# MySQL Replication

PHP[Tek] 2016

Dave Stokes
david.stokes@oracle.com
@stokes
slideshare.net/davidmstokes

## Replication

Past

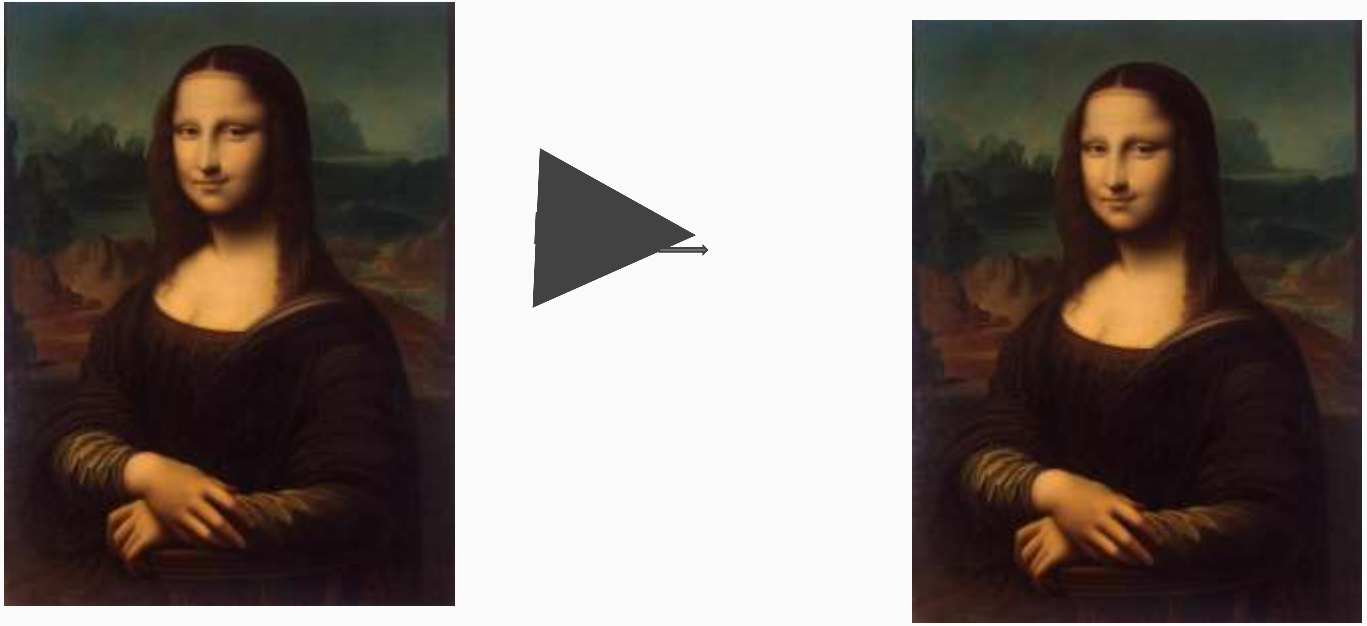
Present

Future

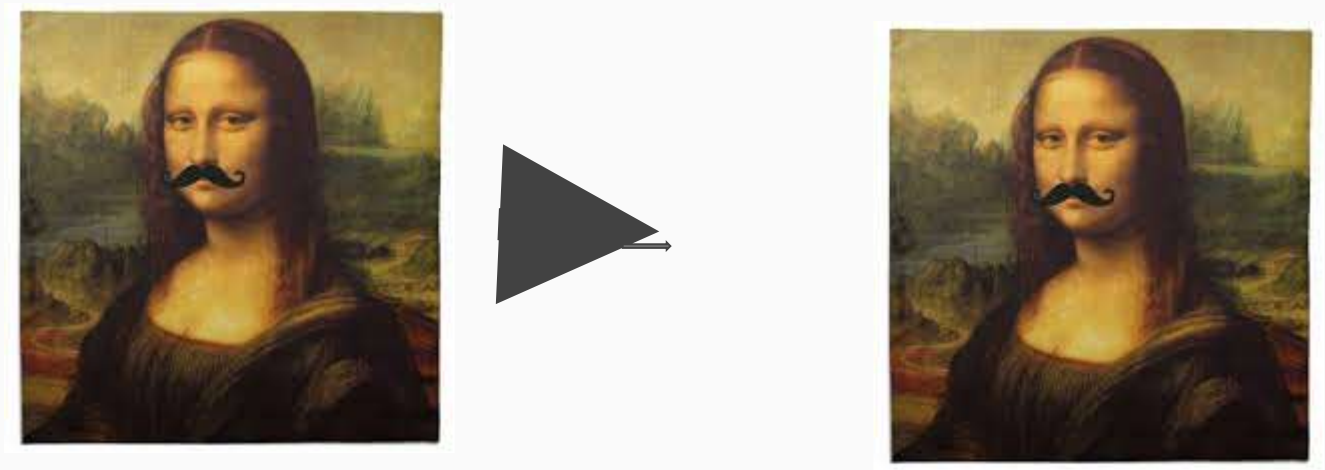
## Basic Premise Behind Replication

I copy all the data off one server and on to another. Then any changes made on the first server get copied over to the second server.

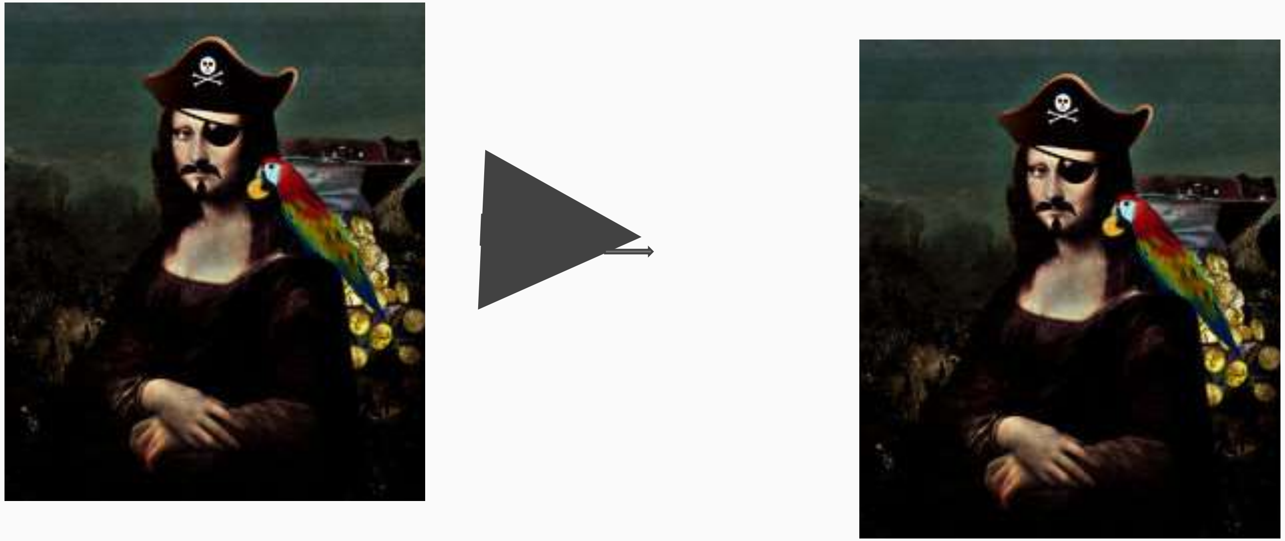
## Make copy of Mona Lisa



## Make copy of changes to Mona Lisa



## Keep making copy of changes to Mona Lisa



## Architecture

Binary Log on master

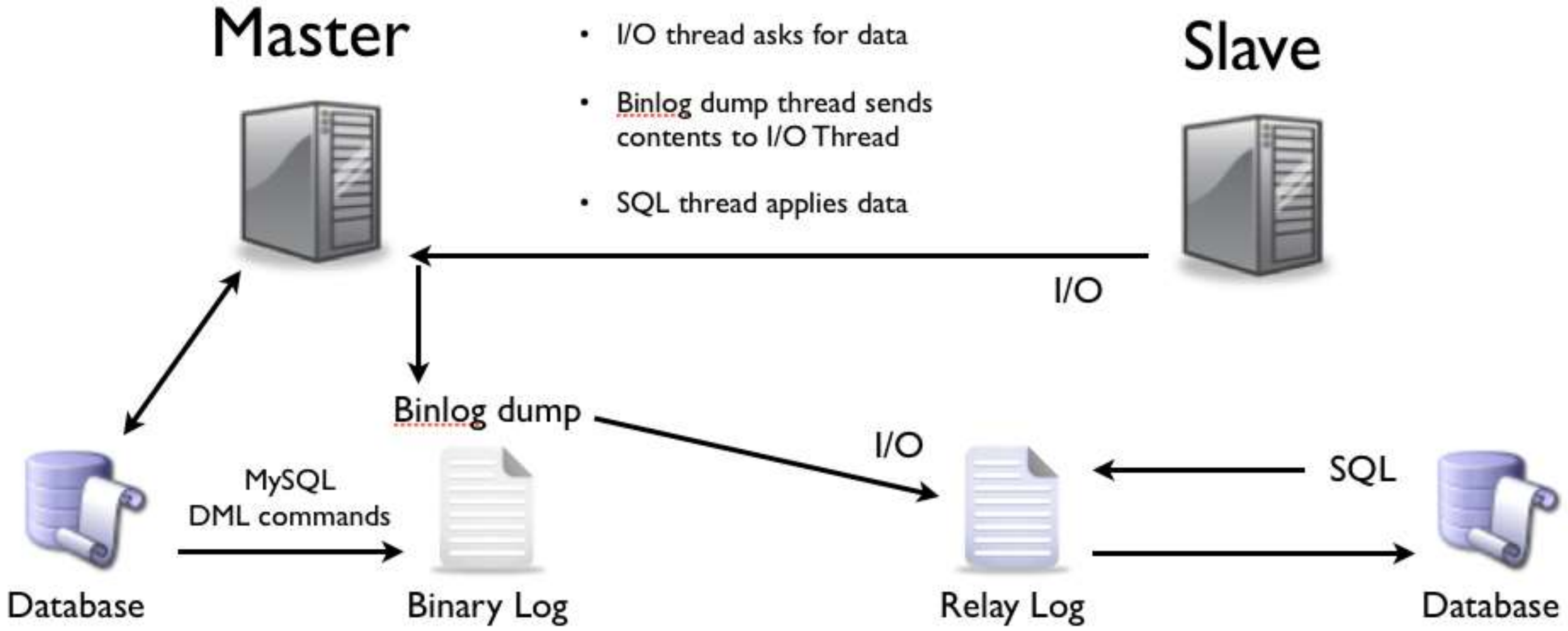
Slave Attaches

I/O Thread Grabs Data

Data copied to slave

Applier thread changes slave’s data

## Graphic overview of replication



- Slave connects to Master
- I/O thread asks for data
- Binlog dump thread sends contents to I/O Thread
- SQL thread applies data

## But what are we replicating?

# Replicating changes to tables and data

SBR: When using statement-based binary logging, the master writes SQL statements to the binary log. Replication of the master to the slave works by executing the SQL statements on the slave. This is called statement-based replication (often abbreviated as SBR), which corresponds to the standard MySQL statement-based binary logging format. Replication capabilities in MySQL version 5.1.4 and earlier used this format exclusively.

Structured Query Language (SQL)

RBR: When using row-based logging, the master writes events to the binary log that indicate how individual table rows are changed. Replication of the master to the slave works by copying the events representing the changes to the table rows to the slave. This is called row-based replication (often abbreviated as RBR).

The Delta

## Or Both

MBR: You can also configure MySQL to use a mix of both statement-based and row-based logging, depending on which is most appropriate for the change to be logged. This is called mixed-format logging. When using mixed-format logging, a statement-based log is used by default. Depending on certain statements, and also the storage engine being used, the log is automatically switched to row-based in particular cases. Replication using the mixed format is often referred to as mixed-based replication or mixed-format replication.

## Notes for the future

With MySQL.5.6 RBR started sending over only the primary key and the changed data (not sending unchanged data) which can drastically cut the amount of data sent to slave servers. This can be huge!!

Many future products will work better with RBR as it more deterministic. So plan accordingly.

## Threading

Before 5.6 MySQL Replication is SINGLE threaded – Airline boarding example

MySQL 5.6 is multi-threaded at the database level

MySQL 5.7 is multi-threaded at the table level

## Syncronicity

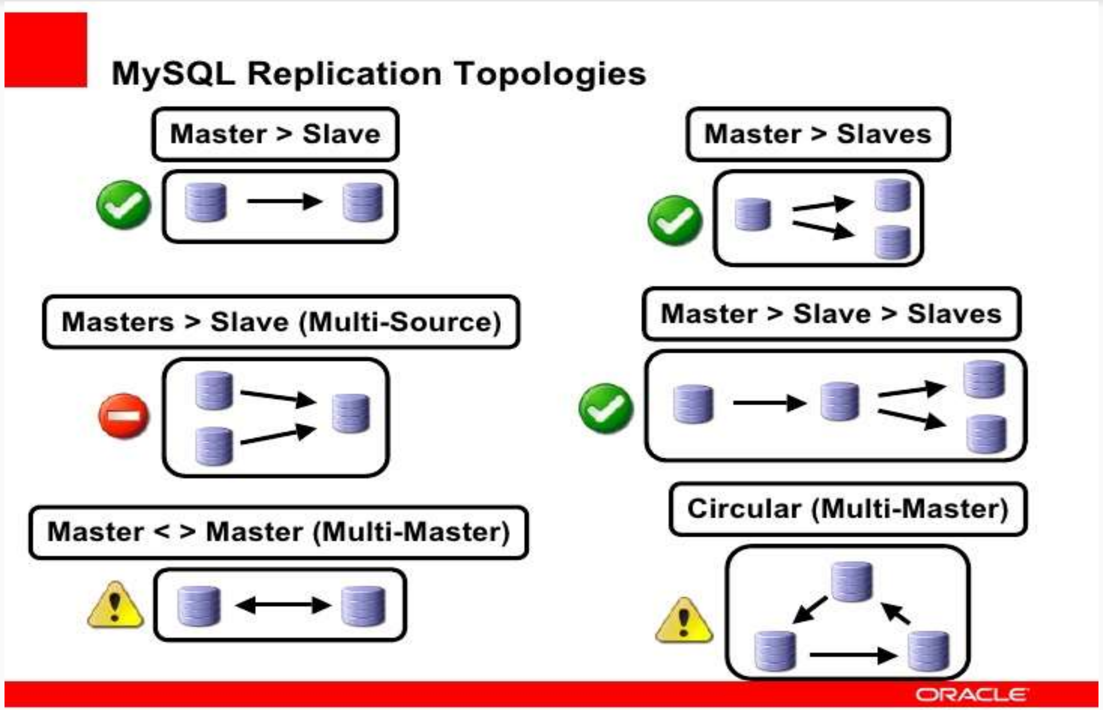
## Async and Semisynchronous

Asynchronous replication -- slave servers retrieve data, master unaware of slave’s consumption.

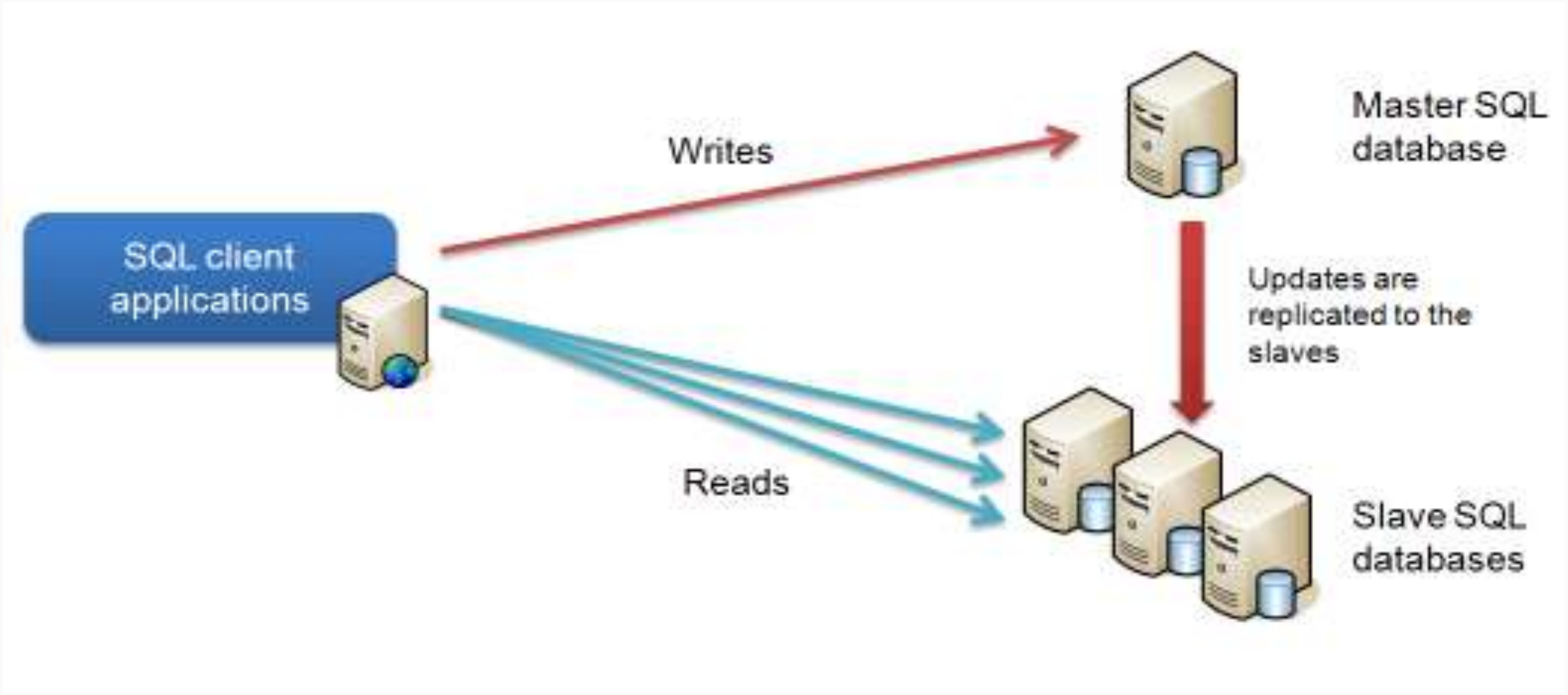
Semisynchronous replication -- a commit performed on the master blocks before returning to the session that performed the transaction until *at least one* slave acknowledges that it has received and logged the events for the transaction  (Note not written to table, just recorded for future).

## Topologies

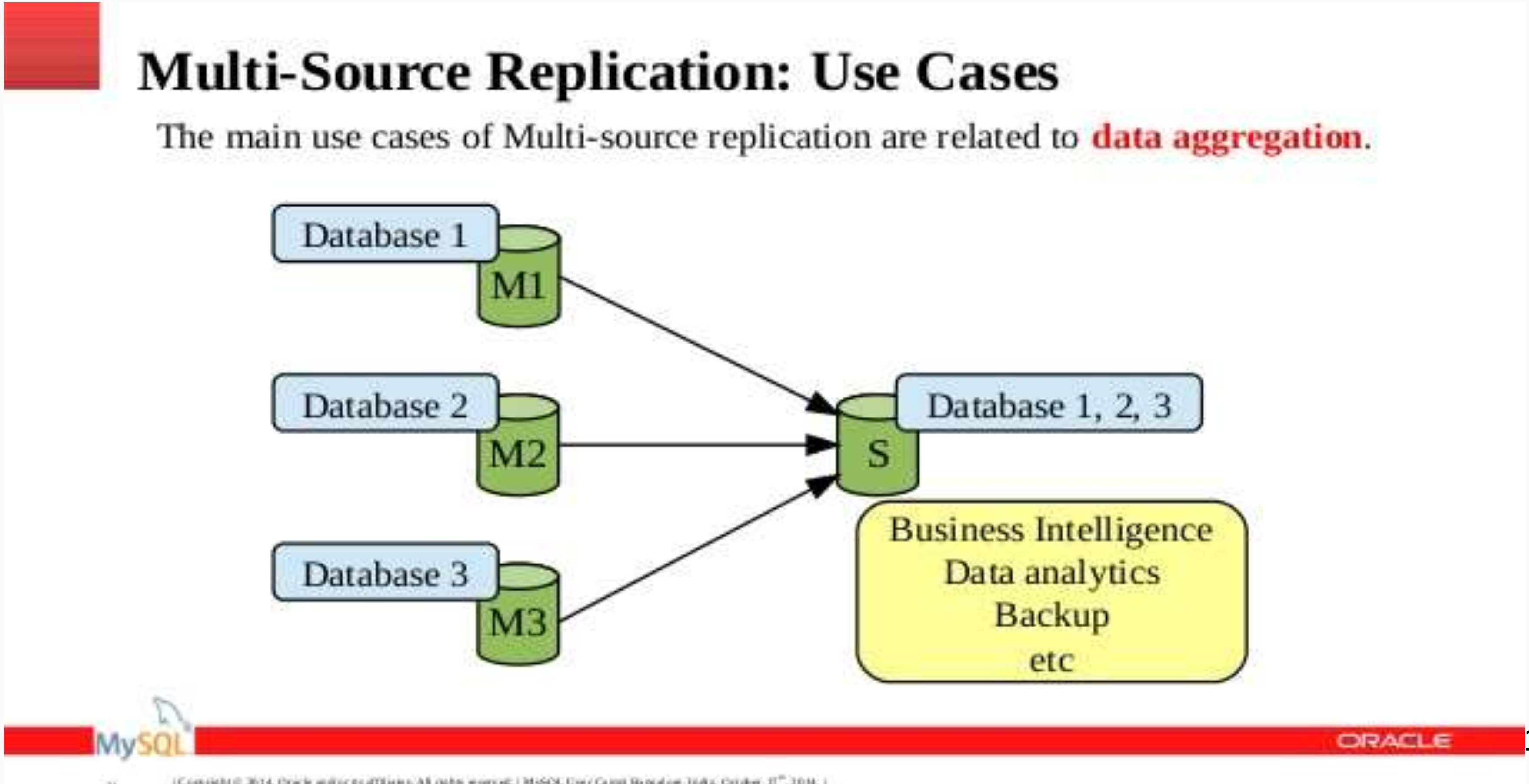
## Topologies -- Before 5.7


MySQL Replication Topologies
Master > Slave
Master > Slaves
Masters > Slave (Multi-Source)
Master > Slave > Slaves
Master < > Master (Multi-Master)
Circular (Multi-Master)
ORACLE

## Common - Read / Write Split



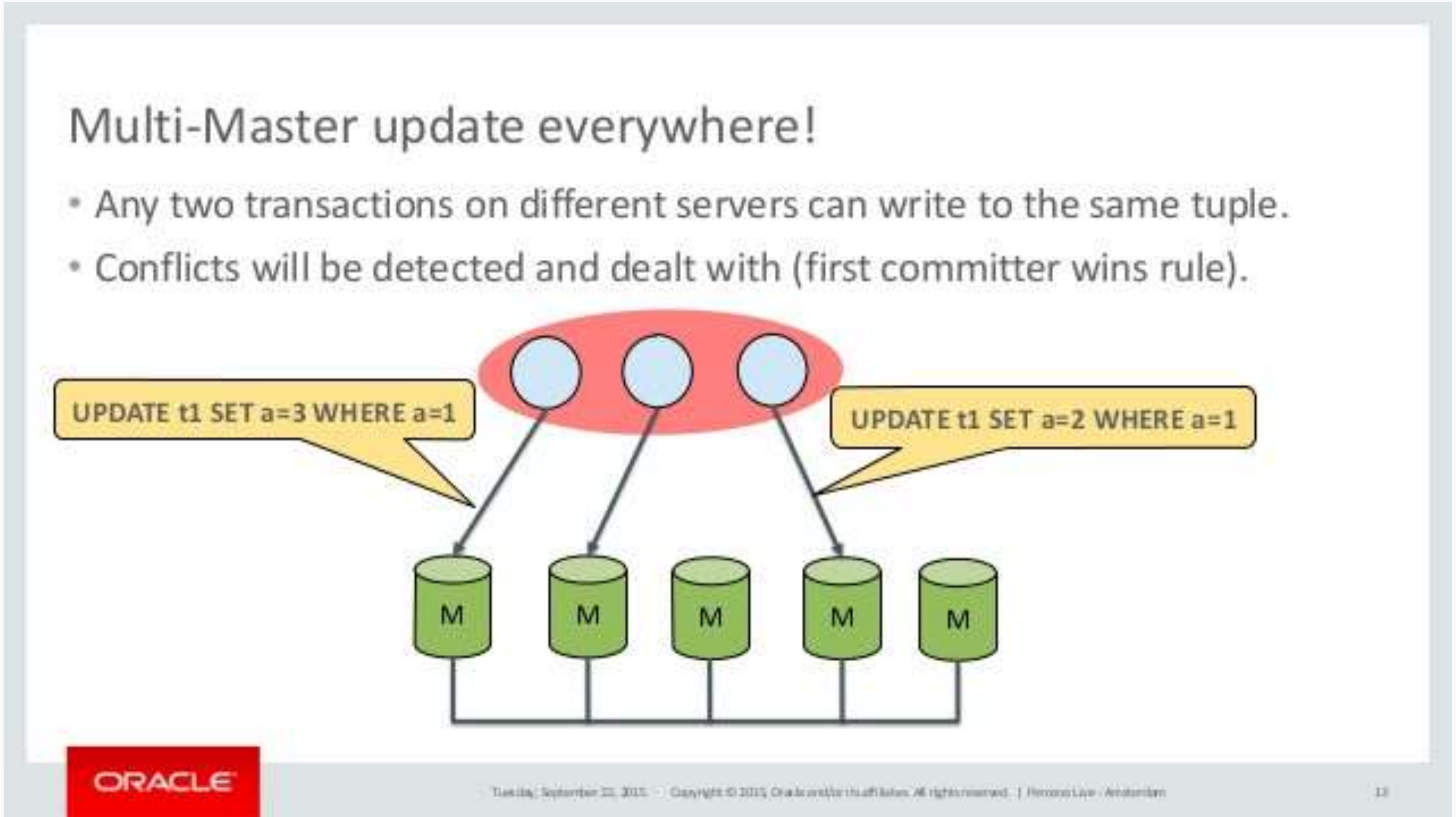
## Multi-source Replication -- MySQL 5.7



### Multi-Source Replication: Use Cases

The main use cases of Multi-source replication are related to **data aggregation**.

## Group Replcation -- Labs.MySQL.Com for evaluation



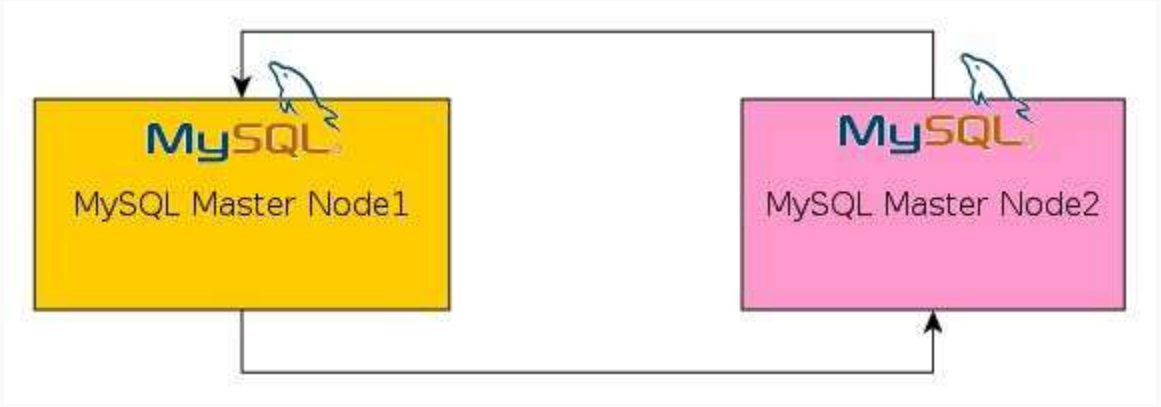
### Multi-Master update everywhere!

- Any two transactions on different servers can write to the same tuple.
- Conflicts will be detected and dealt with (first committer wins rule).

## Multi-Master

Lots of folks want TWO active masters and this can be done but not recomended, You need to have a sharp crew and plan for conflicts.

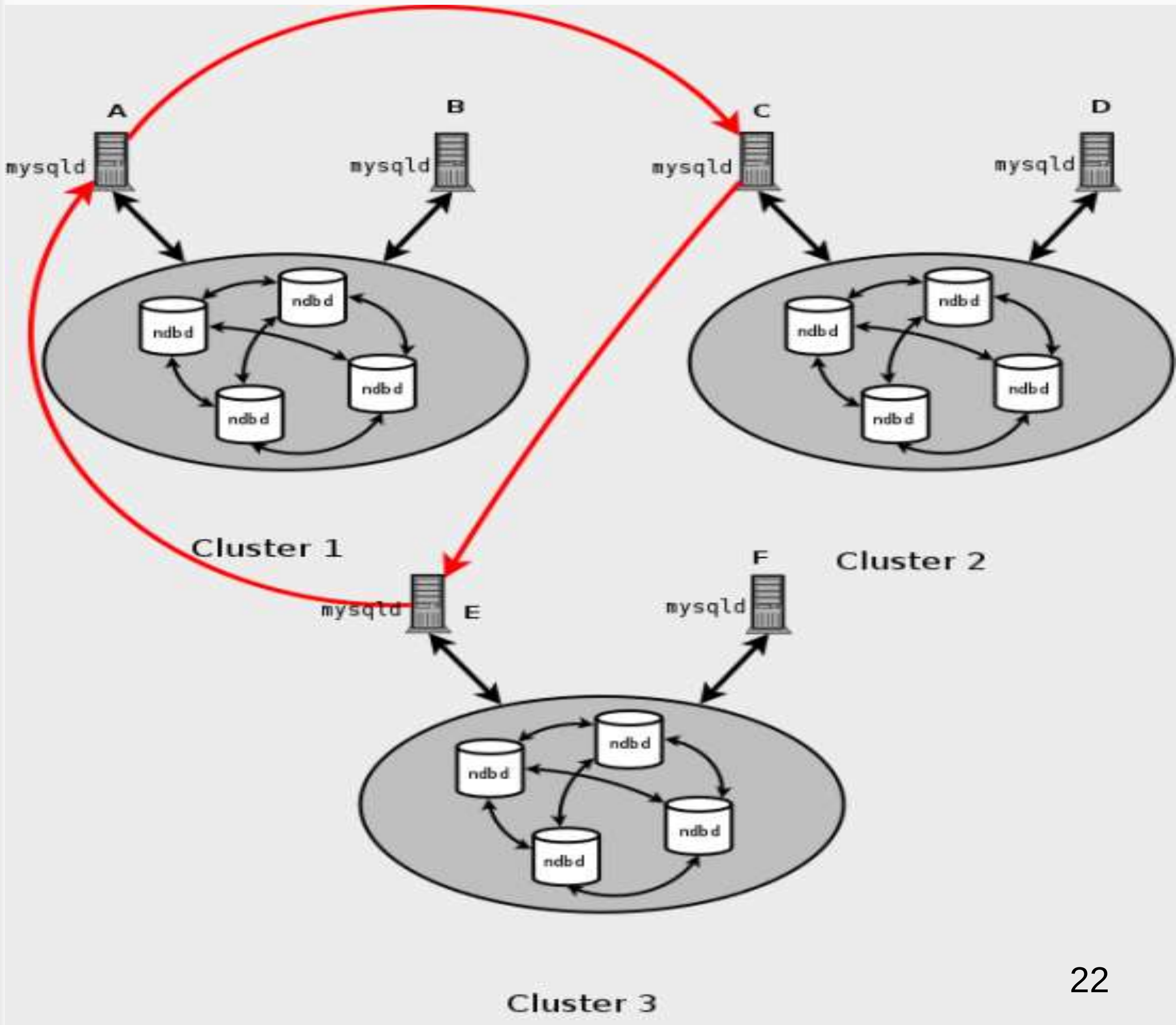
Not recommended



## Multi-Master MySQL Cluster

You can run active-active master-master with MySQL Cluster, even between data centers.

This can be very expensive and MySQl Cluster is not a full featured version of the MySQL Server.

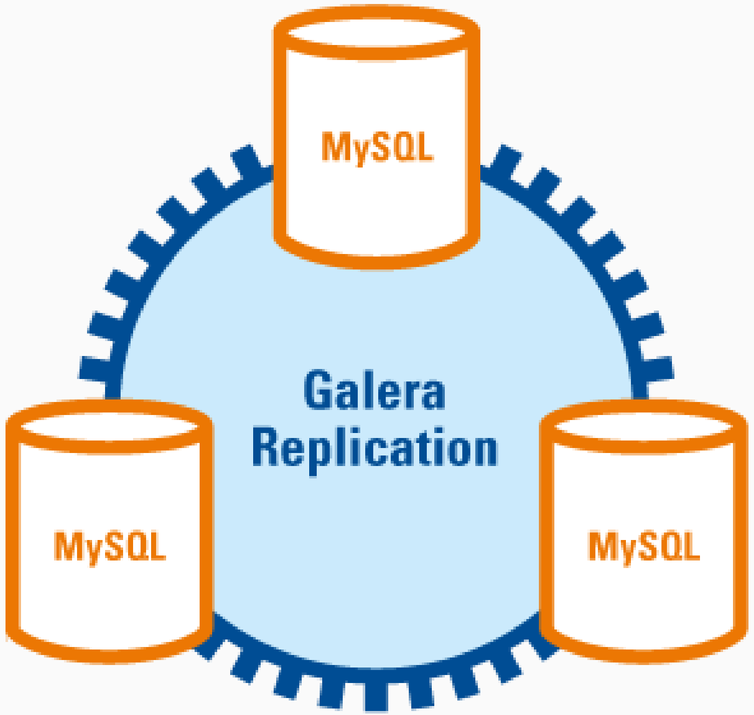


## Galera Cluster

Layer separate from MySQL that is mainly for high availability (not high performance).

Claims to have snapshot isolation on transactions but watch out for ‘first committer wins’ and prepare for rollbacks.

Not low latency



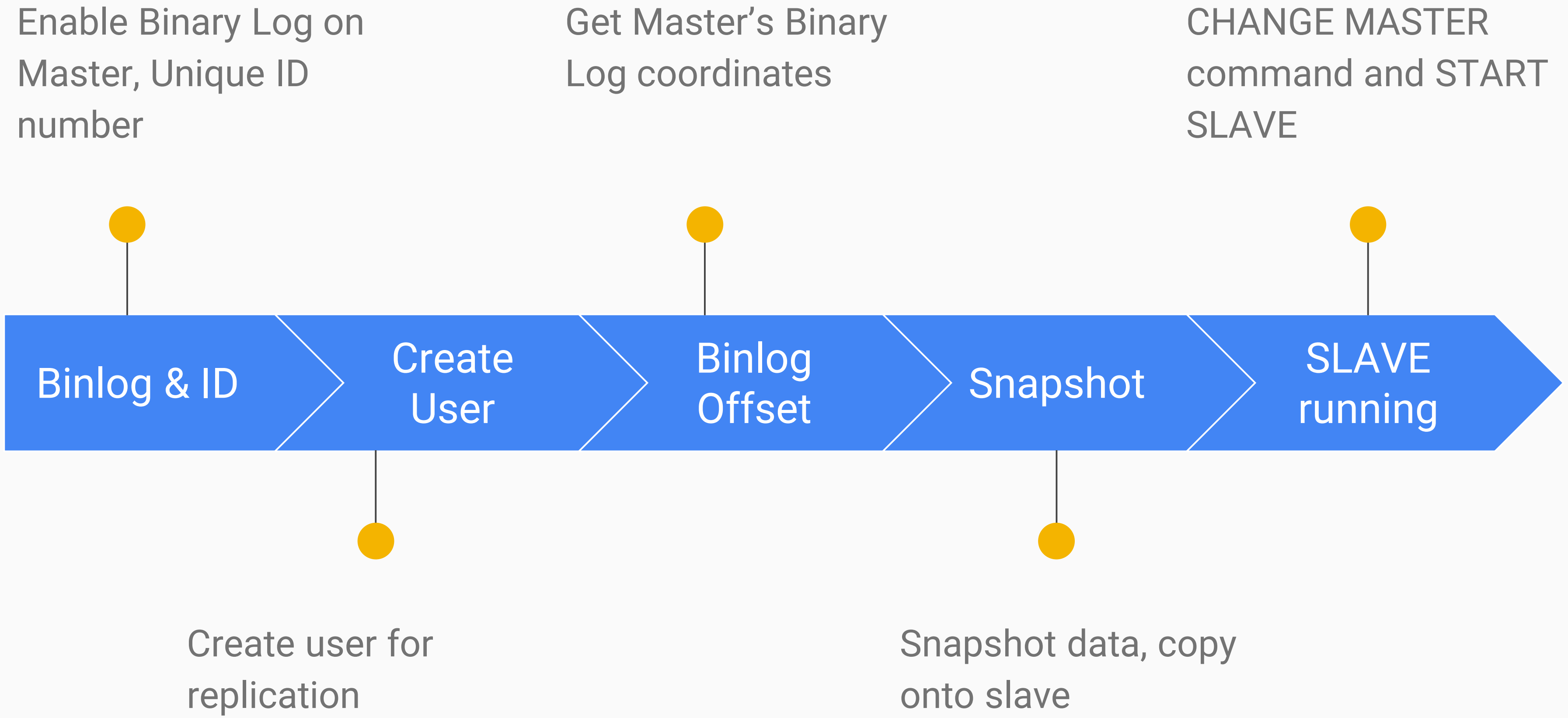
## How to set up MySQL Replication

## Two types of replication w/ & w/o GTIDs

A global transaction identifier (GTID) is a unique identifier created and associated with each transaction committed on the server of origin (master). This identifier is unique not only to the server on which it originated, but is unique across all servers in a given replication setup. There is a 1-to-1 mapping between all transactions and all GTIDs.

```
3E11FA47-71CA-11E1-9E33-C80AA9429562:1-5
```

**Note the 1-5 is a group of transactions**


Enable Binary Log on Master, Unique ID number
Get Master's Binary Log coordinates
CHANGE MASTER command and START SLAVE
Binlog & ID
Create User
Binlog Offset
Snapshot
SLAVE running
Create user for replication
Snapshot data, copy onto slave

## Before GTIDs (MySQL 5.5 and before)

## Enable Binary Log & Unique ID on Master

Edit my.cng file

```
[mysqld]

log-bin=mysql-bin

server-id=1
```

## Create replication user

```
mysql> CREATE USER 'repl'@'%.mydomain.com' IDENTIFIED BY
'slavepass';
mysql> GRANT REPLICATION SLAVE ON *.* TO
'repl'@'%.mydomain.com';
```

## Get binary log position

```
mysql> FLUSH TABLES WITH READ LOCK;

mysql > SHOW MASTER STATUS;

+------------------+----------+--------------+------------------+

| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |

+------------------+----------+--------------+------------------+

| mysql-bin.000003 | 73       | test         | manual,mysql     |

+------------------+----------+--------------+------------------+
```

## Unlock tables

```
ysql> UNLOCK TABLES;
```

## Config slave & load data

No config thee slave server.  Remember server-id must be unique

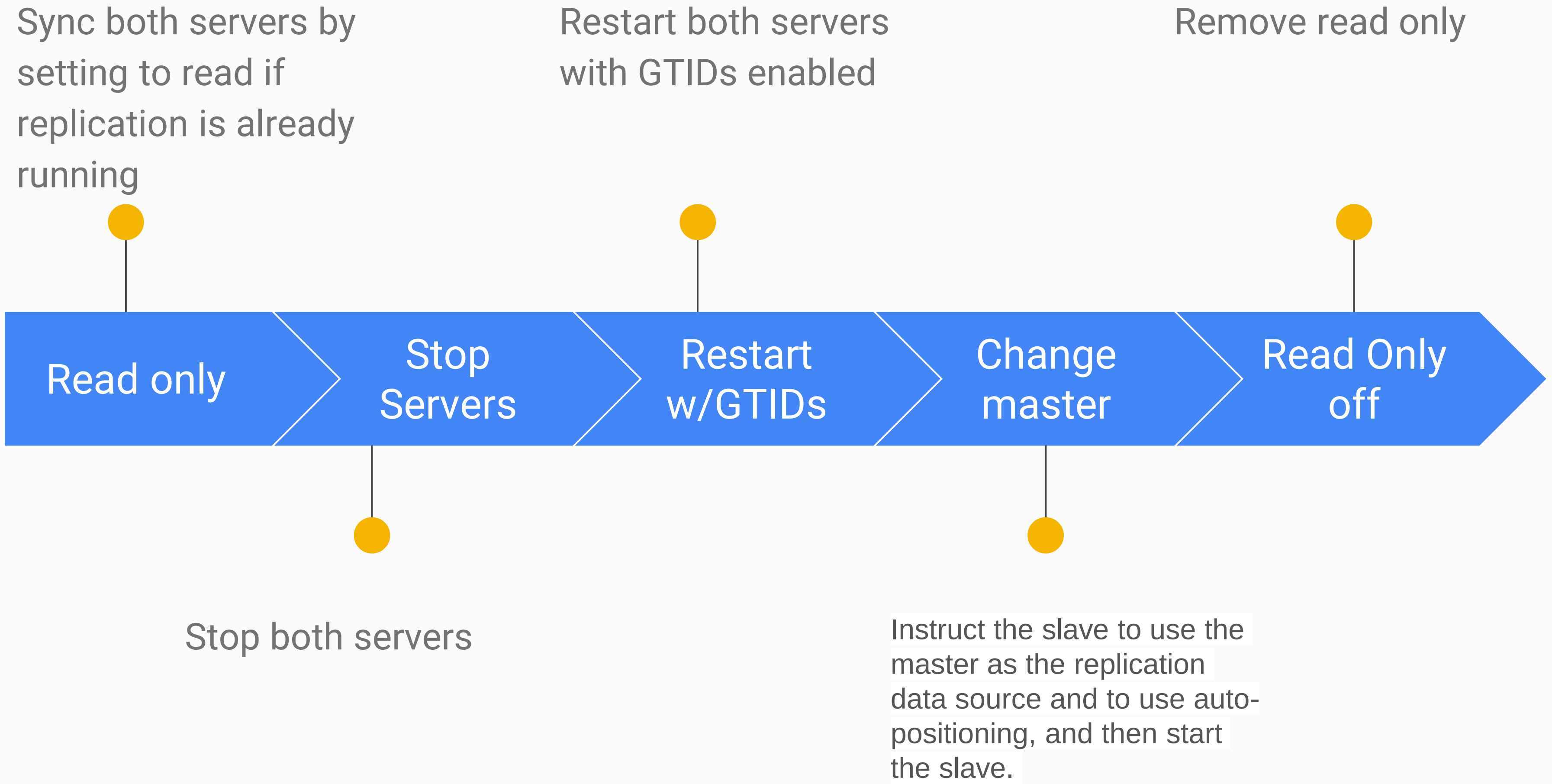
```
[mysqld]
server-id=2
```

```
shell> mysql -h master < fulldb.dump
```

## Change Master & Start Slave

```
mysql> CHANGE MASTER TO
    ->         MASTER_HOST='master_host_name',
    ->         MASTER_USER='replication_user_name',
    ->         MASTER_PASSWORD='replication_password',
    ->         MASTER_LOG_FILE='recorded_log_file_name',
    ->         MASTER_LOG_POS=recorded_log_position;


mysql> START SLAVE;
```

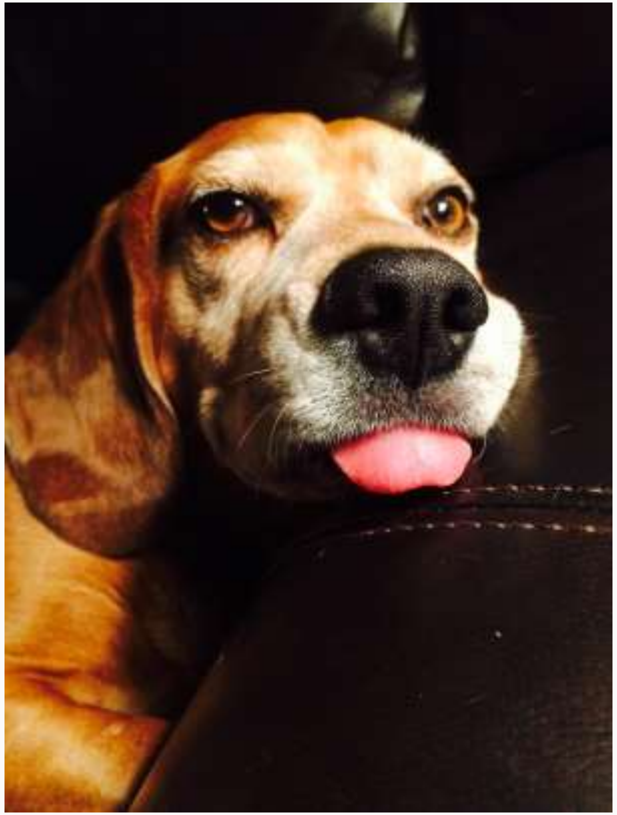

Sync both servers by setting to read if replication is already running
Restart both servers with GTIDs enabled
Remove read only
Read only
Stop Servers
Restart w/GTIDs
Change master
Read Only off
Stop both servers
Instruct the slave to use the master as the replication data source and to use auto-positioning, and then start the slave.

With GTIDs (MySQL 5.6 and later)

## Replication Setup with GTIDs

```
mysql> SET @@global.read_only = ON;shell> mysqladmin -uusername -p shutdown
shell> mysqld --gtid-mode=ON --enforce-gtid-consistency &
mysql> CHANGE MASTER TO
     >     MASTER_HOST = host,
     >     MASTER_PORT = port,
     >     MASTER_USER = user,
     >     MASTER_PASSWORD = password,
     >     MASTER_AUTO_POSITION = 1;
mysql> START SLAVE;
mysql> SET @@global.read_only = OFF;
```

## 30 Minutes



30 Minutes!  Can’t Cover all of MySQL Replication in 30 minutes!!!!!

## MySQL Utilities

FREE Scripts written in Python, used with MySQL Workbench or stand alone

1. Copy, diff databases
2. Disk usage, grants, copy users
3. Search for processed and kill ‘em
4. Setup replication and failover

# Mysqlrplcheck -- check replication setup

```
shell> mysqlrplcheck --master=root@host1:3310 --slave=root@host2:3311
# master on host1: ... connected.
# slave on host2: ... connected.
Test Description                                                  Status
------------------------------------------------------------------------
Checking for binary logging on master                             [pass]
Are there binlog exceptions?                                      [pass]
Replication user exists?                                          [pass]
Checking server_id values                                         [pass]
Is slave connected to master?                                     [pass]
Check master information file                                     [pass]
Checking InnoDB compatibility                                     [pass]
Checking storage engines compatibility                            [pass]
Checking lower_case_table_names settings                          [pass]
Checking slave delay (seconds behind master)                      [pass]
# ...done.
```

## Mysqlrplcheck -- replication checker

```
shell> mysqlrplsync --master=user:pass@localhost:3310 \
          --slaves=rpl:pass@localhost:3311,rpl:pass@localhost:3312
#
# GTID differences between Master and Slaves:
# - Slave 'localhost@3311' is 15 transactions behind Master.
# - Slave 'localhost@3312' is 12 transactions behind Master.
#
# Checking data consistency.
#
# Using Master 'localhost@3310' as base server for comparison.
# Checking 'test_rplsync_db' database...
# - Checking 't0' table data...
#   [OK] `test_rplsync_db`.`t0` checksum for server 'localhost@3311'.
#   [OK] `test_rplsync_db`.`t0` checksum for server 'localhost@3312'.
# - Checking 't1' table data...
#   [OK] `test_rplsync_db`.`t1` checksum for server 'localhost@3311'.
#   [OK] `test_rplsync_db`.`t1` checksum for server 'localhost@3312'.
# Checking 'test_db' database...
```

## Mysqlslavetrx -- skip bad transactions

```
shell> mysqlslavetrx --gtid-set=af6b22ee-7b0b-11e4-aa8d-606720440b68:7-9 \
          --slaves=user:pass@localhost:3311,user:pass@localhost:3312
WARNING: Using a password on the command line interface can be insecure.
#
# GTID set to be skipped for each server:
# - localhost@3311: af6b22ee-7b0b-11e4-aa8d-606720440b68:7-9
# - localhost@3312: af6b22ee-7b0b-11e4-aa8d-606720440b68:7-9
#
# Injecting empty transactions for 'localhost:3311'...
# Injecting empty transactions for 'localhost:3312'...
#
#...done.
```

## Mysqlfailover -- ser up automatic failover

```
shell> mysqlfailover --master=root@localhost:3331 --discover-slaves-login=root --
log=log.txt

MySQL Replication Monitor and Failover Utility
Failover Mode = auto     Next Interval = Mon Mar 19 15:56:03 2012

Master Information
------------------
Binary Log File   Position  Binlog_Do_DB  Binlog_Ignore_DB
mysql-bin.000001  571

GTID Executed Set
2A67DE00-2DA1-11E2-A711-00764F2BE90F:1-7 [...]
```

# Finished!

Hard to get a lot of info in in such a short time. Please get details from the MySQL Manual

David.Stokes@oracle.com

@stoker

slideshare.net/davidmstokes