Python MySQL Replication Documentation

Release 0.14

Julien Duponchelle

Sep 11, 2017

Contents

1 Use cases **3**

2 Contents **5**

2.1 Installation 5
2.2 Limitations 5
2.3 BinLogStreamReader 6
2.4 Events 6
2.5 Examples 8
2.6 Support 8
2.7 Developement 8
2.8 Licence 9

3 Indices and tables **11**

Python Module Index **13**

Pure Python Implementation of MySQL replication protocol build on top of PyMYSQL. This allow you to receive event like insert, update, delete with their datas and raw SQL queries.

CHAPTER 1

Use cases

- MySQL to NoSQL database replication
- MySQL to search engine replication
- Invalidate cache when something change in database
- Audit
- Real time analytics

CHAPTER 2

Contents

Installation

Python MySQL Replication is available on PyPi. You can install it with:

```
pip install mysql-replication
```

Limitations

GEOMETRY

GEOMETRY field is not decoded you will get the raw data.

binlog_row_image

Only [binlog_row_image=full](http://dev.mysql.com/doc/refman/5.6/en/replication-options-binary-log.html#sysvar_binlog_row_image) is supported (it's the default value).

BOOLEAN and BOOL

Boolean is returned as TINYINT(1) because it's the reality.

http://dev.mysql.com/doc/refman/5.6/en/numeric-type-overview.html

Our discussion about it: https://github.com/noplay/python-mysql-replication/pull/16

BinLogStreamReader

class pymysqlreplication.binlogstream.**BinLogStreamReader**(*connection_settings*, *server_id*, *ctl_connection_settings=None*, *resume_stream=False*, *blocking=False*, *only_events=None*, *log_file=None*, *log_pos=None*, *filter_non_implemented_events=True*, *ignored_events=None*, *auto_position=None*, *only_tables=None*, *ignored_tables=None*, *only_schemas=None*, *ignored_schemas=None*, *freeze_schema=False*, *skip_to_timestamp=None*, *report_slave=None*, *slave_uuid=None*, *pymysql_wrapper=None*, *fail_on_table_metadata_unavailable=False*, *slave_heartbeat=None*)

Connect to replication stream and read event

class pymysqlreplication.binlogstream.**ReportSlave**(*value*)

Represent the values that you may report when connecting as a slave to a master. SHOW SLAVE HOSTS related

encoded(*server_id*, *master_id=0*)

server_id: the slave server-id master_id: usually 0. Appears as "master id" in SHOW SLAVE HOSTS on the master. Unknown what else it impacts.

Events

class pymysqlreplication.event.**BeginLoadQueryEvent**(*from_packet*, *event_size*, *table_map*, *ctl_connection*, ***kwargs*)

Attributes: file_id block-data

class pymysqlreplication.event.**ExecuteLoadQueryEvent**(*from_packet*, *event_size*, *table_map*, *ctl_connection*, ***kwargs*)

Attributes: slave_proxy_id execution_time schema_length error_code status_vars_length file_id start_pos end_pos dup_handling_flags

class pymysqlreplication.event.**GtidEvent**(*from_packet*, *event_size*, *table_map*, *ctl_connection*, ***kwargs*)

GTID change in binlog event

gtid

GTID = source_id:transaction_id Eg: 3E11FA47-71CA-11E1-9E33-C80AA9429562:23 See: http://dev.mysql.com/doc/refman/5.6/en/replication-gtids-concepts.html

class pymysqlreplication.event.**HeartbeatLogEvent**(*from_packet*, *event_size*, *table_map*, *ctl_connection*, ***kwargs*)

A Heartbeat event Heartbeats are sent by the master only if there are no unsent events in the binary log file for a period longer than the interval defined by MASTER_HEARTBEAT_PERIOD connection setting.

A mysql server will also play those to the slave for each skipped events in the log. I (baloo) believe the intention is to make the slave bump its position so that if a disconnection occurs, the slave only reconnects from the last skipped position (see Binlog_sender::send_events in sql/rpl_binlog_sender.cc). That makes 106 bytes of data for skipped event in the binlog. *this is also the case with GTID replication*. To mitigate such behavior, you are expected to keep the binlog small (see max_binlog_size, defaults to 1G). In any case, the timestamp is 0 (as in 1970-01-01T00:00:00).

Attributes: ident: Name of the current binlog

class pymysqlreplication.event.**IntvarEvent**(*from_packet*, *event_size*, *table_map*, *ctl_connection*, ***kwargs*)

Attributes: type value

class pymysqlreplication.event.**QueryEvent**(*from_packet*, *event_size*, *table_map*, *ctl_connection*, ***kwargs*)

This evenement is trigger when a query is run of the database. Only replicated queries are logged.

class pymysqlreplication.event.**RotateEvent**(*from_packet*, *event_size*, *table_map*, *ctl_connection*, ***kwargs*)

Change MySQL bin log file

Attributes: position: Position inside next binlog next_binlog: Name of next binlog file

class pymysqlreplication.event.**XidEvent**(*from_packet*, *event_size*, *table_map*, *ctl_connection*, ***kwargs*)

A COMMIT event

Attributes: xid: Transaction ID for 2PC

Row events

This events are send by MySQL when data are modified.

class pymysqlreplication.row_event.**DeleteRowsEvent**(*from_packet*, *event_size*, *table_map*, *ctl_connection*, ***kwargs*)

This event is trigger when a row in the database is removed

For each row you have a hash with a single key: values which contain the data of the removed line.

class pymysqlreplication.row_event.**TableMapEvent**(*from_packet*, *event_size*, *table_map*, *ctl_connection*, ***kwargs*)

This evenement describe the structure of a table. It's sent before a change happens on a table. An end user of the lib should have no usage of this

class pymysqlreplication.row_event.**UpdateRowsEvent**(*from_packet*, *event_size*, *table_map*, *ctl_connection*, ***kwargs*)

This event is triggered when a row in the database is changed

For each row you got a hash with two keys:

- before_values
- after_values

Depending of your MySQL configuration the hash can contains the full row or only the changes: http://dev.mysql.com/doc/refman/5.6/en/replication-options-binary-log.html#sysvar_binlog_row_image

class pymysqlreplication.row_event.**WriteRowsEvent**(*from_packet*, *event_size*, *table_map*, *ctl_connection*, ***kwargs*)

This event is triggered when a row in database is added

For each row you have a hash with a single key: values which contain the data of the new line.

Examples

You can found a list of working examples here: https://github.com/noplay/python-mysql-replication/tree/master/examples

Prerequisites

The user, you plan to use for the BinaryLogClient, must have REPLICATION SLAVE privilege. To get binlog filename and position, he must be granted at least one of REPLICATION CLIENT or SUPER as well. To get table info of mysql server, he also need SELECT privilege on information_schema.COLUMNS. We suggest grant below privileges to the user:

```
GRANT REPLICATION SLAVE, REPLICATION CLIENT, SELECT ON *.* TO 'user'@'host'
```

Support

You can get support and discuss about new features on: https://groups.google.com/d/forum/python-mysql-replication

You can browse and report issues on: https://github.com/noplay/python-mysql-replication/issues

Developement

Contributions

You can report issues and contribute to the project on: https://github.com/noplay/python-mysql-replication

The standard way to contribute code to the project is to fork the Github project and open a pull request with your changes: https://github.com/noplay/python-mysql-replication

Don't hesitate to open an issue with what you want to changes if you want to discuss about it before coding.

Tests

When it's possible we have an unit test.

pymysqlreplication/tests/ contains the test suite. The test suite use the standard *unittest* Python module.

Be carefull tests will reset the binary log of your MySQL server.

Make sure you have the following configuration set in your mysql config file (usually my.cnf on development env):

```
log-bin=mysql-bin
server-id=1
binlog-format    = row #Very important if you want to receive write, update and␣
↪delete row events
```

```
gtid_mode=ON
log-slave_updates=true
enforce_gtid_consistency
```

Run tests with

```
py.test -k "not test_no_trailing_rotate_event"
```

This will skip the `test_no_trailing_rotate_event` which requires that the user running the test have permission to alter the binary log files.

Running mysql in docker (main):

```
docker run --name python-mysql-replication-tests -e MYSQL_ALLOW_EMPTY_PASSWORD=true -
↪p 3306:3306 --rm percona:latest --log-bin=mysql-bin.log --server-id 1 --binlog-
↪format=row --gtid_mode=on --enforce-gtid-consistency=on --log_slave_updates
```

Running mysql in docker (for ctl server):

```
docker run --name python-mysql-replication-tests-ctl --expose=3307 -e MYSQL_ALLOW_
↪EMPTY_PASSWORD=true -p 3307:3307 --rm percona:latest --log-bin=mysql-bin.log --
↪server-id 1 --binlog-format=row --gtid_mode=on --enforce-gtid-consistency=on --log_
↪slave-updates -P 3307
```

Each pull request is tested on Travis CI: https://travis-ci.org/noplay/python-mysql-replication

Build the documentation

The documentation is available in docs folder. You can build it using Sphinx:

```
cd docs
pip install sphinx
make html
```

Licence

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

pymysqlreplication.binlogstream, 6
pymysqlreplication.event, 6
pymysqlreplication.row_event, 7

Index

B

BeginLoadQueryEvent (class in pymysqlreplication.event), 6
BinLogStreamReader (class in pymysqlreplication.binlogstream), 6

D

DeleteRowsEvent (class in pymysqlreplication.row_event), 7

E

encoded() (pymysqlreplication.binlogstream.ReportSlave method), 6
ExecuteLoadQueryEvent (class in pymysqlreplication.event), 6

G

gtid (pymysqlreplication.event.GtidEvent attribute), 6
GtidEvent (class in pymysqlreplication.event), 6

H

HeartbeatLogEvent (class in pymysqlreplication.event), 6

I

IntvarEvent (class in pymysqlreplication.event), 7

P

pymysqlreplication.binlogstream (module), 6
pymysqlreplication.event (module), 6
pymysqlreplication.row_event (module), 7

Q

QueryEvent (class in pymysqlreplication.event), 7

R

ReportSlave (class in pymysqlreplication.binlogstream), 6
RotateEvent (class in pymysqlreplication.event), 7

T

TableMapEvent (class in pymysqlreplication.row_event), 7

U

UpdateRowsEvent (class in pymysqlreplication.row_event), 7

W

WriteRowsEvent (class in pymysqlreplication.row_event), 7

X

XidEvent (class in pymysqlreplication.event), 7