Using ClustrixDB as a Replication Slave

The following topics describe configuring ClustrixDB as a Slave to a MySQL Master.

Creating and Maintaining a Slave Configuration

ClustrixDB supports multiple Slave processes, each with its own configuration. Because this functionality exceeds that provided by MySQL, ClustrixDB extends replication syntax to support a named Slave configuration. Each replication Slave operates independently. If one Slave encounters an error and stops, other Slaves are unaffected. Ensure that Slaves do not update the database in a conflicting manner. To determine where replication for a specified slave is running, query the nodeid column in the system.mysql_slave_stats table.

For compatibility with MySQL, the standard syntax operates on a Slave named "default."

For example, operating on the "default" Slave, you can issue the following commands. Use the login information specifically established for the replication process when specifying MASTER_USER and MASTER_PASSWORD.

```
slave> STOP SLAVE;
slave> CHANGE MASTER TO MASTER_LOG_FILE =
'master_log_name',
                        MASTER_LOG_POS = 1
234,
                        MASTER_HOST = 'hos
t_name',
                        MASTER_USER = 'use
r_name',
                        MASTER_PASSWORD =
'password',
                        MASTER_PORT = 3306
;
slave> START SLAVE;
slave> SHOW SLAVE STATUS;
```

Note the use of the CHANGE MASTER command in the preceding example. This command always refers to the default Slave and is included for compatibility with MySQL. CHANGE MASTER is an alias for CHANGE SLAVE 'default'.

To define named Slaves for multiple Slave instances, use the following syntax. The login information for MASTER_USER and MASTER_PASSWORD should be that of the established replication user.

```
slave> CREATE SLAVE 'slave_name' MASTER_LOG_FILE =
'master_log_name',
                                   MASTER_LOG_POS = n
nn
                                [, MASTER_HOST = 'str
ing']
                                [, MASTER_USER = 'str
ing']
                                [, MASTER_PASSWORD =
'string']
                                [, MASTER_PORT = nnn]
;
```

The following code creates a new slave configuration entry with the specified Master settings. You use the specified slave_name in commands that you issue to control or monitor this slave instance.

```
slave> STOP SLAVE 'foo'; (if applicable)
slave> CREATE SLAVE 'foo' MASTER_LOG_FILE =
'my_master_log',
                          MASTER_LOG_POS = 1
234,
                          MASTER_HOST = 'myh
ost',
                          MASTER_USER = 'rep
lication',
                          MASTER_PASSWORD =
'clustrix',
                          MASTER_PORT = 3306
;
slave> START SLAVE 'foo';
slave> SHOW SLAVE STATUS 'foo';
```

To update the configuration of the Slave, specifying settings such as hostname, port, log sequence, and log filename, issue the following command:

```
slave> CHANGE SLAVE 'slave_name'
TO {master_option} [,
{master_option}];
```

Starting and Stopping a Slave Process

To start a specified Slave, issue the following command:

```
slave> START SLAVE
'slave_name';
```

If the Slave is already running, this command has no effect. To start any Slaves that are configured on this cluster and not already running, issue the following command. To load-balance replication traffic, Slaves are started in a round-robin fashion across the nodes in the cluster which means that the slaves are not consistently bound to one specific node. This doesn't have an impact to replication as the slave is configured to communicate to a specific master host, but replication errors are logged on the node that the slave process is running from.

```
slave> START SLAVE ALL;
```

To stop a specified Slave, issue the following command:

```
slave> STOP SLAVE
'slave_name';
```

If the Slave is not running, this command has no effect.

To stop all running Slaves, issue the following command:

```
slave> STOP SLAVE ALL;
```

Alternatively, ClustrixDB also supports the legacy MySQL Syntax for SLAVE START and SLAVE STOP.

Displaying the Status of a Slave

To display the status of a specified Slave, issue the following command.

```
slave> SHOW SLAVE STATUS
'slave_name';
```

To display the status of all Slaves, issue the following command.

```
slave> SHOW SLAVE STATUS;
```

Skipping Statements for a Stopped Slave

To skip one or more pending replication statements (for example, when dealing with a bad query), issue the following command:

```
slave> START SLAVE [
'slave_name'] SKIP N;
```

where N is the number of statements to skip. The Slave skips the specified number of statements and attempts to execute the following statements. If the statement after N+1 fails for any reason, the Slave remains in the same state as before the SKIP command was issued. For example, if there are three consecutive failing queries in a row, SKIP 1 or SKIP 2 has no effect on the Slave position, but SKIP 3 enables replication to resume.

Specifying Ignored and Included Databases

The ClustrixDB replication Slave supports the following modes:

- Replicate everything, skipping specified databases (default)
- Replicate only specified databases, skipping everything else

The following examples, using databases named "one", "two" and "three", illustrate the modes. In the following example, statements for database "one" and "three" are executed. Statements for database "two" are skipped.

```
slave> SET GLOBAL mysql_default_db_replication_policy = true;
slave> INSERT INTO system.mysql_slave_db_replication_policy (slave_name, dbname, allow) VALUES ('foo', 'one',
true);
slave> INSERT INTO system.mysql_slave_db_replication_policy (slave_name, dbname, allow) VALUES ('foo', 'two',
false);
```

In the following example, statements for database "one" are executed. Statements for databases "two" and "three" are skipped.

```
slave> SET GLOBAL mysql_default_db_replication_policy = false;
slave> INSERT INTO system.mysql_slave_db_replication_policy (slave_name, dbname, allow) VALUES ('foo', 'one',
true);
slave> INSERT INTO system.mysql_slave_db_replication_policy (slave_name, dbname, allow) VALUES ('foo', 'two',
false);
```


- These "ignore" and "include" directives affect named Slave instances globally.
- Only the current database of a statement determines whether it is executed or skipped as part of this policy.
- Rows in which the allow column matches the policy variable are treated as if they are not present (that is, according to the policy).
- Stop the Slave manually before changing policies. This configuration is read when the Slave starts (and periodically when its buffer fills).

Controlling Slave Behavior on Errors

ClustrixDB offers two ways to control slave behavior on errors:

- Inserting error codes into system.mysql_slave_skip_errors
- Setting the slave_exec_mode global variable

Inserting error codes into system.mysql_slave_skip_errors

You can configure the slave not stop on certain errors by inserting the desired error code into the table system.mysql_slave_skip_errors. This applies to all slaves: it cannot be configured per-slave.
Note: Insert the mysql_error_code, NOT the result_code into mysql_slave_skip_errors.

To obtain a list of available error codes, run the following query:

```
slave> select * from system.mysql_error_codes natural join error_codes;
+-------------+------------------+---------+----------------------
+-------------------------------------------------+
| result_code | mysql_error_code | family  | code                  |
message                                           |
+-------------+------------------+---------+----------------------
+-------------------------------------------------+
|        7168 |             1452 | dml     | DML_FK_INSERT_ERR     | Foreign key constraint violation on
insert       |
|       60417 |             1213 | lockman | LOCKMAN_RC_DEADLOCK   | Lock manager deadlock
detected                                          |
|       26645 |             1062 | rigr    | RIGR_RC_DUPLICATE_KEY | Duplicate key in
representation                                    |
|       12309 |             1061 | ddl     | DDL_RC_INDEX_CONFLICT | Index name
conflict                                          |
|        7172 |             1701 | dml     | DML_FK_TRUNCATE_ERR   | Foreign key contraint on
truncate                                          |
|       11267 |             1049 | trans   | NO_SUCH_DATABASE      | No such
database                                          |
|       11281 |             1044 | trans   | DB_PERMISSION_DENIED  | Insufficient user permissions to access
database |
|       12307 |             1007 | ddl     | DDL_RC_DB_CONFLICT    | Database name
conflict                                          |
|       34816 |             1064 | parser  | ERRCODE_SYNTAX_ERROR  | syntax
error                                             |
|        7171 |             1048 | dml     | DML_NOTNULL_ERR       | NOT NULL constraint
violation                                         |
...
```

The only errors supported for Row-Based_Replication (RBR) are 1451 and 1452 (Cannot delete or update parent row, Cannot add or update child row, respectively).

To insert an error code into system.mysql_slave_skip_errors, stop the slaves and then run a query such as the following that inserts error code 1062 into the table:

```
slave> insert into system.mysql_slave_skip_errors values (1062);
Query OK, 1 row affected (0.02 sec)
```

Setting the slave_exec_mode global variable

MySQL row-based replication specifies the expected (pre-update) contents of each row to be updated or deleted. In case the row specified does not exactly match the row present on ClustrixDB, there are two modes of operation, determined by the setting of the global variable slave_exec_mode:

- STRICT: In this mode, as long as the PRIMARY KEY or first UNIQUE key of the row matches, the update/delete is applied. This is the default setting, and also matches MySQL's default behavior.
- EXACT: In this mode, if the row specified does not exactly match (for all columns) a row present in the table, the slave will error (with "Row Not Found"), and stop. Note that this mode is unique to ClustrixDB.

Replicating Identically-Named Databases

To replicate identically-named databases from different Masters, you can create rewrite rules that map one database name to another. To remap database names, add a rule of the form (slave_name, from_db, to_db) to the table system.mysql_slave_rewrite_db. The following example remaps "db1234" to "otherdb."

```
slave> INSERT INTO system.mysql_slave_rewrite_db VALUES('slaveA','db1234','otherdb');
Query OK, 1 row affected (0.02 sec)
```

- You can define only one remapping for any Slave/database combination.
- You must stop the Slave before creating or modifying a rule for it.
- You cannot perform cross-database updates.

Deleting a Slave Configuration

To delete a previously configured Slave, issue the following command:

```
slave> DROP SLAVE
'slave_name';
```

You must stop the Slave before deleting it.

Changing a Slave's Location

Use the following syntax to assign a slave to a different node. This may be useful to experiment with the effects of slave load on the cluster by manually moving a slave without causing a group change. (See Group Changes for more information.)

```
slave> CHANGE SLAVE
'slave_name' to
nodeid=nodeid;
```

Replicating from MariaDB

ClustrixDB does not provide support for MariaDB GTIDs, but can replicate from MariaDB 10.2+ using its support for old-style replication. The normal syntax for creating a replication slave can be used to set ClustrixDB up as a replication slave.

Replicating from MySQL 5.7

ClustrixDB does not provide support for MySQL GTIDs and ignores GTID info it encounters in binlogs.

When replicating from MySQL 5.7, the gtid and server_uuid variables will be ignored when encountered in the binlog.

Additional information is available at MySQL 5.7 Replication and GTIDs.