Handling Failover with MySQL 5.6 and Global Transaction IDs

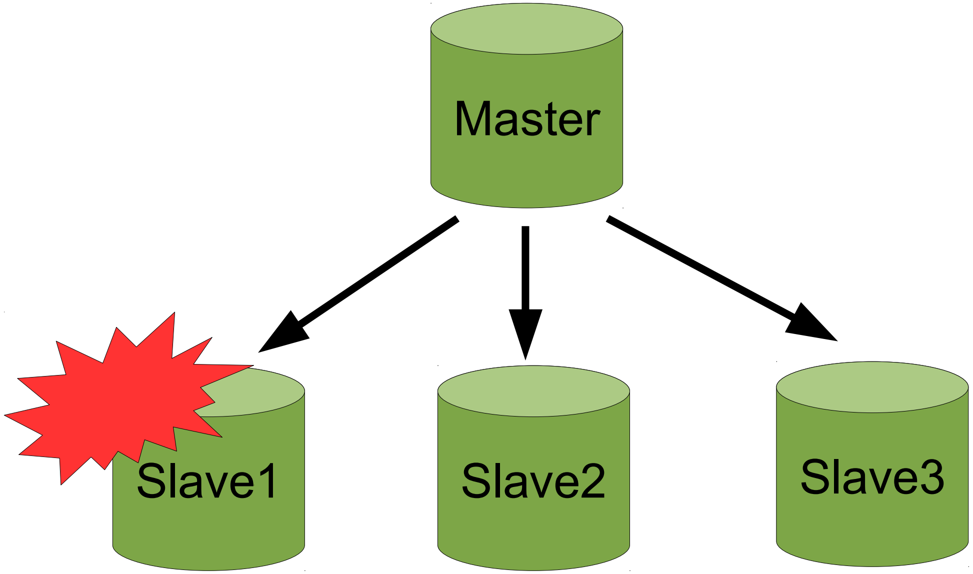
Stéphane Combaudon
FOSDEM
February 1st, 2014

Agenda

- Failover with position-based replication
- Quick introduction to Global Transactions IDs
- MySQL Utilities
- Other solutions

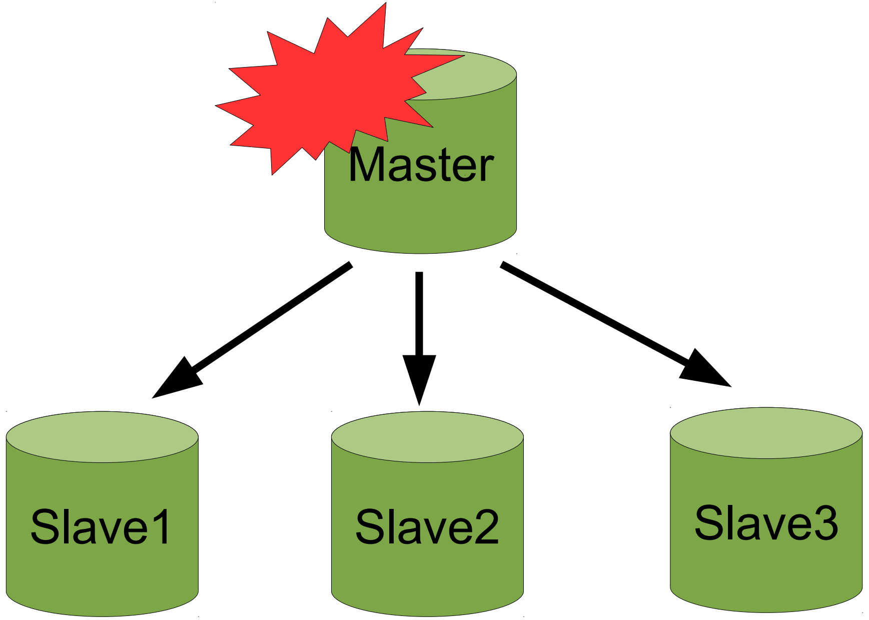
Failover with position-based replication

Slave crash



- NOT critical
  - Read capacity is affected
  - But writes still go to the master and are replicated to other slaves

Master crash



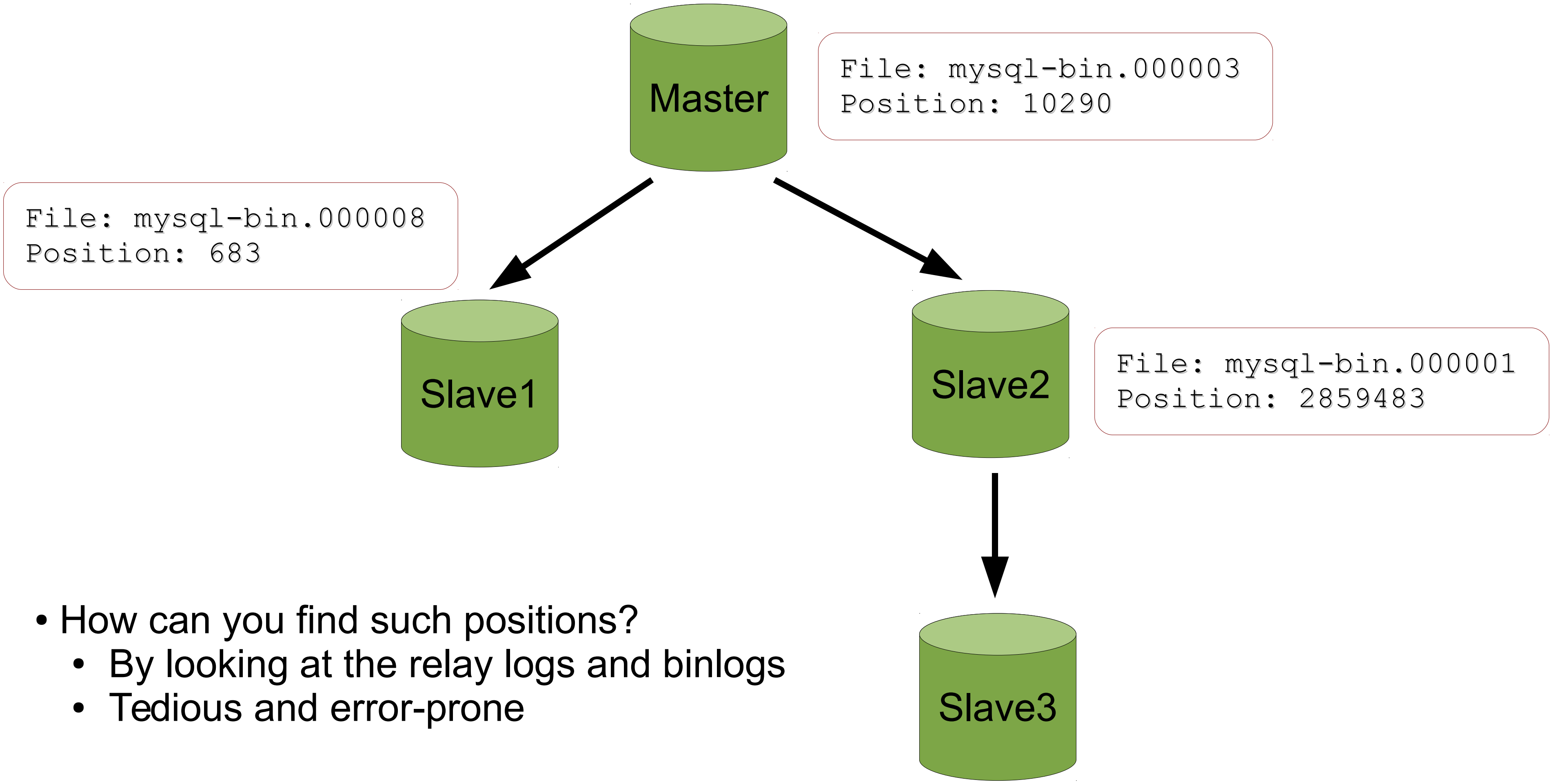
- Critical!
  - Reads can be done on slaves
  - But writes are stopped
  - A slave needs to be promoted: this is failover!

Failover (simplified)

- Select a candidate to become the master
  - The most up-to-date is a good candidate
- Reconfigure the other slaves to replicate from the new master
  - With `CHANGE MASTER TO`
- This is where it becomes interesting!

Finding position for an event

- Same event has several binlog positions



```
File: mysql-bin.000003
Position: 10290
```

```
File: mysql-bin.000008
Position: 683
```

```
File: mysql-bin.000001
Position: 2859483
```

- How can you find such positions?
  - By looking at the relay logs and binlogs
  - Tedious and error-prone

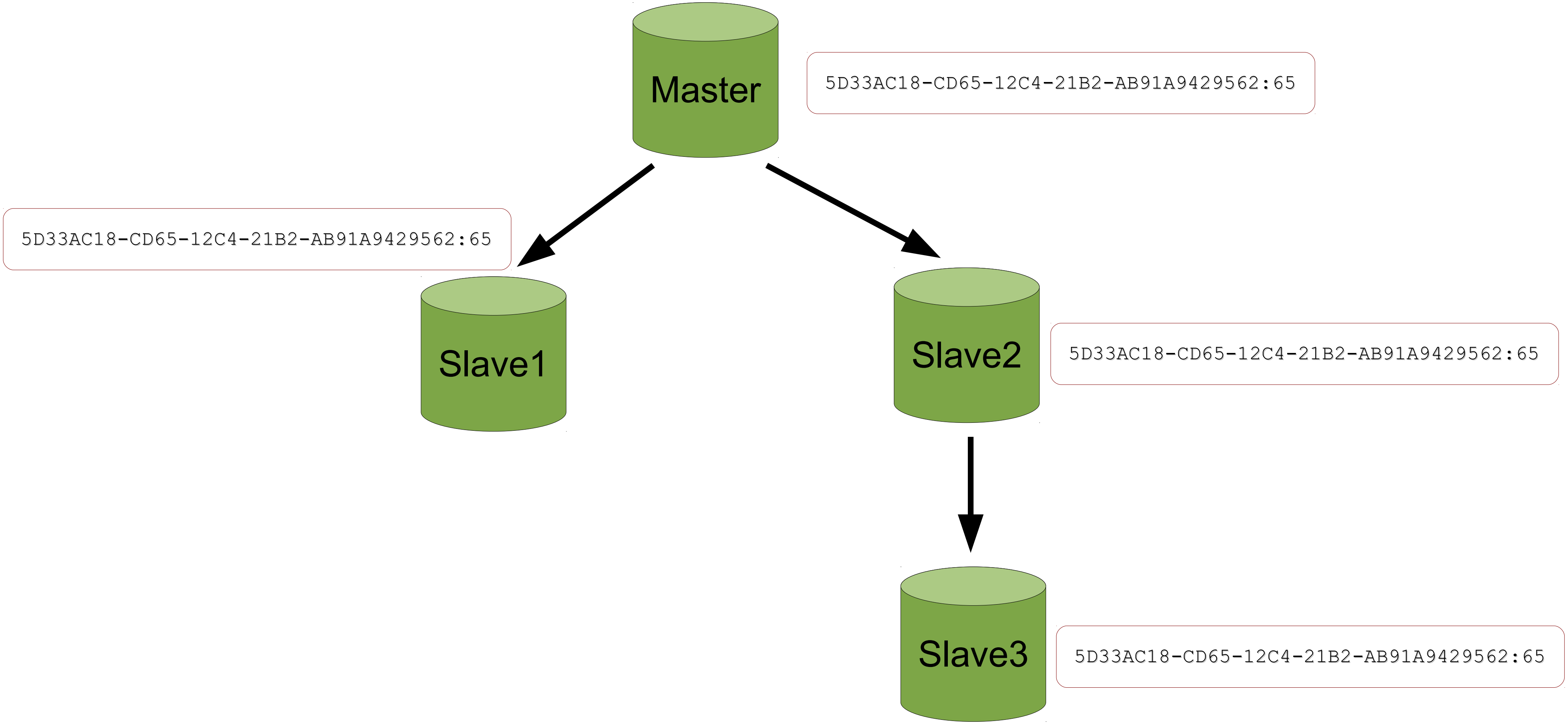
Quick introduction to Global Transactions IDs

What is a GTID?

- Unique identifier of a transaction across all servers of a replication setup
- Available from MySQL 5.6
- A GTID has 2 parts: `source_id:transaction_id`
- A sequence of GTIDs is simply
  - `source_id:trx_start-trx_stop`
  - Eg `3E11FA47-71CA-11E1-9E33-C80AA9429562:1-5`

Finding the position of an event

- Easy, same for all servers!!



Master

5D33AC18-CD65-12C4-21B2-AB91A9429562:65

Slave1

5D33AC18-CD65-12C4-21B2-AB91A9429562:65

Slave2

5D33AC18-CD65-12C4-21B2-AB91A9429562:65

Slave3

5D33AC18-CD65-12C4-21B2-AB91A9429562:65

Limitations

- Switching to GTID-based replication involves stopping all servers at the same time

- Binlog + log_slave_updates should be enabled on all slaves
  - Some performance penalty

- Some rough edges

MySQL Utilities

MySQL Utilities

- Set of Python scripts to ease administration of MySQL servers
- Free and open source, developed by Oracle
- http://dev.mysql.com/doc/workbench/en/mysql-utilities.html

Overview of mysqlfailover

- Health monitoring and automatic failover
  - Target topology: 1 master, N slaves

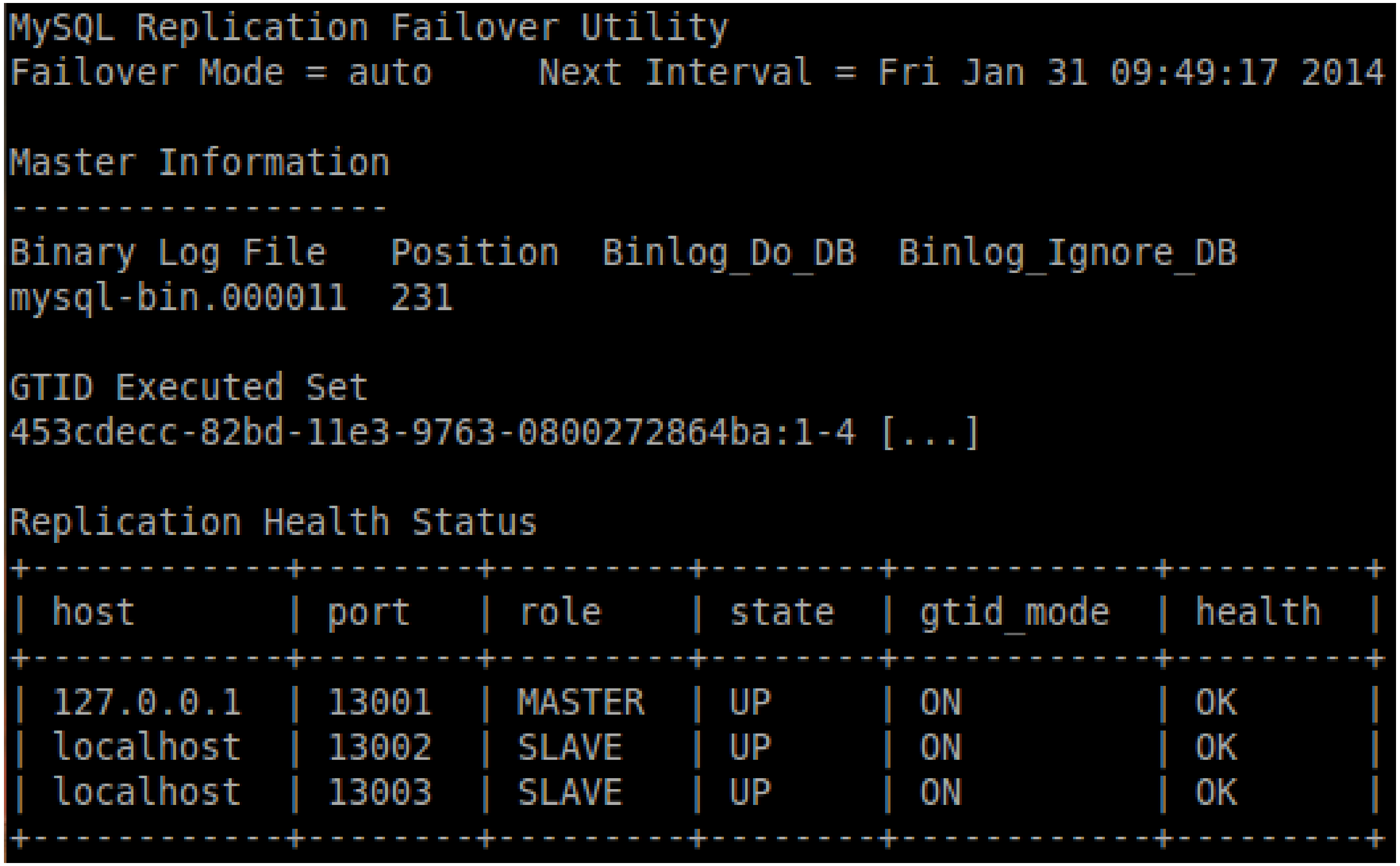
- A few MySQL settings are required
  - `--log-slave-updates, --enforce-gtid-consistency, gtid_mode = ON`
  - `--report-host, --report-port`
  - `--master-info-repository=TABLE`

Different modes

- Elect
  - Chooses a candidate from a list. If none can be promoted, exits with an error
- Auto (default)
  - Same as elect, but if no candidate is suitable, any other slave can be promoted
- Fail
  - Perform health monitoring, exits with an error if the master fails

Example of execution

```
mysqlfailover --discover-slaves-login=root:root \
--master=root:root@127.0.0.1:13001
```



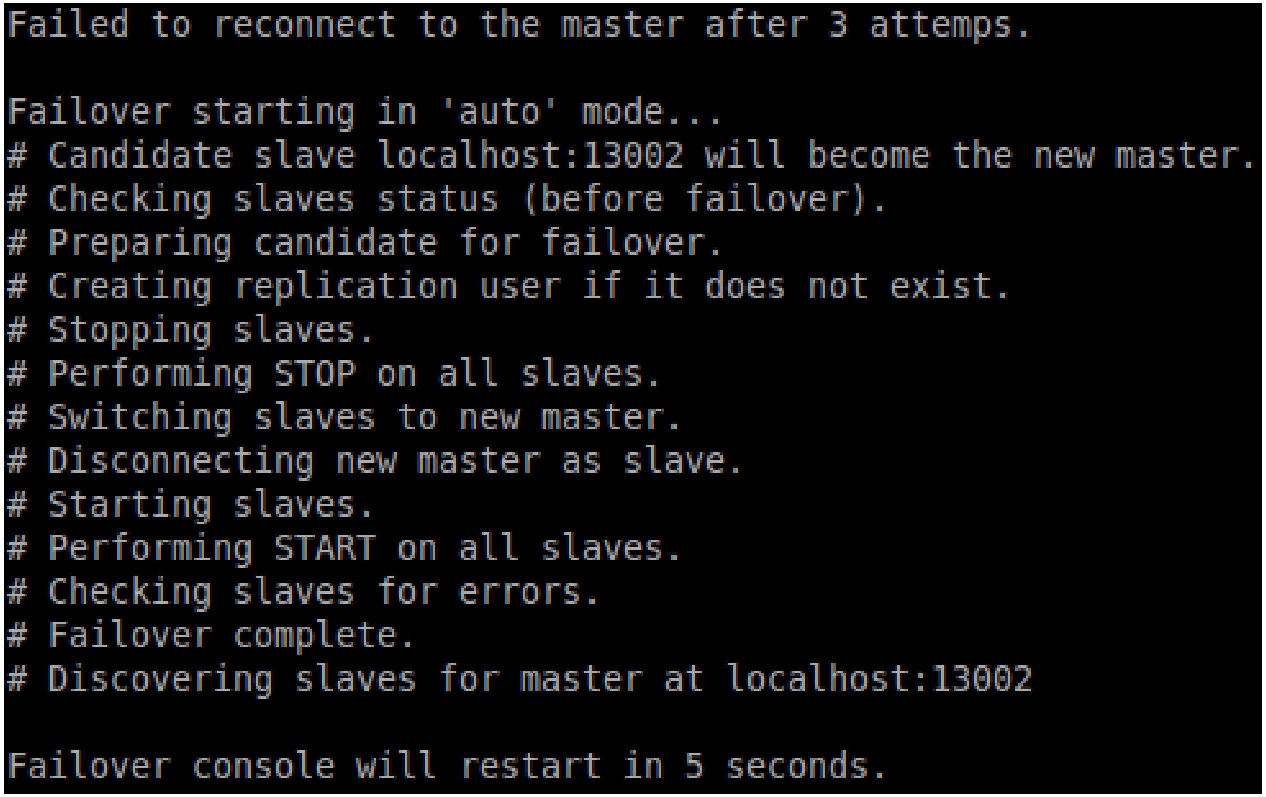
```
MySQL Replication Failover Utility
Failover Mode = auto     Next Interval = Fri Jan 31 09:49:17 2014

Master Information
------------------
Binary Log File   Position  Binlog_Do_DB  Binlog_Ignore_DB
mysql-bin.000011  231

GTID Executed Set
453cdecc-82bd-11e3-9763-0800272864ba:1-4 [...]

Replication Health Status
+------------+--------+---------+--------+------------+---------+
| host       | port   | role    | state  | gtid_mode  | health  |
+------------+--------+---------+--------+------------+---------+
| 127.0.0.1  | 13001  | MASTER  | UP     | ON         | OK      |
| localhost  | 13002  | SLAVE   | UP     | ON         | OK      |
| localhost  | 13003  | SLAVE   | UP     | ON         | OK      |
+------------+--------+---------+--------+------------+---------+
```

If the master fails...

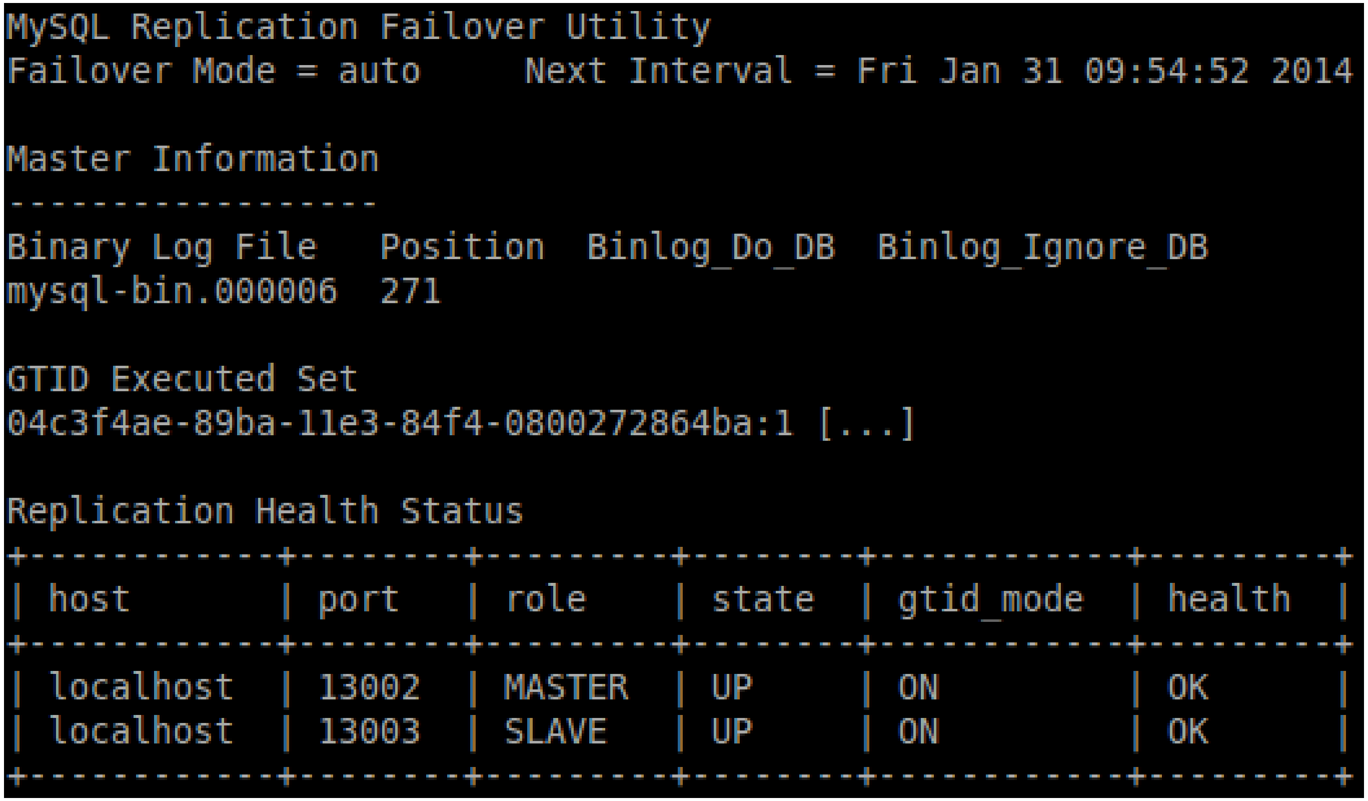


```
Failed to reconnect to the master after 3 attemps.

Failover starting in 'auto' mode...
# Candidate slave localhost:13002 will become the new master.
# Checking slaves status (before failover).
# Preparing candidate for failover.
# Creating replication user if it does not exist.
# Stopping slaves.
# Performing STOP on all slaves.
# Switching slaves to new master.
# Disconnecting new master as slave.
# Starting slaves.
# Performing START on all slaves.
# Checking slaves for errors.
# Failover complete.
# Discovering slaves for master at localhost:13002

Failover console will restart in 5 seconds.
```

When failover is done



```
MySQL Replication Failover Utility
Failover Mode = auto     Next Interval = Fri Jan 31 09:54:52 2014

Master Information
------------------
Binary Log File   Position  Binlog_Do_DB  Binlog_Ignore_DB
mysql-bin.000006  271

GTID Executed Set
04c3f4ae-89ba-11e3-84f4-0800272864ba:1 [...]

Replication Health Status
+------------+--------+---------+--------+------------+---------+
| host       | port   | role    | state  | gtid_mode  | health  |
+------------+--------+---------+--------+------------+---------+
| localhost  | 13002  | MASTER  | UP     | ON         | OK      |
| localhost  | 13003  | SLAVE   | UP     | ON         | OK      |
+------------+--------+---------+--------+------------+---------+
```

Limitations

- Monitoring node is a single point of failure with `mysqlfailover`
  - Manual failover with `mysqlrpladmin` may be preferred
- Errant transactions prevent failover
  - Use `--pedantic` to get an error when starting mysqlfailover
  - Fix manually

Manual failover with mysqlrpladmin

- Planned promotion (switchover)

```
mysqlrpladmin --master=root:root@127.0.0.1:13002 \
--new-master=root:root@127.0.0.1:13001 \
--discover-slaves-login=root:root –demote-master \
switchover
```

- Unplanned promotion (failover)

```
mysqlrpladmin
--slaves=root:root@127.0.0.1:13002,root:root@127.0.0.
1:13003 --candidates=root:root@localhost:13002
failover
```

Other solutions

MHA

- Perl scripts automating slave reconfiguration
- Pros
  - MySQL 5.0+, no need for GTID
  - Tries hard to minimize data loss
  - External to the DB, no change is required
- Cons
  - Monitoring node is a SPOF, manual failover recommended

Galera

- External replication library
- Pros
  - Virtually synchronous replication
  - Automatic failover
  - Automatic provisioning
- Cons
  - Still young, some rough edges
  - A few prerequisites (MySAM, foreign keys & large transactions not recommended)

Pacemaker

- HA resource manager, not limited to MySQL
- Pros
  - MySQL 5.0+, no need for GTID
  - Very mature
  - Lots of feature
  - No single point of failure
- Cons
  - Can be complex to set up and debug
  - Not a lightweight solution

Q&A

Thanks for attending!

stephane.combaudon@percona.com