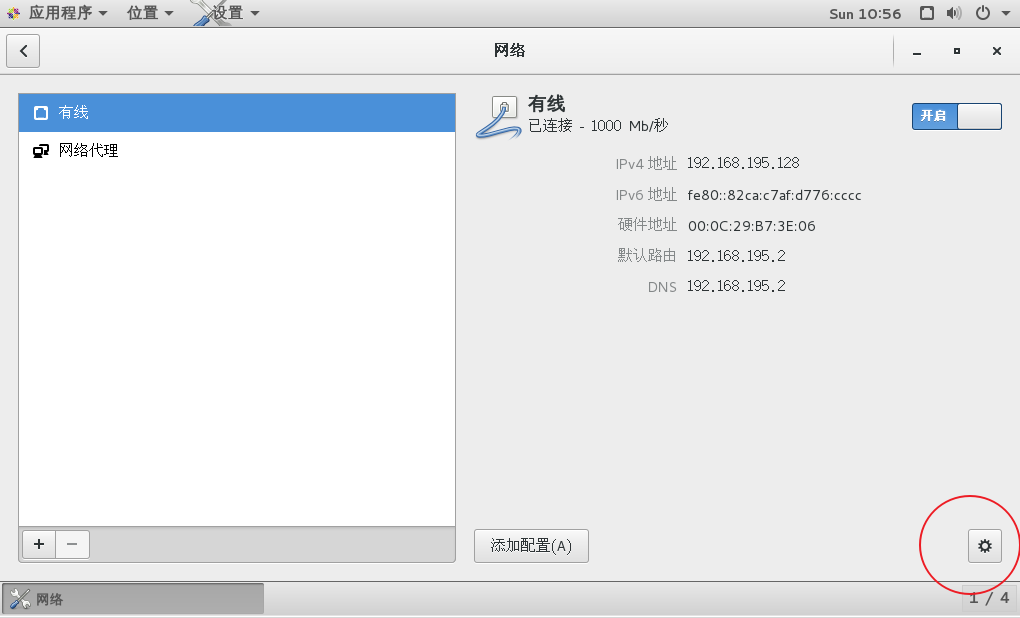
选择更改设置

选中桥接模式，将桥接到这里的网卡信息改成本机网卡Realtek PCIe GBE Family Controller

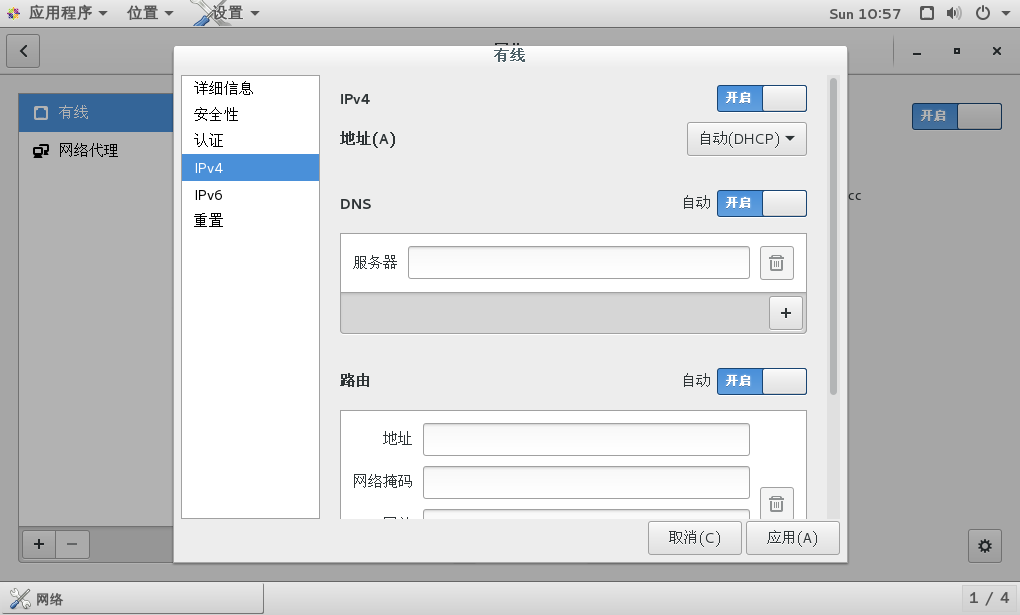


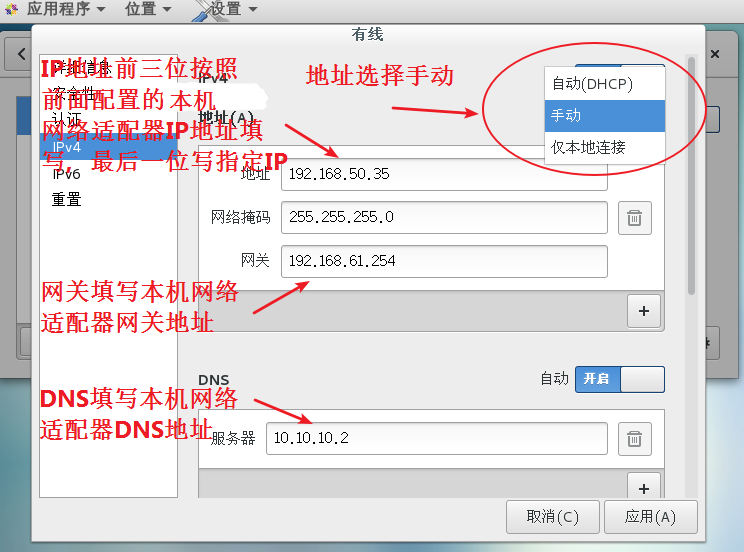
##### CentOS7

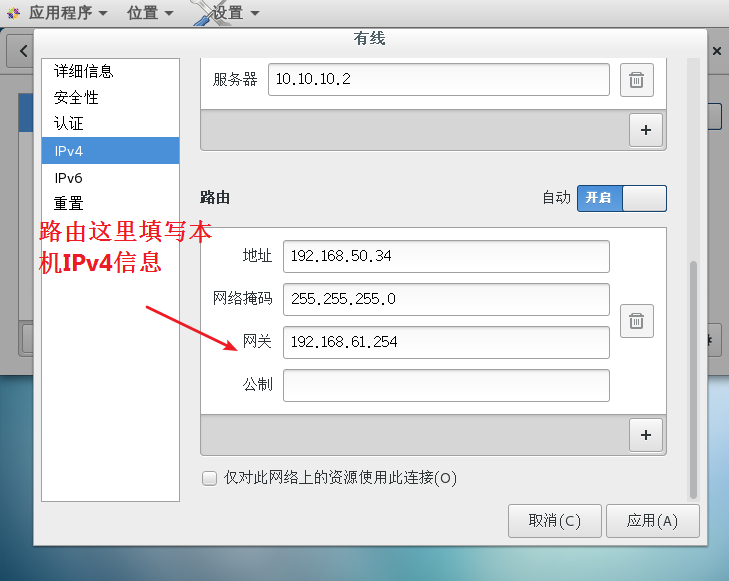
在CentOS7里面打开左上角应用程序，找到系统工具-设置-网络-有线，点击右下角设置按钮



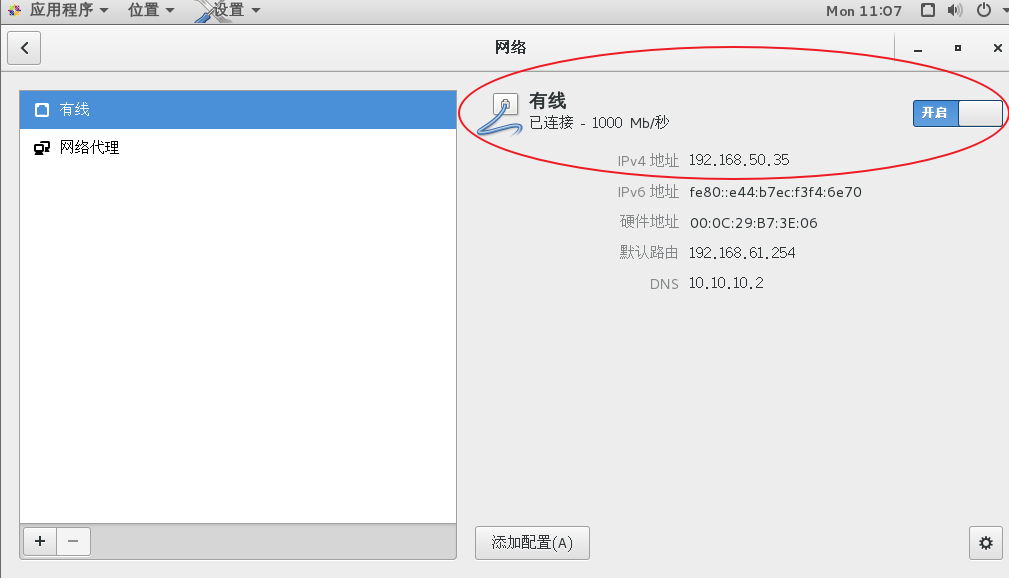
选择Ipv4







填写完成点击应用，并重新开启有线连接

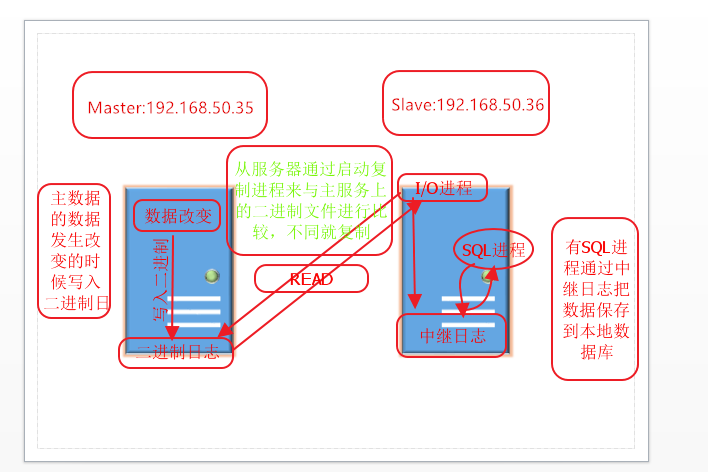


这时候就配置好静态IP并且可以正常访问网络了，从服务器上按照上述步骤将IP地址

设置为192.168.50.36，配置好后记得在Xshell和Navicat for MySQL上更改下IP信息

#### 单主单从

##### 架构图



主服务器Master ：192.168.50.35

从服务器Slave ：192.168.50.36

##### 主配置

###### 配置文件

编辑MySQL配置文件

vim /etc/my.cnf



symbolic-links=0 配置符号链接,0是关闭,1是开启

在最后一排添加以下内容：

#数据库服务ID,必须唯一,通常设置为本机IP地址最后一位

server-id=35

#主服务器二进制日志文件前缀名,用来查看数据库的变更历史、数据库增量备份和恢复、MySQL的复制（主从数据库的复制）

log-bin=master-bin

#主服务器二进制日志文件binlog索引名

log-bin-index=master-bin.index

#binlog日志文件格式,MySQL5.7默认采用ROW

binlog\_format=ROW

#ROW模式该值默认为full，表示记录每一行的变更，包含所有列，会带来磁盘IO上的开销，同时由于binlog日志变大，网络开销也变大，改为minimal后只记录影响后的行

binlog\_row\_image=minimal

#binlog日志文件过期清理时间

expire\_logs\_days=7

#binlog缓存大小

binlog\_cache\_size=4M

#最大binlog缓存大小

max\_binlog\_cache\_size=512M

#单个binlog文件最大值, 最大和默认值是1GB，该设置并不能严格控制binlog的大小，尤其是binlog比较靠近最大值而又遇到一个比较大事务时，为了保证事务的完整性，不可能做切换日志的动作，只能将该事务的所有SQL都记录进当前日志，直到事务结束

max\_binlog\_size=512M

#设置为1,标志支持分布式事物,支持内部xa事务也就是主要保证支持binlog与innodb redo log之间数据一致性保持一致性,属于同步操作

innodb\_support\_xa=1

#binlog-do-db=需要同步的数据库名字,如果是多个数据库,就此格式再写多行

#binlog-ignore-db=不需要同步的数据库名字,如果是多个数据库,就此格式再写多行

#如果binlog-do-db和binlog-ignore-db不加的话，那么默认是同步复制整个MySQL数据库

#推荐设置--高性能：每次事务提交时MySQL都会把log buffer的数据写入log file,但是flush(刷到磁盘)操作并不会同时进行,在该模式下MySQL会每秒执行一次 flush(刷到磁盘)操作;更安全：只有在操作系统崩溃或者系统掉电的情况下,上一秒钟所有事务数据才可能丢失

innodb\_flush\_log\_at\_trx\_commit=2

sync\_binlog=1000

#禁止MySQL对连接的客户端进行DNS反向解析,只能允许IP访问,加快远程连接速度

skip-name-resolve

#设置默认字符集为UTF-8

character\_set\_server=utf8

#关闭密码策略插件

validate\_password=off

#设置MySQL大小写不敏感

lower\_case\_table\_names=1

#关闭SSL连接

disable\_ssl

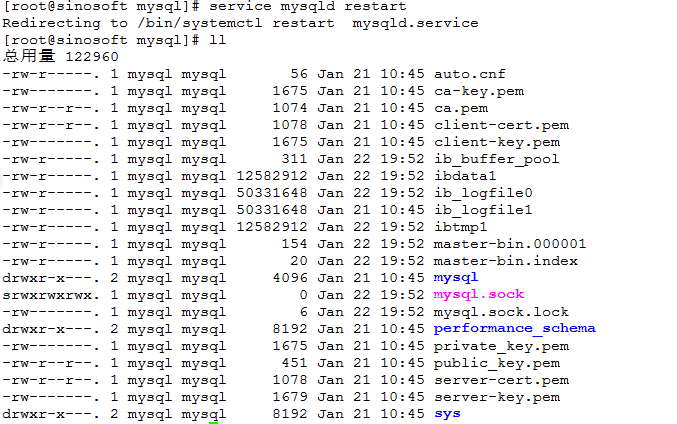
粘贴完成后将每行前面第一格多余的#号去掉，保持与上文格式一样

一些具体配置参数解释见[配置注解](#_配置注解)

保存退出并重启mysql服务

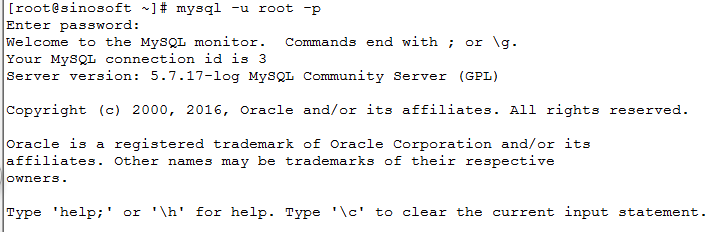
service mysqld restart

配置完这些后重新启动MySQL服务，会在在数据目录/var/lib/mysql下多出了两个文件： master-bin.000001，master-bin.index，每次重启Mysql服务都会生成新的master-bin.xxxxxx文件，这两个文件在主从同步中起作用。



###### 授权账号

用root用户登录主服务器master的MySQL数据库



开始创建同步账号并授权在从服务器slave上登录访问

创建用户testdb密码123456指定在192.168.50.36主机上可以登录

create user 'testdb'@'192.168.50.36' identified by '123456';

移除testdb这个用户在192.168.50.36主机上的所有权限

revoke all privileges ,grant option from 'testdb'@'192.168.50.36';

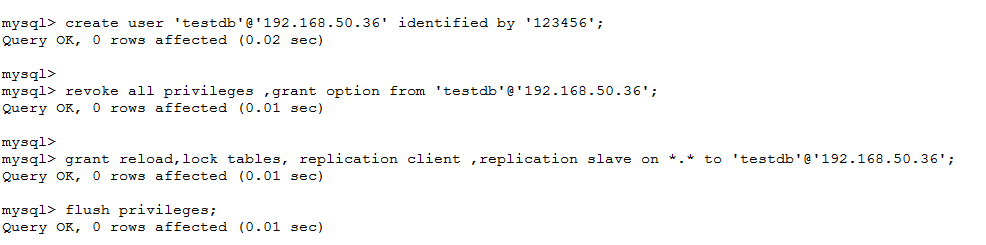
授权重载，锁表，复制客户端，复制从属权限给testdb用户在192.168.50.36主机的所有数据库里

grant reload,lock tables, replication client ,replication slave on \*.\* to 'testdb'@'192.168.50.36';

这个用户权限配置管理grant和revoke的语法详情见[用户权限配置管理](#_用户权限配置管理)

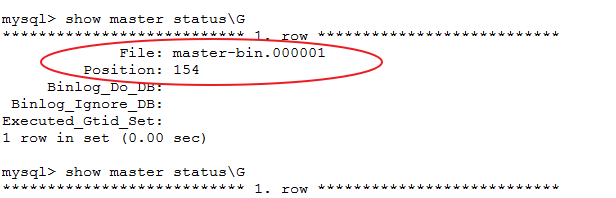
创建完后刷新同步账号的权限

flush privileges;



查看主服务器MySQL的状态

show master status\G



记住file和position的值，待会同步要用，锁定主数据库DDL操作，避免Position值发生变化

flush tables with read lock;



###### 参数意义

innodb\_flush\_log\_at\_trx\_commit

innodb\_flush\_log\_at\_trx\_commit和sync\_binlog 这两个参数是控制MySQL 磁盘写入策略以及数据安全性的关键参数。本文从参数含义、性能、安全角度阐述两个参数为不同的值时对db 性能、数据的影响。

0，log buffer将每秒一次地写入log file中，并且log file的flush(刷到磁盘)操作同时进行.该模式下，在事务提交的时候，不会主动触发写入磁盘的操作。

1，每次事务提交时MySQL都会把log buffer的数据写入log file，并且flush(刷到磁盘)中去。

2，每次事务提交时MySQL都会把log buffer的数据写入log file.但是flush(刷到磁盘)操作并不会同时进行。该模式下,MySQL会每秒执行一次 flush(刷到磁盘)操作。

注意：

由于进程调度策略问题,这个“每秒执行一次 flush(刷到磁盘)操作”并不是保证100%的“每秒”。

sync\_binlog

默认值是0，像操作系统刷其他文件的机制一样，MySQL不会同步到磁盘中去而是依赖操作系统来刷新binary log。

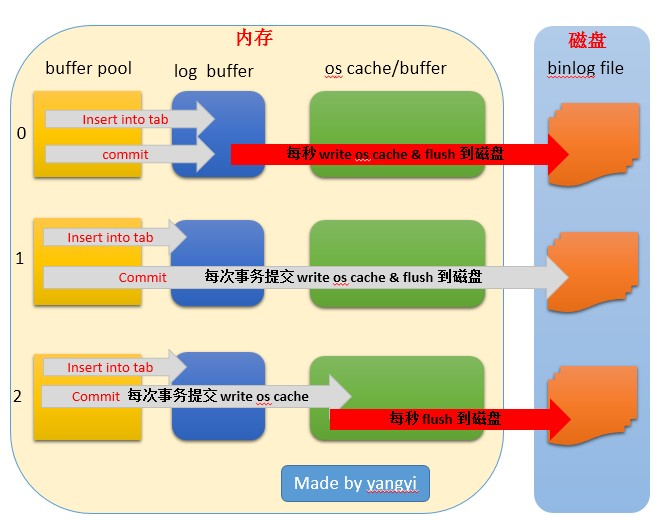
当sync\_binlog =N (N>0) ，MySQL 在每写 N次 二进制日志binary log时，会使用fdatasync()函数将它的写二进制日志binary log同步到磁盘中去。

注:

   如果启用了autocommit，那么每一个语句statement就会有一次写操作；否则每个事务对应一个写操作。

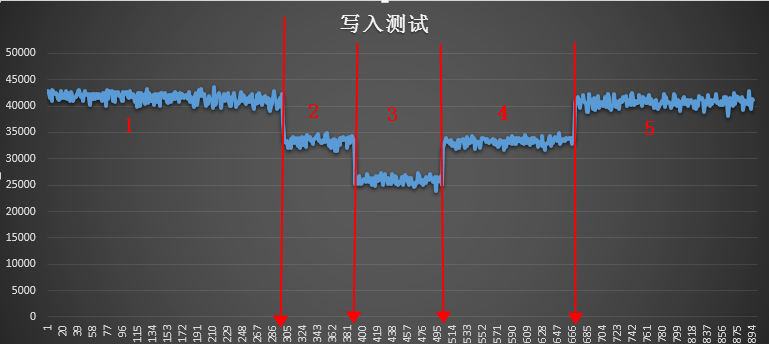
参数设置测试

根据上述描述，做了一张图，可以方便大家查看。



**性能**

两个参数在不同值时对db的纯写入的影响表现如下：



 测试场景1

innodb\_flush\_log\_at\_trx\_commit=2

sync\_binlog=1000

 测试场景2

innodb\_flush\_log\_at\_trx\_commit=1

sync\_binlog=1000

 测试场景3

innodb\_flush\_log\_at\_trx\_commit=1

sync\_binlog=1000

 测试场景4

innodb\_flush\_log\_at\_trx\_commit=1

sync\_binlog=1000

 测试场景5

innodb\_flush\_log\_at\_trx\_commit=2

sync\_binlog=1000

|  |  |
| --- | --- |
| 场景 | TPS |
| 场景1 | 41000 |
| 场景2 | 33000 |
| 场景3 | 26000 |
| 场景4 | 33000 |

**由此可见，当两个参数设置为双1的时候，写入性能最差，sync\_binlog=N (N>1 ) innodb\_flush\_log\_at\_trx\_commit=2 时，(在当前模式下)MySQL的写操作才能达到最高性能。**

**安全**

当innodb\_flush\_log\_at\_trx\_commit和sync\_binlog  都为 1 时是最安全的，在mysqld 服务崩溃或者服务器主机crash的情况下，binary log 只有可能丢失最多一个语句或者一个事务。但是鱼与熊掌不可兼得，双1 会导致频繁的io操作，因此该模式也是最慢的一种方式。

当innodb\_flush\_log\_at\_trx\_commit设置为0，mysqld进程的崩溃会导致上一秒钟所有事务数据的丢失。

当innodb\_flush\_log\_at\_trx\_commit设置为2，只有在操作系统崩溃或者系统掉电的情况下，上一秒钟所有事务数据才可能丢失。

**双1适合数据安全性要求非常高，而且磁盘IO写能力足够支持业务，比如订单,交易,充值,支付消费系统。双1模式下，当磁盘IO无法满足业务需求时 比如11.11 活动的压力。推荐的做法是 innodb\_flush\_log\_at\_trx\_commit=2 ，sync\_binlog=N (N为500 或1000) 且使用带蓄电池后备电源的缓存cache，防止系统断电异常。**

系统性能和数据安全是业务系统高可用稳定的必要因素。我们对系统的优化需要寻找一个平衡点，合适的才是最好的，根据不同的业务场景需求，可以将两个参数做组合调整，以便是db系统的性能达到最优化。

**本文地址：**<http://blog.itpub.net/22664653/viewspace-1063134>

binlog\_format

**MySQL复制方式**

**MySQL复制主要有三种方式：基于SQL语句的复制(statement-based replication, SBR)，基于行的复制(row-based replication, RBR)，混合模式复制(mixed-based replication, MBR)。对应的，binlog的格式也有三种：STATEMENT，ROW，MIXED。**

**STATEMENT模式（SBR）**

每一条会修改数据的sql语句会记录到binlog中。优点是并不需要记录每一条sql语句和每一行的数据变化，减少了binlog日志量，节约IO，提高性能。缺点是在某些情况下会导致master-slave中的数据不一致(如sleep()函数， last\_insert\_id()，以及user-defined functions(udf)等会出现问题)

**ROW模式（RBR）**

不记录每条sql语句的上下文信息，仅需记录哪条数据被修改了，修改成什么样了。而且不会出现某些特定情况下的存储过程、或function、或trigger的调用和触发无法被正确复制的问题。缺点是会产生大量的日志，尤其是alter table的时候会让日志暴涨。

**MIXED模式（MBR）**

以上两种模式的混合使用，一般的复制使用STATEMENT模式保存binlog，对于STATEMENT模式无法复制的操作使用ROW模式保存binlog，MySQL会根据执行的SQL语句选择日志保存方式。

**binlog复制配置**

在mysql的配置文件my.cnf中，可以通过一下选项配置binglog相关

|  |  |
| --- | --- |
| **代码如下** | 复制代码 |
| binlog\_format=MIXED //binlog日志格式,mysql默认采用statement,建议使用mixed log-bin=/data/mysql/mysql-bin.log   //binlog日志文件 expire\_logs\_days=7                           //binlog过期清理时间 max\_binlog\_size=100m                    //binlog每个日志文件大小 binlog\_cache\_size= 4m                    //binlog缓存大小 max\_binlog\_cache\_size= 512m         //最大binlog缓存大小 | |

**复制方式优缺点**

**MIXED说明**

对于执行的SQL语句中包含now()这样的时间函数，会在日志中产生对应的unix\_timestamp()\*1000的时间字符串，slave在完成同步时，取用的是sqlEvent发生的时间来保证数据的准确性。另外对于一些功能性函数slave能完成相应的数据同步，而对于上面指定的一些类似于UDF函数，导致Slave无法知晓的情况，则会采用ROW格式存储这些Binlog，以保证产生的Binlog可以供Slave完成数据同步。

现在来比较以下 SBR 和 RBR 2中模式各自的优缺点：

**SBR 的优点：**

历史悠久，技术成熟  
binlog文件较小  
binlog中包含了所有[数据库](http://www.111cn.net/database/database.html" \t "_blank)更改信息，可以据此来审核数据库的安全等情况  
binlog可以用于实时的还原，而不仅仅用于复制  
主从版本可以不一样，从服务器版本可以比主服务器版本高

**SBR 的缺点：**

不是所有的UPDATE语句都能被复制，尤其是包含不确定操作的时候。  
调用具有不确定因素的 UDF 时复制也可能出问题  
使用以下函数的语句也无法被复制：

**\* LOAD\_FILE()  
\* UUID()  
\* USER()  
\* FOUND\_ROWS()  
\* SYSDATE() (除非启动时启用了 --sysdate-is-now 选项)  
 INSERT ... SELECT 会产生比 RBR 更多的行级锁**复制需要进行全表扫描(WHERE 语句中没有使用到索引)的 UPDATE 时，需要比 RBR 请求更多的行级锁  
对于有 AUTO\_INCREMENT 字段的 InnoDB表而言，INSERT 语句会阻塞其他 INSERT 语句  
对于一些复杂的语句，在从服务器上的耗资源情况会更严重，而 RBR 模式下，只会对那个发生变化的记录产生影响  
存储函数(不是存储过程)在被调用的同时也会执行一次 NOW() 函数，这个可以说是坏事也可能是好事  
确定了的 UDF 也需要在从服务器上执行  
数据表必须几乎和主服务器保持一致才行，否则可能会导致复制出错  
执行复杂语句如果出错的话，会消耗更多资源

**RBR 的优点：**

任何情况都可以被复制，这对复制来说是最安全可靠的  
和其他大多数数据库系统的复制技术一样  
多数情况下，从服务器上的表如果有主键的话，复制就会快了很多  
复制以下几种语句时的行锁更少：  
**\* INSERT ... SELECT  
\* 包含 AUTO\_INCREMENT 字段的 INSERT  
\* 没有附带条件或者并没有修改很多记录的 UPDATE 或 DELETE 语句  
执行 INSERT，UPDATE，DELETE 语句时锁更少**从服务器上采用多线程来执行复制成为可能

**RBR 的缺点：**

binlog 大了很多  
复杂的回滚时 binlog 中会包含大量的数据  
主服务器上执行 UPDATE 语句时，所有发生变化的记录都会写到 binlog 中，而 SBR 只会写一次，这会导致频繁发生 binlog 的并发写问题  
UDF 产生的大 BLOB 值会导致复制变慢  
无法从 binlog 中看到都复制了写什么语句  
当在非事务表上执行一段堆积的SQL语句时，最好采用 SBR 模式，否则很容易导致主从服务器的数据不一致情况发生

**另外，针对系统库 mysql 里面的表发生变化时的处理规则如下：  
如果是采用 INSERT，UPDATE，DELETE 直接操作表的情况，则日志格式根据 binlog\_format 的设定而记录  
如果是采用 GRANT，REVOKE，SET PASSWORD 等管理语句来做的话，那么无论如何都采用 SBR 模式记录  
注：采用 RBR 模式后，能解决很多原先出现的主键重复问题。**

**本文地址：**<http://www.cnblogs.com/langtianya/p/5504774.html>

授权配置介绍

**本文地址：**<http://blog.useasp.net/archive/2013/06/19/mysql-user-privileges-configuration-and-management-by-show-grant-and-revoke-syntax.aspx>

<http://tec.5lulu.com/detail/104k3n1e6z27g8d40.html>

##### 从配置

###### 配置文件

编辑MySQL配置文件

vim /etc/my.cnf



symbolic-links=0 配置符号链接,0是关闭,1是开启

在最后一排添加以下内容：

#数据库服务ID,必须唯一,通常设置为本机IP地址最后一位

server-id=36

#中继日志二进制日志文件前缀名

relay-log=slave-relay-bin

#中继日志二进制日志文件relay\_log索引名

relay-log-index=slave-relay-bin.index

#开启lay\_log记录主库同步的操作日志

log\_slave\_updates=1

#设定从库为只读模式,可以限定普通用户进行数据修改的操作,但不会限定具有super权限的用户如root的数据修改操作;在MySQL中设置read\_only=1后,普通的应用用户进行insert、update、delete等会产生数据变化的DML操作时,都会报出数据库处于只读模式不能发生数据变化的错误,但具有super权限的用户,例如在本地或远程通过root用户登录到数据库,还是可以进行数据变化的DML操作

read\_only=1

#禁止MySQL对连接的客户端进行DNS反向解析,只能允许IP访问,加快远程连接速度

skip-name-resolve

#设置默认字符集为UTF-8

character\_set\_server=utf8

#关闭密码策略插件

validate\_password=off

#设置MySQL大小写不敏感

lower\_case\_table\_names=1

#关闭SSL连接

disable\_ssl

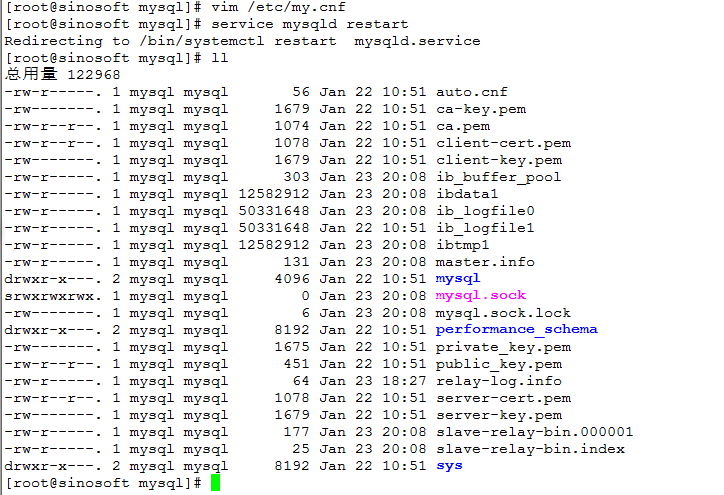
粘贴完成后将每行前面第一格多余的#号去掉，保持与上文格式一样

一些具体配置参数解释见[配置注解](#_配置注解_1)

保存退出并重启mysql服务

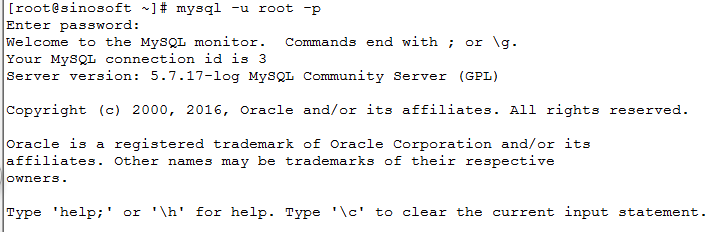
service mysqld restart

配置完这些后重新启动MySQL服务，会在在数据目录/var/lib/mysql下多出了两个文件： slave-relay-bin.000001，slave-relay-bin.index，每次重启Mysql服务都会生成新的slave-relay-bin.xxxxxx文件，这两个文件在主从同步中起作用。



###### 指定主从

用root用户登录从服务器slave的MySQL数据库



停止中继日志

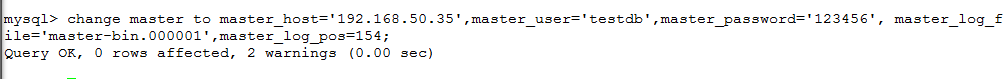
stop slave;



使用 CHANGE MASTER TO 命令将从库指向主库

语法：change master to master\_host='主服务器master的IP',master\_user='前面配置好的同步账号',master\_password='同步账号的密码', master\_log\_file='前面查出来主服务器master状态的File值',master\_log\_pos=前面查出来主服务器master状态的Position值;

change master to master\_host='192.168.50.35',master\_user='testdb',master\_password='123456', master\_log\_file='master-bin.000001',master\_log\_pos=154;



启动中继日志

start slave;



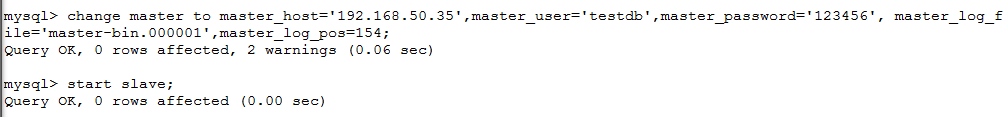
若报下面错



输入reset slave;



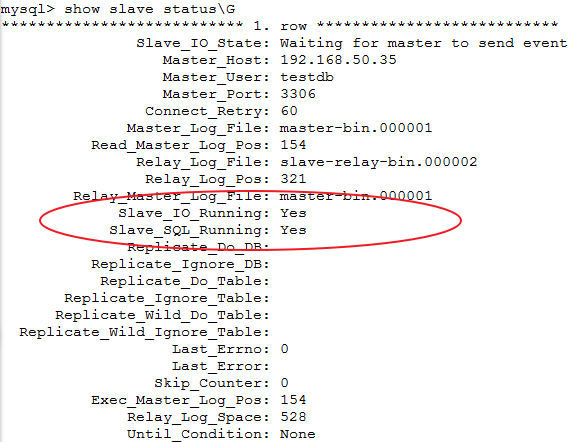
再重新使用 CHANGE MASTER TO 命令将从库指向主库，并启动从服务器slave



###### 查看状态

查看slave状态

show slave status\G

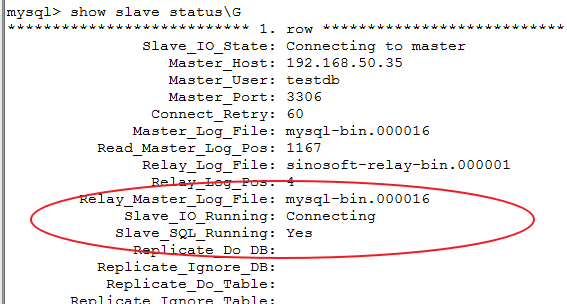


若这里Slave\_IO\_Running和Slave\_IO\_Running的状态都是yes，证明配置成功

导致slave\_IO\_Running 为connecting 的原因主要有以下 3 个方面：

1、网络不通  
2、密码不对  
3、pos不对

出现这种问题的原一般先检查写入是字母是否正确，其次去检查网络原因。



Slave\_IO\_Running：连接到主库，并读取主库的日志到本地，生成本地日志文件

Slave\_SQL\_Running:读取本地日志文件，并执行日志里的SQL命令。

###### 参数意义

read\_only

在MySQL数据库中，在进行数据迁移和从库只读状态设置时，都会涉及到只读状态和Master-slave的设置和关系。

经过实际测试，对于MySQL单实例数据库和master库，如果需要设置为只读状态，需要进行如下操作和设置：

将MySQL设置为只读状态的命令：

# mysql -uroot -p

mysql> show global variables like "%read\_only%";

mysql> flush tables with read lock;

mysql> set global read\_only=1;

mysql> show global variables like "%read\_only%";

将MySQL从只读设置为读写状态的命令：

mysql> unlock tables;

mysql> set global read\_only=0;

对于需要保证master-slave主从同步的salve库，如果要设置为只读状态，需要执行的命令为：

mysql> set global read\_only=1;

将salve库从只读状态变为读写状态，需要执行的命令是：

mysql> set global read\_only=0;

      对于数据库读写状态，主要靠 “read\_only”全局参数来设定；默认情况下，数据库是用于读写操作的，所以read\_only参数也是0或faluse状态，这时候不论是本地用户还是远程访问数据库的用户，都可以进行读写操作；如需设置为只读状态，将该read\_only参数设置为1或TRUE状态，但设置 read\_only=1 状态有两个需要注意的地方：

read\_only=1只读模式，不会影响slave同步复制的功能，所以在MySQL slave库中设定了read\_only=1后，通过 show slave status\G 命令查看salve状态，可以看到salve仍然会读取master上的日志，并且在slave库中应用日志，保证主从数据库同步一致；

可以限定普通用户进行数据修改的操作，但不会限定具有super权限的用户的数据修改操作；在MySQL中设置read\_only=1后，普通的应用用户进行insert、update、delete等会产生数据变化的DML操作时，都会报出数据库处于只读模式不能发生数据变化的错误，但具有super权限的用户，例如在本地或远程通过root用户登录到数据库，还是可以进行数据变化的DML操作；

**锁表：**

      为了确保所有用户，包括具有super权限的用户也不能进行读写操作，就需要执行给所有的表加读锁的命令 “flush tables with read lock;”，这样使用具有super权限的用户登录数据库，想要发生数据变化的操作时，也会提示表被锁定不能修改的报错。

        这样通过 设置“read\_only=1”和“flush tables with read lock;”两条命令，就可以确保数据库处于只读模式，不会发生任何数据改变，在MySQL进行数据库迁移时，限定master主库不能有任何数据变化，就可以通过这种方式来设定。

       但同时由于加表锁的命令对数据库表限定非常严格，如果再slave从库上执行这个命令后，slave库可以从master读取binlog日志，但不能够应用日志，slave库不能发生数据改变，当然也不能够实现主从同步了，这时如果使用 “unlock tables;”解除全局的表读锁，slave就会应用从master读取到的binlog日志，继续保证主从库数据库一致同步。

       为了保证主从同步可以一直进行，在slave库上要保证具有super权限的root等用户只能在本地登录，不会发生数据变化，其他远程连接的应用用户只按需分配为select,insert,update,delete等权限，保证没有super权限，则只需要将salve设定“read\_only=1”模式，即可保证主从同步，又可以实现从库只读。

       相对的，设定“read\_only=1”只读模式开启的解锁命令为设定“read\_only=0”;设定全局锁“flush tables with read lock;”，对应的解锁模式命令为：“unlock tables;”.

      当然设定了read\_only=1后，所有的select查询操作都是可以正常进行的

**本文地址：**<http://www.cnblogs.com/wlmq/p/6084880.html>

log\_slave\_updates

说明：最近部署了MySQL的集群环境，详细如下M01和M02为主主复制，M01和R01为主从复制；在测试的过程中发现了以下问题：

1、M01和M02的主主复制是没有问题的（从M01写入数据能同步到M02，从M02写入数据能够同步到M01);

2、主从同步的时候，当从M01写入的时候，数据可以写入到R01；

3、当从M02写入的时候，数据就不能写入到R01；

问题的原因：log\_slave\_updates参数的状态为NO

MySQL的官网说明如下：

a) M01同步从M02同步数据过来的时候，log\_slave\_updates参数用来控制M01是否把所有的操作写入到binary log，默认的情况下mysql是关闭的;

b) R01数据的更新需要通过读取到M01的binary log才能进行更新，这个时候M01是没有写binary log的，所以当数据从M02写入的时候，R01也就没有更新了。

问题的解决方法：

log\_slave\_updates：默认值为OFF; Dynamic Variable：NO 处理方法：修改/etc/my.cnf，增加一行log\_slave\_updates=1，重启数据库后就可以了； 总结：设置完该参数后，数据库的架构就可以设置成M01和M02为主主同步，R01通过M01进行主从同步； 应用的写操作中M02上面进行，读操作中R01上面进行（如果读操作很多的话，可以在M01上面架设多台只读数据库），当M02发生故障后，系统的写操作自动迁移到M01上面。这种架构基本可以保证大部分公司的应用需求；

**本文地址：**<http://www.linuxidc.com/Linux/2014-11/109029.htm>

##### 测试同步

此时配置Mysql单主多从成功，需要解锁表，才能对主数据库DDL操作

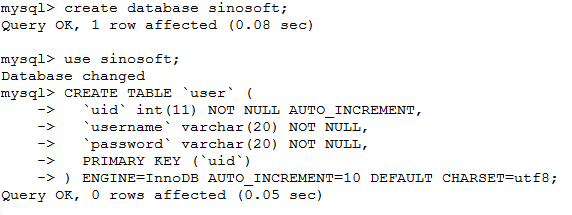
登录主库的mysql命令窗口，执行解锁表命令

unlock tables

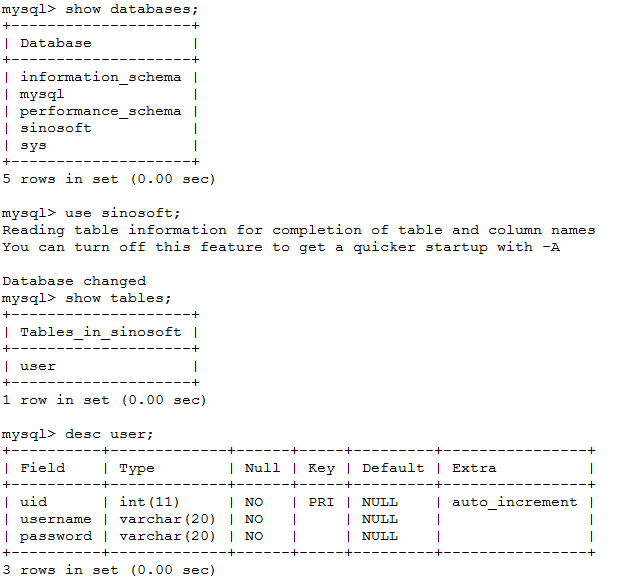


###### 新建库表

在主库里面创建sionsoft数据库和user表



再在从服务器里面查询，看下刚创建的库和表是否存在



剩下的增删改查数据测试为了节省篇幅就不一一写在该文档里了，可以自己一一测试。

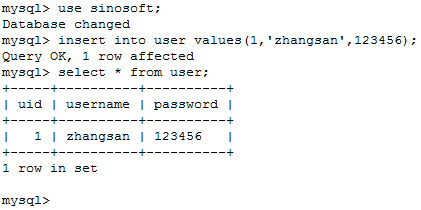
###### 宕机测试

模拟从库服务器宕机，此时测试主库服务器进行数据操作，然后恢复从库，查看数据是否从主库同步到从库

先停掉从库MySQL服务，模拟从库服务器宕机



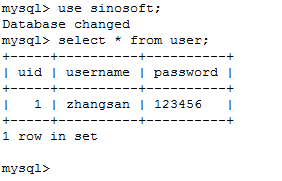
使用root用户登录主库，切换到sinosoft库，插入一条数据到user表



然后再开启从库的MySQL服务



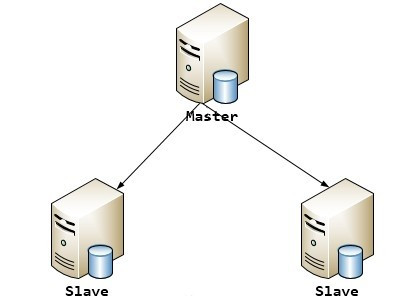
使用root用户登录主库，切换到sinosoft库，查询user表里面数据是否同步过来



此时数据已经同步到了从服务器，测试成功

#### 单主多从

##### 架构图



MasterA 192.168.50.35

SlaveA 192.168.50.36

SlaveB 192.168.50.37

##### 配置文件

###### MasterA

编辑MySQL配置文件

vim /etc/my.cnf



symbolic-links=0 配置符号链接,0是关闭,1是开启

在最后一排添加以下内容：

#数据库服务ID,必须唯一,通常设置为本机IP地址最后一位

server-id=35

#主服务器二进制日志文件前缀名,用来查看数据库的变更历史、数据库增量备份和恢复、MySQL的复制（主从数据库的复制）

log-bin=master-bin

#主服务器二进制日志文件binlog索引名

log-bin-index=master-bin.index

#binlog日志文件格式,MySQL5.7默认采用ROW

binlog\_format=ROW

#ROW模式该值默认为full，表示记录每一行的变更，包含所有列，会带来磁盘IO上的开销，同时由于binlog日志变大，网络开销也变大，改为minimal后只记录影响后的行

binlog\_row\_image=minimal

#binlog日志文件过期清理时间

expire\_logs\_days=7

#binlog缓存大小

binlog\_cache\_size=4M

#最大binlog缓存大小

max\_binlog\_cache\_size=512M

#单个binlog文件最大值, 最大和默认值是1GB，该设置并不能严格控制binlog的大小，尤其是binlog比较靠近最大值而又遇到一个比较大事务时，为了保证事务的完整性，不可能做切换日志的动作，只能将该事务的所有SQL都记录进当前日志，直到事务结束

max\_binlog\_size=512M

#设置为1,标志支持分布式事物,支持内部xa事务也就是主要保证支持binlog与innodb redo log之间数据一致性保持一致性,属于同步操作

innodb\_support\_xa=1

#binlog-do-db=需要同步的数据库名字,如果是多个数据库,就此格式再写多行

#binlog-ignore-db=不需要同步的数据库名字,如果是多个数据库,就此格式再写多行

#如果binlog-do-db和binlog-ignore-db不加的话，那么默认是同步复制整个MySQL数据库

#推荐设置--高性能：每次事务提交时MySQL都会把log buffer的数据写入log file,但是flush(刷到磁盘)操作并不会同时进行,在该模式下MySQL会每秒执行一次 flush(刷到磁盘)操作;更安全：只有在操作系统崩溃或者系统掉电的情况下,上一秒钟所有事务数据才可能丢失

innodb\_flush\_log\_at\_trx\_commit=2

sync\_binlog=1000

#禁止MySQL对连接的客户端进行DNS反向解析,只能允许IP访问,加快远程连接速度

skip-name-resolve

#设置默认字符集为UTF-8

character\_set\_server=utf8

#关闭密码策略插件

validate\_password=off

#设置MySQL大小写不敏感

lower\_case\_table\_names=1

#关闭SSL连接

disable\_ssl

粘贴完成后将每行前面第一格多余的#号去掉，保持与上文格式一样

一些具体配置参数解释见[参数意义](#_参数意义)

保存退出并重启mysql服务

service mysqld restart

###### SlaveA

编辑MySQL配置文件

vim /etc/my.cnf



symbolic-links=0 配置符号链接,0是关闭,1是开启

在最后一排添加以下内容：

#数据库服务ID,必须唯一,通常设置为本机IP地址最后一位

server-id=36

#中继日志二进制日志文件前缀名

relay-log=slaveA-relay-bin

#中继日志二进制日志文件relay\_log索引名

relay-log-index=slaveA-relay-bin.index

#开启lay\_log记录主库同步的操作日志

log\_slave\_updates=1

#禁止MySQL对连接的客户端进行DNS反向解析,只能允许IP访问,加快远程连接速度

skip-name-resolve

#设置默认字符集为UTF-8

character\_set\_server=utf8

#关闭密码策略插件

validate\_password=off

#设置MySQL大小写不敏感

lower\_case\_table\_names=1

#关闭SSL连接

disable\_ssl

粘贴完成后将每行前面第一格多余的#号去掉，保持与上文格式一样

一些具体配置参数解释见[参数意义](#_配置参数解释)

保存退出后重启mysql服务

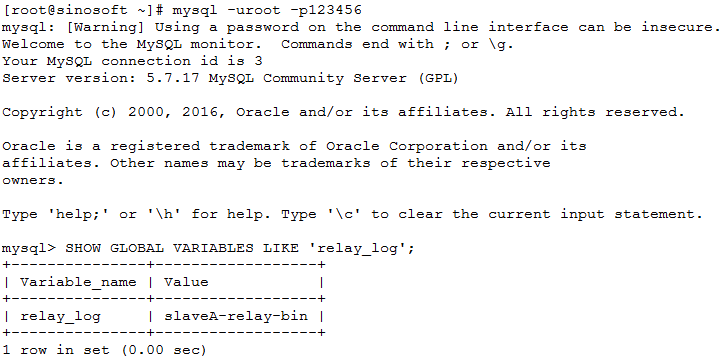
service mysqld restart



使用root用户登录MySQL数据库

查看中继日志选项是否开启

SHOW GLOBAL VARIABLES LIKE 'relay\_log';



###### SlaveB

编辑MySQL配置文件

vim /etc/my.cnf



symbolic-links=0 配置符号链接,0是关闭,1是开启

在最后一排添加以下内容：

#数据库服务ID,必须唯一,通常设置为本机IP地址最后一位

server-id=37

#中继日志二进制日志文件前缀名

relay-log=slaveB-relay-bin

#中继日志二进制日志文件relay\_log索引名

relay-log-index=slaveB-relay-bin.index

#开启lay\_log记录主库同步的操作日志

log\_slave\_updates=1

#禁止MySQL对连接的客户端进行DNS反向解析,只能允许IP访问,加快远程连接速度

skip-name-resolve

#设置默认字符集为UTF-8

character\_set\_server=utf8

#关闭密码策略插件

validate\_password=off

#设置MySQL大小写不敏感

lower\_case\_table\_names=1

#关闭SSL连接

disable\_ssl

粘贴完成后将每行前面第一格多余的#号去掉，保持与上文格式一样

一些具体配置参数解释见[参数意义](#_配置参数解释)

保存退出后重启mysql服务

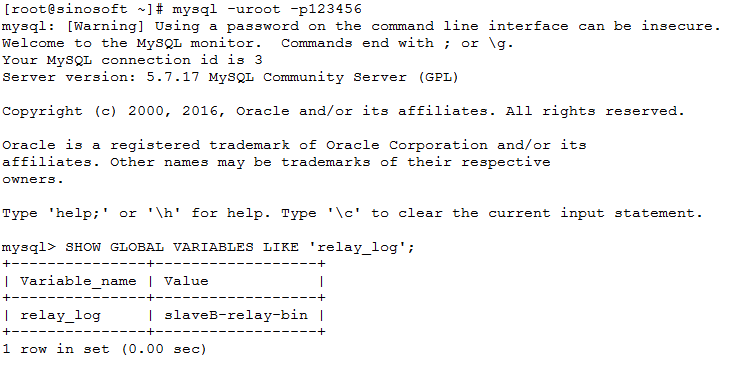
service mysqld restart



使用root用户登录MySQL数据库

查看中继日志选项是否开启

SHOW GLOBAL VARIABLES LIKE 'relay\_log';



##### 主从设置

###### 创建用户

MasterA

用root用户登录上MasterA的数据库

创建有复制权限的用户：

创建用户slaveA密码123456指定在192.168.50.36主机上可以登录

create user 'slaveA'@'192.168.50.36' identified by '123456';

移除slaveA这个用户在192.168.50.36主机上的所有权限

revoke all privileges ,grant option from 'slaveA'@'192.168.50.36';

授权重载，锁表，复制客户端，复制从属权限给masterA用户在192.168.50.36主机的所有数据库里

grant reload,lock tables, replication client ,replication slave on \*.\* to 'slaveA'@'192.168.50.36' ;

这个用户权限配置管理grant和revoke的语法详情见[用户权限配置管理](#_用户权限配置管理)

创建用户slaveB密码123456指定在192.168.50.37主机上可以登录

create user 'slaveB'@'192.168.50.37' identified by '123456';

移除slaveB这个用户在192.168.50.37主机上的所有权限

revoke all privileges ,grant option from 'slaveB'@'192.168.50.37';

授权重载，锁表，复制客户端，复制从属权限给slaveB用户在192.168.50.37主机的所有数据库里

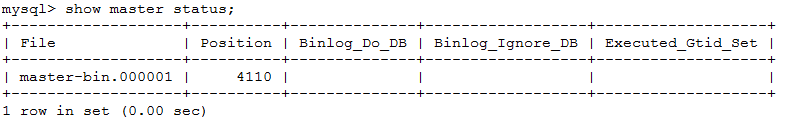
grant reload,lock tables, replication client ,replication slave on \*.\* to 'slaveB'@'192.168.50.37';

创建完后刷新同步账号的权限

flush privileges;

查看MasterA状态

show master status;



记住file和position的值，待会同步要用，此时锁定主数据库DDL操作，避免Position值发生变化

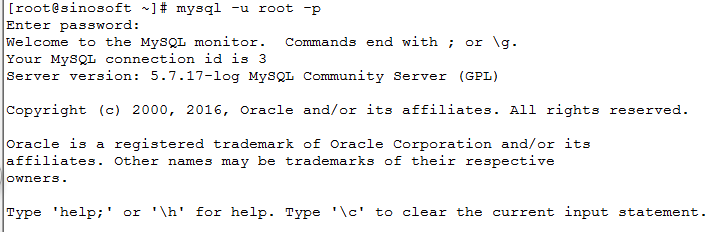
flush tables with read lock;



###### 指定主从

SlaveA

用root用户登录从服务器slave的MySQL数据库



先关闭中继日志

stop slave;



使用 CHANGE MASTER TO 命令将从库指向主库

语法：change master to master\_host='主服务器master的IP',master\_user='前面配置好的同步账号',master\_password='同步账号的密码', master\_log\_file='前面查出来主服务器master状态的File值',master\_log\_pos=前面查出来主服务器master状态的Position值;

change master to master\_user='slaveA',master\_host='192.168.50.35',

master\_password='123456',master\_log\_file='master-bin.000001',master\_log\_pos=4110;



启动中继日志

start slave;



若报下面错



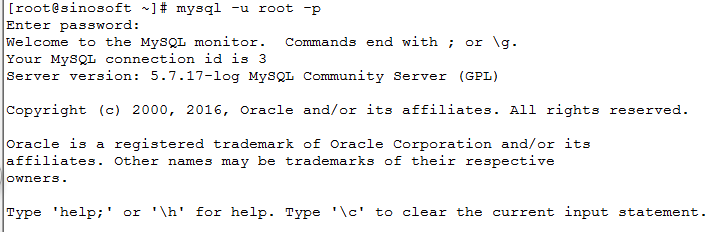
输入reset slave;



再重新使用 change master to 命令将从库指向主库，并启动服务器slave

SlaveB

用root用户登录SlaveB的MySQL数据库



先关闭中继日志

stop slave;

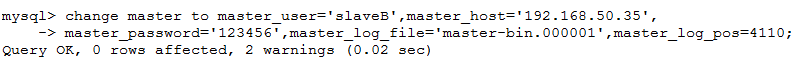


使用 CHANGE MASTER TO 命令将从库指向主库

语法：change master to master\_host='主服务器master的IP',master\_user='前面配置好的同步账号',master\_password='同步账号的密码', master\_log\_file='前面查出来主服务器master状态的File值',master\_log\_pos=前面查出来主服务器master状态的Position值;

change master to master\_user='slaveB',master\_host='192.168.50.35',

master\_password='123456',master\_log\_file='master-bin.000001',master\_log\_pos=4110;



启动中继日志

start slave;

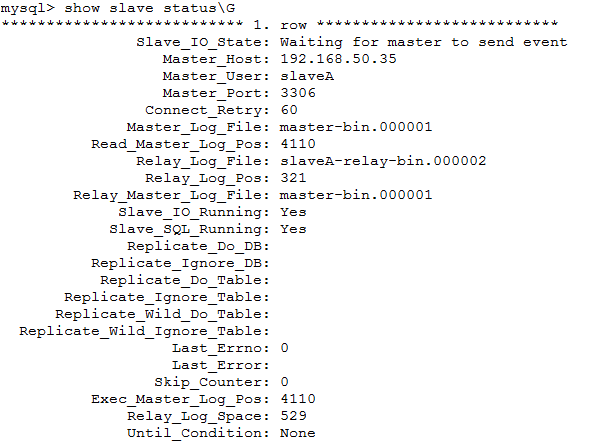


###### 查看状态

SlaveA

查看slave状态

show slave status\G



查看Slave\_IO\_Running、Slave\_SQL\_Running状态，为Yes则表明设置成功。

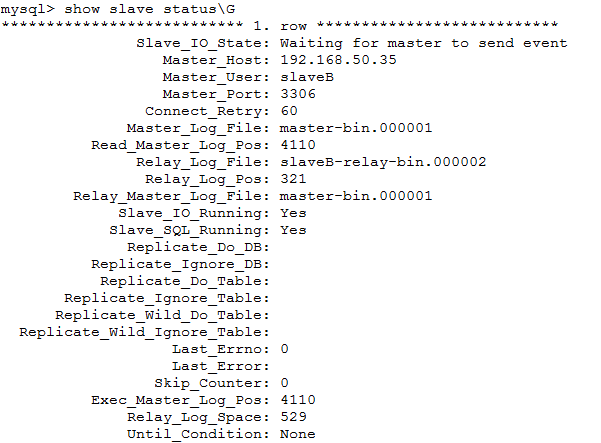
Slave\_IO\_Running：连接到主库，并读取主库的日志到本地，生成本地日志文件

Slave\_SQL\_Running:读取本地日志文件，并执行日志里的SQL命令。

SlaveB

查看slave状态

show slave status\G



查看Slave\_IO\_Running、Slave\_SQL\_Running状态，为Yes则表明设置成功。

Slave\_IO\_Running：连接到主库，并读取主库的日志到本地，生成本地日志文件

Slave\_SQL\_Running:读取本地日志文件，并执行日志里的SQL命令。

##### 测试同步

###### 新建库表

此时配置Mysql单主多从成功，需要解锁表，才能对主数据库DDL操作

登录MasterA的mysql命令窗口，执行解锁表命令

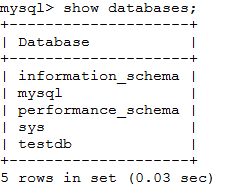
unlock tables



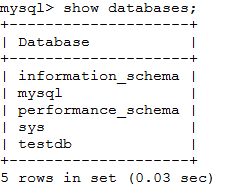
然后创建testdb数据库



然后登录SlaveA的mysql命令窗口，查询是否同步创建了刚才的testdb表



继续登录SlaveB的mysql命令窗口，查询是否同步创建了刚才的testdb表



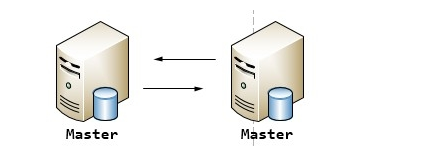
此时Mysql单主多从配置测试成功

###### 宕机测试

同上文1.2.1.2.4.2[宕机测试](#_宕机测试)

#### 双主配置

##### 架构图



IP地址：

MasterA 192.168.50.35

MasterB 192.168.50.37

##### 配置文件

###### MasterA

编辑MySQL配置文件

vim /etc/my.cnf



Symbolic-links=0 配置符号链接,0是关闭,1是开启

在最后一排添加以下内容：

#数据库服务ID,必须唯一,通常设置为本机IP地址最后一位

server-id=35

#主服务器二进制日志文件前缀名,用来查看数据库的变更历史、数据库增量备份和恢复、MySQL的复制（主从数据库的复制）

log-bin=master1-bin

#主服务器二进制日志文件binlog索引

log-bin-index=master1-bin.index

#binlog日志文件格式,MySQL5.7默认采用ROW

binlog\_format=ROW

#ROW模式该值默认为full，表示记录每一行的变更，包含所有列，会带来磁盘IO上的开销，同时由于binlog日志变大，网络开销也变大，改为minimal后只记录影响后的行

binlog\_row\_image=minimal

#binlog日志文件过期清理时间

expire\_logs\_days=7

#binlog缓存大小

binlog\_cache\_size=4M

#最大binlog缓存大小

max\_binlog\_cache\_size=512M

#单个binlog文件最大值, 最大和默认值是1GB，该设置并不能严格控制binlog的大小，尤其是binlog比较靠近最大值而又遇到一个比较大事务时，为了保证事务的完整性，不可能做切换日志的动作，只能将该事务的所有SQL都记录进当前日志，直到事务结束

max\_binlog\_size=500M

#中继日志二进制日志文件前缀名

relay\_log=slave-relay-bin

#开启lay\_log记录主库同步操作日志

log\_slave\_updates=1

#添加下面两项,以避免在MySQL自动为INSERT语句选择不互相冲突的值

auto-increment-offset=1 #起始值

auto-increment-increment=2 #步长

#推荐设置--高性能：每次事务提交时MySQL都会把log buffer的数据写入log file,但是flush(刷到磁盘)操作并不会同时进行,在该模式下MySQL会每秒执行一次 flush(刷到磁盘)操作;更安全：只有在操作系统崩溃或者系统掉电的情况下,上一秒钟所有事务数据才可能丢失

innodb\_flush\_log\_at\_trx\_commit=2

sync\_binlog=1000

#设置为1,标志支持分布式事物,支持内部xa事务也就是主要保证支持binlog与innodb redo log之间数据一致性保持一致性,属于同步操作

innodb\_support\_xa=1

#禁止MySQL对连接的客户端进行DNS反向解析,只能允许IP访问,加快远程连接速度

skip-name-resolve

#设置默认字符集为UTF-8

character\_set\_server=utf8

#关闭密码策略插件

validate\_password=off

#设置MySQL大小写不敏感

lower\_case\_table\_names=1

#关闭SSL连接

disable\_ssl

粘贴完成后将每行前面第一格多余的#号去掉，保持与上文格式一样

一些具体配置参数解释见[参数意义](#_配置参数解释)

保存退出后重启mysql服务

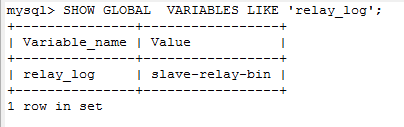
service mysqld restart



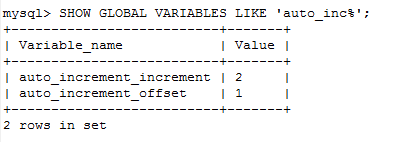
使用root用户登录MySQL数据库

查看中继日志选项是否开启

SHOW GLOBAL VARIABLES LIKE 'relay\_log';



查看自动增长选项是否开启



###### MasterB

编辑MySQL配置文件

vim /etc/my.cnf



symbolic-links=0 配置符号链接,0是关闭,1是开启

在最后一排添加以下内容：

#数据库服务ID,必须唯一,通常设置为本机IP地址最后一位

server-id=37

#主服务器二进制日志文件前缀名,用来查看数据库的变更历史、数据库增量备份和恢复、MySQL的复制（主从数据库的复制）

log-bin=master2-bin

#主服务器二进制日志文件binlog索引

log-bin-index=master2-bin.index

#binlog日志文件格式,MySQL5.7默认采用ROW

binlog\_format=ROW

#ROW模式该值默认为full，表示记录每一行的变更，包含所有列，会带来磁盘IO上的开销，同时由于binlog日志变大，网络开销也变大，改为minimal后只记录影响后的行

binlog\_row\_image=minimal

#binlog日志文件过期清理时间

expire\_logs\_days=7

#binlog缓存大小

binlog\_cache\_size=4M

#最大binlog缓存大小

max\_binlog\_cache\_size=512M

#单个binlog文件最大值, 最大和默认值是1GB，该设置并不能严格控制binlog的大小，尤其是binlog比较靠近最大值而又遇到一个比较大事务时，为了保证事务的完整性，不可能做切换日志的动作，只能将该事务的所有SQL都记录进当前日志，直到事务结束

max\_binlog\_size=500M

#中继日志二进制日志文件前缀名

relay\_log=slave-relay-bin

#开启lay\_log记录主库同步操作日志

log\_slave\_updates=1

#添加下面两项,以避免在MySQL自动为INSERT语句选择不互相冲突的值

auto\_increment\_offset=2 #起始值

auto\_increment\_increment=2 #步长

#推荐设置--高性能：每次事务提交时MySQL都会把log buffer的数据写入log file,但是flush(刷到磁盘)操作并不会同时进行,在该模式下MySQL会每秒执行一次 flush(刷到磁盘)操作;更安全：只有在操作系统崩溃或者系统掉电的情况下,上一秒钟所有事务数据才可能丢失

innodb\_flush\_log\_at\_trx\_commit=2

sync\_binlog=1000

#设置为1,标志支持分布式事物,支持内部xa事务也就是主要保证支持binlog与innodb redo log之间数据一致性保持一致性,属于同步操作

innodb\_support\_xa=1

#禁止MySQL对连接的客户端进行DNS反向解析,只能允许IP访问,加快远程连接速度

skip-name-resolve

#设置默认字符集为UTF-8

character\_set\_server=utf8

#关闭密码策略插件

validate\_password=off

#设置MySQL大小写不敏感

lower\_case\_table\_names=1

#关闭SSL连接

disable\_ssl

粘贴完成后将每行前面第一格多余的#号去掉，保持与上文格式一样

一些具体配置参数解释见[参数意义](#_配置参数解释)

保存退出后重启mysql服务

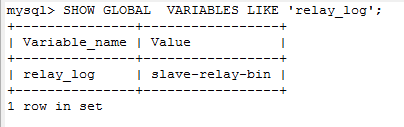
service mysqld restart



使用root用户登录MySQL数据库

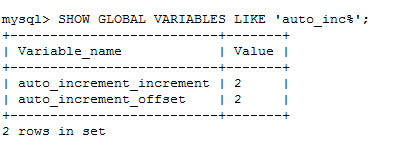
查看中继日志选项是否开启

SHOW GLOBAL VARIABLES LIKE 'relay\_log';



查看自动增长选项是否开启：

SHOW GLOBAL VARIABLES LIKE 'auto\_inc%';



##### 主从设置

###### 创建用户

MasterA

用root用户登录上MasterA的数据库

创建有复制权限的用户：

创建用户masterA密码123456指定在192.168.50.37主机上可以登录

create user 'masterA'@'192.168.50.37' identified by '123456';

移除masterA这个用户在192.168.50.37主机上的所有权限

revoke all privileges ,grant option from 'masterA'@'192.168.50.37';

授权重载，锁表，复制客户端，复制从属权限给masterA用户在192.168.50.37主机的所有数据库里

grant reload,lock tables, replication client ,replication slave on \*.\* to 'masterA'@'192.168.50.37';

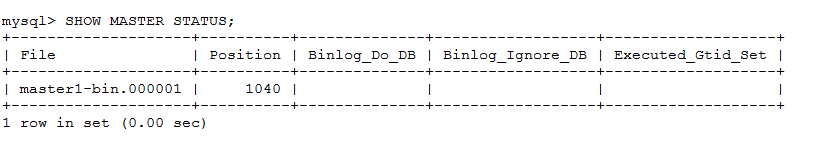
这个用户权限配置管理grant和revoke的语法详情见[用户权限配置管理](#_用户权限配置管理)

创建完后刷新同步账号的权限

flush privileges;

查看MasterA状态

show master status;



记住file和position的值，待会同步要用，此时不要再对主服务器MySQL做任何操作，避免Position值发生变化

MasterB

用root用户登录上MasterA的数据库

创建有复制权限的用户：

创建用户masterB密码123456指定在192.168.50.35主机上可以登录

CREATE USER 'masterB'@'192.168.50.35' IDENTIFIED BY '123456';

移除masterB这个用户在192.168.50.35主机上的所有权限

REVOKE ALL PRIVILEGES ,GRANT OPTION FROM 'masterB'@'192.168.50.35';

授权重载，锁表，复制客户端，复制从属权限给masterB用户在192.168.50.35主机的所有数据库里

GRANT RELOAD,LOCK TABLES, REPLICATION CLIENT ,REPLICATION SLAVE ON \*.\* TO 'masterB'@'192.168.50.35';

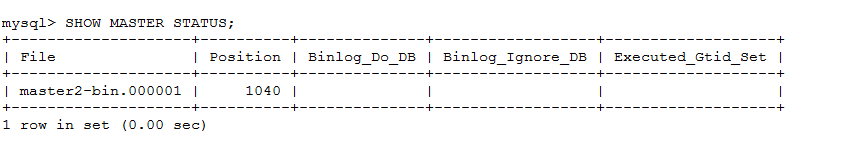
这个用户权限配置管理grant和revoke的语法详情见[用户权限配置管理](#_用户权限配置管理)

创建完后刷新同步账号的权限

FLUSH PRIVILEGES;

查看MasterB状态

SHOW MASTER STATUS;



记住file和position的值，待会同步要用，此时锁定主数据库DDL操作，避免Position值发生变化

flush tables with read lock;



###### 指定主从

MasterA

先关闭中继日志

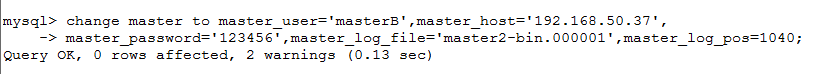
stop slave;



接下来指定MasterB服务器为自己的主服务器

change master to master\_user='masterB',master\_host='192.168.50.37',

master\_password='123456',master\_log\_file='master2-bin.000001',master\_log\_pos=1040;



MasterB

先关闭中继日志

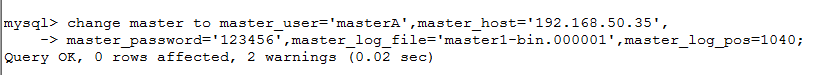
stop slave;



接下来指定MasterA服务器为自己的主服务器

change master to master\_user='masterA',master\_host='192.168.50.35',

master\_password='123456',master\_log\_file='master1-bin.000001',master\_log\_pos=1040;



###### 查看状态

MasterA

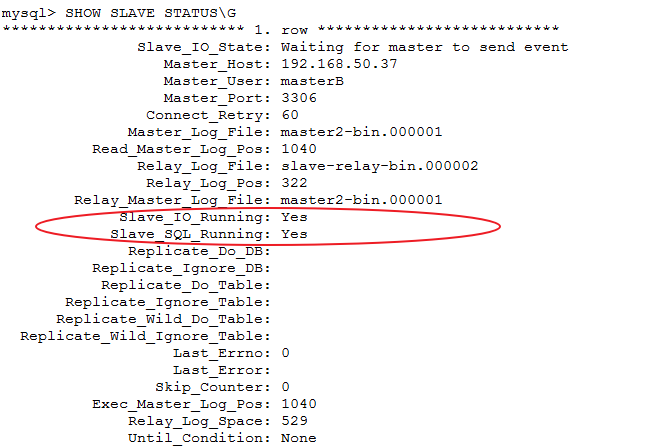
开启中继日志

start slave;



查看复制状态

show slave status\G



查看Slave\_IO\_Running、Slave\_SQL\_Running状态，为Yes则表明设置成功。

Slave\_IO\_Running：连接到主库，并读取主库的日志到本地，生成本地日志文件

Slave\_SQL\_Running:读取本地日志文件，并执行日志里的SQL命令。

MasterB

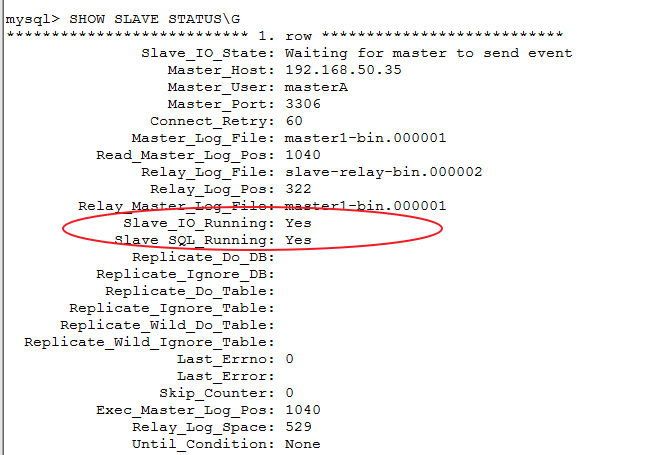
开启中继日志

start slave;



查看复制状态

show slave status\g



查看Slave\_IO\_Running、Slave\_SQL\_Running状态，为Yes则表明设置成功。

Slave\_IO\_Running：连接到主库，并读取主库的日志到本地，生成本地日志文件

Slave\_SQL\_Running:读取本地日志文件，并执行日志里的SQL命令。

##### 测试同步

###### 新建库表

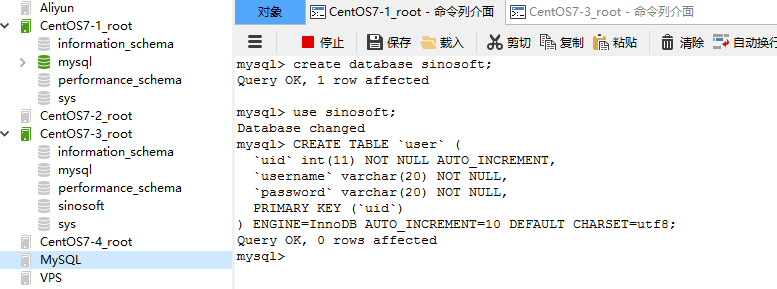
此时配置Mysql单主多从成功，需要解锁表，才能对主数据库DDL操作

登录MasterA的mysql命令窗口，执行解锁表命令

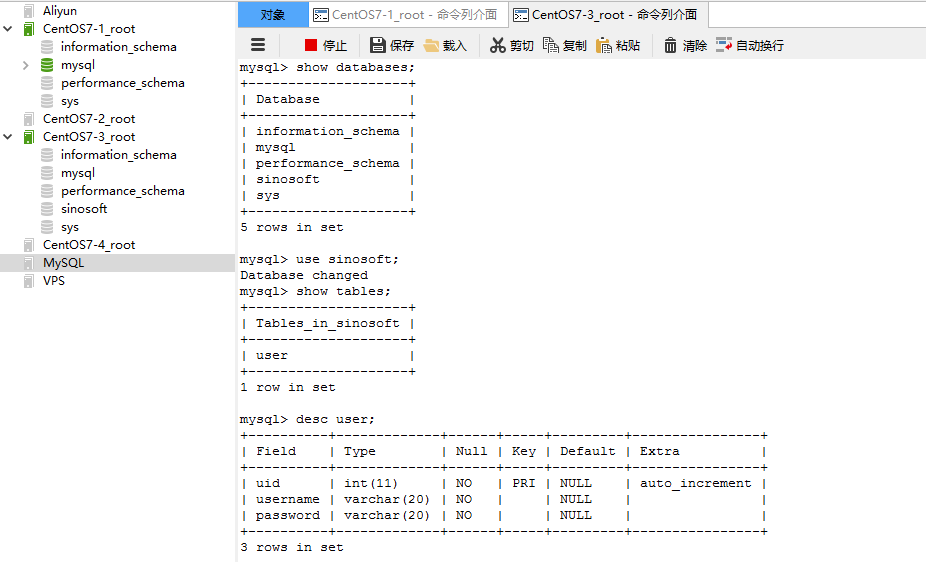
unlock tables



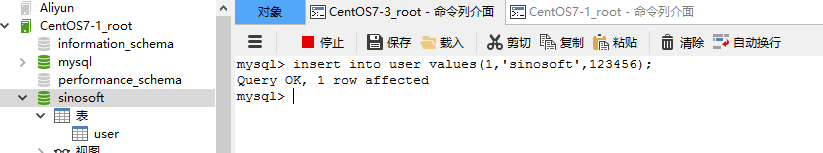
在MasterA里面创建数据库sinosoft和user表



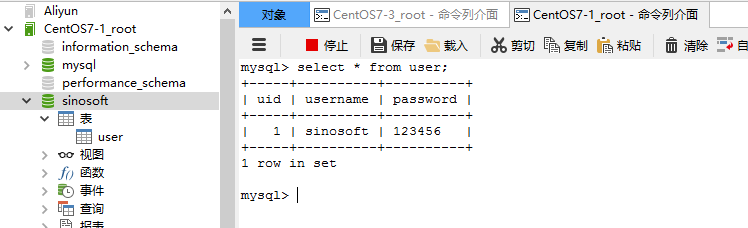
在MasterB里面查询是否有刚创建的sinosoft库和user表



从MasterB中插入一条数据到sinosoft.user表



再在MasterA中查询sinosoft.user表中是否有刚才插入的数据



此时双主配置成功，可以再自己测试数据改查等功能

###### 宕机测试

同上文1.2.1.2.4.2[宕机测试](#_宕机测试)

##### 配置注解

###### auto\_increment\_offset和auto\_increment\_increment配置

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.

这样才可以避免两台服务器同时做更新时自增长字段的值之间发生冲突。

更多信息请参考官方文档： <http://dev.mysql.com/doc/refman/5.0/en/replication-options-master.html>

#### 双主双从

IP地址：

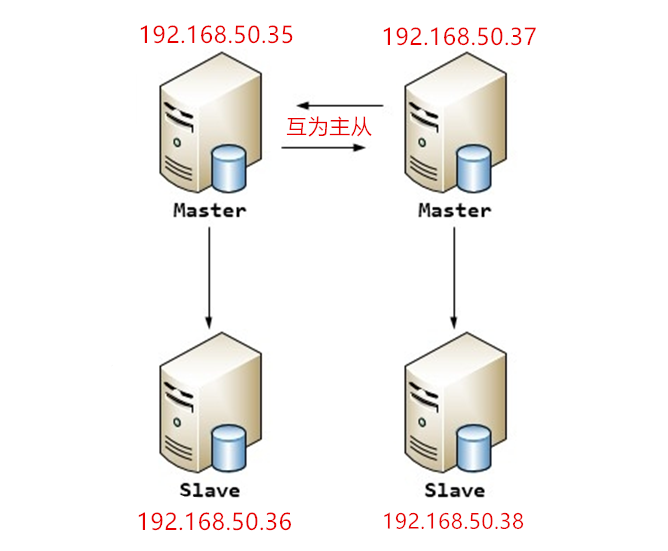
MasterA 192.168.50.35

MasterB 192.168.50.37

SlaveA 192.168.50.36

SlaveB 192.168.50.38

##### 架构图



##### 配置文件

###### MasterA

编辑MySQL配置文件

vim /etc/my.cnf



symbolic-links=0 配置符号链接,0是关闭,1是开启

在最后一排添加以下内容：

#数据库服务ID,必须唯一,通常设置为本机IP地址最后一位

server-id=35

#主服务器二进制日志文件前缀名,用来查看数据库的变更历史、数据库增量备份和恢复、MySQL的复制（主从数据库的复制）

log-bin=master1-bin

#主服务器二进制日志文件binlog索引

log-bin-index=master1-bin.index

#binlog日志文件格式,MySQL5.7默认采用ROW

binlog\_format=ROW

#ROW模式该值默认为full，表示记录每一行的变更，包含所有列，会带来磁盘IO上的开销，同时由于binlog日志变大，网络开销也变大，改为minimal后只记录影响后的行

binlog\_row\_image=minimal

#binlog日志文件过期清理时间

expire\_logs\_days=7

#binlog缓存大小

binlog\_cache\_size=4M

#最大binlog缓存大小

max\_binlog\_cache\_size=512M

#单个binlog文件最大值, 最大和默认值是1GB，该设置并不能严格控制binlog的大小，尤其是binlog比较靠近最大值而又遇到一个比较大事务时，为了保证事务的完整性，不可能做切换日志的动作，只能将该事务的所有SQL都记录进当前日志，直到事务结束

max\_binlog\_size=500M

#中继日志二进制日志文件前缀名

relay\_log=slave-relay-bin

#开启lay\_log记录主库同步操作日志

log\_slave\_updates=1

#添加下面两项,以避免在MySQL自动为INSERT语句选择不互相冲突的值

auto-increment-offset=1 #起始值

auto-increment-increment=2 #步长

#推荐设置--高性能：每次事务提交时MySQL都会把log buffer的数据写入log file,但是flush(刷到磁盘)操作并不会同时进行,在该模式下MySQL会每秒执行一次 flush(刷到磁盘)操作;更安全：只有在操作系统崩溃或者系统掉电的情况下,上一秒钟所有事务数据才可能丢失

innodb\_flush\_log\_at\_trx\_commit=2

sync\_binlog=1000

#设置为1,标志支持分布式事物,支持内部xa事务也就是主要保证支持binlog与innodb redo log之间数据一致性保持一致性,属于同步操作

innodb\_support\_xa=1

#禁止MySQL对连接的客户端进行DNS反向解析,只能允许IP访问,加快远程连接速度

skip-name-resolve

#设置默认字符集为UTF-8

character\_set\_server=utf8

#关闭密码策略插件

validate\_password=off

#设置MySQL大小写不敏感

lower\_case\_table\_names=1

#关闭SSL连接

disable\_ssl

粘贴完成后将每行前面第一格多余的#号去掉，保持与上文格式一样

一些具体配置参数解释见[配置注解](#_配置注解_2)

保存退出后重启mysql服务

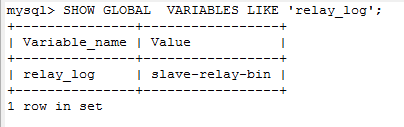
service mysqld restart



使用root用户登录MySQL数据库

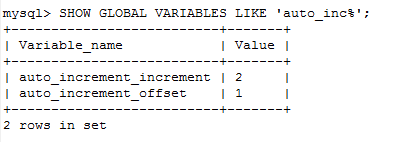
查看中继日志选项是否开启

SHOW GLOBAL VARIABLES LIKE 'relay\_log';



查看自动增长选项是否开启

SHOW GLOBAL VARIABLES LIKE 'auto\_inc%';



###### MasterB

编辑MySQL配置文件

vim /etc/my.cnf



symbolic-links=0 配置符号链接,0是关闭,1是开启

在最后一排添加以下内容：

#数据库服务ID,必须唯一,通常设置为本机IP地址最后一位

server-id=37

#主服务器二进制日志文件前缀名,用来查看数据库的变更历史、数据库增量备份和恢复、MySQL的复制（主从数据库的复制）

log-bin=master2-bin

#主服务器二进制日志文件binlog索引

log-bin-index=master2-bin.index

#binlog日志文件格式,MySQL5.7默认采用ROW

binlog\_format=ROW

#ROW模式该值默认为full，表示记录每一行的变更，包含所有列，会带来磁盘IO上的开销，同时由于binlog日志变大，网络开销也变大，改为minimal后只记录影响后的行

binlog\_row\_image=minimal

#binlog日志文件过期清理时间

expire\_logs\_days=7

#binlog缓存大小

binlog\_cache\_size=4M

#最大binlog缓存大小

max\_binlog\_cache\_size=512M

#单个binlog文件最大值, 最大和默认值是1GB，该设置并不能严格控制binlog的大小，尤其是binlog比较靠近最大值而又遇到一个比较大事务时，为了保证事务的完整性，不可能做切换日志的动作，只能将该事务的所有SQL都记录进当前日志，直到事务结束

max\_binlog\_size=500M

#中继日志二进制日志文件前缀名

relay\_log=slave-relay-bin

#开启lay\_log记录主库同步操作日志

log\_slave\_updates=1

#添加下面两项,以避免在MySQL自动为INSERT语句选择不互相冲突的值

auto-increment-offset=2 #起始值

auto-increment-increment=2 #步长

#推荐设置--高性能：每次事务提交时MySQL都会把log buffer的数据写入log file,但是flush(刷到磁盘)操作并不会同时进行,在该模式下MySQL会每秒执行一次 flush(刷到磁盘)操作;更安全：只有在操作系统崩溃或者系统掉电的情况下,上一秒钟所有事务数据才可能丢失

innodb\_flush\_log\_at\_trx\_commit=2

sync\_binlog=1000

#设置为1,标志支持分布式事物,支持内部xa事务也就是主要保证支持binlog与innodb redo log之间数据一致性保持一致性,属于同步操作

innodb\_support\_xa=1

#禁止MySQL对连接的客户端进行DNS反向解析,只能允许IP访问,加快远程连接速度

skip-name-resolve

#设置默认字符集为UTF-8

character\_set\_server=utf8

#关闭密码策略插件

validate\_password=off

#设置MySQL大小写不敏感

lower\_case\_table\_names=1

#关闭SSL连接

disable\_ssl

粘贴完成后将每行前面第一格多余的#号去掉，保持与上文格式一样

一些具体配置参数解释见[配置注解](#_配置注解_2)

保存退出后重启mysql服务

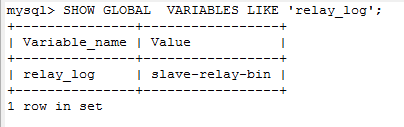
service mysqld restart



使用root用户登录MySQL数据库

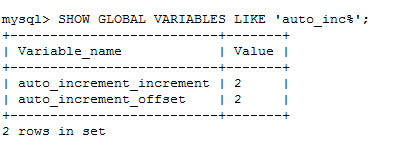
查看中继日志选项是否开启

SHOW GLOBAL VARIABLES LIKE 'relay\_log';



查看自动增长选项是否开启：

SHOW GLOBAL VARIABLES LIKE 'auto\_inc%';



###### SlaveA

编辑MySQL配置文件

vim /etc/my.cnf



symbolic-links=0 配置符号链接,0是关闭,1是开启

在最后一排添加以下内容：

#数据库服务ID,必须唯一,通常设置为本机IP地址最后一位

server-id=36

#中继日志二进制日志文件前缀名

relay-log=slave1-relay-bin

#中继日志二进制日志文件relay\_log索引名

relay-log-index=slave1-relay-bin.index

#开启lay\_log记录主库同步的操作日志

log\_slave\_updates=1

#设定从库为只读模式,可以限定普通用户进行数据修改的操作,但不会限定具有super权限的用户如root的数据修改操作;在MySQL中设置read\_only=1后,普通的应用用户进行insert、update、delete等会产生数据变化的DML操作时,都会报出数据库处于只读模式不能发生数据变化的错误,但具有super权限的用户,例如在本地或远程通过root用户登录到数据库,还是可以进行数据变化的DML操作

read\_only=1

#禁止MySQL对连接的客户端进行DNS反向解析,只能允许IP访问,加快远程连接速度

skip-name-resolve

#设置默认字符集为UTF-8

character\_set\_server=utf8

#关闭密码策略插件

validate\_password=off

#设置MySQL大小写不敏感

lower\_case\_table\_names=1

#关闭SSL连接

disable\_ssl

粘贴完成后将每行前面第一格多余的#号去掉，保持与上文格式一样

一些具体配置参数解释见[配置注解](#_配置注解_1)

保存退出并重启mysql服务

service mysqld restart

###### SlaveB

编辑MySQL配置文件

vim /etc/my.cnf



symbolic-links=0 配置符号链接,0是关闭,1是开启

在最后一排添加以下内容：

#数据库服务ID,必须唯一,通常设置为本机IP地址最后一位

server-id=38

#中继日志二进制日志文件前缀名

relay-log=slave2-relay-bin

#中继日志二进制日志文件relay\_log索引名

relay-log-index=slave2-relay-bin.index

#开启lay\_log记录主库同步的操作日志

log\_slave\_updates=1

#设定从库为只读模式,可以限定普通用户进行数据修改的操作,但不会限定具有super权限的用户如root的数据修改操作;在MySQL中设置read\_only=1后,普通的应用用户进行insert、update、delete等会产生数据变化的DML操作时,都会报出数据库处于只读模式不能发生数据变化的错误,但具有super权限的用户,例如在本地或远程通过root用户登录到数据库,还是可以进行数据变化的DML操作

read\_only=1

#禁止MySQL对连接的客户端进行DNS反向解析,只能允许IP访问,加快远程连接速度

skip-name-resolve

#设置默认字符集为UTF-8

character\_set\_server=utf8

#关闭密码策略插件

validate\_password=off

#设置MySQL大小写不敏感

lower\_case\_table\_names=1

#关闭SSL连接

disable\_ssl

粘贴完成后将每行前面第一格多余的#号去掉，保持与上文格式一样

一些具体配置参数解释见[配置注解](#_配置注解_1)

保存退出并重启mysql服务

service mysqld restart

##### 主从设置

###### 创建用户

MasterA

用root用户登录上MasterA的数据库

创建有复制权限的用户：

创建用户masterA密码123456指定在192.168.50.37主机上可以登录

CREATE USER 'masterA'@'192.168.50.37' IDENTIFIED BY '123456';

移除masterA这个用户在192.168.50.37主机上的所有权限

REVOKE ALL PRIVILEGES ,GRANT OPTION FROM 'masterA'@'192.168.50.37';

授权重载，锁表，复制客户端，复制从属权限给masterA用户在192.168.50.37主机的所有数据库里

GRANT RELOAD,LOCK TABLES, REPLICATION CLIENT ,REPLICATION SLAVE ON \*.\* TO 'masterA'@'192.168.50.37';

这个用户权限配置管理grant和revoke的语法详情见[用户权限配置管理](#_用户权限配置管理)

创建完后刷新同步账号的权限

FLUSH PRIVILEGES;

MasterB

用root用户登录上MasterB的数据库

创建有复制权限的用户：

创建用户masterB密码123456指定在192.168.50.35主机上可以登录

CREATE USER 'masterB'@'192.168.50.35' IDENTIFIED BY '123456';

移除masterB这个用户在192.168.50.35主机上的所有权限

REVOKE ALL PRIVILEGES ,GRANT OPTION FROM 'masterB'@'192.168.50.35';

授权重载，锁表，复制客户端，复制从属权限给masterB用户在192.168.50.35主机的所有数据库里

GRANT RELOAD,LOCK TABLES, REPLICATION CLIENT ,REPLICATION SLAVE ON \*.\* TO 'masterB'@'192.168.50.35';

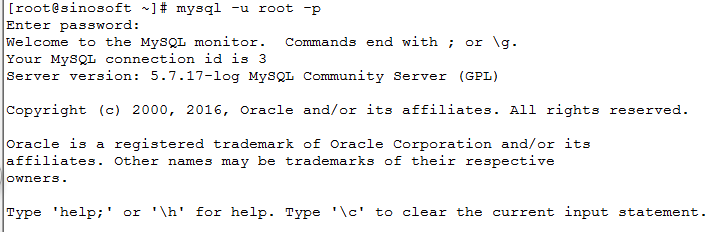
这个用户权限配置管理grant和revoke的语法详情见[用户权限配置管理](#_用户权限配置管理)

创建完后刷新同步账号的权限

FLUSH PRIVILEGES;

SlaveA

用root用户登录主服务器masterA的MySQL数据库



开始创建同步账号并授权在从服务器slaveA上登录访问,创建完后刷新同步账号的权限

创建用户slaveA密码123456指定在192.168.50.36主机上可以登录

create user 'slaveA'@'192.168.50.36' identified by '123456';

移除slaveA这个用户在192.168.50.36主机上的所有权限

revoke all privileges ,grant option from 'slaveA'@'192.168.50.36';

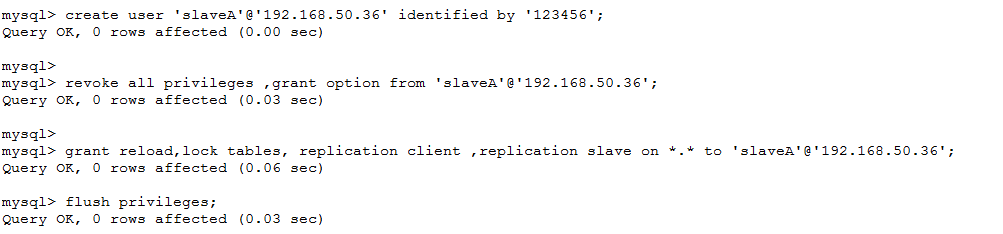
授权重载，锁表，复制客户端，复制从属权限给slaveA用户在192.168.50.36主机的所有数据库里

grant reload,lock tables, replication client ,replication slave on \*.\* to 'slaveA'@'192.168.50.36';

这个用户权限配置管理grant和revoke的语法详情见[用户权限配置管理](#_用户权限配置管理)

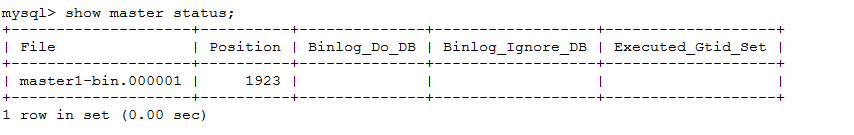
创建完后刷新同步账号的权限

flush privileges;



查看主服务器MySQL的状态

show master status;

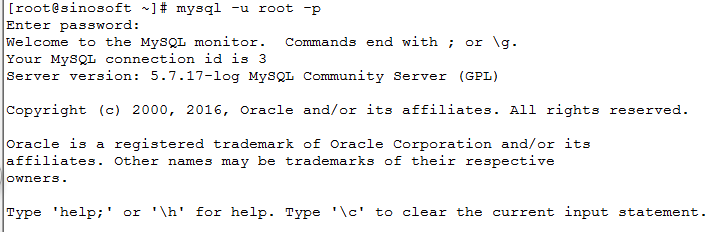


记住file和position的值，待会同步要用

此时不要再对主服务器MySQL做任何操作，避免Position值发生变化

SlaveB

用root用户登录主服务器masterB的MySQL数据库



开始创建同步账号并授权在从服务器slaveB上登录访问, 创建完后刷新同步账号的权限

创建用户slaveB密码123456指定在192.168.50.38主机上可以登录

create user 'slaveB'@'192.168.50.38' identified by '123456';

移除slaveB这个用户在192.168.50.38主机上的所有权限

revoke all privileges ,grant option from 'slaveB'@'192.168.50.38';

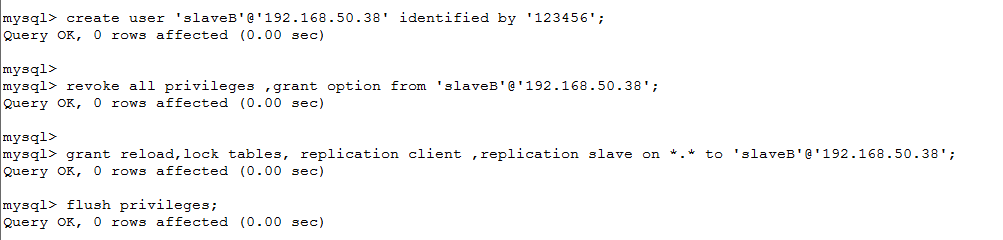
授权重载，锁表，复制客户端，复制从属权限给slaveA用户在192.168.50.36主机的所有数据库里

grant reload,lock tables, replication client ,replication slave on \*.\* to 'slaveB'@'192.168.50.38';

这个用户权限配置管理grant和revoke的语法详情见[用户权限配置管理](#_用户权限配置管理)

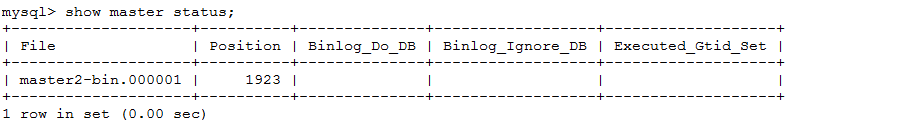
创建完后刷新同步账号的权限

flush privileges;



查看主服务器MySQL的状态

show master status;



记住file和position的值，待会同步要用

此时不要再对主服务器MySQL做任何操作，避免Position值发生变化

###### 指定主从

MasterA

切换到MasterA的MySQL命令窗口

先关闭中继日志

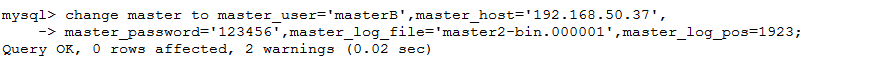
stop slave;



接下来指定MasterB服务器为自己的主服务器

change master to master\_user='masterB',master\_host='192.168.50.37',

master\_password='123456',master\_log\_file='master2-bin.000001',master\_log\_pos=1923;



MasterB

切换到MasterB的MySQL命令窗口

先关闭中继日志

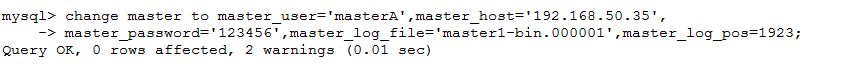
stop slave;



接下来指定MasterA服务器为自己的主服务器

change master to master\_user='masterA',master\_host='192.168.50.35',

master\_password='123456',master\_log\_file='master1-bin.000001',master\_log\_pos=1923;



SlaveA

切换到SlaveA的MySQL命令窗口

先关闭中继日志

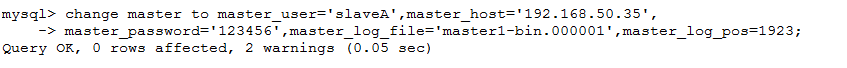
stop slave;



接下来指定MasterA服务器为自己的主服务器

change master to master\_user='slaveA',master\_host='192.168.50.35',

master\_password='123456',master\_log\_file='master1-bin.000001',master\_log\_pos=1923;



SlaveB

切换到SlaveB的MySQL命令窗口

先关闭中继日志

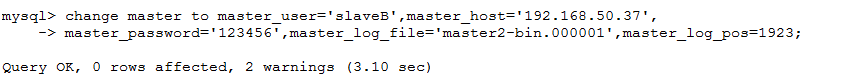
stop slave;



接下来指定MasterB服务器为自己的主服务器

change master to master\_user='slaveB',master\_host='192.168.50.37',

master\_password='123456',master\_log\_file='master2-bin.000001',master\_log\_pos=1923;



###### 查看状态

MasterA

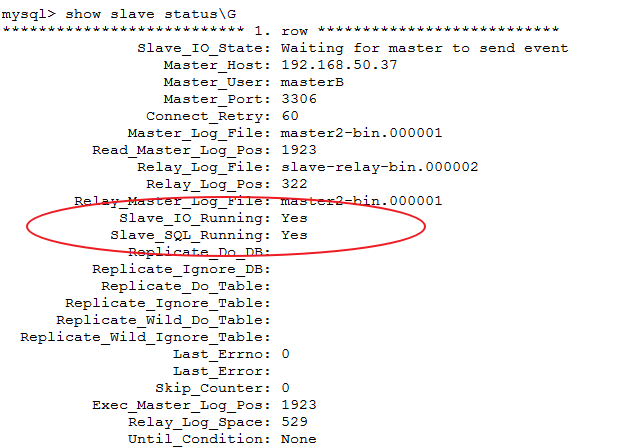
开启中继日志

start slave;



查看复制状态

show slave status\G



查看Slave\_IO\_Running、Slave\_SQL\_Running状态，为Yes则表明设置成功。

Slave\_IO\_Running：连接到主库，并读取主库的日志到本地，生成本地日志文件

Slave\_SQL\_Running:读取本地日志文件，并执行日志里的SQL命令。

MasterB

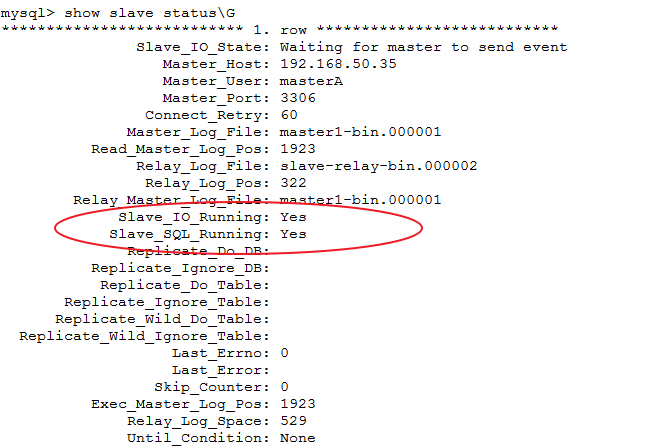
开启中继日志

start slave;



查看复制状态

show slave status\G



查看Slave\_IO\_Running、Slave\_SQL\_Running状态，为Yes则表明设置成功。

Slave\_IO\_Running：连接到主库，并读取主库的日志到本地，生成本地日志文件

Slave\_SQL\_Running:读取本地日志文件，并执行日志里的SQL命令。

SlaveA

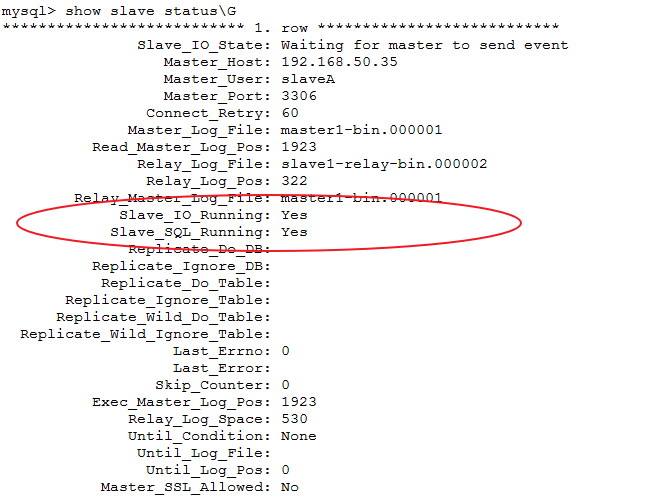
开启中继日志

start slave;



查看复制状态

show slave status\G



查看Slave\_IO\_Running、Slave\_SQL\_Running状态，为Yes则表明设置成功。

Slave\_IO\_Running：连接到主库，并读取主库的日志到本地，生成本地日志文件

Slave\_SQL\_Running:读取本地日志文件，并执行日志里的SQL命令。

SlaveB

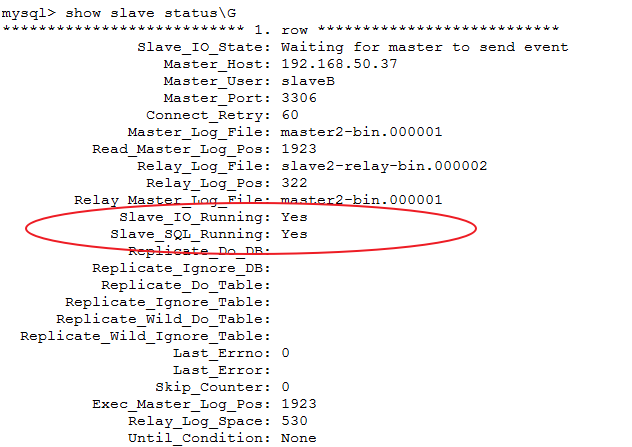
开启中继日志

start slave;



查看复制状态

show slave status\G



查看Slave\_IO\_Running、Slave\_SQL\_Running状态，为Yes则表明设置成功。

Slave\_IO\_Running：连接到主库，并读取主库的日志到本地，生成本地日志文件

Slave\_SQL\_Running:读取本地日志文件，并执行日志里的SQL命令。

##### 测试同步

此时配置Mysql单主多从成功，需要解锁表，才能对主数据库DDL操作

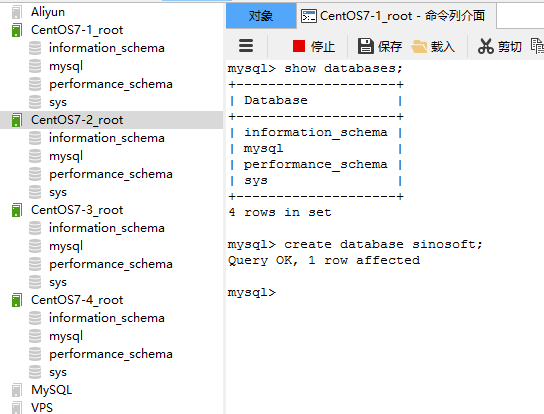
登录MasterA的mysql命令窗口，执行解锁表命令

unlock tables

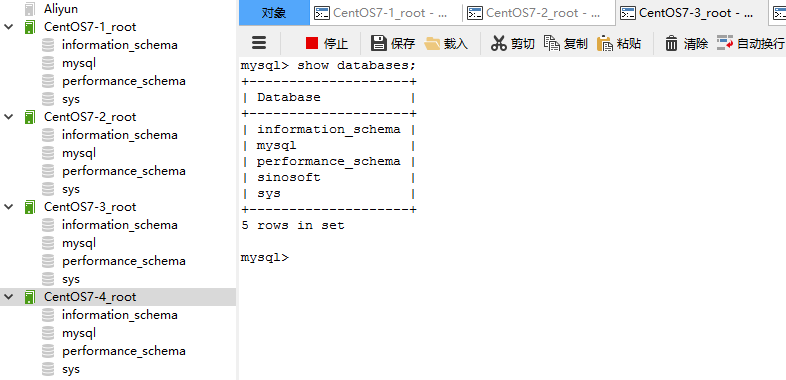


###### 新建库表

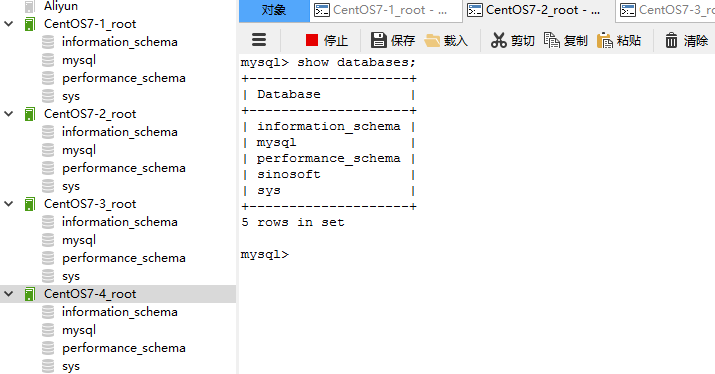
在MasterA里面创建sinosoft数据库



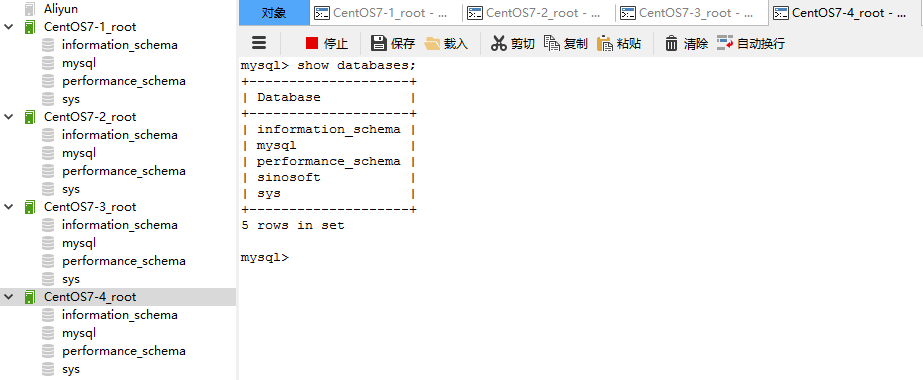
在MasterB里面查询



在SlaveA里面查询



在SlaveB里面查询



测试成功，剩下可以自己测试对数据的增删改查操作。

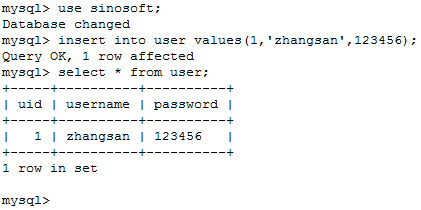
###### 宕机测试

模拟MasterB、SlaveA、SlaveB服务器宕机，此时测试MasterA服务器进行数据操作，然后分别恢复MasetB、SlaveA、SlaveB，查看数据是否从MasterA同步到MasterB、SlaveA、SlaveB

先停掉MasterB、SlaveA、SlaveB的MySQL服务，模拟从库服务器宕机



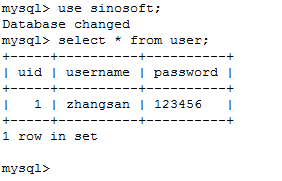
使用root用户登录主库，切换到sinosoft库，插入一条数据到user表



然后再开启SlaveA的MySQL服务



使用root用户登录主库，切换到sinosoft库，查询user表里面数据是否同步过来

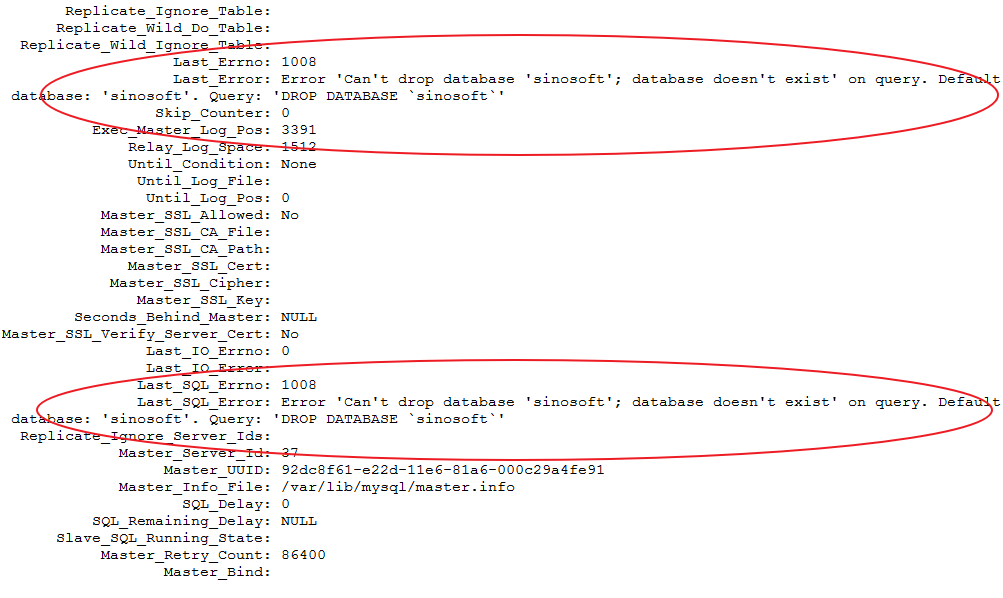


接着依次按照上面步骤开启MasterB、SlaveB的MySQL服务，查询刚添加的数据是否已经同步过来，若数据已经同步到了从服务器，测试成功

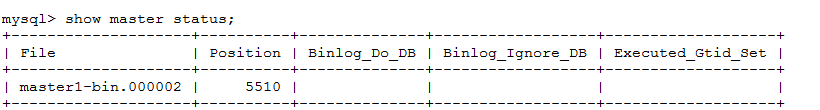
#### 特别注意

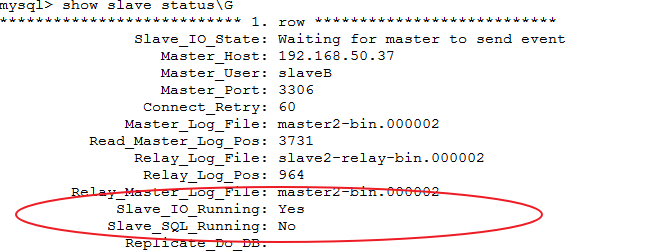
##### 错误问题

配置主从时所有MySQL服务器里面要同步的数据库内容必须保持一致，不然操作同步时会报错！！！类似下面这种：



2.在查看完主服务器的状态后尽量不要再对主服务器进行任何操作，避免Position值发生变化！！！导致类似下面这种错误：





##### 解决方案

碰到Slave\_IO\_Running、Slave\_SQL\_Running状态为No问题解决方案：

先在删除每一个MySQL服务器的mysql.user表里前面创建的授权用户，

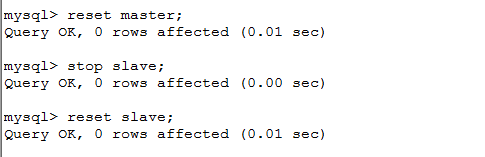
比如 drop user masterA@192.168.50.37;

再用root用户登入每一个MySQL服务器里面执行这三行

reset master;

stop slave;

reset slave;



然后按照每个主从关系的创建用户步骤重新开始设置，这样就设置成功了！

### Windows服务器

*需要有截图。*

正文……

# MySQL Cluster

## 概述

MySQL Cluster 是MySQL 官方集群部署方案，它的历史较久。支持通过自动分片支持读写扩展，通过实时备份冗余数据，是可用性最高的方案，声称可做到99.999%的可用性。

# MySQL监控

相关内容已转至MyCAT安装配置文档.docx