PERCONA

MySQL 5.6 GTID in a nutshell

Miguel Ángel Nieto
Percona Live University - Toronto

Who am I?

- **Miguel** Ángel
- I live in the north of **Spain**
- Support Engineer at **Percona**
- Hobbies:
  - **Scuba Diving**
  - Videogames
  - American TV Series
  - Beers

Agenda

I’m going to answer the following questions and give a detailed overview that will let us to start working with it:

- What is GTID?
- What problems GTID solves?
- How can I implement it?
- How can I repair it?
- How can I use it for HA and Failover?
- Take in account...

What is GTID?

What is GTID?

```
8182213e-7c1e-11e2-a6e2-080027635ef5:1
```

What is GTID?

And that thing is a GTID?

8182213e-7c1e-11e2-a6e2-080027635e

Not impressed

What is GTID?

```
8182213e-7c1e-11e2-a6e2-080027635ef5:1
```

SID. This is the server's 128 bit identification number (**SERVER_UUID**). It identifies where the transaction was originated. Every server has its own **SERVER_UUID**.

What is GTID?

`8182213e-7c1e-11e2-a6e2-080027635ef5:1`

GNO. This is the transaction identification number. It is a sequence number that increments with every new transaction.

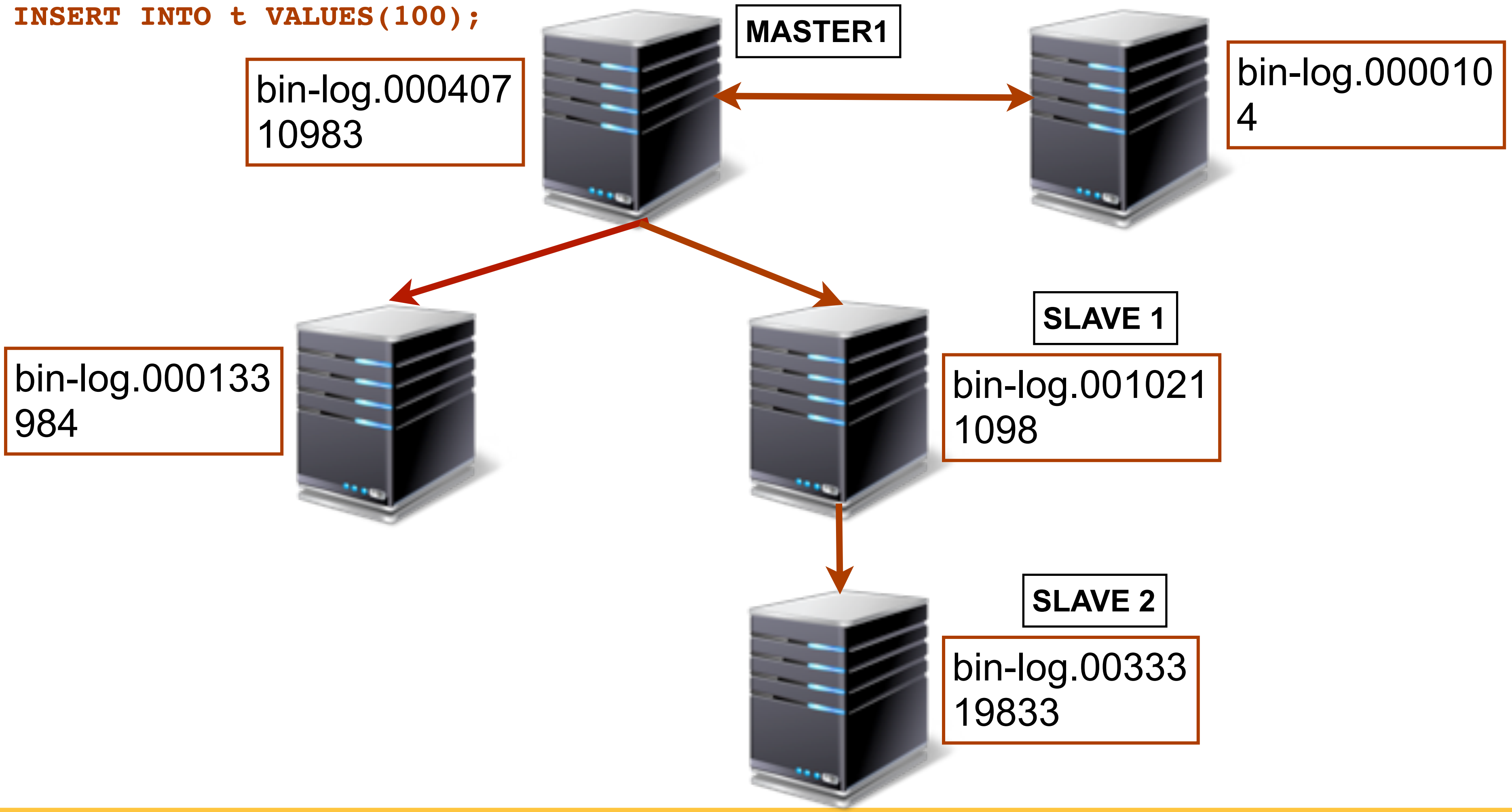
What is GTID?

- This is how we can see the GTID inside the binary logs:

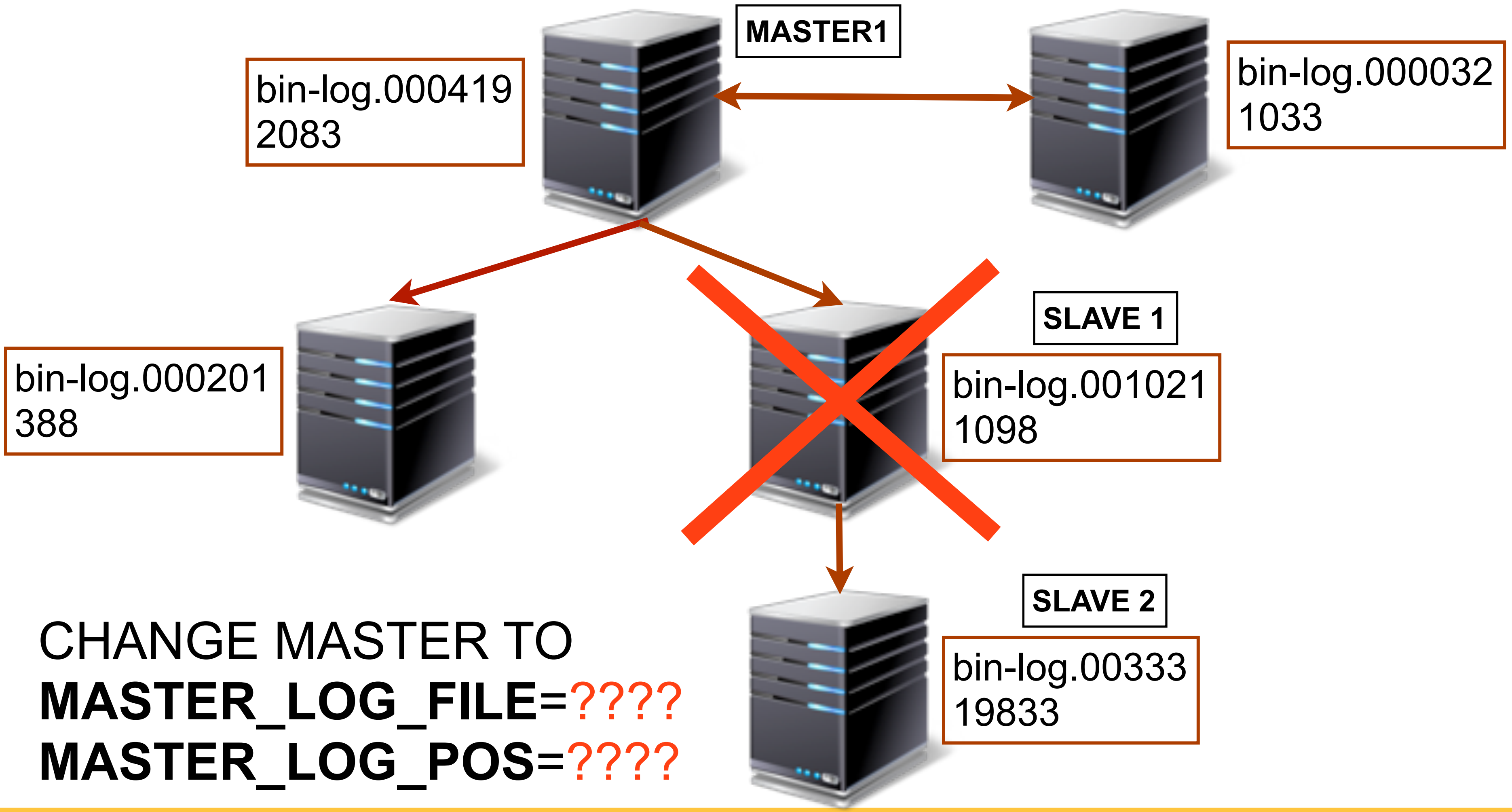
```
# at 300
#130221 13:08:58 server id 101  end_log_pos 348 CRC32 0xc18cdbda        GTID [commit=yes]
SET @@SESSION.GTID_NEXT= '8182213e-7c1e-11e2-a6e2-080027635ef5:2'/*!*/;
# at 348
BEGIN
insert into t values(1)
COMMIT/*!*/;
# at 565
#130221 13:09:03 server id 101  end_log_pos 613 CRC32 0x5b25189e        GTID [commit=yes]
SET @@SESSION.GTID_NEXT= '8182213e-7c1e-11e2-a6e2-080027635ef5:3'/*!*/;
# at 697
BEGIN
insert into t values(100)
COMMIT/*!*/;
```

- The GTID is replicated to Slave servers.

What problems GTID solves?



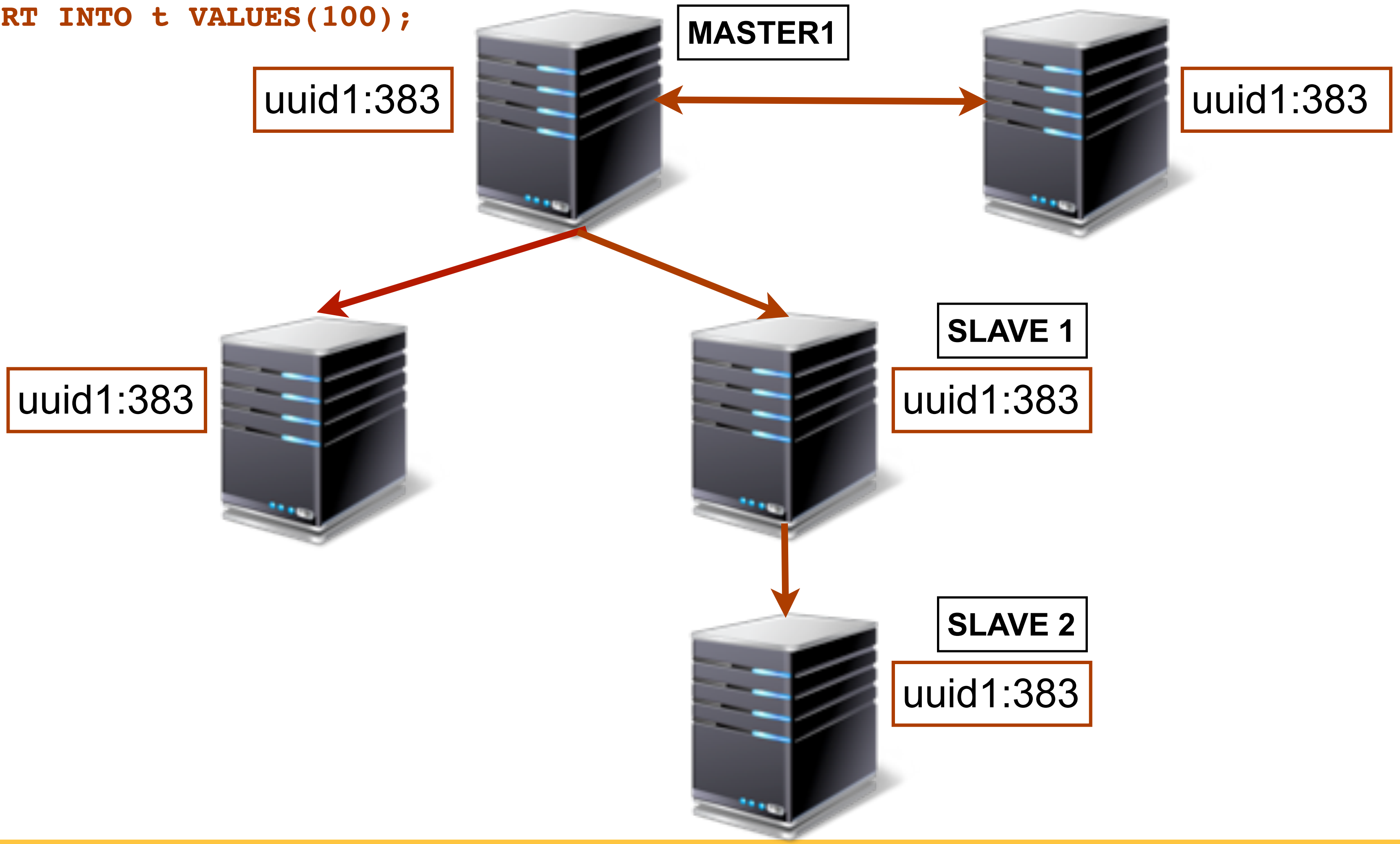
What problems GTID solves?



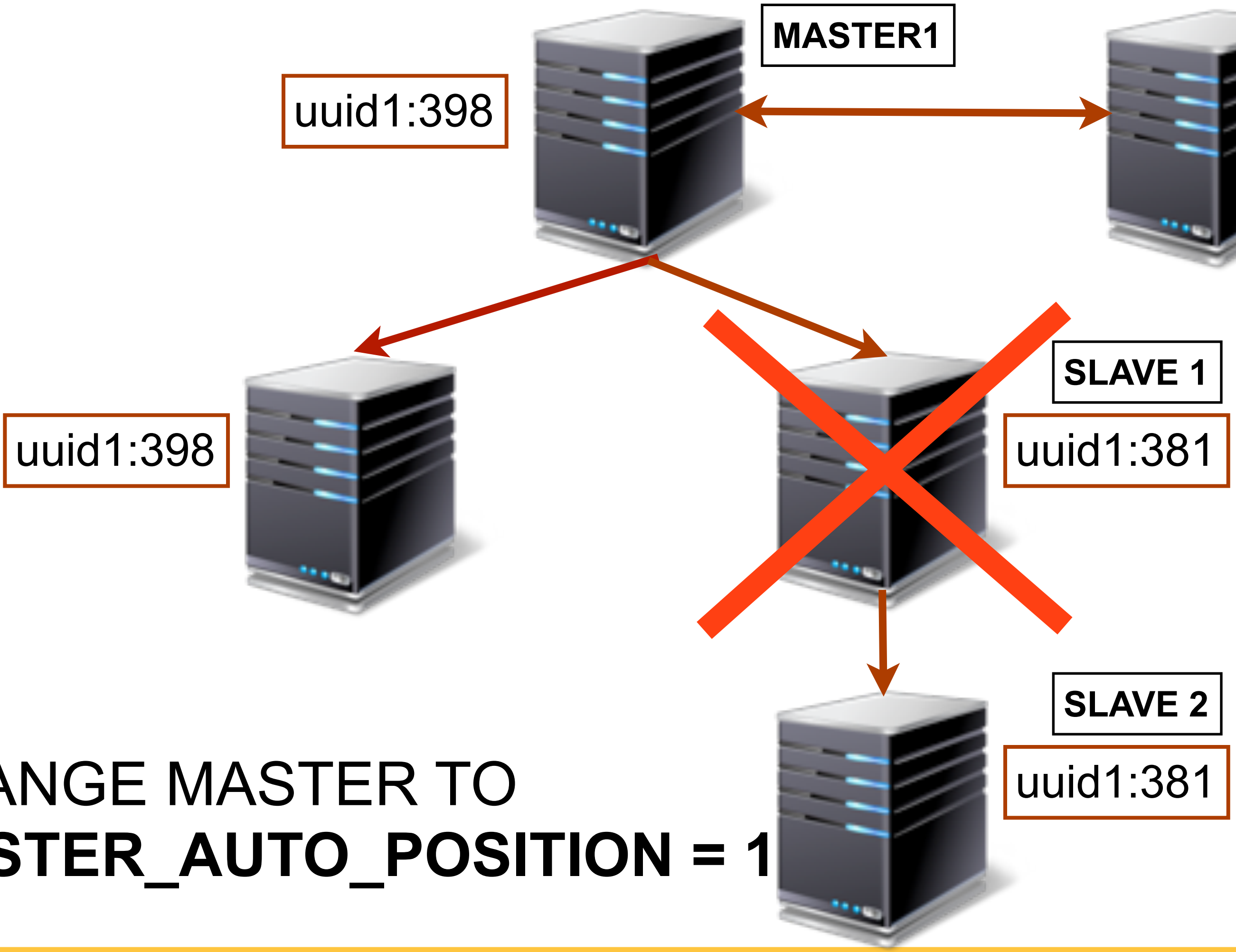
CHANGE MASTER TO
MASTER_LOG_FILE=????
MASTER_LOG_POS=????

What problems GTID solves?

```
INSERT INTO t VALUES(100);
```



What problems GTID solves?



CHANGE MASTER TO
MASTER_AUTO_POSITION = 1

What problems GTID solves?

- It is possible to identify a transaction **uniquely** across the replication servers.

- Make the automation of failover process much easier. There is no need to do calculations, inspect the binary log and so on. Just **MASTER_AUTO_POSITION=1.**

- At application level it is easier to do WRITE/READ split. After a write on the MASTER you have a GTID so just check if that GTID has been executed on the SLAVE that you use for reads.

- Development of new automation tools isn't a pain now.

How can I implement it?

- Three variables are needed in ALL servers of the replication chain

- **gtid_mode:** It can be ON or OFF (not 1 or 0). It enables the GTID on the server.
- **log_bin:** Enable binary logs. Mandatory to create a replication environment.
- **log-slave-updates:** Slave servers must log the changes that comes from the master in its own binary log.
- **enforce-gtid-consistency:** Statements that can't be logged in a transactionally safe manner are denied by the server.

How can I implement it?

- **enforce-gtid-consistency**
  - CREATE TABLE ... SELECT statements.

```
ERROR 1786 (HY000): CREATE TABLE ... SELECT is forbidden when
ENFORCE_GTID_CONSISTENCY = 1.
```

  - CREATE TEMPORARY TABLE inside transactions.

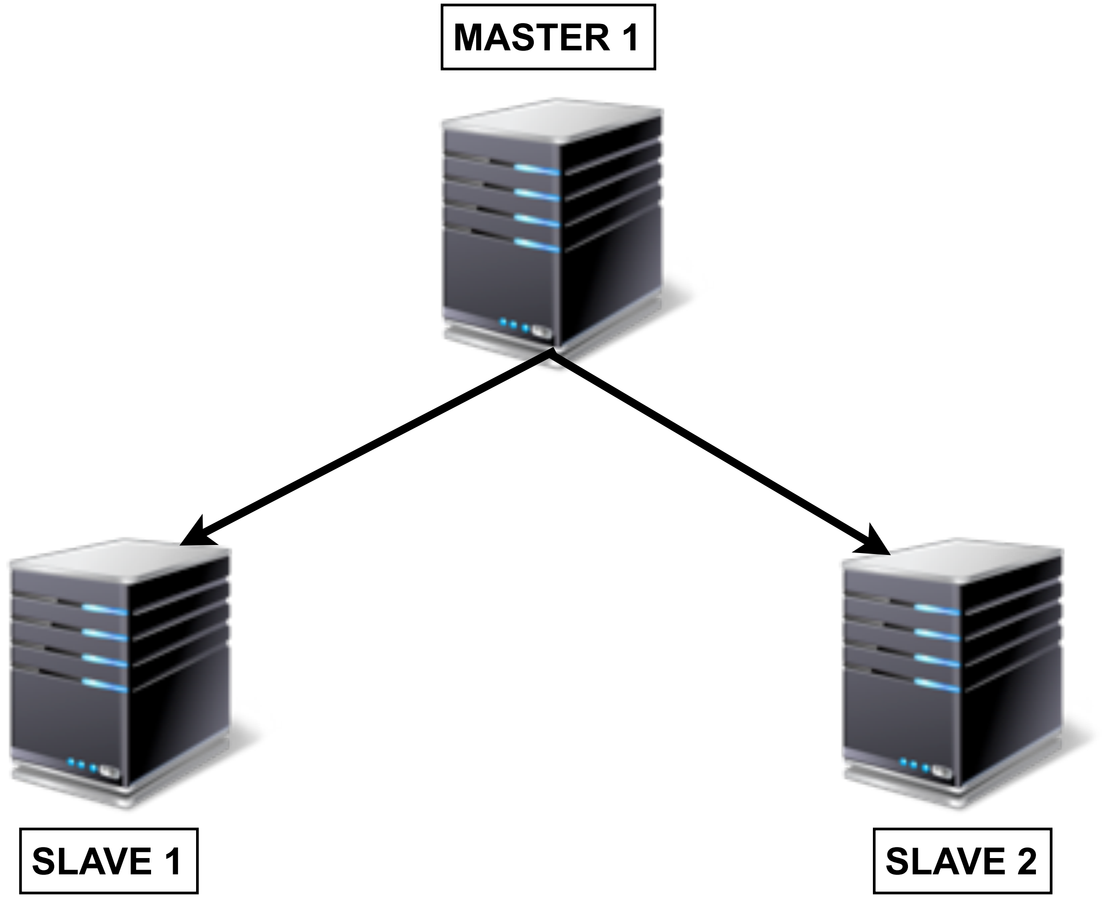
```
ERROR 1787 (HY000): When ENFORCE_GTID_CONSISTENCY = 1, the statements CREATE
TEMPORARY TABLE and DROP TEMPORARY TABLE can be executed in a non-transactional
context only, and require that AUTOCOMMIT = 1.
```

  - Transactions that mixes updates on transactional and non-transactional tables.

```
ERROR 1785 (HY000): When ENFORCE_GTID_CONSISTENCY = 1, updates to non-
transactional tables can only be done in either autocommitted statements or single-statement
transactions, and never in the same statement as updates to transactional tables.
```

How can I implement it?

New replication from scratch

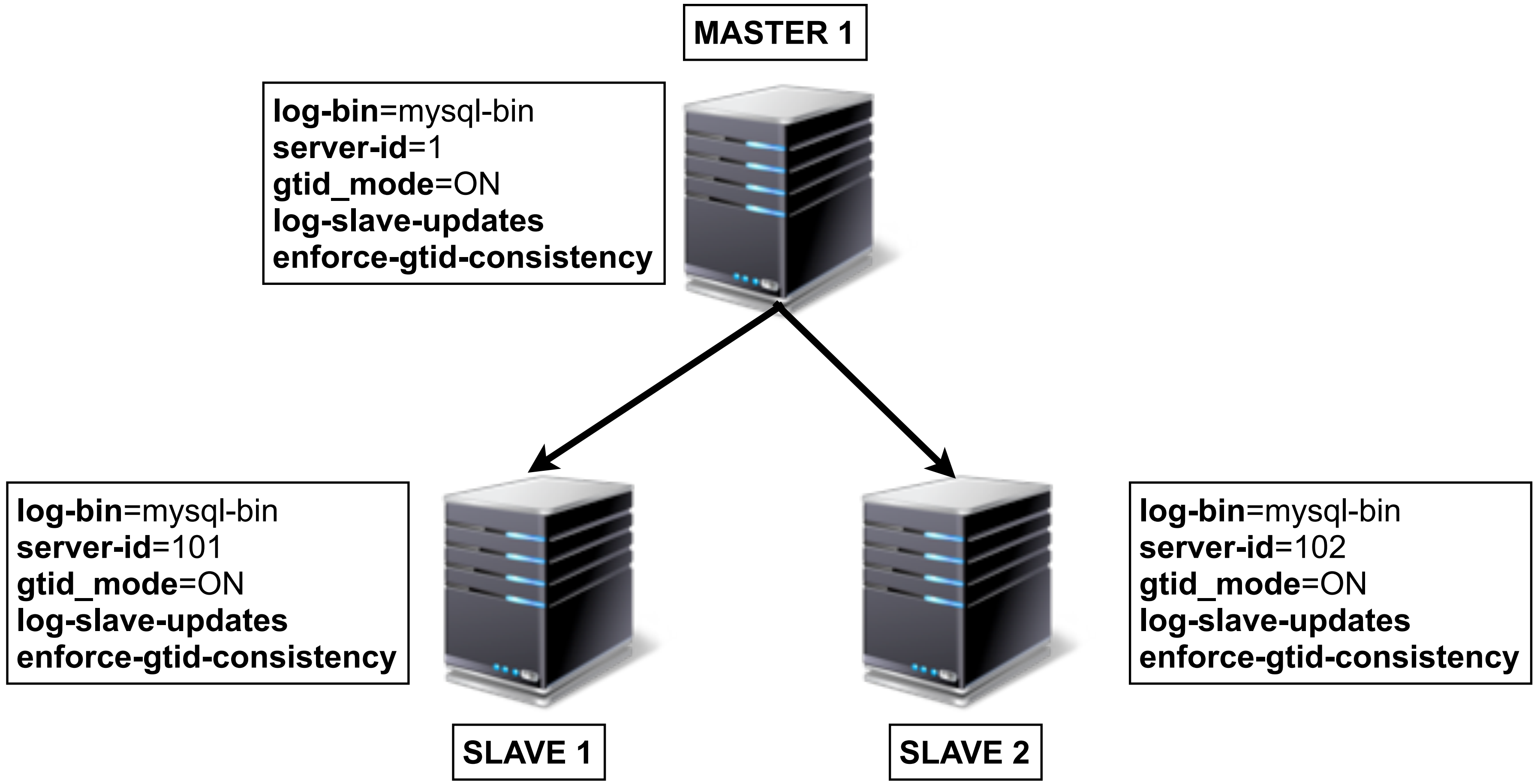


How can I implement it?

New replication from scratch

1) Create replication user on the master server.
2) Configure the parameters on all three servers:
   - gtid_mode
   - log_bin
   - log-slave-updates
   - enforce-gtid-consistency
   - server_id
3) Start all mysql services.
4) CHANGE MASTER TO... with **MASTER_AUTO_POSITION=1** on the two slave servers.

How can I implement it?



MASTER 1:

```
log-bin=mysql-bin
server-id=1
gtid_mode=ON
log-slave-updates
enforce-gtid-consistency
```

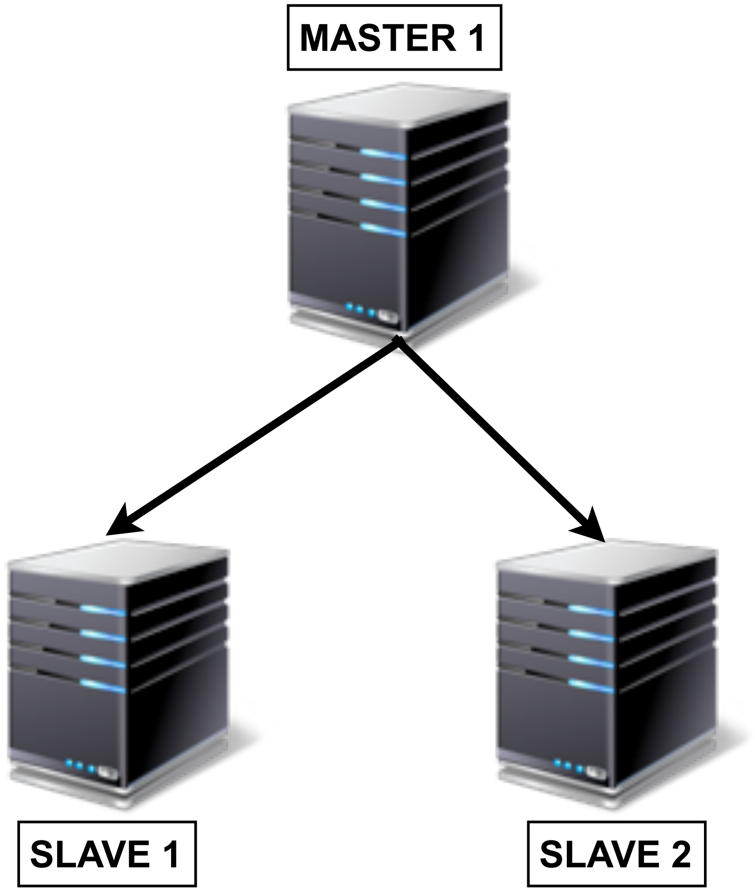
SLAVE 1:

```
log-bin=mysql-bin
server-id=101
gtid_mode=ON
log-slave-updates
enforce-gtid-consistency
```

SLAVE 2:

```
log-bin=mysql-bin
server-id=102
gtid_mode=ON
log-slave-updates
enforce-gtid-consistency
```

How can I implement it?



```
CHANGE MASTER TO MASTER_HOST="127.0.0.1", MASTER_PORT=18675,
MASTER_USER="msandbox", MASTER_PASSWORD="msandbox", MASTER_AUTO_POSITION=1;
```

How can I implement it?

Move already running replication to GTID

1. Set the master as **read_only** and wait until the slaves catch up.
2. Stop all servers.
3. Configure the GTID variables in my.cnf.
4. Start all the servers:
   - Master should start in **read_only** mode.
   - Slaves should start with **skip_slave_start.**
5. **CHANGE MASTER** with **MASTER_AUTO_POSITION**=1 on the slaves.
6. **START SLAVE;** on slave servers.
7. **SET GLOBAL read_only=0;** on master server.

How can I implement it?

- Now we run two transactions on the master:
  CREATE TABLE t (i INT);
  INSERT INTO t VALUES(1);
- This is the status of slaves:

```
slave1 > show slave status\G
[…]
             Master_Log_File: mysql-bin.000001
         Read_Master_Log_Pos: 550
              Relay_Log_File: mysql_sandbox18676-relay-bin.000002
               Relay_Log_Pos: 760
       Relay_Master_Log_File: mysql-bin.000001
[…]
            Master_Server_Id: 1
                 Master_UUID: 1c9cdcc8-7c33-11e2-a769-080027635ef5

[…]
          Retrieved_Gtid_Set: 1c9cdcc8-7c33-11e2-a769-080027635ef5:1-2
           Executed_Gtid_Set: 1c9cdcc8-7c33-11e2-a769-080027635ef5:1-2
               Auto_Position: 1
```

How can I implement it?

- Now we have new variables to check:
  - **gtid_executed (ro)**: shows the transactions that have been executed in this server.
    1c9cdcc8-7c33-11e2-a769-080027635ef5:1-3

  - **gtid_purged (ro)**: shows the transactions that have been purged from the binary log (purge binary logs to...).
    1c9cdcc8-7c33-11e2-a769-080027635ef5:1-2

  - **gtid_next:** the next GTID that will be used.
    SET @@SESSION.GTID_NEXT= '8182213e-7c1e-11e2-a6e2-080027635ef5:2'/*!*/;

How can I repair it?

- Even with GTID we have the same problem. MySQL replication can easily fail.
- The procedure to repair a replication is slightly different from the regular replication based on binary log position.
- There is a very good blog post written by a very good blogger that explains how to repair it:

  http://www.mysqlperformanceblog.com/2013/02/08/how-to-createrestore-a-slave-using-gtid-replication-in-mysql-5-6/

How can I repair it?

- ERROR!

```
Slave_IO_Running: No
Slave_SQL_Running: Yes
Last_IO_Error: Got fatal error 1236 from master when reading data from binary log:
'The slave is connecting using CHANGE MASTER TO MASTER_AUTO_POSITION = 1, but the
master has purged binary logs containing GTIDs that the slave
requires.'
```

- **mysqldump** supports GTID:

```
# mysqldump --all-databases --single-transaction --triggers --routines --
host=127.0.0.1 --port=18675 --user=msandbox --password=msandbox > dump.sql

# grep PURGED dump.sql
SET @@GLOBAL.GTID_PURGED='9a511b7b-7059-11e2-9a24-08002762b8af:
1-13';
```

- **Xtrabackup's** support for GTID is in **Work in Progress.**

How can I repair it?

- The server was already running as slave, so **GTID_EXECUTED** and **GTID_PURGED** has values:

```
slave1 > source test.sql;
ERROR 1840 (HY000): GTID_PURGED can only be set when GTID_EXECUTED is
empty.
```

- So, let's empty **GITD_EXECUTED**. But... How? It is a read only variable!

```
slave1 > reset master;
slave1 > show global variables like 'GTID_EXECUTED';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| gtid_executed |       |
+---------------+-------+
slave1> source test.sql;
slave1> start slave;
```

How can I repair it?

- Another way, injecting empty transactions

  http://dev.mysql.com/doc/refman/5.6/en/replication-gtids-failover.html#replication-gtids-failover-empty

- **SQL_SLAVE_SKIP_COUNTER** doesn’t work anymore with GTID

- We need to find what transaction is causing the replication to fail
  - From binary logs
  - From SHOW SLAVE STATUS (retrieved vs executed)

How can I repair it?

- Slave failed:

```
               Last_SQL_Error: Error 'Duplicate entry '4' for key 'PRIMARY'' on
query. Default database: 'test'. Query: 'insert into t VALUES(NULL,'salazar')'
           Retrieved_Gtid_Set: 7d72f9b4-8577-11e2-a3d7-080027635ef5:1-5
            Executed_Gtid_Set: 7d72f9b4-8577-11e2-a3d7-080027635ef5:1-4
```

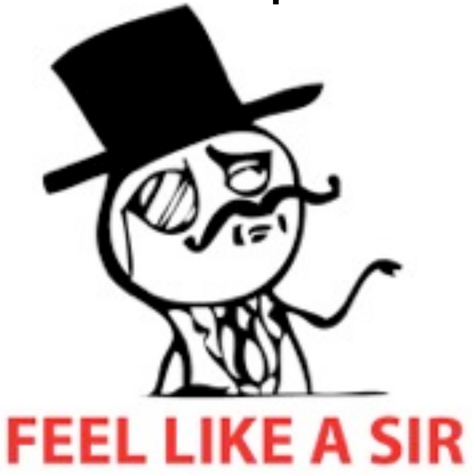
- So, this slave has retrieved transactions from 1 to 5 but only 1 to 4 has been applied. Seems that transaction 4 is the problem here.

```
STOP SLAVE;
SET GTID_NEXT="7d72f9b4-8577-11e2-a3d7-080027635ef5:5";
BEGIN; COMMIT;
SET GTID_NEXT="AUTOMATIC";
START SLAVE;
[...]
Retrieved_Gtid_Set: 7d72f9b4-8577-11e2-a3d7-080027635ef5:1-5
Executed_Gtid_Set: 7d72f9b4-8577-11e2-a3d7-080027635ef5:1-5
```

How can I repair it?

- mysqldump can be also used to create new slaves.
- It is a new slave so GTID_EXECUTED and GTID_PURGED are empty. No RESET MASTER is needed.

- Now we know how to create and repair a replication with GTID.



How can I use it for HA and Failover?

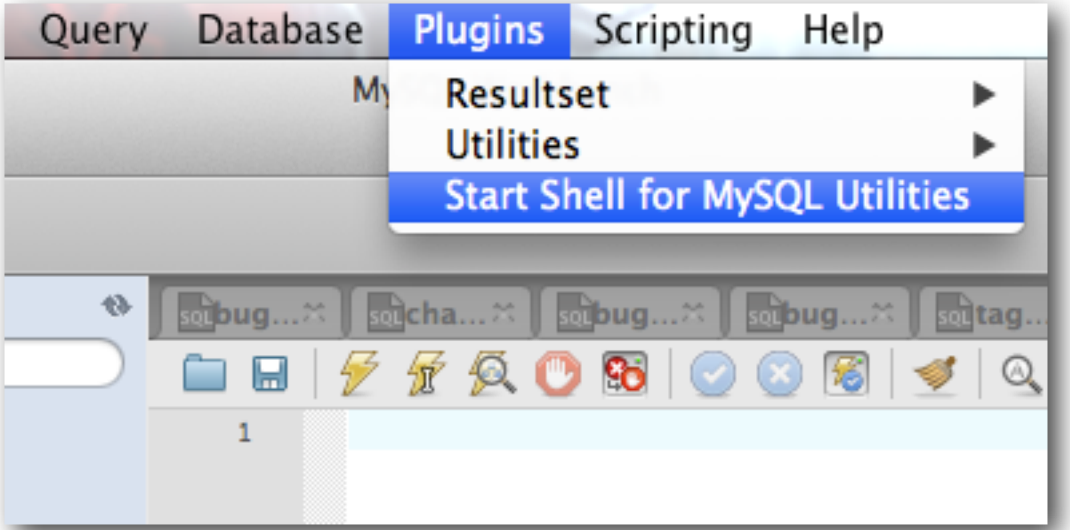
- We have seen how to implement and repair a GTID based replication.
- Now we are going to see it can help us with HA and failover:

  mysqlrpladmin: replication administration tool. For failover and switchover.

  mysqlfailover: replication heath heck and automatic failover tool.

How can I use it for HA and Failover?

- Where can I download these tools?
- https://launchpad.net/mysql-utilities
- MySQL Workbench:



How can I use it for HA and Failover?

- **mysqlrpladmin** is a tool used to perform planned maintenance tasks in our replication environment:
  - **switchover**: a planned stop for a master. Slave is promoted to new master. No possibility of transaction loss.
  - **failover**: a non-planned stop of the master and a slave promoted to new master. Last transaction can be lost. The tool will chose the most up-to-date slave.

How can I use it for HA and Failover?

- Some pre-requisites:
  - To make the autodiscover work the master-info-repository=TABLE should be enabled.
  - Slaves should have the replication user created in order to be elected as new masters.

```
# mysqlrpladmin --master=root:msandbox@master:18675
--discover-slaves-login=root:msandbox health
# Discovering slaves for master at 127.0.0.1:18675
# Checking privileges.
# Replication Topology Health:
+-------------+---------+----------+---------+-------------+----------+
| host        | port    | role     | state   | gtid_mode   | health   |
+-------------+---------+----------+---------+-------------+----------+
| master      | 18675   | MASTER   | UP      | ON          | OK       |
| SBslave1    | 18676   | SLAVE    | UP      | ON          | OK       |
| SBslave2    | 18677   | SLAVE    | UP      | ON          | OK       |
| SBslave3    | 18678   | SLAVE    | UP      | ON          | OK       |
+-------------+---------+----------+---------+-------------+----------+
```

How can I use it for HA and Failover?

- **health**: shows the health status of the replication servers.
- **elect**: shows which slave server should be elected as new master in case of a failover.
- **failover**: performs a failover selecting the most up-to-date slave.
- **gtid**: shows GTID information from all nodes
- **reset, start, stop**: reset, start or stop command on all slaves.
- **switchover**: do a slave promotion using the --new-master parameter.

How can I use it for HA and Failover?

- We need to remove the master from the replication and promote a slave to a new master:

```
# mysqlrpladmin --demote-master --
master=msandbox:msandbox@master:18675 --new-
master=root:msandbox@sbslave1:18676 --
slaves=root:msandbox@sbslave1:18676,root:msandbox@sbslave2:
18677,root:msandbox@sbslave3:18678 switchover
# Performing switchover from master at master:18675 to slave at
sbslave1:18676.
# Switchover complete.
+-----------+--------+---------+--------+------------+---------+
| host      | port   | role    | state  | gtid_mode  | health  |
+-----------+--------+---------+--------+------------+---------+
| sbslave1  | 18676  | MASTER  | UP     | ON         | OK      |
| master    | 18675  | SLAVE   | UP     | ON         | OK      |
| sbslave2  | 18677  | SLAVE   | UP     | ON         | OK      |
| sbslave3  | 18678  | SLAVE   | UP     | ON         | OK      |
+-----------+--------+---------+--------+------------+---------+
```

How can I use it for HA and Failover?

- **mysqlfailover** tool do a health check on the replication servers and run a failover automatically in case it is necessary.

- It monitors the master and in case of a failure on the health check it selects the best slave and performs the failover.

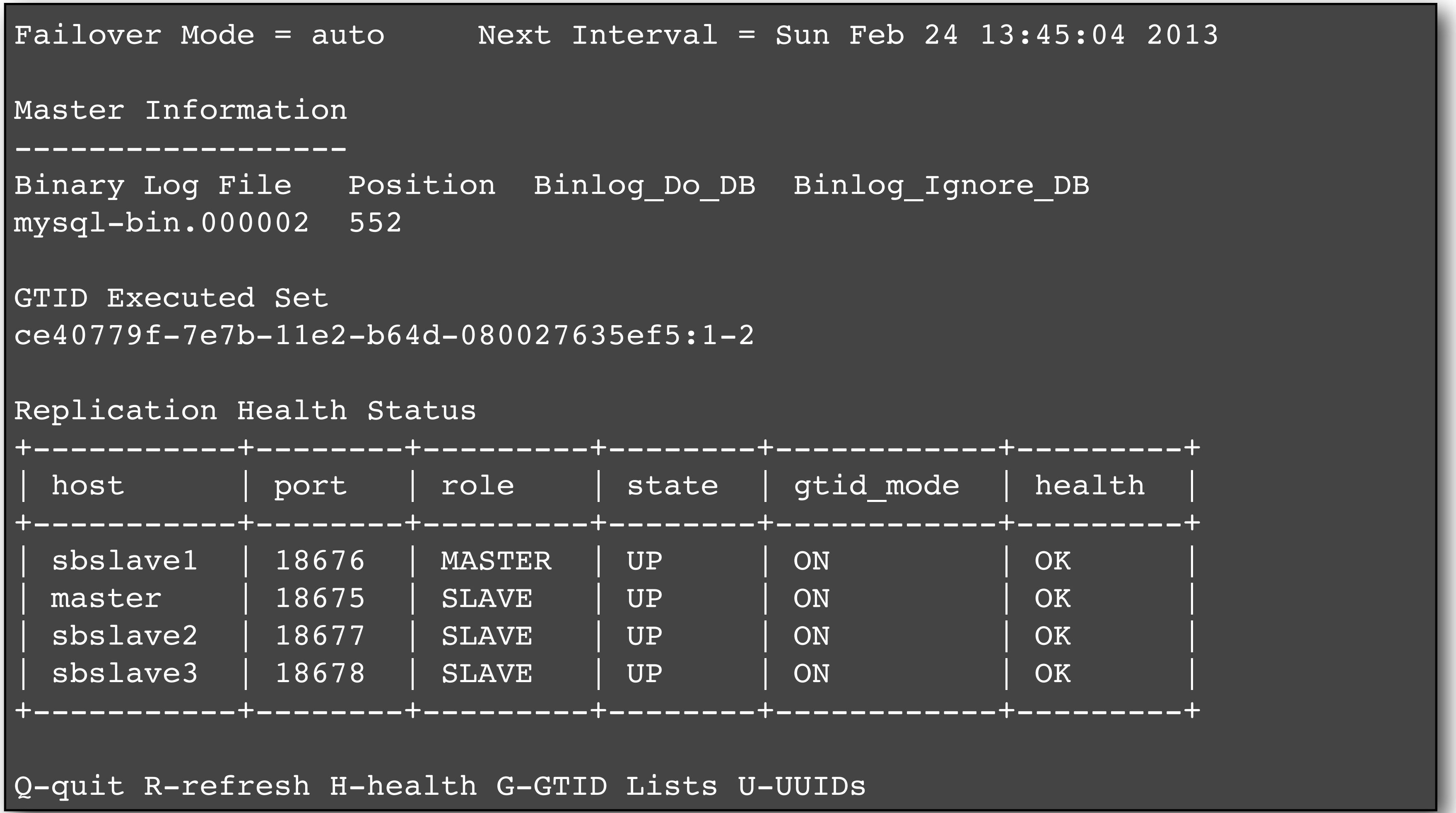
- You can give the tool a list of slaves that should be taken in account for master promotion. For example in case of different hardware characteristics.

How can I use it for HA and Failover?

- **auto**: performs an automatic failover.
- **elect**: the same as auto but if no candidates on the candidates list are viable it shows an error and exists.
- **fail**: doesn't perform any failover, just shows an error an exist.

```
mysqlfailover --master=msandbox:msandbox@sbslave1:18676 --
slaves=root:msandbox@master:
18675,root:msandbox@sbslave2:18677,root:msandbox@sbslave3:18678 auto
```

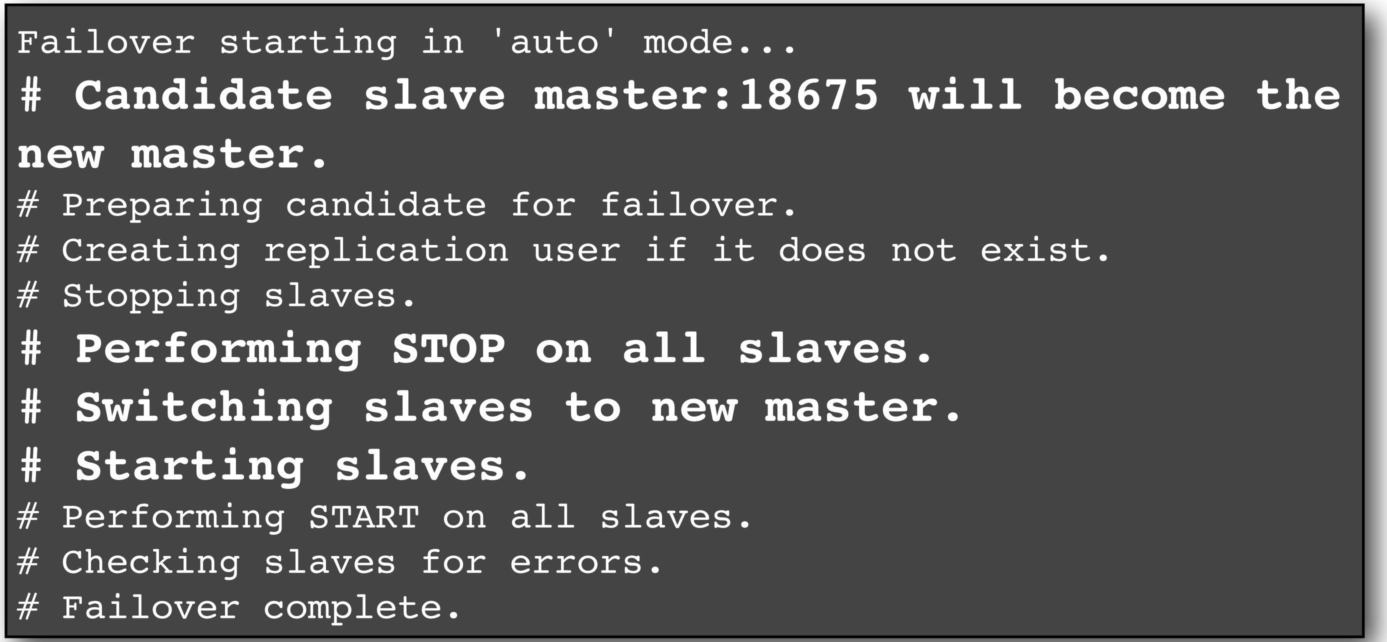
How can I use it for HA and Failover?



```
Failover Mode = auto     Next Interval = Sun Feb 24 13:45:04 2013

Master Information
------------------
Binary Log File   Position  Binlog_Do_DB  Binlog_Ignore_DB
mysql-bin.000002  552

GTID Executed Set
ce40779f-7e7b-11e2-b64d-080027635ef5:1-2

Replication Health Status
+-----------+--------+---------+--------+------------+---------+
| host      | port   | role    | state  | gtid_mode  | health  |
+-----------+--------+---------+--------+------------+---------+
| sbslave1  | 18676  | MASTER  | UP     | ON         | OK      |
| master    | 18675  | SLAVE   | UP     | ON         | OK      |
| sbslave2  | 18677  | SLAVE   | UP     | ON         | OK      |
| sbslave3  | 18678  | SLAVE   | UP     | ON         | OK      |
+-----------+--------+---------+--------+------------+---------+


Q-quit R-refresh H-health G-GTID Lists U-UUIDs
```

How can I use it for HA and Failover?

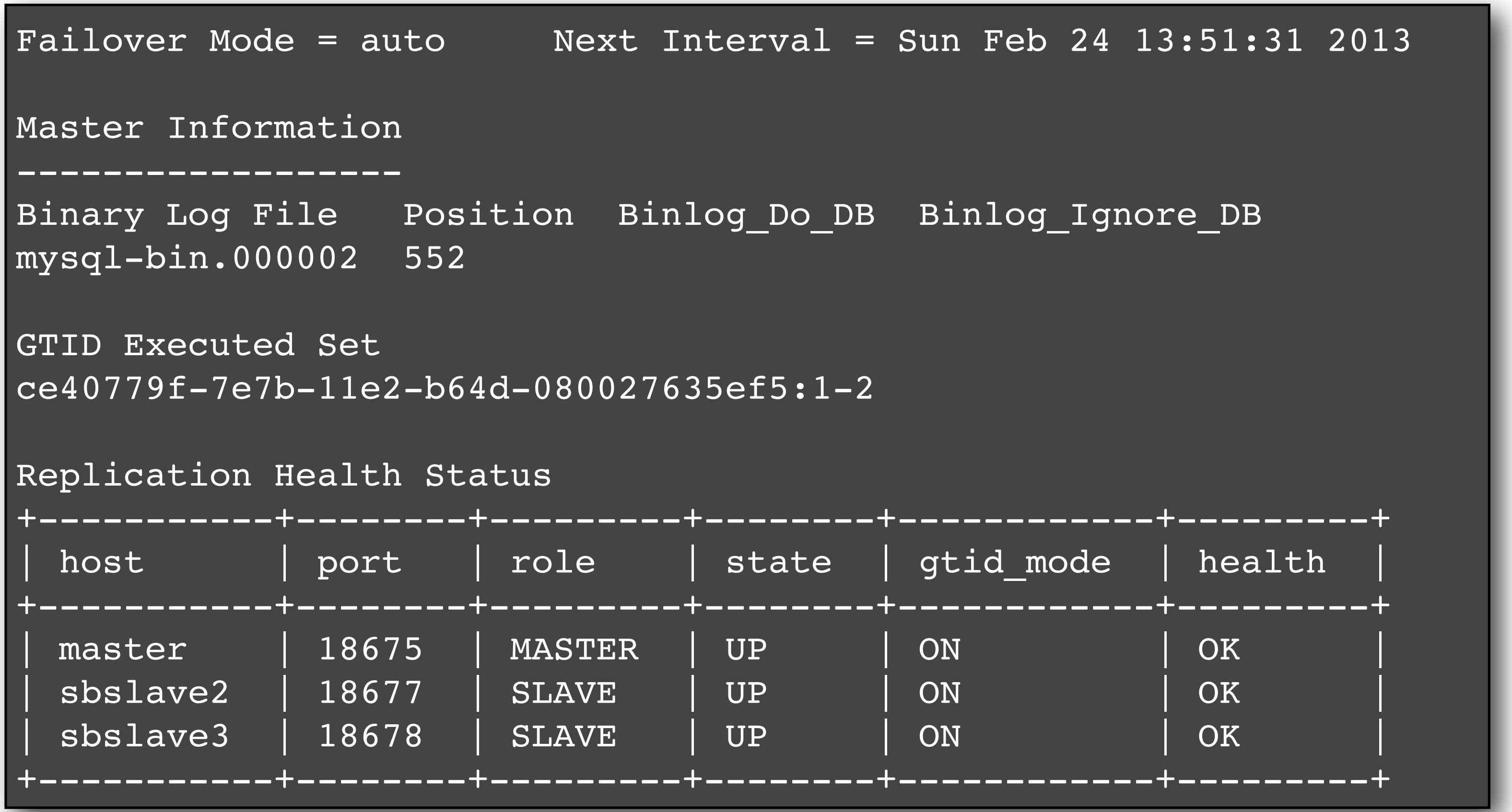
- We kill the slave server and the automatic failover starts:



```
Failover starting in 'auto' mode...
# Candidate slave master:18675 will become the
new master.
# Preparing candidate for failover.
# Creating replication user if it does not exist.
# Stopping slaves.
# Performing STOP on all slaves.
# Switching slaves to new master.
# Starting slaves.
# Performing START on all slaves.
# Checking slaves for errors.
# Failover complete.
```

How can I use it for HA and Failover?

- Failover done:



```
Failover Mode = auto     Next Interval = Sun Feb 24 13:51:31 2013

Master Information
------------------
Binary Log File   Position  Binlog_Do_DB  Binlog_Ignore_DB
mysql-bin.000002  552

GTID Executed Set
ce40779f-7e7b-11e2-b64d-080027635ef5:1-2

Replication Health Status
+-----------+--------+---------+--------+------------+---------+
| host      | port   | role    | state  | gtid_mode  | health  |
+-----------+--------+---------+--------+------------+---------+
| master    | 18675  | MASTER  | UP     | ON         | OK      |
| sbslave2  | 18677  | SLAVE   | UP     | ON         | OK      |
| sbslave3  | 18678  | SLAVE   | UP     | ON         | OK      |
+-----------+--------+---------+--------+------------+---------+
```

How can I use it for HA and Failover?

- **--exec-after**
- **--exec-before**

- You can use your own -pre and -post scripts to do a variety of different tasks:
  - Send a mail.
  - Move a virtual IP.
  - Electrocute the DBA.
  - ...

Take in account...

- 5.6 is a new GA release so there can be bugs... http://bugs.mysql.com/bug.php?id=68460
- There have been some problems to make GTID compatible with MyISAM. From 5.6.9 it is possible to run single statements updating MyISAM tables. Be cautious.
- mysql_upgrade --write-binlog=ON can't connect to a server with GTID enabled. So from 5.6 by default mysql_upgrade has write-binlog disabled.

That’s all Folks!

miguel.nieto@percona.com