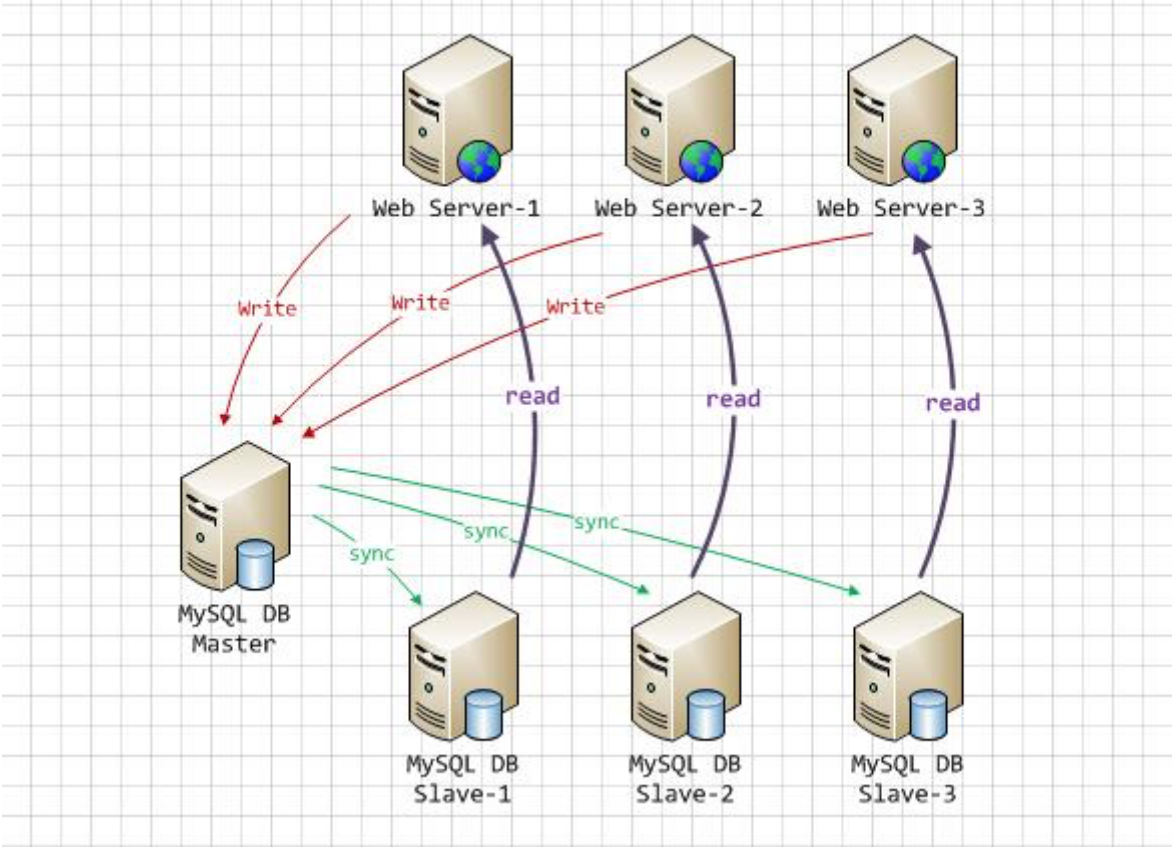
MySQL主从复制

一. 通过bin-log方式进行主从复制

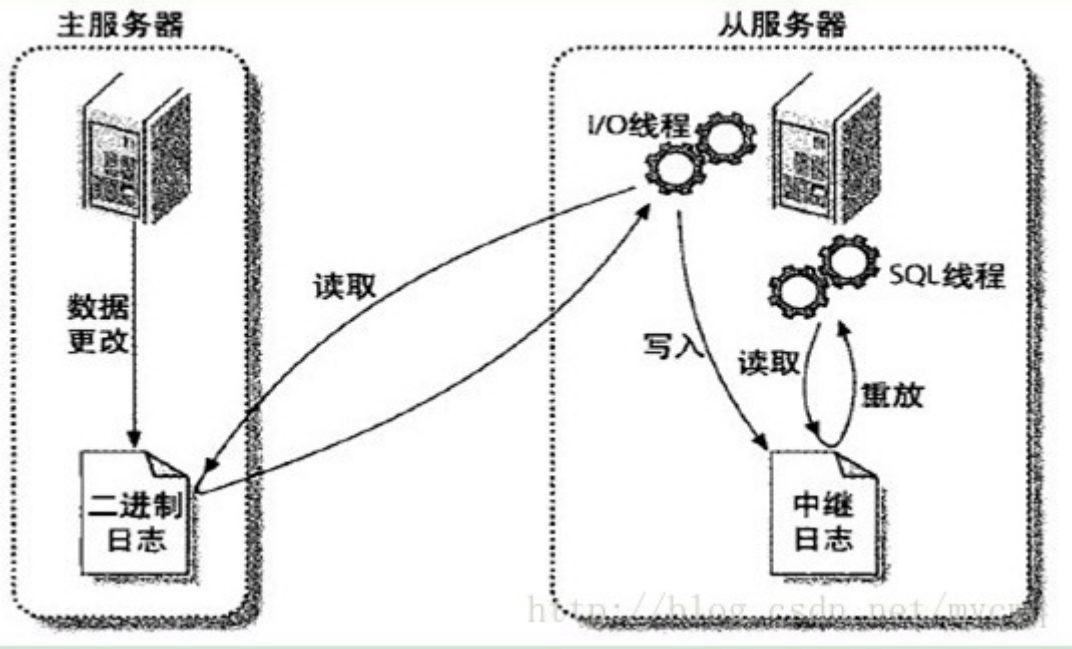
概述：

MySQL数据库自身提供的主从复制功能可以方便的实现数据的多处自动备份，实现数据库的拓展. 多个数据备份不仅可以加强数据的安全性，通过实现读写分离还能进一步提升数据库的负载性能.

数据库读写分离模型图：



在一主多从的数据库体系中，多个从服务器采用异步的方式更新主数据库的变化，业务服务器在执行写或者相关修改数据库的操作是在主服务器上进行的，读操作则是在各从服务器上进行. 主从复制的方式有两种，一种称之为Bin-log，一种为gtid. Mysql主从复制的实现原理图大致如下：



MySQL之间数据复制的基础是二进制日志文件(binary log file)，一台MySQL数据库一旦启用二进制日志后，其作为master，它的数据库中所有操作都会以"事件"的方式记录在二进制日志中，其他数据库作为slave通过一个I/O线程与主服务器保持通信，并监控master的二进制日志文件的变化，如果发现master二进制日志文件发生变化，则会把变化复制到自己的中继日志中，然后slave的一个SQL线程会把相关的"事件"执行到自己的数据库中，以此实现从数据库和主数据库的一致性，也就实现了主从复制.

1.2 主从复制配置流程：

主服务器：
    * 开启二进制日志
    * 配置唯一的server-id
    * 获得master二进制日志文件名及位置
    * 创建一个用于slave和master通信的用户账号

从服务器：
    * 配置唯一的server-id
    * 使用master分配的用户账号读取master二进制日志
    * 启用slave服务

1.3 开始进行主从复制

实验拓扑：
    Mysql Master: 172.16.70.233
    Mysql Slave: 172.16.70.241

Master服务器配置

1.3.1 修改Mysql配置文件/etc/my.cnf，在配置文件中插入：
    [mysqld]
    log-bin=mysql-bin
    server-id=1

1.3.2 重启mysql并创建用于主从复制的用户账号

```
[root@gitlab my.cnf.d]# systemctl restart mysqld
MariaDB [(none)]> CREATE USER 'repl'@'%' IDENTIFIED BY 'repl';     #创建账号
MariaDB [(none)]> GRANT REPLICATION SLAVE ON *.* TO 'repl'@'%';    #为账号添加主从复制权限
mysql>flush privileges;    #刷新权限
```

1.3.3 查看master状态，记录二进制文件名(mysql-bin.000002)和位置(507)：

```
MariaDB [(none)]> show master status;
+------------------+----------+--------------+------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+------------------+----------+--------------+------------------+
| mysql-bin.000002 |      507 |              |                  |
+------------------+----------+--------------+------------------+
1 row in set (0.00 sec)
```

SLAVE服务器配置

1.3.4 修改/etc/my.cnf配置文件，加入以下内容：

```
    [mysqld]
    server-id=2  #设置server-id 主从复制集群内部必须为1
```

1.3.5 重启mysql，打开mysql会话，执行同步SQL语句(需要主服务器主机名 登陆凭据 二进制文件的名称和位置)：

```
    MariaDB [(none)]> CHANGE MASTER TO MASTER_HOST='172.16.70.233',
MASTER_USER='repl',  MASTER_PASSWORD='repl', MASTER_LOG_FILE='mysql-bin.000001',
MASTER_LOG_POS=762;

    MariaDB [(none)]> start slave;
```

1.3.6 查看slave状态

```
MariaDB [(none)]> show slave status\G;
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 172.16.70.233
                  Master_User: repl
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: mysql-bin.000002
          Read_Master_Log_Pos: 507
               Relay_Log_File: kube-node1-relay-bin.000002
                Relay_Log_Pos: 555
        Relay_Master_Log_File: mysql-bin.000002
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Master_Log_Pos: 507
              Relay_Log_Space: 869
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Master_SSL_Allowed: No
           Master_SSL_CA_File:
           Master_SSL_CA_Path:
              Master_SSL_Cert:
```

```
            Master_SSL_Cipher:
               Master_SSL_Key:
        Seconds_Behind_Master: 0
Master_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 0
                Last_IO_Error:
               Last_SQL_Errno: 0
               Last_SQL_Error:
  Replicate_Ignore_Server_Ids:
             Master_Server_Id: 1
               Master_SSL_Crl:
           Master_SSL_Crlpath:
                   Using_Gtid: No
                  Gtid_IO_Pos:
      Replicate_Do_Domain_Ids:
  Replicate_Ignore_Domain_Ids:
                Parallel_Mode: conservative
                    SQL_Delay: 0
          SQL_Remaining_Delay: NULL
      Slave_SQL_Running_State: Slave has read all relay log; waiting for the slave
I/O thread to update it
1 row in set (0.00 sec)

ERROR: No query specified

MariaDB [(none)]>
```

当Slave_IO_Running和Slave_SQL_Running都为YES的时候就表示主从同步设置成功了，接下来就可以进行一些验证了，比如在主master数据库的test数据库的一张表中插入一条数据，在slave的test库的相同数据表中查看是否有新增的数据即可验证主从复制功能是否有效，还可以关闭slave(mysql>stop slave);然后再修改master，看slave是否也相应修改(停止slave后，master的修改不会同步到slave)，就可以完成主从复制功能的验证了.

Master开启二进制日志后默认记录所有库所有表的操作，可以通过配置来指定只记录指定的数据库甚至指定的表的操作，具体在mysql配置文件的[mysqld] 可添加修改如下选项：

```
#不同步哪些数据库
binlog-ignore-db = mysql
binlog-ignore-db = test
binlog-ignore-db = information_schema

#只同步哪些数据库，除此之外，其他不同步
binlog-do-db = game
```

In [ ]:

二. 通过GTID方式进行主从复制

2.1 什么是GTID

从MySQL5.6.5开始新增了一种基于GTID的主从复制方式，通过GTID保证了每个在主库上提交的事务在集群中有一个唯一的ID，这种方式强化了数据库的主备一致性，故障恢复以及容错能力. 在原来基于二进制日志的复制中，从库需要告知主库要从哪个偏移量进行增量同步，如果指定错误会造成数据的遗漏，从而造成数据的不一致. 借助GTID，在发生主备切换的情况下，MySQL的其它从库可以自动在新主库上找到正确的复制位置，这大大简化了复杂复制拓扑下集群的维护，也减少了人为设置复制位置发生误操作的风险. 另外，基于GTID的复制可以忽略已经执行过的事务，减少了数据发生不一致的风险.

GTID(Global Transaction ID) 是对于一个已提交事务的编号，并且是一个全局唯一的编号. GTID实际上是由 UUID+TID 组成的，其中 UUID 是一个MySQL实例的唯一标识，TID代表了该实例上已经提交的事务数量，并且随着事务提交单调递增. 下面是一个GTID的具体形式：

```
mysql> show master status;
+------------------+----------+--------------+------------------+-------------------------------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set                         |
+------------------+----------+--------------+------------------+-------------------------------------------+
| master-bin.000001 |     1065 |              |                  | 2ba1494b-6709-11e9-befd-000c293043a7:1-5 |
+------------------+----------+--------------+------------------+-------------------------------------------+
1 row in set (0.00 sec)
```

2.2 GTID的作用

GTID 的使用不单单是用单独的标识符替换旧的二进制日志文件和位置，它也采用了新的复制协议．旧的协议往往简单直接，首先从服务器上在一个特定的偏移量位置连接到主服务器上一个给定的二进制日志文件，然后主服务器再从给定的连接点开始发送所有的事件．

新协议稍有不同：支持以全局统一事务ID(GTID)为基础的复制，当在主库上提交事务或者被从库应用时，可以定位和追踪每一个事务．GTID复制是全部以事务为基础，使得检查主从一致性变得非常简单．如果所有主库上提交的事务也同样提交到从库上，一致性就得到了保证．GTID相关操作，默认情况下将一个事务记录进二进制文件时，首先记录它的GTID，而且GTID和事务相关信息一并要发送给从服务器，由从服务器在本地应用认证，但是绝对不会改变原来的事务ID号．因此在GTID的架构上就算有了N层架构，复制是N级架构．事务ID依然不会改变，有效的保证了数据的完整和安全性．

2.3 配置GTID主从复制

实验环境：
master: 172.16.70.221
slave: 172.16.70.222

2.3.1 修改Master数据库的配置文件 /etc/my.cnf

```
[mysqld]
server-id=1
binlog_format=row
gtid_mode=on
enforce_gtid_consistency=true
slave-parallel-workers=2
binlog-checksum=CRC32
slave-sql-verify-checksum=1
master-verify-checksum=1
report-port=3306
report-host=master.contoso.com
log-slave-updates=true
master-info-repository=TABLE
relay-log-info-repository=TABLE
binlog-rows-query-log_events=1
sync-master-info=1
log-bin=master-bin
```

2.3.2 修改Slave数据库的配置文件/etc/my.cnf

```
[mysqld]
binlog_format=row
log-bin=master-bin
log-slave-updates=true
gtid_mode=on
enforce_gtid_consistency=true
master-info-repository=TABLE
relay-log-info-repository=TABLE
sync-master-info=1
slave-parallel-workers=2
server-id=3322548
report-port=3306
```

```
report-host=slave3322548.contoso.com
binlog-checksum=CRC32
master-verify-checksum=1
slave-sql-verify-checksum=1
binlog-rows-query-log_events=1
```

参数解释：

```
server-id=1 #主从复制集群中的唯一标识
binlog_format #你可以使用基于语句的或基于行的复制与GTID，但是为了获得最佳效果我们建议你使用基于行的格式
gtid_mode=on  #开启GTID
enforce_gtid_consistency=true #GTID强一致
slave-parallel-workers=2  #设定从服务器的SQL线程数，0表示关闭多线程复制功能
binlog-checksum=CRC32
master-verify-checksum=1
slave-sql-verify-checksum=1  #启动与复制有关的校验功能
log-slave-updates=true
report-port=3306
report-host=master.contoso.com #主从复制时使用的名字与端口
master-info-repository=TABLE
relay-log-info-repository=TABLE #用于实现在崩溃时保证二进制及从服务器安全的功能
binlog-rows-query-log_events=1 #用于在二进制日志记录事件相关的信息，可降低故障排除的复杂度
sync-master-info=1 #启用之可确保无信息丢失
log-bin=master-bin #启用二进制日志，这是保证复制功能的基本前提
```

2.3.3 在Master服务器上创建复制账号并设置权限

```
mysql>  CREATE USER 'repl'@'%' IDENTIFIED BY 'repl';    #创建账号
mysql>  GRANT REPLICATION SLAVE ON *.* TO 'repl'@'%';   #为账号添加主从复制权限
mysql>  flush privileges;   #刷新权限
```

2.3.4 验证主库gtid相关配置

```
mysql> show variables like "%gtid%";
+---------------------------------+-----------+
| Variable_name                   | Value     |
+---------------------------------+-----------+
| binlog_gtid_simple_recovery     | ON        |
| enforce_gtid_consistency        | ON        |
| gtid_executed_compression_period | 1000      |
| gtid_mode                       | ON        |
| gtid_next                       | AUTOMATIC |
| gtid_owned                      |           |
| gtid_purged                     |           |
| session_track_gtids             | OFF       |
+---------------------------------+-----------+
8 rows in set (0.00 sec)
```

```
mysql> show variables like '%gtid_next%';
+---------------+-----------+
| Variable_name | Value     |
+---------------+-----------+
| gtid_next     | AUTOMATIC |
+---------------+-----------+
1 row in set (0.00 sec)
```

```
mysql> show master status;
+------------------+----------+--------------+------------------+-------------------------------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set                         |
+------------------+----------+--------------+------------------+-------------------------------------------+
| master-bin.000001 |     1065 |              |                  | 2ba1494b-6709-11e9-befd-000c293043a7:1-5 |
+------------------+----------+--------------+------------------+-------------------------------------------+
1 row in set (0.00 sec)
```

2.3.5 从库连接主库复制

```
mysql> CHANGE MASTER TO
MASTER_HOST='172.16.70.221',MASTER_USER='repl',MASTER_PASSWORD='repl',MASTER_AUTO_
POSITION=1;
mysql> start slave;
```

2.3.6 验证主从复制

1) 在Master数据库上查看slave

```
mysql> show slave hosts
    -> ;
+-----------+-------------------------+------+-----------+--------------------------------------+
| Server_id | Host                    | Port | Master_id | Slave_UUID                           |
+-----------+-------------------------+------+-----------+--------------------------------------+
|   3322548 | slave3322548.contoso.com | 3306 |         1 | ef8d134c-6709-11e9-8521-000c295f0a0b |
+-----------+-------------------------+------+-----------+--------------------------------------+
1 row in set (0.00 sec)
```

2)在Slave数据库上查看状态

```
mysql> show slave status\G;
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 172.16.70.221
                  Master_User: repl
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: master-bin.000001
          Read_Master_Log_Pos: 1065
               Relay_Log_File: haproxy2-relay-bin.000002
                Relay_Log_Pos: 417
        Relay_Master_Log_File: master-bin.000001
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
                ......
```

```
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
```

以上两项都为YES则代表主从复制配置成功!