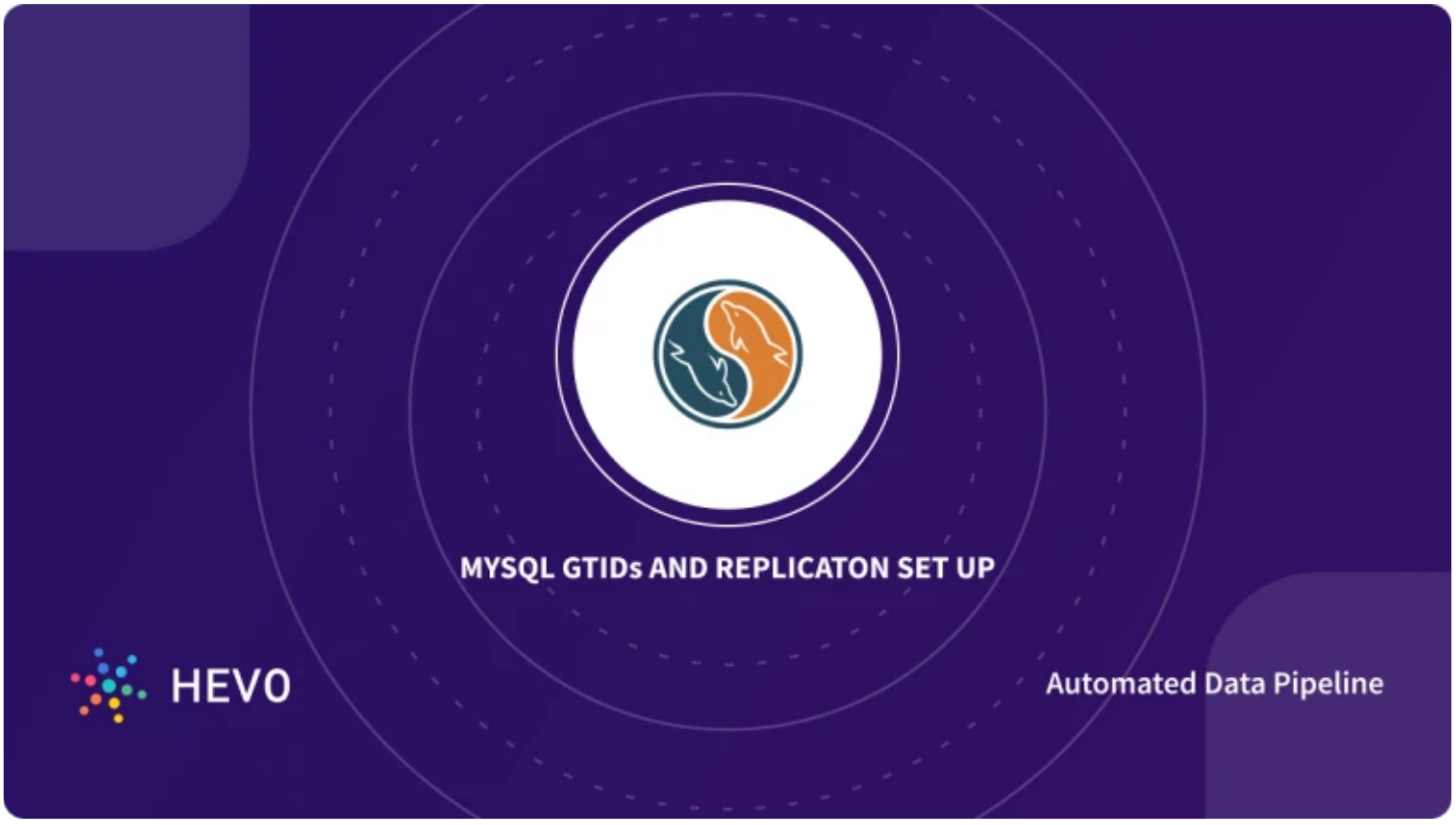
MySQL GTIDs Replication Set Up: Made Easy

Shruti Garg on Tutorials • August 17th, 2020 • **WRITE FOR HEVO**



Are you trying to set up MySQL replication for your database? If that’s the case, you have landed on the right page. A great way to set up replication for your database is by using MySQL GTIDs. The present article aims at providing a step-by-step guide to help you set up MySQL Replication using GTIDs and help you replicate your MySQL data with ease. A complete walkthrough of the content will help you develop the skill to set up MySQL Replication in a matter of minutes.

Table of Contents

Introduction to MySQL

MySQL is an open-source database management system that is distributed, supported and developed by Oracle Incorporation (INC). MySQL is a relational database which stores the data in the form of tables and views. It supports various database objects such as tables, stored procedures, functions, triggers, views, indexes, and even cursors.

MySQL database server runs comfortably on any laptop or desktop even with various web applications or servers installed on the system. MySQL database server is designed to support large databases which may contain data of many organizations. MySQL database servers support a wide range of functions and multiple web API’s.

It performs exceptionally well & securely accesses various databases on the internet. It ensures connectivity with servers and devices at all times.

For further information, you can check the official MySQL website here.

Introduction to MySQL GTIDs

GTID stands for global transaction identifier (GTID) which uniquely identifies a transaction committed on the server of origin (master). A unique GTID is created when any transaction occurs. The GTID is not just unique to the server on which it originates, but also across the servers in any given replication setup. In other words, each transaction is mapped to a GTID.

MySQL GTIDs are displayed as a pair of coordinates, separated by a colon character (:), as shown here:

```
GTID = source_id:transaction_id
```


Understanding MySQL GTIDs Replication & Why You Should Use It

MySQL, starting with version 5.6 and above, allows users to make use of GTIDs to set up replication for

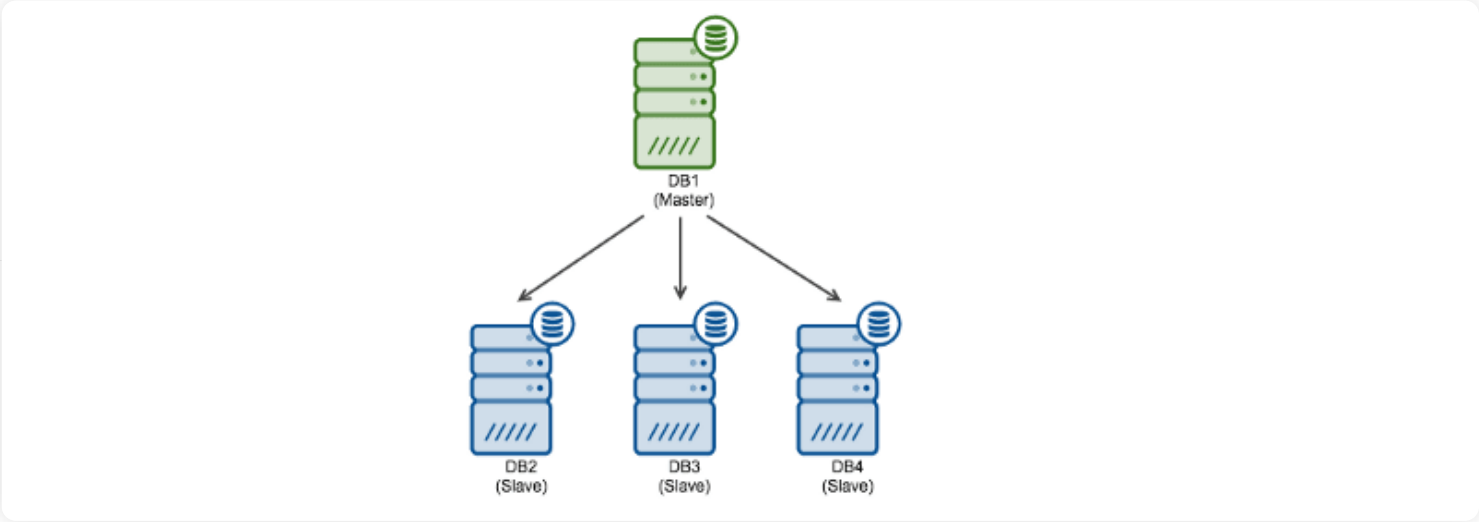
their MySQL database. Each GTID acts as a unique Id that MySQL generates and associates with every transaction that takes place within your database. It makes use of a unique transaction number and master server's UUID to identify each transaction. It thereby ensures a simple failover and recovery process by keeping track of each replication-based transaction between the master and slave server.

When the replication takes place, the slave makes use of the same GTIDs, irrespective of whether it acts as a master for any other nodes or not. With each transaction replication, the same GTIDs and transaction numbers also come along from the master and the slave will write these to the binlog if it's configured to write its data events.

To ensure a smooth, consistent and fault-tolerant replication, the slave will then inform the master of the GTIDs that were a part of the execution, which helps master node identify if any transaction did not take place. The master node then informs the slave to carry out the left-over transactions and thereby ensures that data replication takes place accurately.

This is how MySQL GTIDs Replication provides an efficient way to start replicating your MySQL data.

Using MySQL GTIDs for Replication



You can set up MySQL GTIDs Replication using the following steps to replicating your MySQL data:

- Step 1: Synchronizing Master & Slave Servers
- Step 2: Stopping Master & Slave Servers
- Step 3: Configuring the Master Server
- Step 4: Creating a Replication User
- Step 5: Using mysqldump to Back up Master Server
- Step 6: Configuring the Master Server
- Step 7: Executing the Change Master
- Step 8: Verifying the Replication Process

Step 1: Synchronizing Master & Slave Servers

Synchronize both servers by setting them to read-only if the replication is running already by using the following command:

```
mysql> SET @@GLOBAL.read_only = ON;
```

You must allow all ongoing transactions to commit or roll back first. Then, you may allow the slave to catch up with the master.

Step 2: Stopping Master & Slave Server

Both the Master and Slave Servers should be stopped by using the following command:

```
sudo service mysqld stop
```

Step 3: Configuring the Master Server

The Master server needs to be started with GTID mode enabled by setting the **gtid_mode variable** to ON. It is also essential that the **enforce_gtid_consistency** variable is enabled to make sure that only the statements which are safe for MySQL GTIDs Replication are logged. These changes can be made by accessing the given file:

```
vi /etc/my.cnf
```

The following parameters should be added under the [mysqld] section of my.cnf file

```
server-id = 1
log-bin = mysql-bin
binlog_format = row
gtid-mode=ON
enforce-gtid-consistency
log-slave-updates
```

The next step is to start the mysqld server process, using the following command:

```
sudo service mysqld start
```

Step 4: Creating a Replication User

You must now create a replication user for the slave server through the given command:

```
create user 'repl_user'@'%' identified by 'XXXXXXXXXX';
Grant replication slave on *.* to 'repl_user'@'%';
```

User must be created with required user name, password and permissions.

Use the following command to get the master binary log coordinates:

```
mysql> show master status ;
```

```
+------------------+----------+--------------+------------------+
-------------------------------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
Executed_Gtid_Set                          |
+------------------+----------+--------------+------------------+
-------------------------------------------+
| mysql-bin.000001 |     1195 |              |                  |
82e9927e-7645-11e9-8e8d-0e651b9198f4:1-5 |
+------------------+----------+--------------+------------------+
-------------------------------------------+
```

```
mysql> show global variables like 'gtid_executed';
```

```
+---------------+-----------------------------------------+
| Variable_name | Value                                   |
+---------------+-----------------------------------------+
| gtid_executed | 82e9927e-7645-11e9-8e8d-0e651b9198f4:1-5 |
+---------------+-----------------------------------------+
```

Step 5: Using mysqldump to Back up Master Server

In this example, we will be using mysqldump to back up the Master. You can do this by using the following command:

```
mysqldump --all-databases -flush-privileges --single-transaction --flush-logs --triggers --routine
# head -n30 mysqlbackup_dump.sql
-- MySQL dump 10.13  Distrib 5.7.26, for Linux (x86_64)
--
-- Host: 54.89.242.211    Database:
-- ------------------------------------------------------
-- Server version       5.7.26-log
/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
```

```
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;
SET @MYSQLDUMP_TEMP_LOG_BIN = @@SESSION.SQL_LOG_BIN;
SET @@SESSION.SQL_LOG_BIN= 0;
--
-- GTID state at the beginning of the backup
--SET @@GLOBAL.GTID_PURGED='82e9927e-7645-11e9-8e8d-0e651b9198f4:1-5';
```

Step 6: Configuring the Master Server

Once the Master has been configured, the next step is to configure the Slave server: The following parameters must be added under the [mysqld] section of my.cnf file:

```
server-id = 2
log-bin = mysql-bin
relay-log = relay-log-server
relay-log = relay-log-server
read-only = ON
gtid-mode=ON
enforce-gtid-consistency
log-slave-updates
```

Additionally, you can also start slaves with the –skip-slave-start option before configuring the slave settings.

Now you may start the mysqld server process using the given command:

```
sudo service mysqld start
```

Next step is to load the mysqldump in slave server:

```
mysql> show global variables like 'gtid_executed';
```

```
+-----------------+--------+
| Variable_name | Value |
+-----------------+--------+
| gtid_executed |  |
+-----------------+--------+
```

```
mysql> source mysqlbackup_dump.sql ;
mysql> show global variables like 'gtid_executed';
```

```
+---------------+-----------------------------------------+
| Variable_name | Value                                   |
+---------------+-----------------------------------------+
| gtid_executed | 82e9927e-7645-11e9-8e8d-0e651b9198f4:1-5 |
+---------------+-----------------------------------------+
```

Step 7: Executing the Change Master

The slave should be configured to use the master with GTID based transactions as the source for data replication and to use GTID-based auto-positioning rather than file-based positioning.

```
CHANGE MASTER TO
MASTER_HOST = '54.89.xx.xx',
MASTER_PORT = 3306,
MASTER_USER = 'repl_user',
MASTER_PASSWORD = 'XXXXXXXXX',
MASTER_AUTO_POSITION = 1;
```

It is now time to begin the slave process:

```
start slave;
```

Step 8: Verifying the Replication Process

The final step is the check your replication process using the given command:

```
mysql> show slave status G
```

You must also disable the read-only mode if it is enabled:

```
mysql> SET @@GLOBAL.read_only = OFF;
```

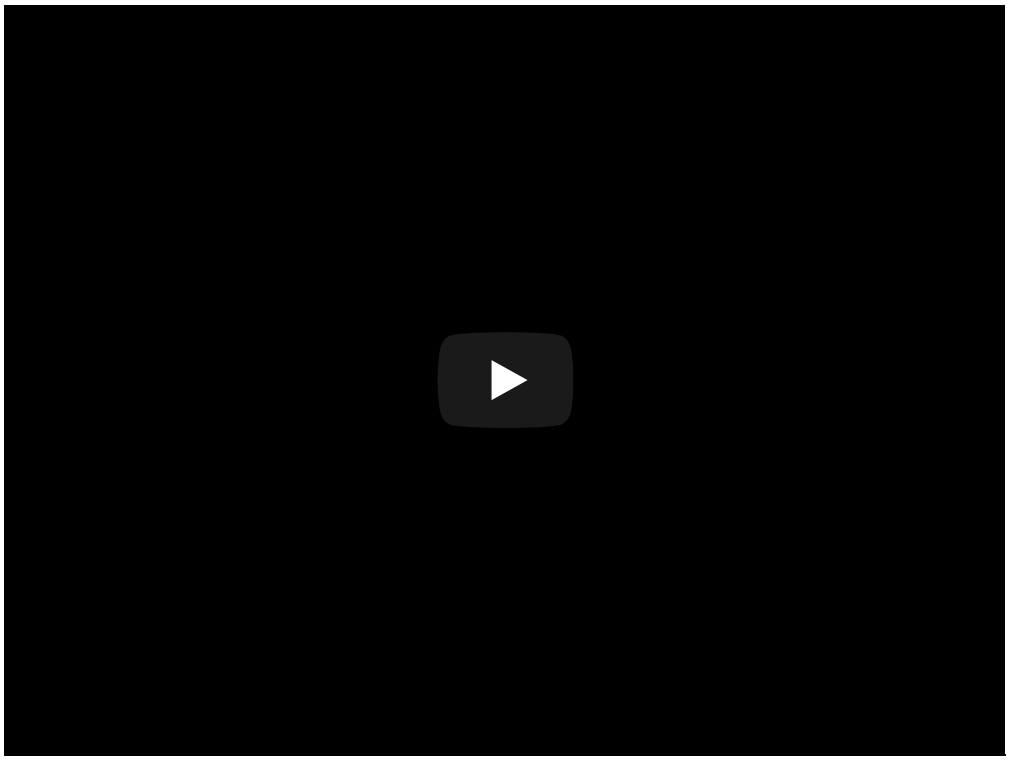
This is how you can set up MySQL GTIDs Replication with ease.

Conclusion

In this blog post, you have learnt how to set up MySQL GTIDs Replication using GTID. However, if you are tired of making compromises and want a simple, easy-to-use, and completely managed MySQL database replication experience, you should have a look at what Hevo offers.

Hevo is a No-code Data Pipeline. It is a fully automated platform. Hevo also offers pre-built integrations from 100+ data sources at an amazing price.

Sign up for a 14-day free trial and experience efficient and effective ETL.



Have any further queries? Get in touch with us in the comments section below.