1、配置文件中的auto_increment_incremenet auto_increment_offset 参数说明

两个节点的auto_increment_incremenet(自增步长)和auto_increment_offset(自增起始点)设为不同值。目的为了避免master意外宕机，可能会有部分binlog未能及时复制到slave上被应用，从而导致slave新写入数据的自增值和原先的master冲突，从offset起始点开始就错开了，避免了主键id的冲突，当然，如有合适的容错机制解决冲突话，也可以不这么设置

2、如果遇到主从延迟怎么解决？

①首先需要通过show slave statusG中 Seconds_Behind_Master观察主从之间延迟的状态

②slave节点服务器硬件配置不能与master节点相差太大，会大大导致复制的延迟

③如果对延迟问题很敏感，可以考虑更换mariadb分支版本，或者直接上线mysql5.7最新版本，利用多线程复制的方式可以很大程度降低复制延迟

mysql> show global variables like 'slave_paralle%';

Variable_name	Value
slave_parallel_type	DATABASE
slave_parallel_workers	0

slave_parallel_workers：默认为0，表示为单线程

slave_parallel_type：默认多线程机制为一个线程处理一个DATABASE

mysql> set global slave_parallel_workers=4; #修改为四个线程操作

mysql> set global slave_parallel_type='logical_clock'; #修改为并行复制

④调整master节点服务器DDL速度还有就是主库是写，对数据安全性较高，比如sync_binlog=1，innodb_flush_log_at_trx_commit= 1 之类的设置，而slave则不需要这么高的数据安全，完全可以讲sync_binlog设置为0或者关闭binlog，innodb_flushlog也可以设置为0来提高sql的执行效率。另外就是使用比主库更好的硬件设备作为slave

3、关于从库show slave status 中的Retrieved_Gtid_Set 和 Executed_Gtid_Set.

Retrieved_Gtid_Set ： 从库已经接收到主库的事务编号

Executed_Gtid_Set ： 从库自身已经执行的事务编号

4、查看server-uuid

```
mysql> show variables like '%uuid%';
+---------------+--------------------------------------+
| Variable_name | Value                                |
+---------------+--------------------------------------+
| server_uuid   | cd5d521b-b1c7-11e9-a1dc-000c2980d8b3 |
+---------------+--------------------------------------+
1 row in set (0.01 sec)
```

5、备份参数

--single-transaction: 基于此选项能实现热备InnoDB表；因此，不需要同时使用--lock-all-tables;

--master-data=2 记录备份那一时刻的二进制日志的位置，并且注释掉，1是不注释的

--databases hellodb 指定备份的数据库

真正在生产环境中，我们应该导出的是整个mysql服务器中的数据，而不是单个库，因此应该使用--all-databases

6、my.cnf开启gtid 部分配置

binlog_format = ROW //建议row

log-slave-updates=true //在从服务器进入主服务器传入过来的修改日志所使用，在Mysql5.7之前主从架构上使用gtid模式的话，必须使用此选项，在Mysql5.7取消了，会增加系统负载。

enforce-gtid-consistency=true // 强制gtid一直性，用于保证启动gitd后事务的安全;

gtid-mode=on //开启gtid模式

master_info_repository=TABLE

relay_log_info_repository=TABLE //指定中继日志的存储方式，默认是文件，这样配置是使用了 两个表，是INNODB存储引擎，好处是当出现数据库崩溃时，利用INNODE事务引擎的特点，对这两个表进行恢复，以保证从服务器可以从正确位置恢复数据。

sync-master-info=1 //同步master_info,任何事物提交以后都必须要把事务提交以后的二进制日志事件的位置对应的文件名称，记录到master_info中，下次启动自动读取，保证数据无丢失

slave-parallel-workers=2 //设定从服务器的启动线程数，0表示不启动

binlog-checksum=CRC32　　//主服务端在启动的时候要不要校验bin-log本身的校验码

master-verify-checksum=1　　//都是在服务器出现故障情况下，读取对服务器有用的数据

slave-sql-verify-checksum=1

binlog-rows-query-log_events=1　　//启用后，可用于在二进制日志记录事件相关信息，可降低故障排除复杂度（并非强制启动）

7 mysqldump的关键参数

-B：指定多个库，在备份文件中增加建库语句和use语句

--compact：去掉备份文件中的注释，适合调试，生产场景不用

-A：备份所有库

-F：刷新binlog日志

--master-data：在备份文件中增加binlog日志文件名及对应的位置点

-x --lock-all-tables：锁表

-l：只读锁表

-d：只备份表结构

-t：只备份数据

--single-transaction：适合innodb事务数据库的备份

InnoDB表在备份时，通常启用选项--single-transaction来保证备份的一致性，原理是设定本次会话的隔离级别为Repeatable read，来保证本次会话（也就是dump）时，不会看到其它会话已经提交了的数据。