MySQL Replication, the Community Sceptic Roundup

Giuseppe Maxia

Quality Assurance Architect

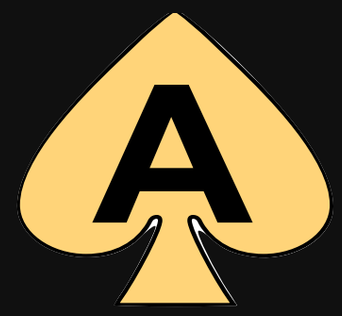
at VMware

@datacharmer

About me

Who's this guy?

- Giuseppe Maxia, a.k.a. "The Data Charmer"
  - QA Architect at VMware
  - 25+ years development and DB experience
  - Long timer MySQL community member.
  - Oracle ACE Director
  - Blog: http://datacharmer.blogspot.com
  - Twitter: @datacharmer



SKEPTIC?

skep·tic

/ˈskeptik/

noun: sceptic

1. a person inclined to question or doubt all accepted opinions.
   synonyms: cynic, doubter

2. PHILOSOPHY
   an ancient or modern philosopher who denies the possibility of knowledge, or even rational belief, in some sphere.

SKEPTIC?

Features are announced.

But not always they are usable.

We verify every claim.

Supporting material and software

http://bit.ly/my-rep-samples

(or check 'datacharmer' on GitHub)

Summary

What will we see in this session

- The concepts of Replication
- Why monitoring matters
- Global Transaction Identifiers
- Multi source replication
- Parallel replication

Actors

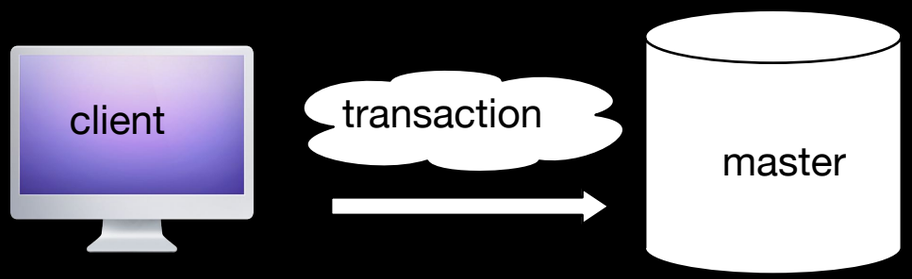
We will see practical examples with the following systems

- MySQL 5.6.29
- MySQL 5.7.12
- MariaDB 10.0.20
- MariaDB 10.1.13

Part 1 - MySQL Replication in a nutshell

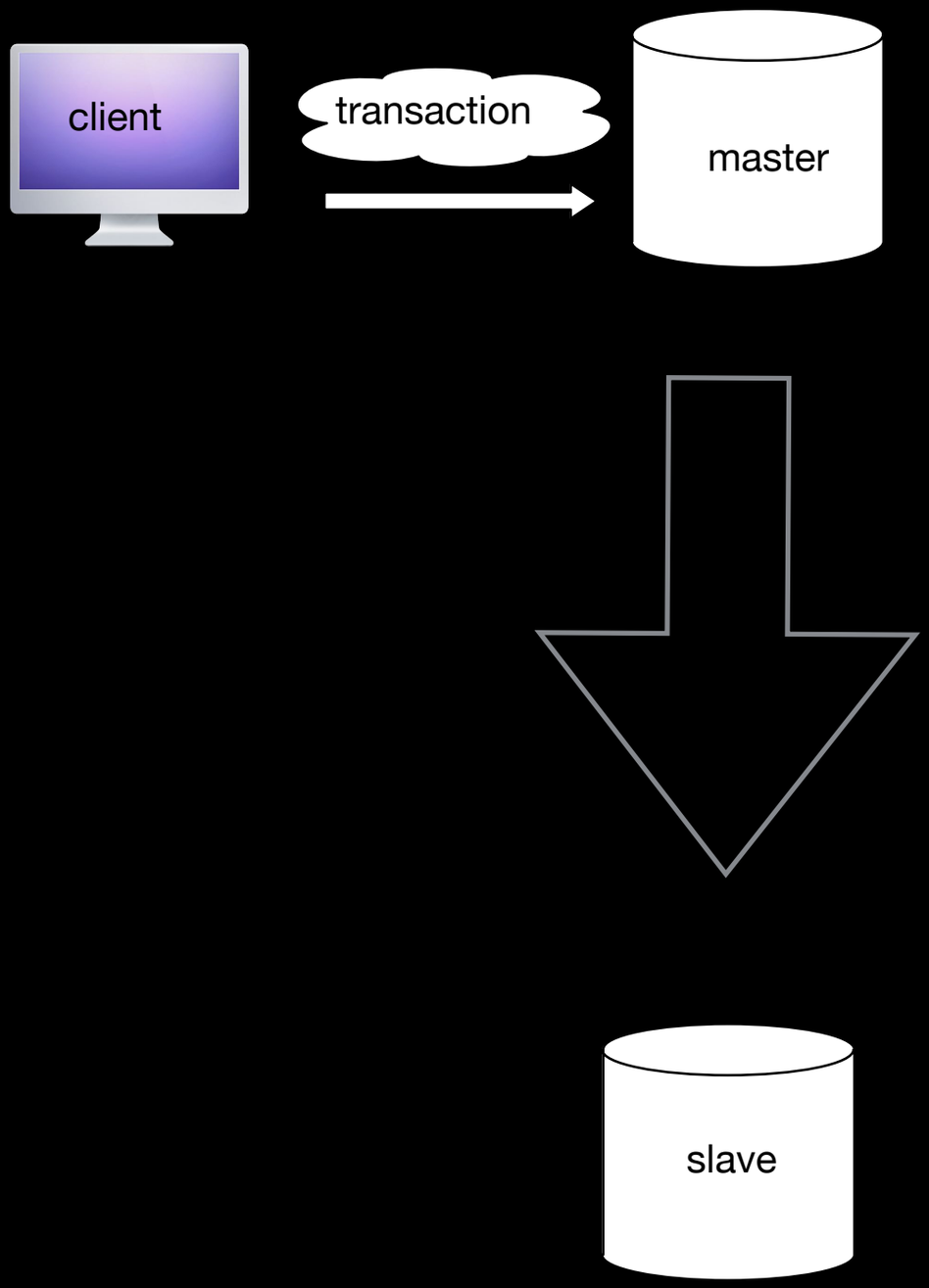
The concepts of replication

The basics



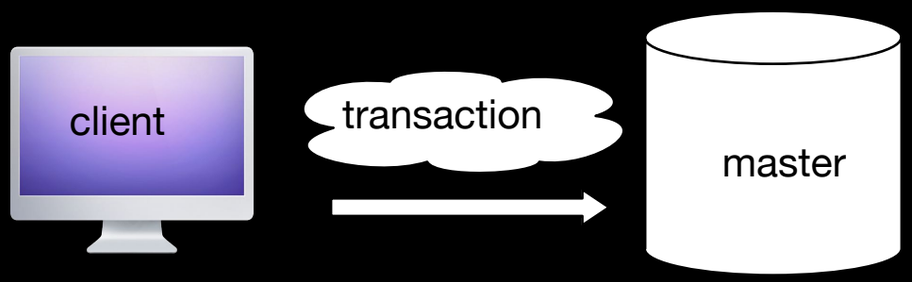
The concepts of replication

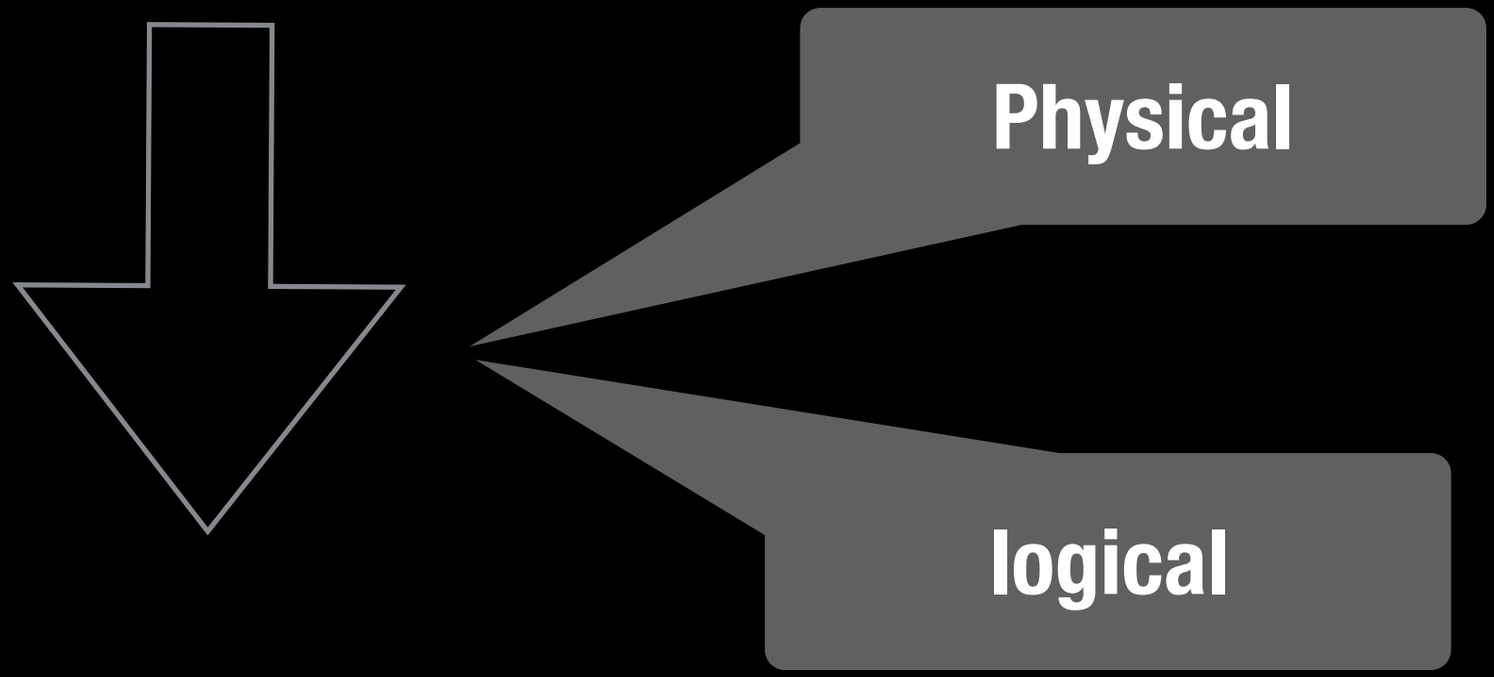
The basics

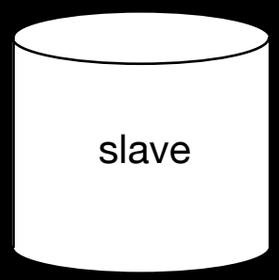


The concepts of replication

The basics

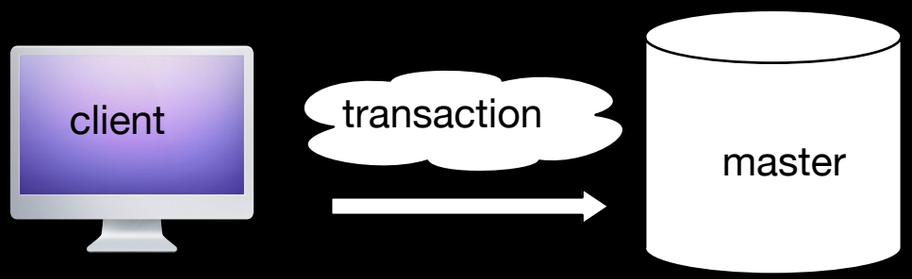


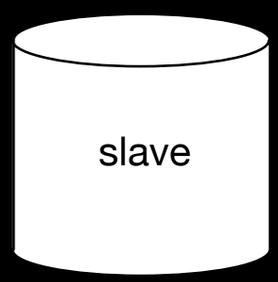




The concepts of replication

The basics

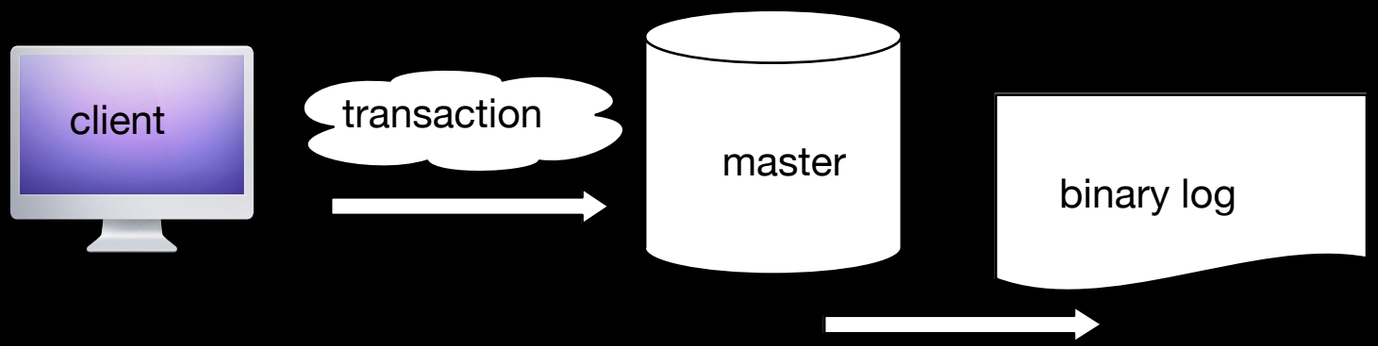


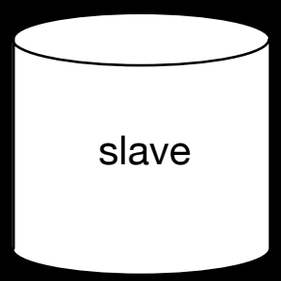


continuent by vmware

The concepts of replication

The basics

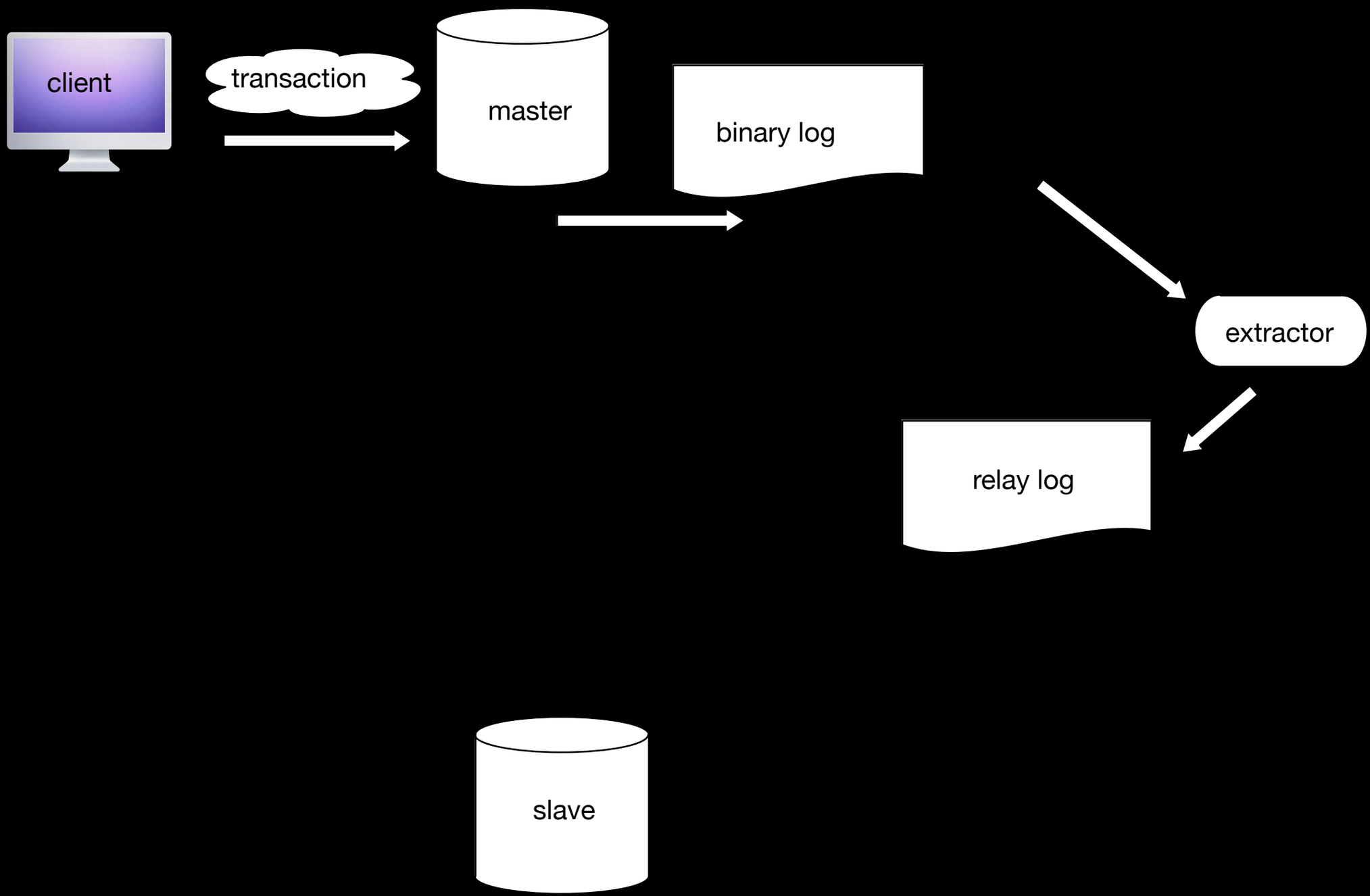




continuent by vmware

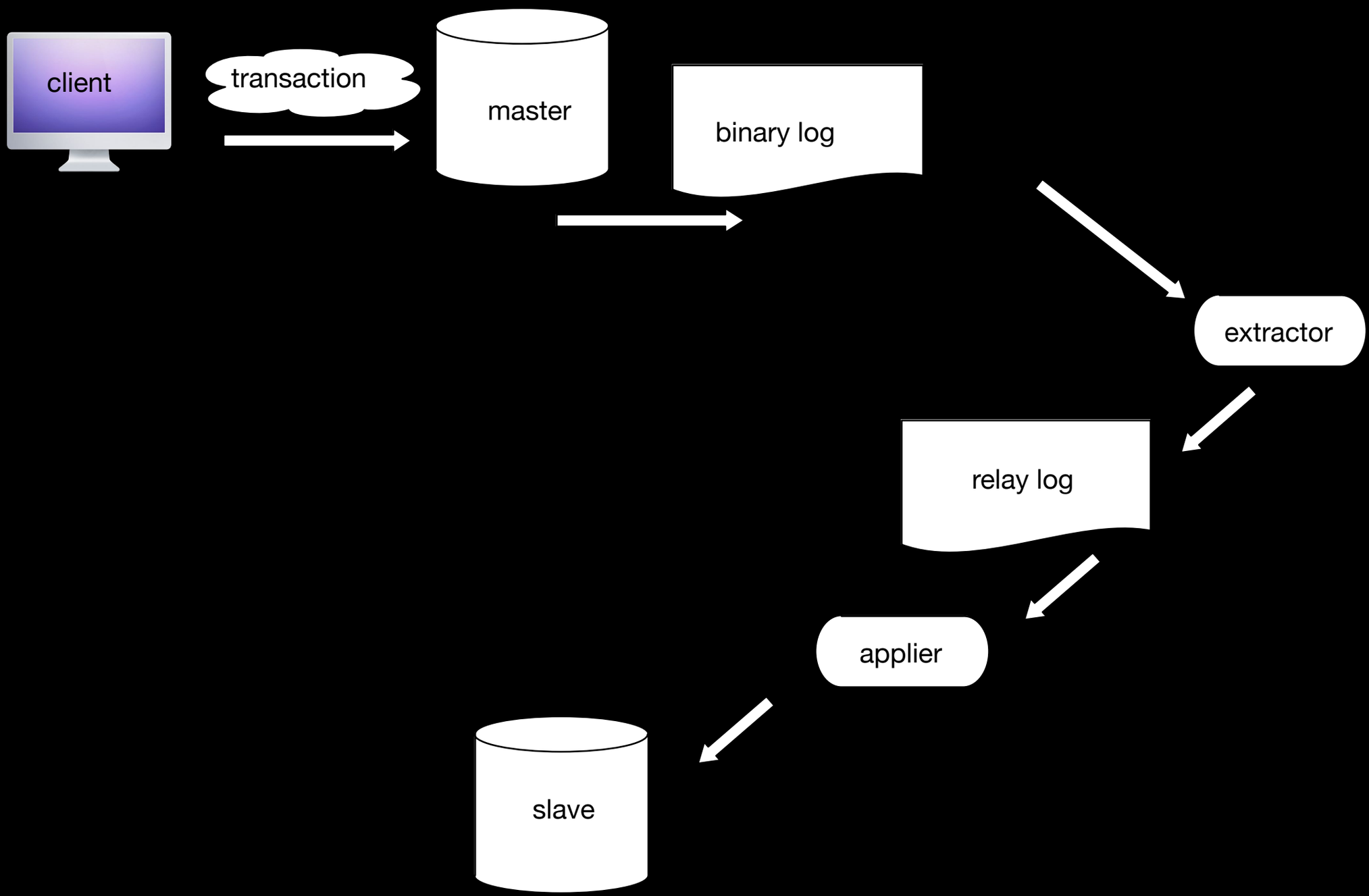
The concepts of replication

The basics



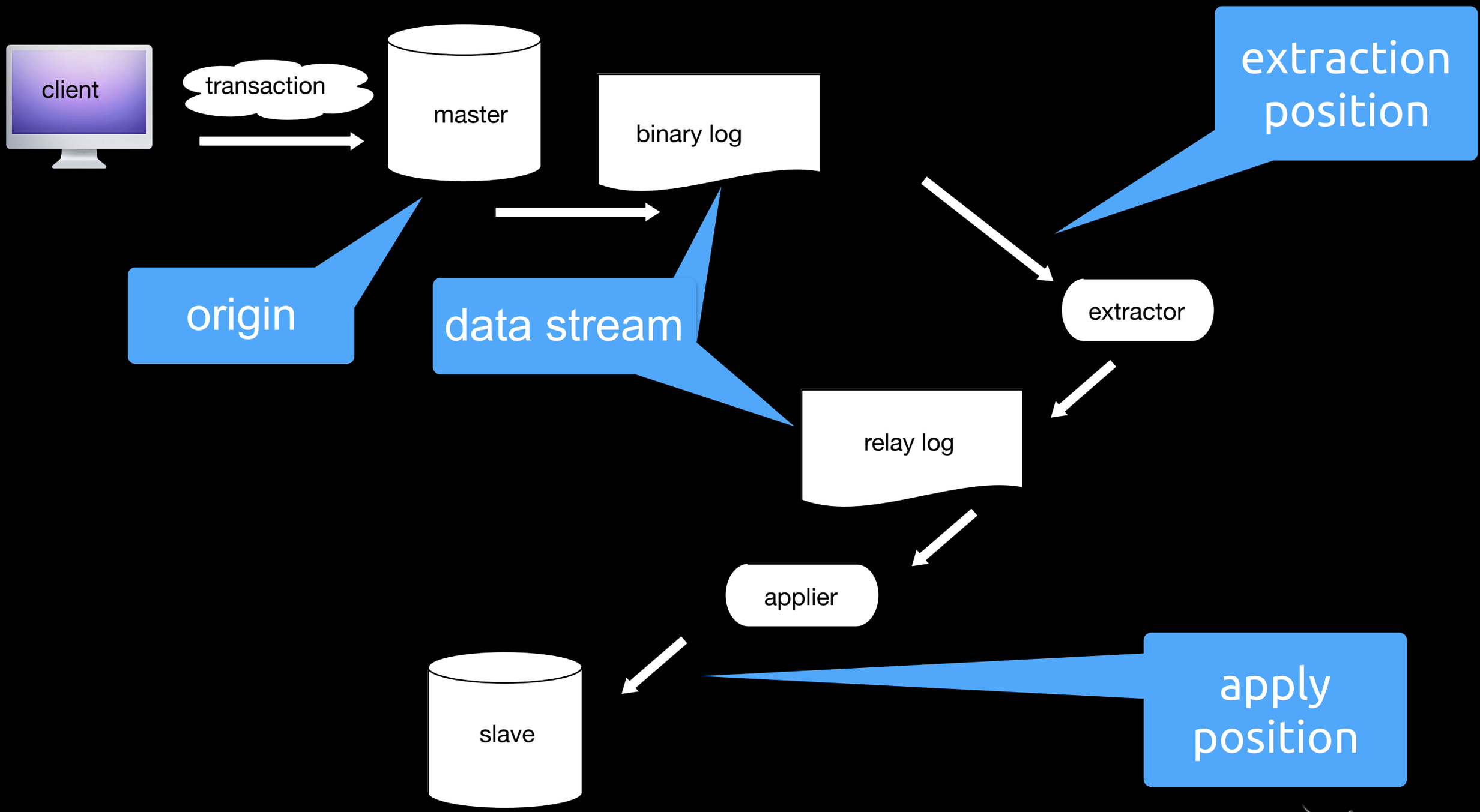
The concepts of replication

The basics



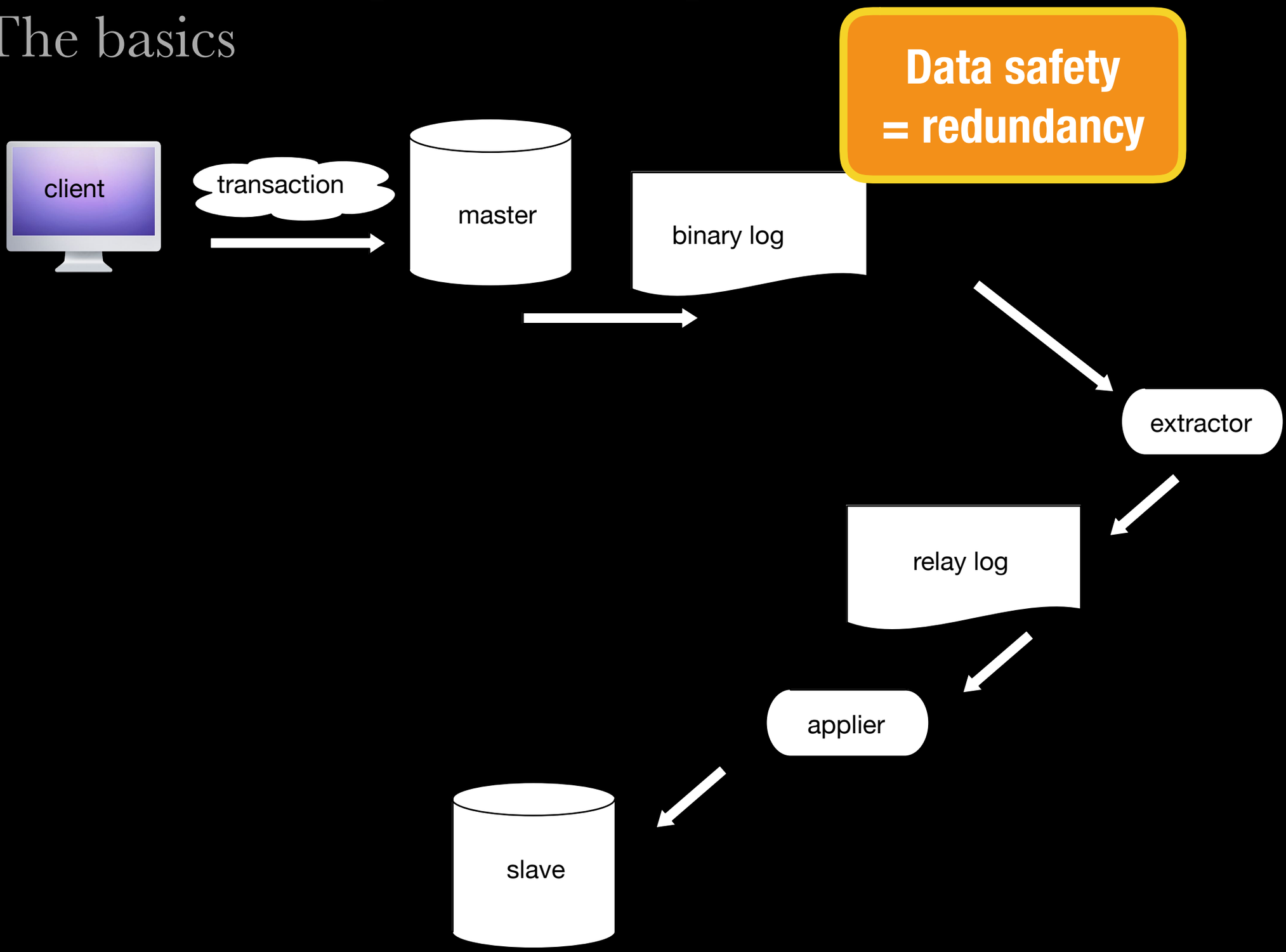
The concepts of replication

The basics



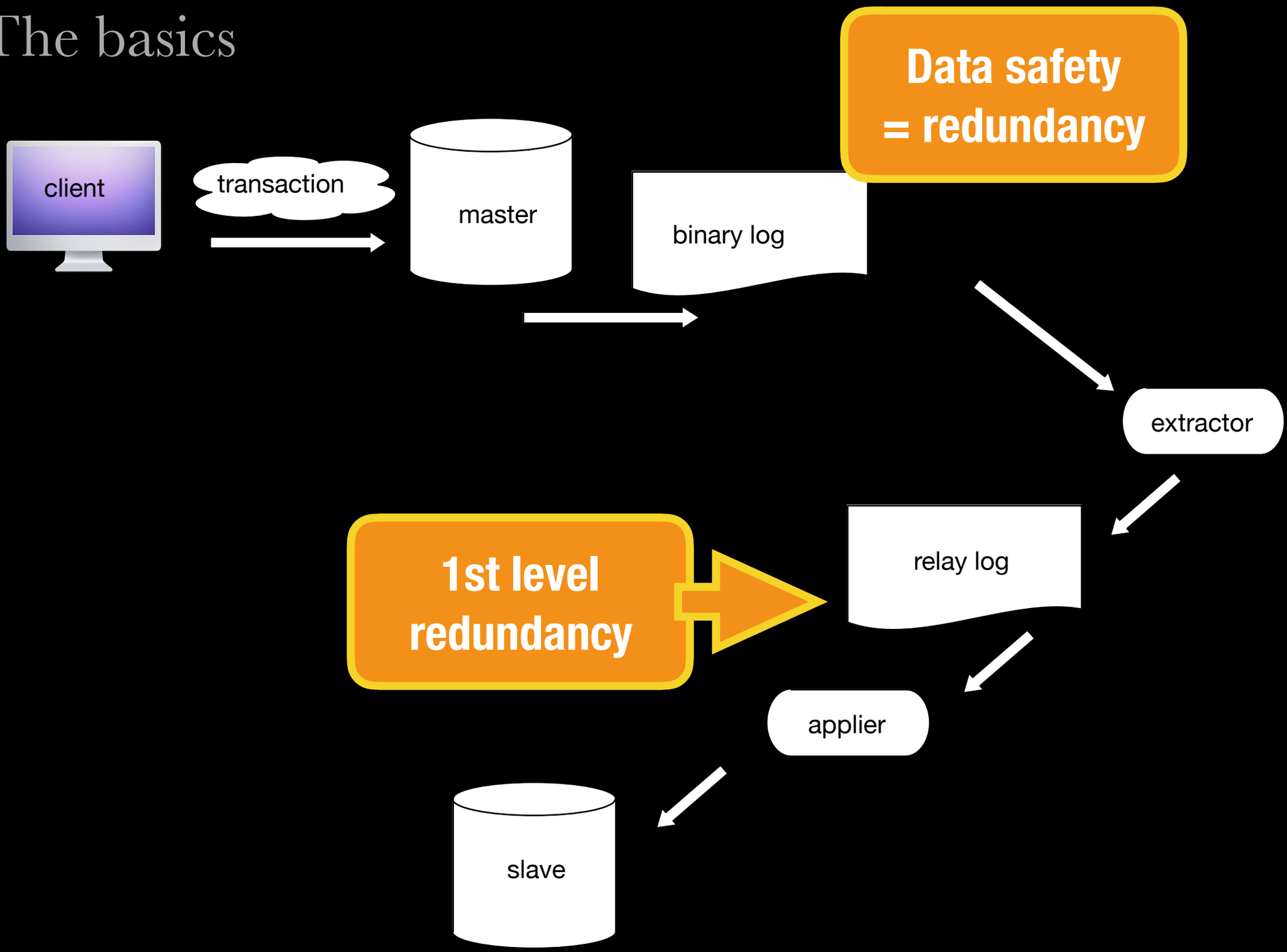
The concepts of replication

The basics



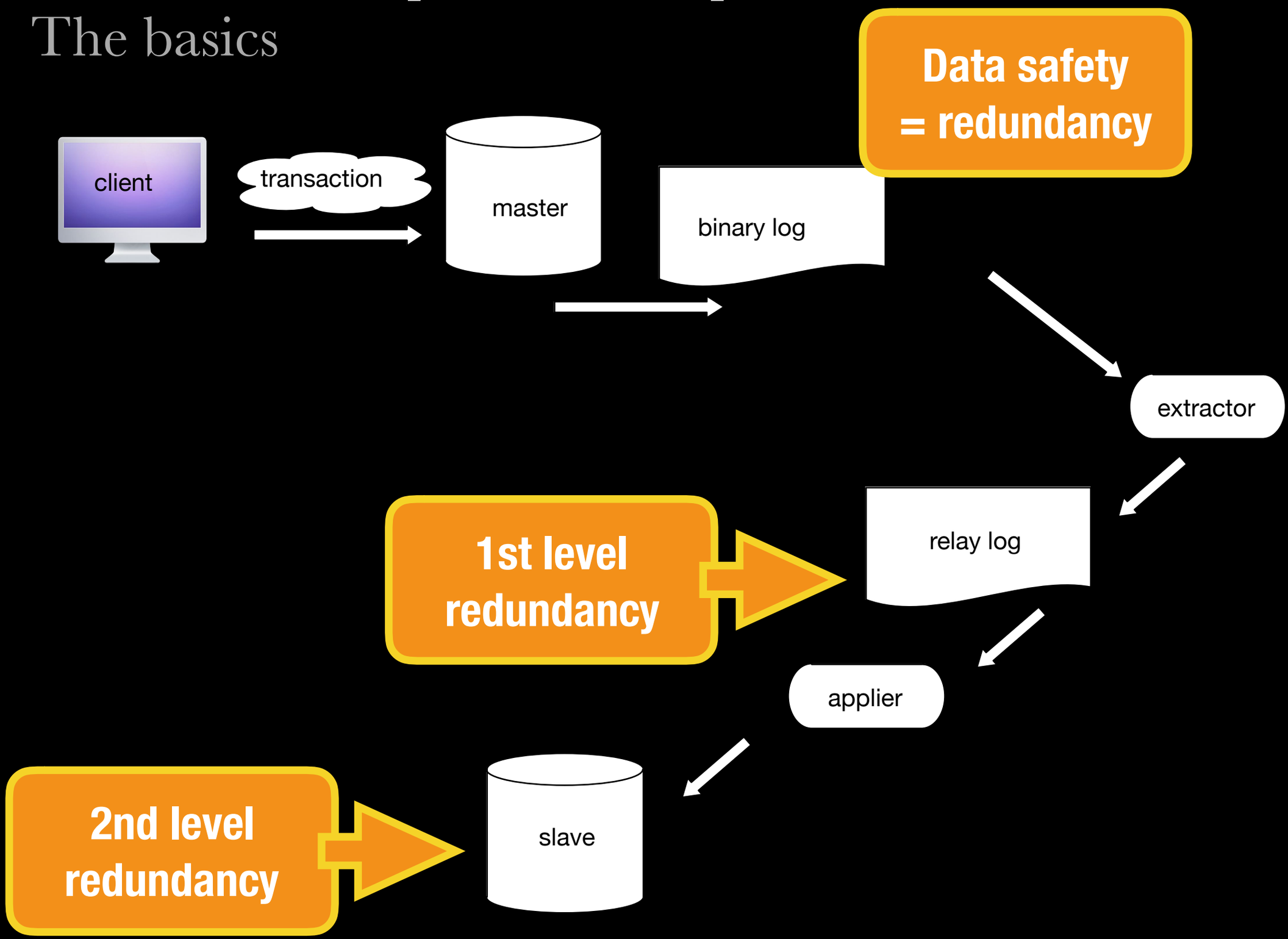
The concepts of replication

The basics



The concepts of replication

The basics



Focus on monitoring

Why we will see lots of monitoring concepts

- Monitoring tells you if replication is working properly
- It also show us how some features work
- If we can see the data moving, we understand it better

Focus on monitoring

The most important reason:

- Replication will fail, sooner or later.
- Good monitoring metadata is what can tell you what the problem is

MySQL replication monitoring

There is more than one way of knowing if replication is working

MySQL replication monitoring

There is more than one way of knowing if replication is working

- Sentinel data : tap tap, is this thing working?

MySQL replication monitoring

There is more than one way of knowing if replication is working

- Sentinel data : tap tap, is this thing working?
- Monitoring: are you still there?

MySQL replication monitoring

There is more than one way of knowing if replication is working

- Sentinel data : tap tap, is this thing working?
- Monitoring: are you still there?
- Latency: are you catching up?

MySQL replication monitoring

There is more than one way of knowing if replication is working

- Sentinel data : tap tap, is this thing working?
- Monitoring: are you still there?
- Latency: are you catching up?
- Status persistence: are you OK, dear?

MySQL replication monitoring

There is more than one way of knowing if replication is working

- Sentinel data : tap tap, is this thing working?
- Monitoring: are you still there?
- Latency: are you catching up?
- Status persistence: are you OK, dear?
- Completeness: did you miss anything?

MySQL replication monitoring

There is more than one way of knowing if replication is working

- Sentinel data : tap tap, is this thing working?
- Monitoring: are you still there?
- Latency: are you catching up?
- Status persistence: are you OK, dear?
- Completeness: did you miss anything?
- Checksum probes: have you got it all?

Sentinel data : tap tap, is this thing on?

The simplest, system-independent method of testing replication

Sentinel data : tap tap, is this thing on?

The simplest, system-independent method of testing replication

1. Make sure the data is not in the master or in the slave

Sentinel data : tap tap, is this thing on?

The simplest, system-independent method of testing replication

1. Make sure the data is not in the master or in the slave
2. Write the data to the master

Sentinel data : tap tap, is this thing on?

The simplest, system-independent method of testing replication

1. Make sure the data is not in the master or in the slave
2. Write the data to the master
3. Retrieve the data in the slave

Sentinel data : tap tap, is this thing on?

The simplest, system-independent method of testing replication

1. Make sure the data is not in the master or in the slave
2. Write the data to the master
3. Retrieve the data in the slave
4. Modify the data in the master

Sentinel data : tap tap, is this thing on?

The simplest, system-independent method of testing replication

1. Make sure the data is not in the master or in the slave
2. Write the data to the master
3. Retrieve the data in the slave
4. Modify the data in the master
5. Check the changes in the slave

Sentinel data : tap tap, is this thing on?

The simplest, system-independent method of testing replication

1. Make sure the data is not in the master or in the slave
2. Write the data to the master
3. Retrieve the data in the slave
4. Modify the data in the master
5. Check the changes in the slave
6. Delete the data from the master

Sentinel data : tap tap, is this thing on?

The simplest, system-independent method of testing replication

1. Make sure the data is not in the master or in the slave
2. Write the data to the master
3. Retrieve the data in the slave
4. Modify the data in the master
5. Check the changes in the slave
6. Delete the data from the master
7. Make sure it was also deleted in the slave

Sentinel data: what is it and isn't

Caveats

- It tells you if replication **CAN** work
- It won’t tell you if replication works **ALWAYS**
- It won’t tell you if **all your data** is replicated

Monitoring: are you still there?

Objectives of monitoring :

Monitoring: are you still there?

Objectives of monitoring :

1. Making sure that the slave is replicating from the intended master.

Monitoring: are you still there?

Objectives of monitoring :

1. Making sure that the slave is replicating from the intended master.
2. Checking that the slave is replicating from the right binary logs.

Monitoring: are you still there?

Objectives of monitoring :

1. Making sure that the slave is replicating from the intended master.
2. Checking that the slave is replicating from the right binary logs.
3. Checking that the data from the master is transferred to the slave.

Monitoring: are you still there?

Objectives of monitoring :

1. Making sure that the slave is replicating from the intended master.
2. Checking that the slave is replicating from the right binary logs.
3. Checking that the data from the master is transferred to the slave.
4. Checking that the slave is applying data without errors.

Monitoring: are you still there?

Objectives of monitoring :

1. Making sure that the slave is replicating from the intended master.
2. Checking that the slave is replicating from the right binary logs.
3. Checking that the data from the master is transferred to the slave.
4. Checking that the slave is applying data without errors.
5. Checking that the slave is keeping up with the master

How monitoring works

To monitor effectively, we need to cover all bases

- We need to know :
  - who the master is
  - what the master is doing
  - what the slave is doing
- And then compare what we have got

Who is the master?

Get information about the master, from the master server

```
mysql> show variables like 'port';
    +---------------+-------+
    | Variable_name | Value |
    +---------------+-------+
| port          | 22786 |
+---------------+-------+
```

Who is the master?

Get information about the master, from the master server

```
mysql> show variables like 'hostname';
    +---------------+-------------+
    | Variable_name | Value       |
    +---------------+-------------+
    | hostname      | localhost |
    +---------------+-------------+
```

What is the master doing?

Get information about the master, from the master server

```
mysql> show master status\G
     *************************** 1. row
***************************
                File: mysql-bin.000003
            Position: 5149170
    Binlog_Do_DB:
Binlog_Ignore_DB:
1 row in set (0.00 sec)
```

What is the slave doing?

This means, usually, running “SHOW SLAVE STATUS”

```
SHOW SLAVE STATUS\G
Master_Host: 127.0.0.1
Master_Port: 22786
Master_Log_File: mysql-bin.000003
Read_Master_Log_Pos: 5149170
Relay_Log_Pos: 2060153
Relay_Master_Log_File: mysql-bin.000003
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
Exec_Master_Log_Pos: 2060007
Relay_Log_Space: 5149528
```

Latency

Are you catching up?

- Delta between
  - Time of commit in the master
  - Time of apply in the slave
- Can be measured with a simple *sentinel* system
  1. Insert a high res timestamp in the master
  2. Retrieve the record from the slave
  3. Measure the interval
  4. Subtract the commit time.

Status persistence

Assume your servers will fail. Prepare for it

- Problem:
  - When server crashes and resumes, replication status on file may not be in sync
- Solution:
  - Crash-safe slave tables
  - Replication status is kept in the database

Completeness: did you miss anything?

Using filters during extraction or apply can have side effects

Completeness: did you miss anything?

Using filters during extraction or apply can have side effects

- Data can be removed by filters
  - in the master bin log (never gets to the slaves)
  - in the slaves (apply rules)

Completeness: did you miss anything?

Using filters during extraction or apply can have side effects

- Data can be removed by filters
  - in the master bin log (never gets to the slaves)
  - in the slaves (apply rules)
- Data can be modified
  - for heterogeneous replication
  - for ETL tasks

Completeness: did you miss anything?

Using filters during extraction or apply can have side effects

- Data can be removed by filters
  - in the master bin log (never gets to the slaves)
  - in the slaves (apply rules)
- Data can be modified
  - for heterogeneous replication
  - for ETL tasks
- **If you have filters, your slave CAN`T become master.**

Checksum probes: have you got it all?

Checking that replicas have the right data is a sound idea

- total probes (expensive: may stop or slow down operations)
- incremental probes (take long time, but have low impact on operations)
- Many methods. A popular one is pt-table-checksum from Percona Toolkit

Global Transaction Identifiers

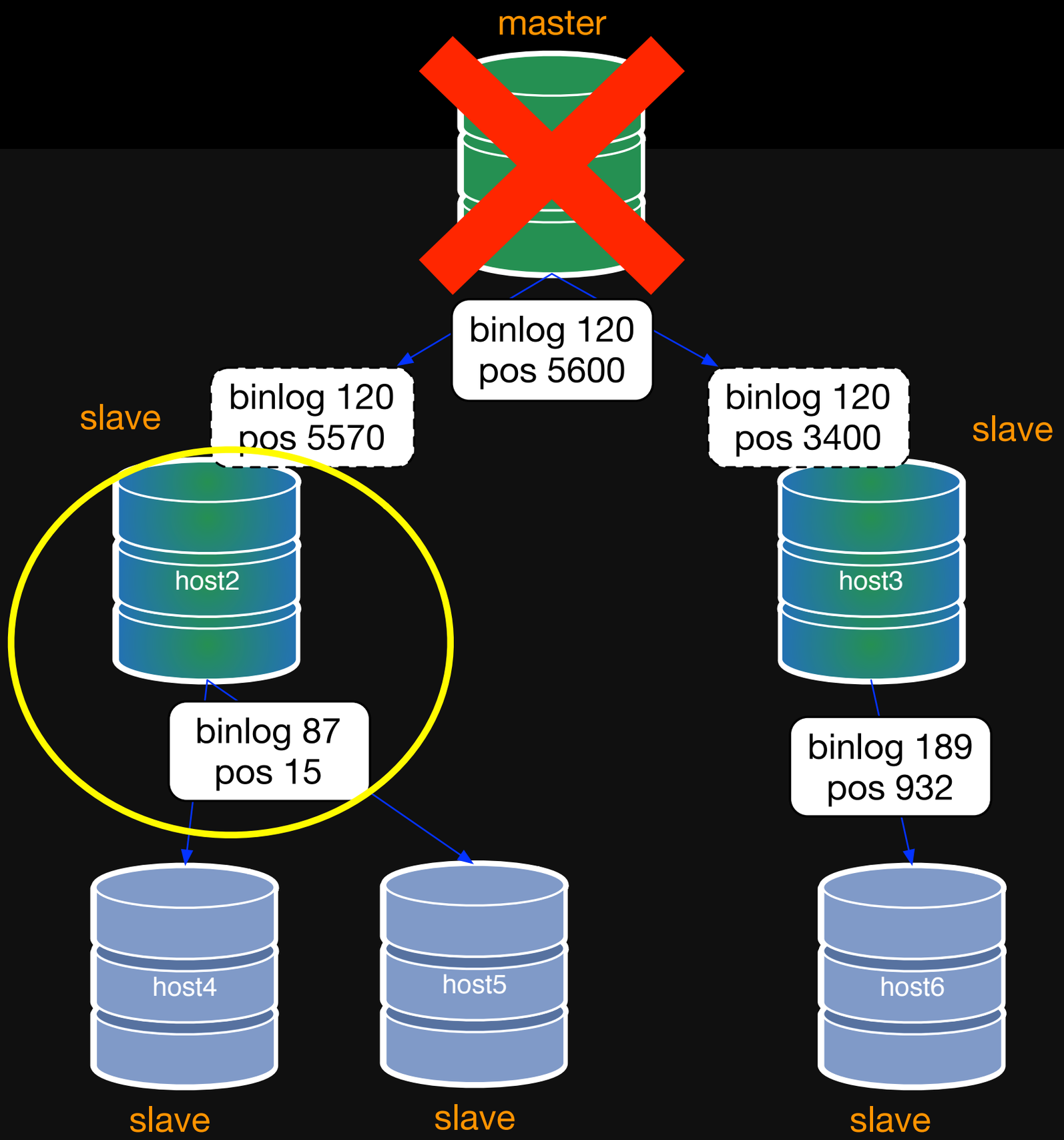
Transactions blues

You think you know where your transactions are … until something unexpected happens

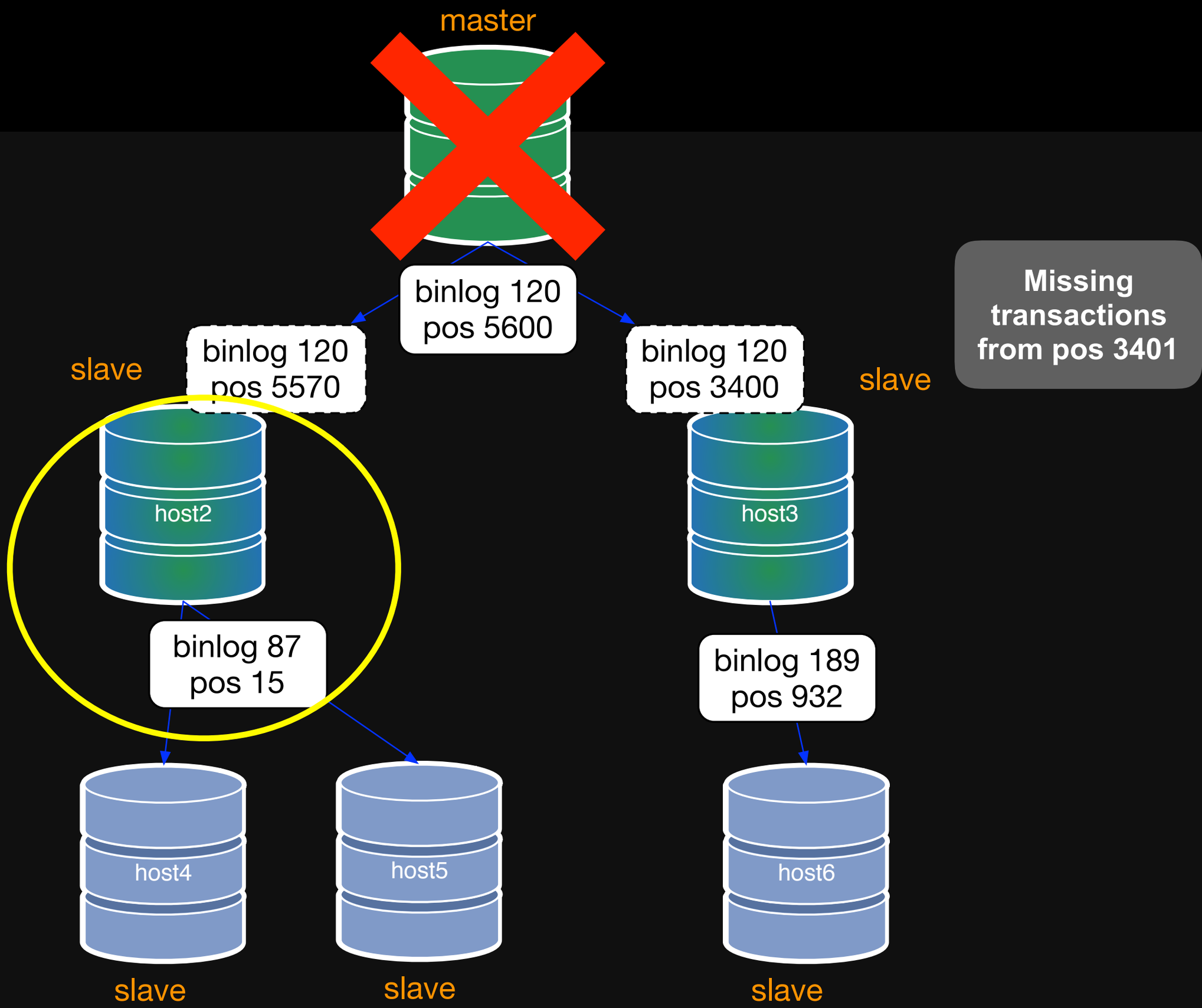
- Problem:
  - MySQL replication identifies transactions with a combination of binary log file name and offset position;
  - When using many possible masters, file names and positions may differ.
  - Practical cases: failover, circular replication, hierarchical replication
- Solution: use a global ID, not related to the file name and position

Transaction problem in a nutshell (1)

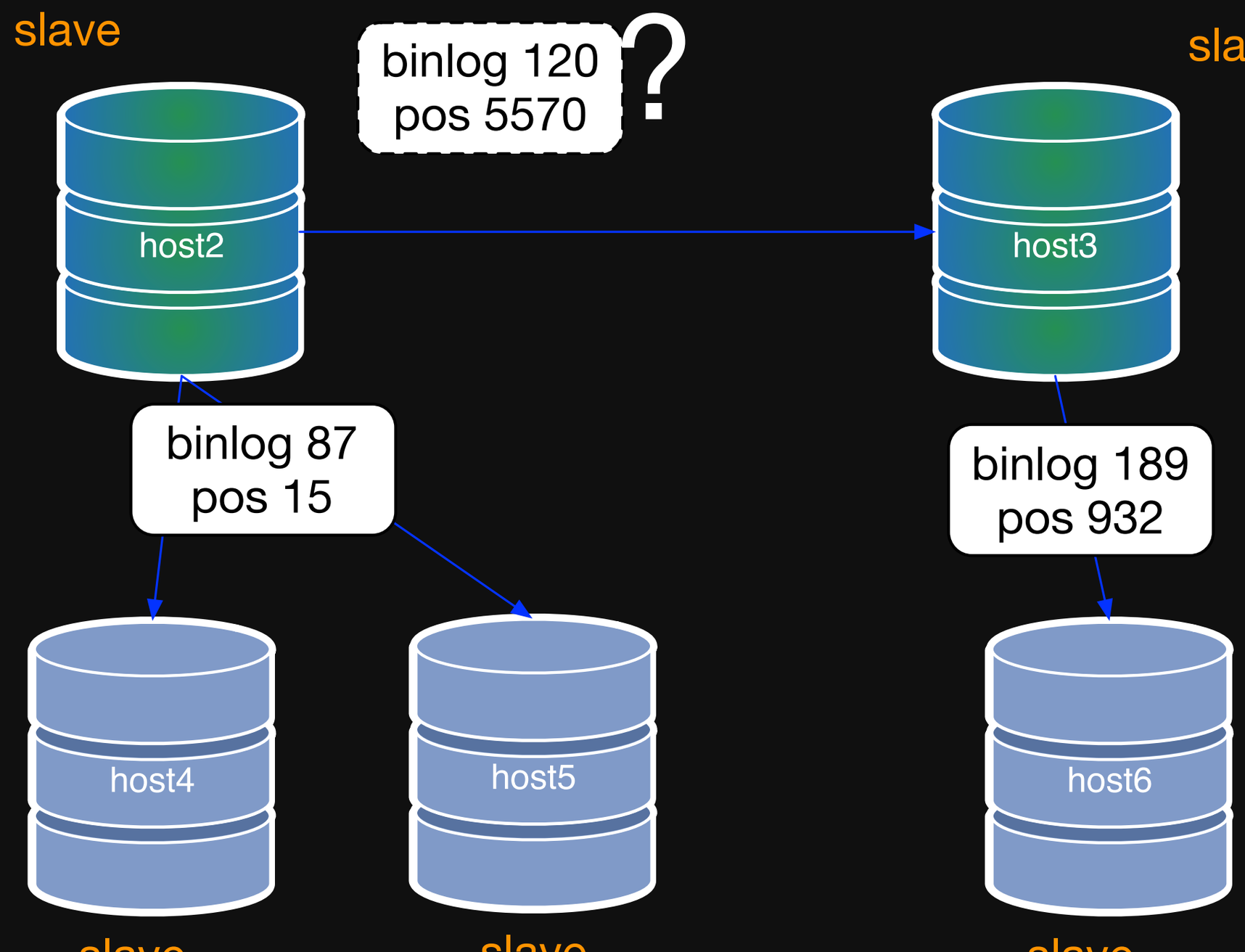
Transaction problem in a nutshell (2)



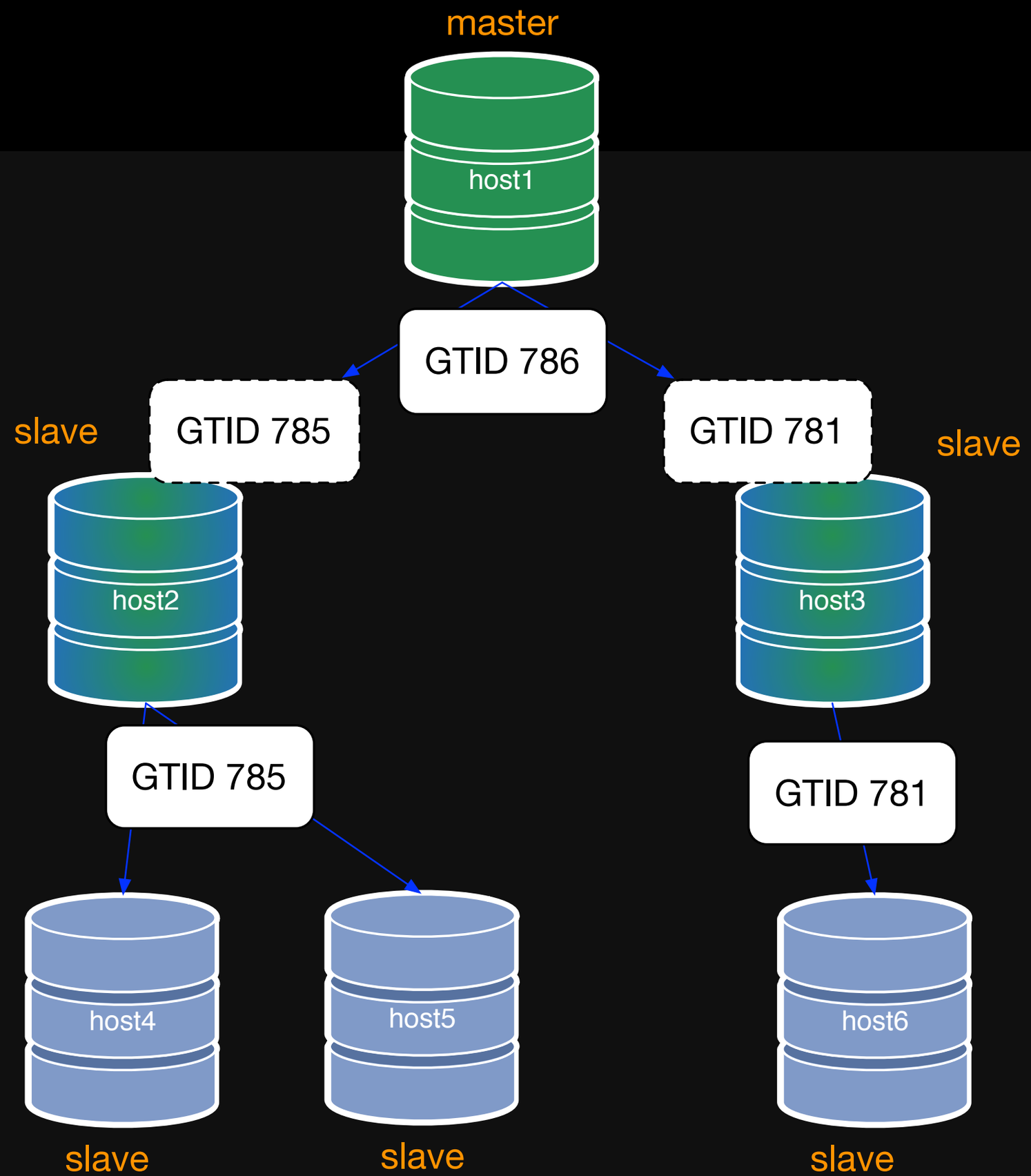
Transaction problem in a nutshell (2)



Transaction problem in a nutshell (3)

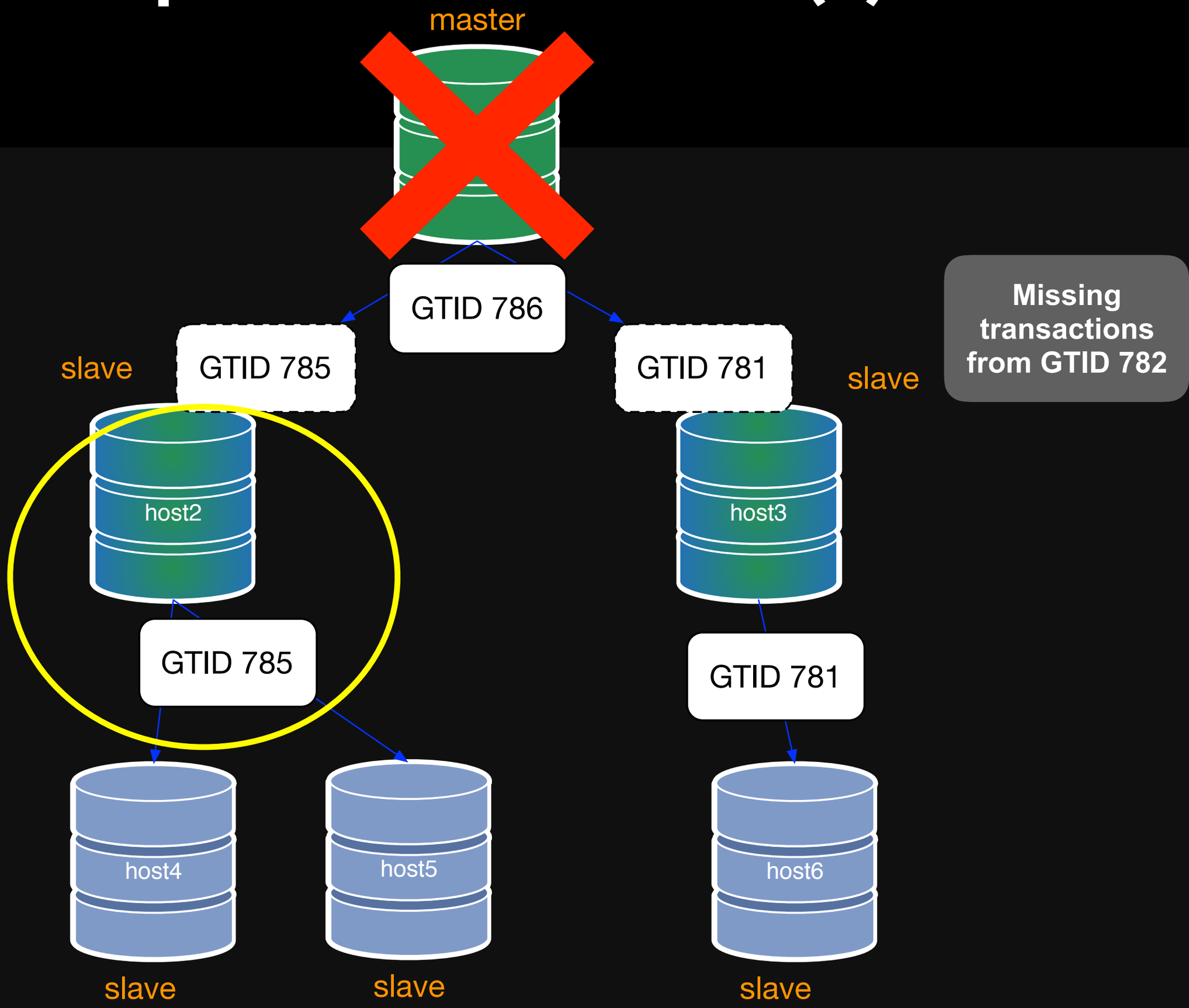


Transaction problem with GTID (1)

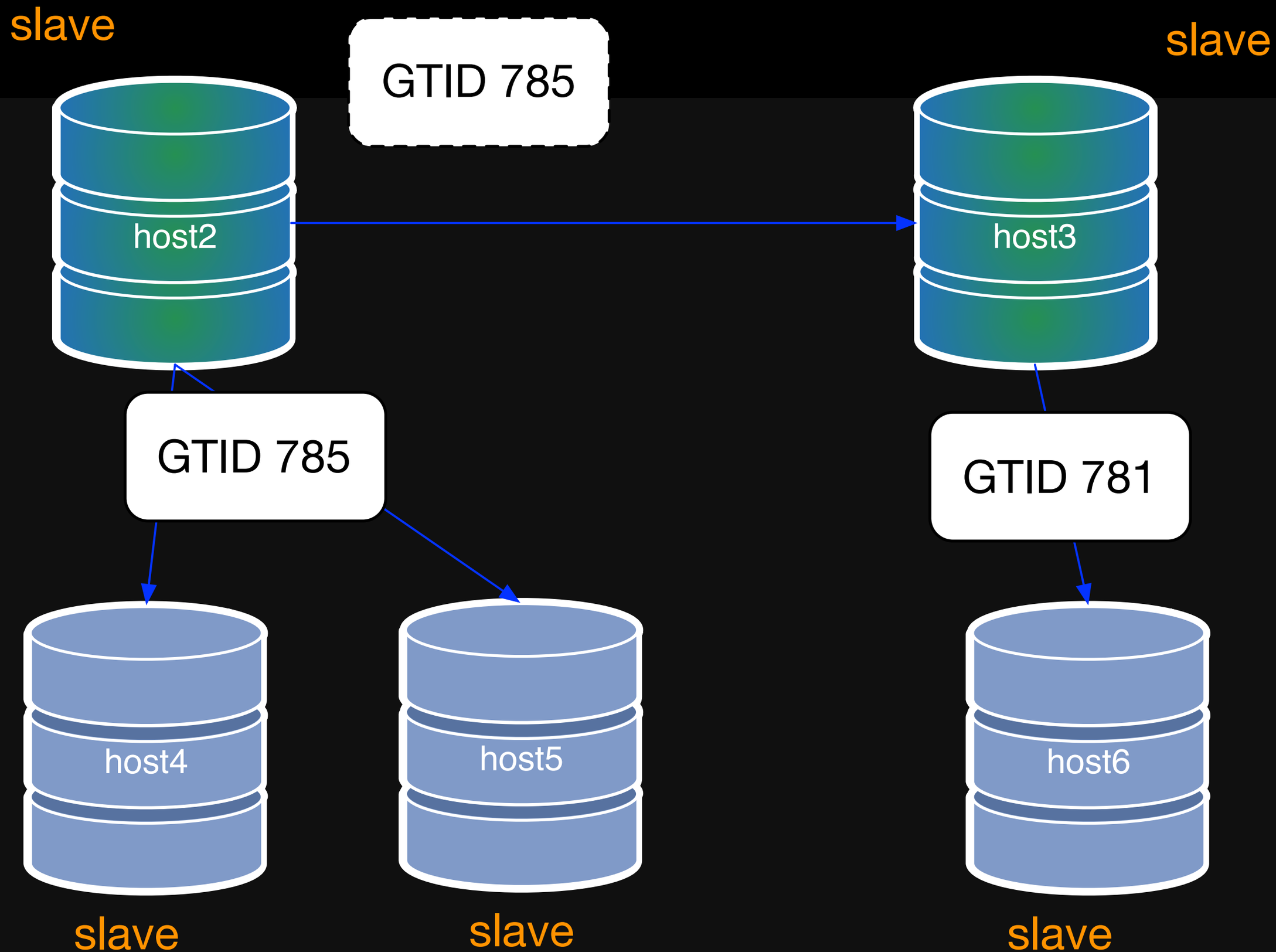


Transaction problem with GTID (2)

Transaction problem with GTID (2)



Transaction problem with GTID (3)



Implementation: (1) MySQL 5.6 & 5.7

A half baked feature, which kind of works

- Made of server UUID + transaction ID
  - (e.g.: “e8679838-b832-11e3-b3fc-017f7cee3849:1”)
- Only transactional engines
- No “create table … select …” supported
- No temporary tables within transactions
- Requires **log-slave-updates** in all nodes (removed in 5.7)

Implementation: (1) MySQL 5.6 & 5.7

A half baked feature, which kind of works

- The good
  - GTID are easily parseable by scripts in the binlog
  - Failover and transaction tracking are easier
- The bad
  - Not enabled by default
  - Hard to read for humans!
  - Little integration between GTID and existing software (ignored in crash-safe tables, parallel replication)
  - makes log-slave updates mandatory (only in 5.6)

GTID in MySQL 5.7.6+

Something was changed ...

- GTID can now be enabled dynamically.
- However, it requires a 9 (NINE!) steps procedure.
- http://mysqlhighavailability.com/enabling-gtids-without-downtime-in-mysql-5-7-6/

MySQL 5.7: What you see in the master

```
show master status\G
            File: mysql-bin.000001
        Position: 1033
    Binlog_Do_DB:
 Binlog_Ignore_DB:
Executed_Gtid_Set: d9f8aeb1-ff3a-11e5-a3d1-0242ac110002:1-4

show global variables like 'gtid_executed'\G
Variable_name: gtid_executed
        Value: d9f8aeb1-ff3a-11e5-a3d1-0242ac110002:1-4
1 row in set (0.00 sec)
```

MySQL 5.7: What you see in the slave

Excerpt from **SHOW SLAVE STATUS**

```
[...]
            Master_Server_Id: 100
                 Master_UUID: d9f8aeb1-ff3a-11e5-a3d1-0242ac110002
            Master_Info_File: mysql.slave_master_info
[ ... ]
  Retrieved_Gtid_Set: d9f8aeb1-ff3a-11e5-a3d1-0242ac110002:1-4
Executed_Gtid_Set: d9f8aeb1-ff3a-11e5-a3d1-0242ac110002:1-4
```

MySQL 5.7: What you see in the slave

Excerpt from **SHOW SLAVE STATUS**

```
[...]
            Master_Server_Id: 100
                 Master_UUID: d9f8aeb1-ff3a-11e5-a3d1-0242ac110002
            Master_Info_File: mysql.slave_master_info
[ ... ]
  Retrieved_Gtid_Set: d9f8aeb1-ff3a-11e5-a3d1-0242ac110002:1-4
Executed_Gtid_Set: d9f8aeb1-ff3a-11e5-a3d1-0242ac110002:1-4
```

Note: we have two pieces of information:

* **retrieved**
* **executed**

MySQL 5.7: What you see in the slave

No GTID info in mysql.slave_relay_log_info

```
select * from slave_relay_log_info\G
*************************** 1. row ***************************
  Number_of_lines: 7
   Relay_log_name: ./mysql-relay.000002
    Relay_log_pos: 1246
  Master_log_name: mysql-bin.000001
   Master_log_pos: 1033
        Sql_delay: 0
Number_of_workers: 0
               Id: 1
     Channel_name:
1 row in set (0.00 sec)
```

More on this topic when we discuss monitoring

Implementation (2) MariaDB 10

A well thought feature, with some questionable choices

- Made of domain ID+server ID + number
  - e.g. (0-101-10)
- Enabled by default
- Uses a crash-safe table
- No limitations
- Lack of integration with old replication coordinates.

MariaDB 10.0: What you see in the master

```
show master status\G
           File: mysql-bin.000001
       Position: 3139
   Binlog_Do_DB:
Binlog_Ignore_DB:
```

```
show variables like '%gtid%pos';
+------------------+--------+
| Variable_name    | Value  |
+------------------+--------+
| gtid_binlog_pos  | 0-1-14 |
| gtid_current_pos | 0-1-14 |
| gtid_slave_pos   |        |
+------------------+--------+
```

MariaDB 10.0: What you see in the slave

Excerpt from **SHOW SLAVE STATUS**

```
[ ... ]
                  Using_Gtid: Current_Pos
                Gtid_IO_Pos: 0-1-14
     Replicate_Do_Domain_Ids:
 Replicate_Ignore_Domain_Ids:
[ ... ]
```

MariaDB 10.0: What you see in the slave

Excerpt from **SHOW SLAVE STATUS**

```
[ ... ]
                  Using_Gtid: Current_Pos
                Gtid_IO_Pos: 0-1-14
     Replicate_Do_Domain_Ids:
 Replicate_Ignore_Domain_Ids:
[ ... ]
```

Note: we have only one piece of information:

* **IO_Pos (= retrieved)**

MariaDB 10.0: What you see in the slave

Excerpt from **SHOW SLAVE STATUS**

```
 select * from mysql.gtid_slave_pos;
+-----------+--------+-----------+--------+
| domain_id | sub_id | server_id | seq_no |
+-----------+--------+-----------+--------+
|         0 |     13 |         1 |     13 |
|         0 |     14 |         1 |     14 |
+-----------+--------+-----------+--------+
```

MariaDB 10.0: What you see in the slave

Excerpt from **SHOW SLAVE STATUS**

```
 select * from mysql.gtid_slave_pos;
+-----------+--------+-----------+--------+
| domain_id | sub_id | server_id | seq_no |
+-----------+--------+-----------+--------+
|         0 |     13 |         1 |     13 |
|         0 |     14 |         1 |     14 |
+-----------+--------+-----------+--------+
```

Note: we have only one piece of information related to the execution of the transaction identified by the GTID

Claim: global transaction identifiers

- Claimed by
  - MySQL 5.6 and 5.7
  - MariaDB 10.0 and 10.1

Sceptic assessment: global transaction identifiers

- MySQL 5.6 and 5.7
  - Not active by default
  - Unfriendly for humans
  - Lack of integration with other features
- MariaDB 10.0 and 10.1
  - Friendlier then MySQL 5.6/5.7
  - Insufficient info for monitoring

Sceptic assessment: global transaction identifiers

CAN DO MUCH BETTER!

- MySQL 5.6 and 5.7
  - Not active by default
  - Unfriendly for humans
  - Lack of integration with other features
- MariaDB 10.0 and 10.1
  - Friendlier then MySQL 5.6/5.7
  - Insufficient info for monitoring

Monitoring (MySQL 5.6+ - MariaDB 10)

The new trend : using tables to monitor

All replication data should be now in tables

- Both MySQL and MariaDB 10 can monitor replication using tables.
- But not all data is available

MySQL 5.6 crash-safe tables

There are tables that can replace files, and SHOW statements ... up to a point

- up to 5.5:
  - **SQL** in the slave
    - show slave status
  - **SQL** in the master
    - show master status
- 5.6 & 5.7:
  - **Tables** in the slave
    - slave_master_info
    - slave_relay_log_info
    - slave_worker_info
    - performance_schema (5.7)
  - **SQL** in the master
    - show master status
    - select @@global.gtid_executed

MySQL tables

Very detailed, but designed in different stages

- One table replaces the file master.info
- Another replaces relay-log.info
- They were designed before introducing GTID
- There is NO GTID in these tables
- They are NOT updated continuously

MySQL 5.7 additional tables in the slave

Performance Schema helps with monitoring

- replication_applier_configuration
- replication_applier_status
- replication_applier_status_by_coordinator
- replication_applier_status_by_worker
- replication_connection_configuration
- replication_connection_status

MySQL 5.7 additional tables in the slave

Performance Schema helps with monitoring

- replication_applier_configuration
- replication_applier_status
- replication_applier_status_by_coordinator
- replication_applier_status_by_worker
- replication_connection_configuration
- replication_connection_status

Despite all these tables,
not all info from SHOW SLAVE STATUS is available

MariaDB 10 crash-safe tables

A complete redesign of the monitoring system, integrated with GTID

- up to 5.5:
  - **SQL** in the slave
    - show slave status
  - **SQL** in the master
    - show master status
- 10.0
  - **Table** in the slave
    - gtid_slave_pos
  - **SQL** in the master
    - show master status
    - select @@gtid_current_pos

MySQL 5.7: tables in the slave

in the mysql database

```
select * from slave_relay_log_info\G
*************************** 1. row ***************************
  Number_of_lines: 7
   Relay_log_name: ./mysql-relay.000002
    Relay_log_pos: 1246
  Master_log_name: mysql-bin.000001
   Master_log_pos: 1033
        Sql_delay: 0
Number_of_workers: 0
               Id: 1
     Channel_name:
1 row in set (0.00 sec)
```

MySQL 5.7: tables in the slave

in the mysql database

```
select * from mysql.slave_master_info\G
*************************** 1. row ***************************
       Number_of_lines: 25
       Master_log_name: mysql-bin.000001
        Master_log_pos: 154
                  Host: 172.17.0.2
             User_name: rdocker
         User_password: rdocker
                  Port: 3306
         Connect_retry: 60
           Enabled_ssl: 0
[...]
             Heartbeat: 30
    Ignored_server_ids: 0
                  Uuid: f4c64510-ff4c-11e5-80f9-0242ac110002
           Retry_count: 86400
```

MySQL 5.7: tables in the slave

in the performance_schema database

```
select * from replication_applier_configuration\G
*************************** 1. row ***************************
 CHANNEL_NAME:
DESIRED_DELAY: 0
1 row in set (0.00 sec)
```

```
select * from replication_applier_status\G
*************************** 1. row ***************************
             CHANNEL_NAME:
            SERVICE_STATE: ON
        REMAINING_DELAY: NULL
COUNT_TRANSACTIONS_RETRIES: 0
```

MySQL 5.7: tables in the slave

in the performance_schema database

```
select * from replication_applier_status_by_coordinator\G
Empty set (0.00 sec)

select * from replication_connection_configuration\G
             CHANNEL_NAME:
                     HOST: 172.17.0.2
                     PORT: 3306
                     USER: rdocker
          NETWORK_INTERFACE:
              AUTO_POSITION: 1
                SSL_ALLOWED: NO
[ ... ]
```

MySQL 5.7: tables in the slave

in the performance_schema database

```
 select * from replication_connection_status\G
*************************** 1. row ***************************
             CHANNEL_NAME:
               GROUP_NAME:
              SOURCE_UUID: f4c64510-ff4c-11e5-80f9-0242ac110002
                THREAD_ID: 33
            SERVICE_STATE: ON
COUNT_RECEIVED_HEARTBEATS: 12
 LAST_HEARTBEAT_TIMESTAMP: 2016-04-10 18:55:56
 RECEIVED_TRANSACTION_SET: f4c64510-ff4c-11e5-80f9-0242ac110002:1-4
        LAST_ERROR_NUMBER: 0
       LAST_ERROR_MESSAGE:
     LAST_ERROR_TIMESTAMP: 0000-00-00 00:00:00
```

MySQL 5.7: tables in the slave

in the performance_schema database

```
 select * from replication_connection_status\G
*************************** 1. row ***************************
             CHANNEL_NAME:
               GROUP_NAME:
              SOURCE_UUID: f4c64510-ff4c-11e5-80f9-0242ac110002
                THREAD_ID: 33
            SERVICE_STATE: ON
COUNT_RECEIVED_HEARTBEATS: 12
 LAST_HEARTBEAT_TIMESTAMP: 2016-04-10 18:55:56
 RECEIVED_TRANSACTION_SET: f4c64510-ff4c-11e5-80f9-0242ac110002:1-4
        LAST_ERROR_NUMBER: 0
       LAST_ERROR_MESSAGE:
     LAST_ERROR_TIMESTAMP: 0000-00-00 00:00:00
```

Note: we have only <u>one</u> piece of information related to the received transaction

Claim: Monitoring in crash-safe tables

- Claimed by
  - MySQL 5.6 and 5.7
  - MariaDB 10.0 and 10.1

Sceptic assessment: monitoring in crash-safe tables

- Both:
  - (+) Yes. The slave is crash safe
  - (-) No replication info tables in the master
  - (-) Split info about received and executed data
- MySQL 5.6 and 5.7
  - (-) Lack of integration with other features
  - (-) Only SHOW SLAVE STATUS has the full picture
- MariaDB 10.0 and 10.1
  - (-) Insufficient info for monitoring
  - (-) Insufficient data in SHOW SLAVE STATUS

Sceptic assessment: monitoring in crash-safe tables

- Both:
  - (+) Yes. The slave is crash safe
  - (-) No replication info tables in the master
  - (-) Split info about received and executed data
- MySQL 5.6 and 5.7
  - (-) Lack of integration with other features
  - (-) Only SHOW SLAVE STATUS has the full picture
- MariaDB 10.0 and 10.1
  - (-) Insufficient info for monitoring
  - (-) Insufficient data in SHOW SLAVE STATUS

CAN DO MUCH, MUCH BETTER!

Monitoring and GTID demo

Multi-source replication

What is it?

The dream of every DBA is to have a group of database servers that behave like a single server

- Traditional replication allows master/slave and chain replication (a.k.a. circular or ring)
- Up to MySQL 5.6, a slave cannot have more than one master.
- Multi source is the ability of replicating from more than one master at once.
- Implemented in Tungsten Replicator (2009), MySQL 5.7 (2015), MariaDB 10 (2013).

Implementation (1) MySQL 5.7

Introduced in MySQL 5.7.7

- New syntax: CHANGE MASTER TO … FOR CHANNEL “name”
- SHOW SLAVE STATUS FOR CHANNEL “name”
- START/STOP SLAVE FOR CHANNEL “name”
- Includes replication tables in performance_schema
- Requires GTID and crash-safe tables to be enabled

MySQL 5.7 example

Setting several channels

```
CHANGE MASTER TO
    MASTER_HOST='foo.example.com', MASTER_PORT=3306,
    MASTER_USER='repl_user',
    MASTER_PASSWORD='repl_pass',
    MASTER_AUTO_POSITION=1
    for channel 'sl_foo';
START SLAVE for channel 'sl_foo';

CHANGE MASTER TO
    MASTER_HOST='bar.example.com', MASTER_PORT=3306,
    MASTER_USER='repl_user',
    MASTER_PASSWORD='repl_pass',
    MASTER_AUTO_POSITION=1
    for channel 'sl_bar'
START SLAVE for channel 'sl_bar';
```

implementation (2) : MariaDB 10

Now GA, the multi source was well planned and executed

- New syntax “CHANGE MASTER “name” …”
- START/STOP/RESET SLAVE “name”
- SHOW SLAVE “name” STATUS
- SHOW ALL SLAVE**S** STATUS

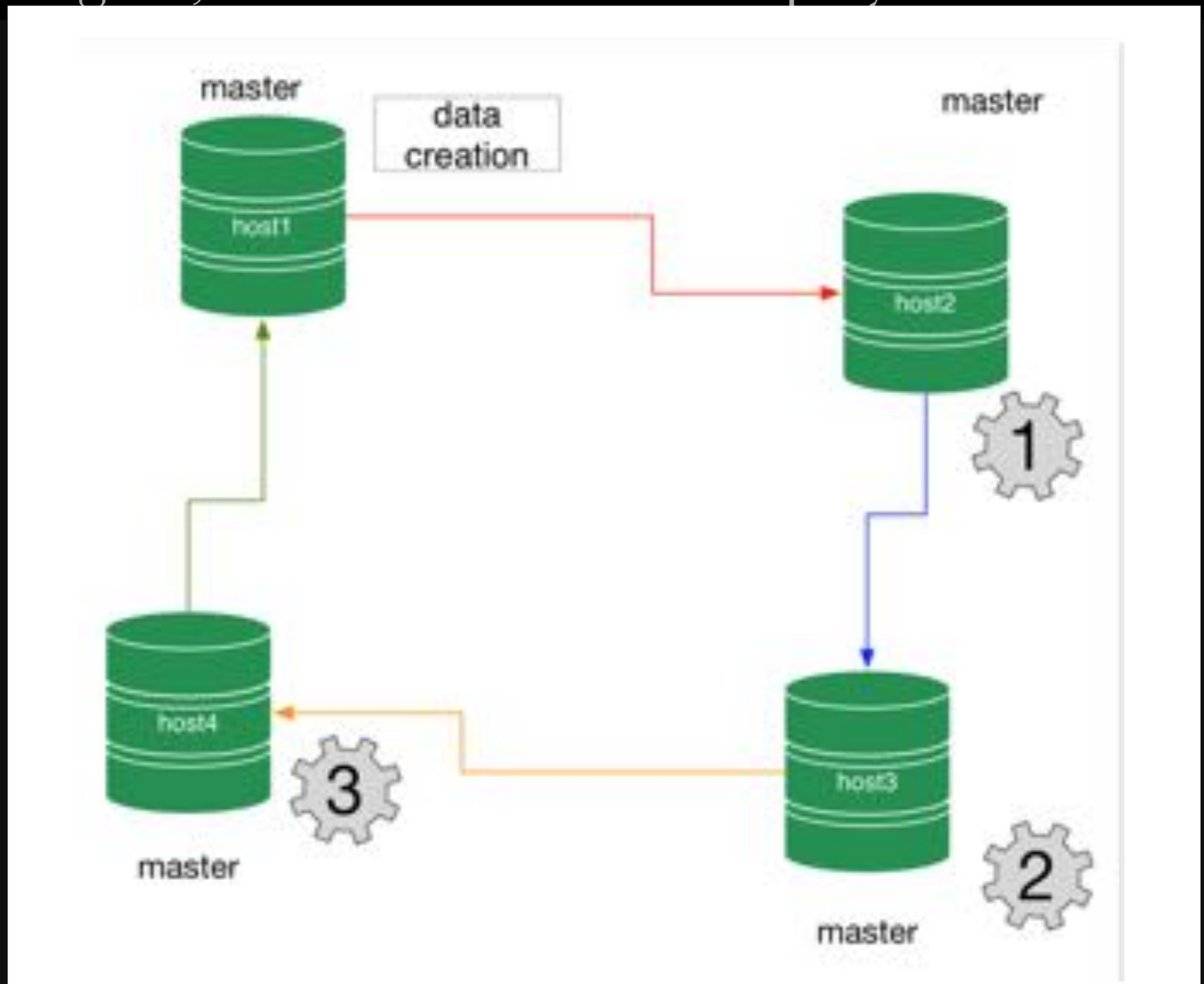
MariaDB 10.1 example

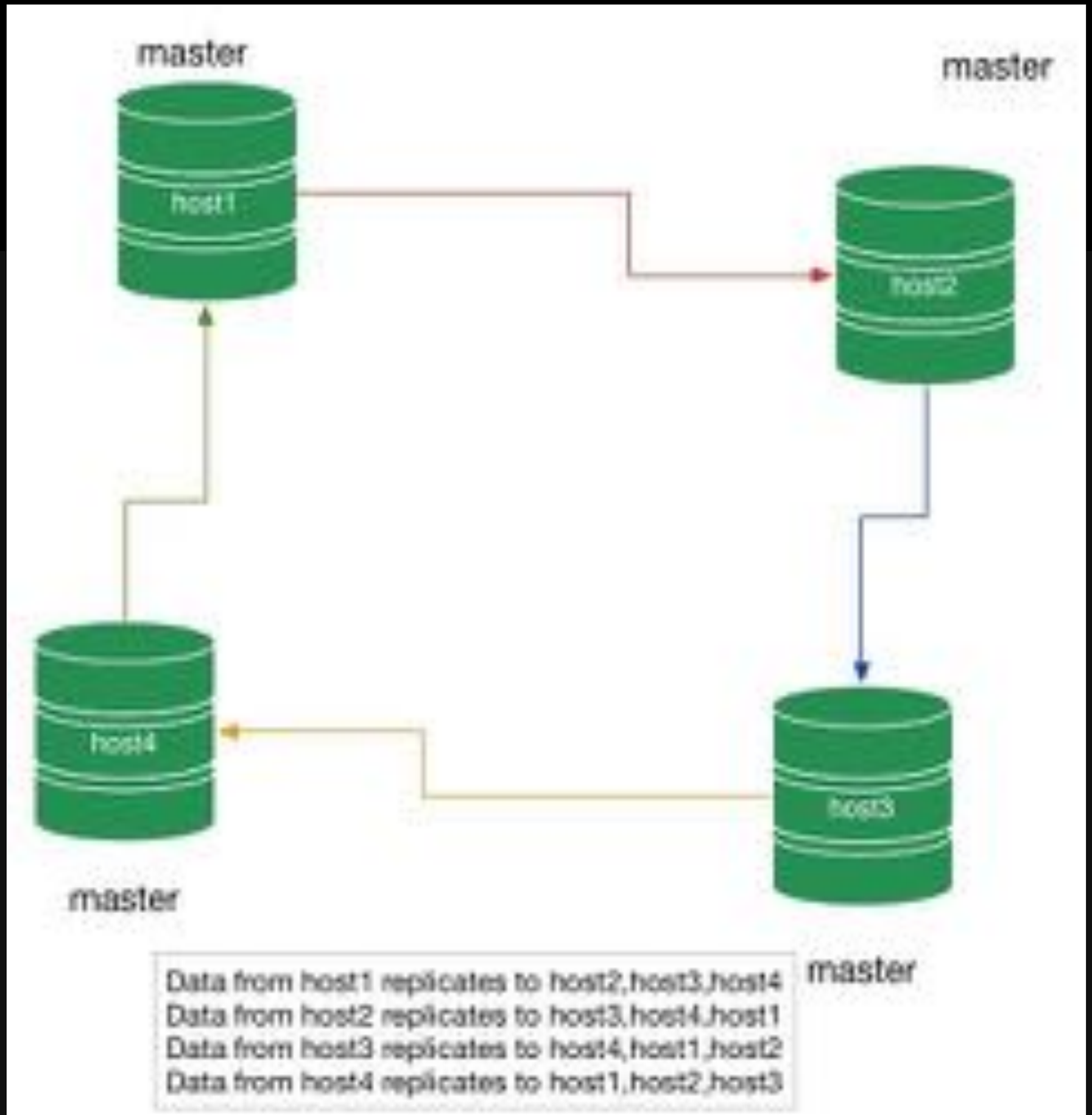
Setting several channels

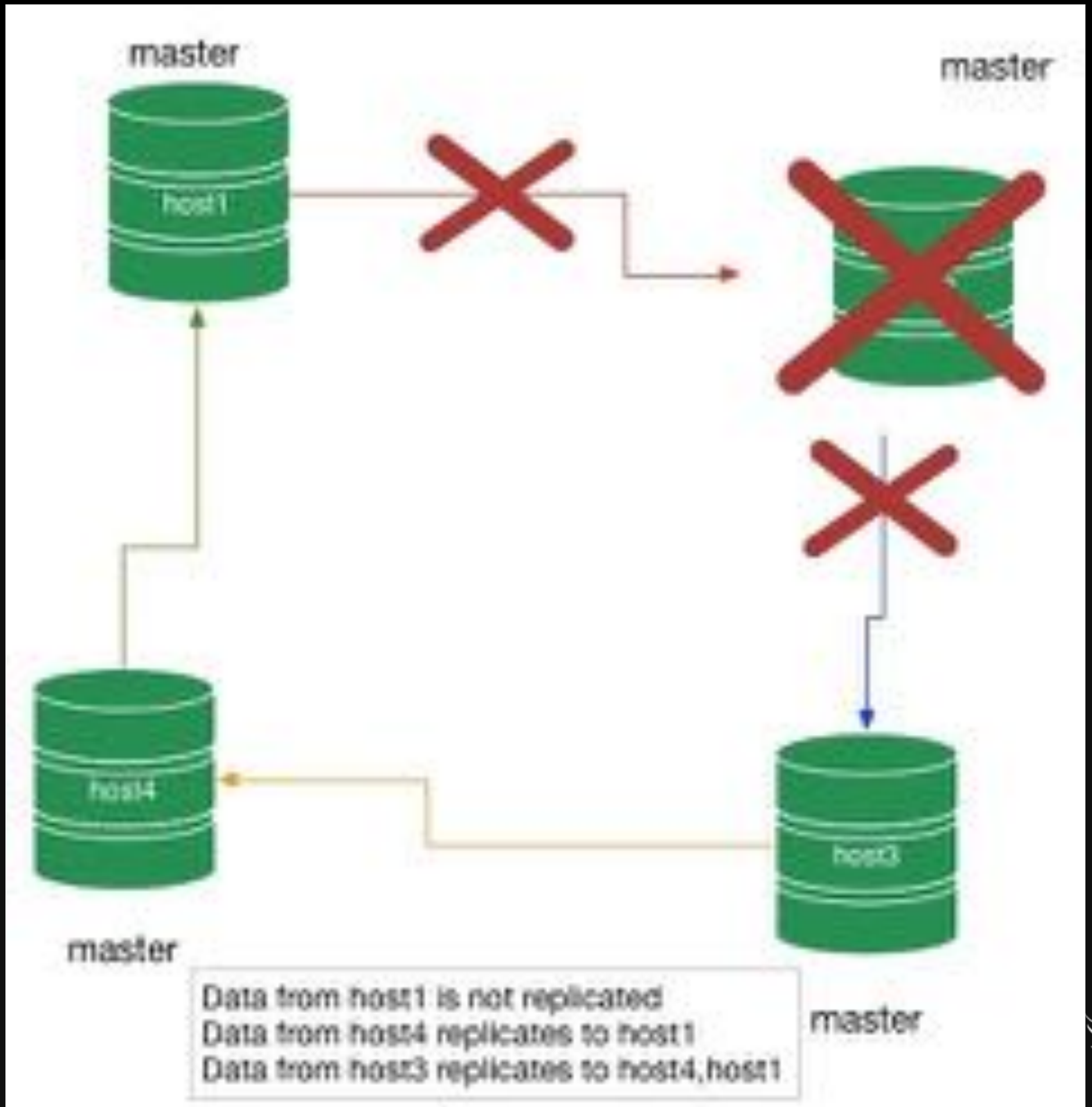
```
CHANGE MASTER 'sl_foo' TO
    MASTER_HOST='foo.example.com', MASTER_PORT=3306,
    MASTER_USER='repl_user',
    MASTER_PASSWORD='repl_pass',
    MASTER_USE_GTID=current_pos;
START SLAVE 'sl_foo';

CHANGE MASTER 'sl_bar' TO
    MASTER_HOST='bar.example.com', MASTER_PORT=3306,
    MASTER_USER='repl_user',
    MASTER_PASSWORD='repl_pass',
    MASTER_USE_GTID=current_pos;
START SLAVE 'sl_bar';
```

Full slave replay (circular)

When the data is applied, saved to a binary log, and then replicated again, we have a full slave replay

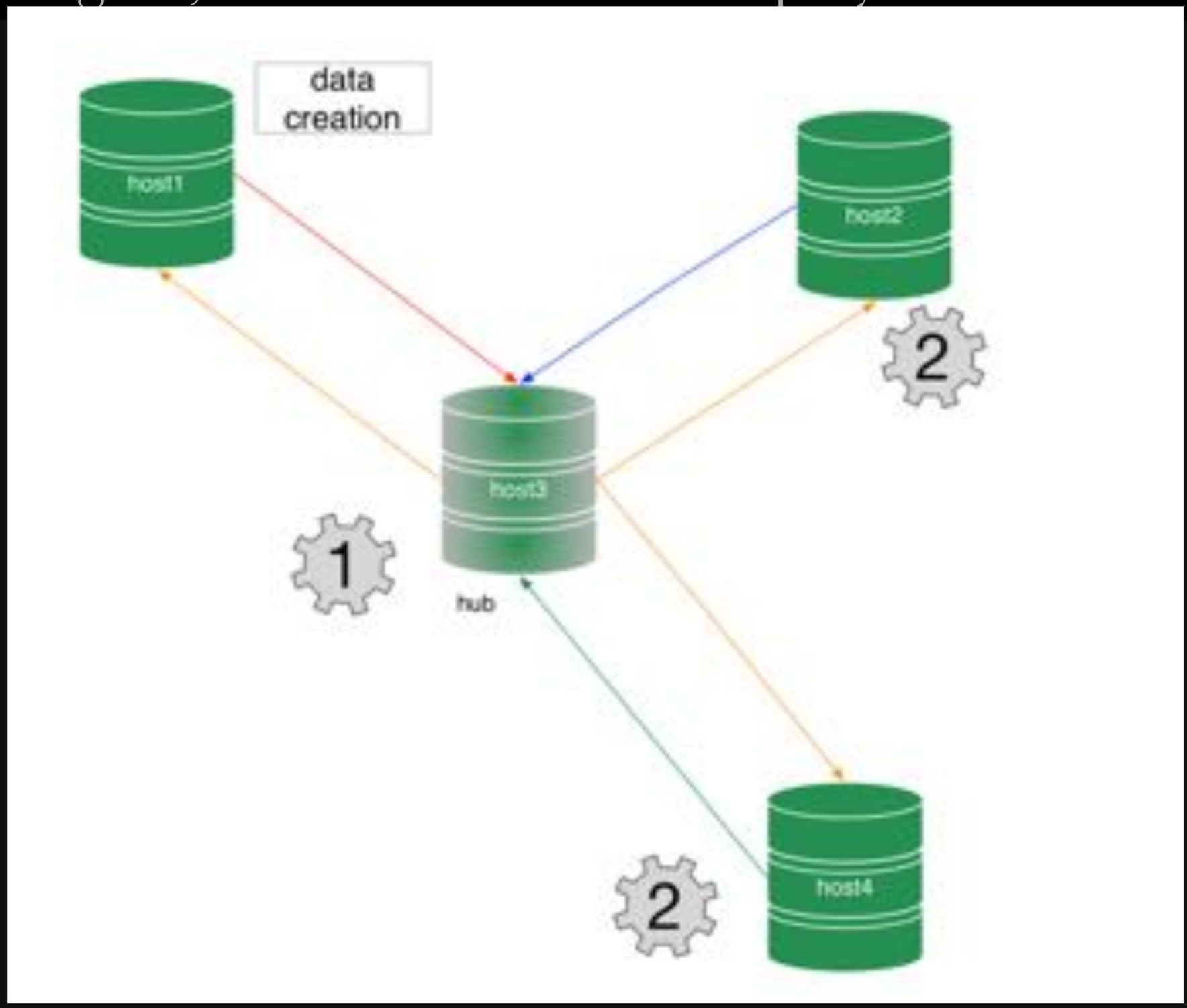



master
master
host1
host2
host4
host3
master
master
Data from host1 replicates to host2,host3,host4
Data from host2 replicates to host3,host4,host1
Data from host3 replicates to host4,host1,host2
Data from host4 replicates to host1,host2,host3


master
master
host1
host4
host3
master
master
Data from host1 is not replicated!
Data from host4 replicates to host1
Data from host3 replicates to host4,host1

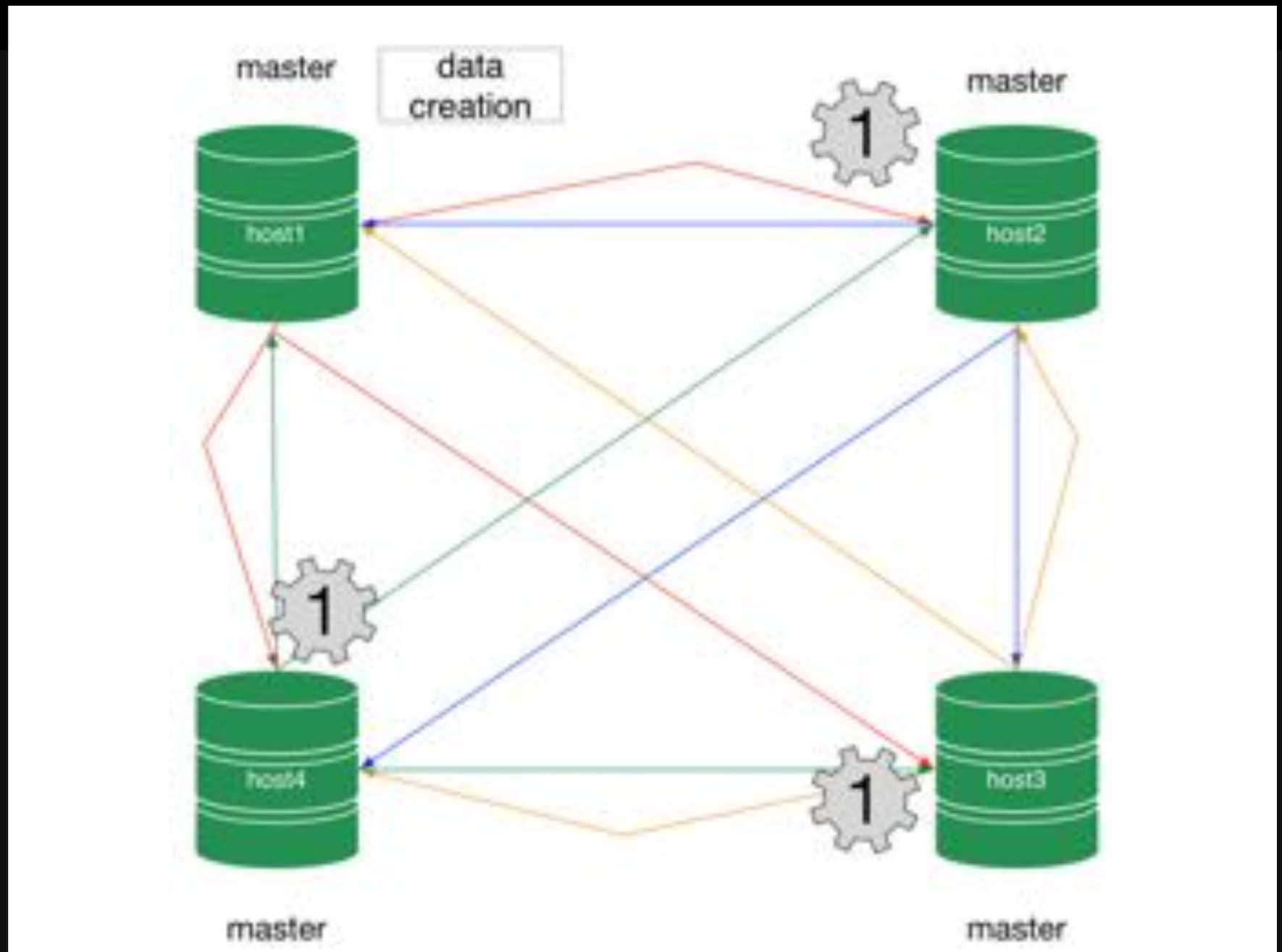
Full slave replay (star)

When the data is applied, saved to a binary log, and then replicated again, we have a full slave replay



Point-to-point replication

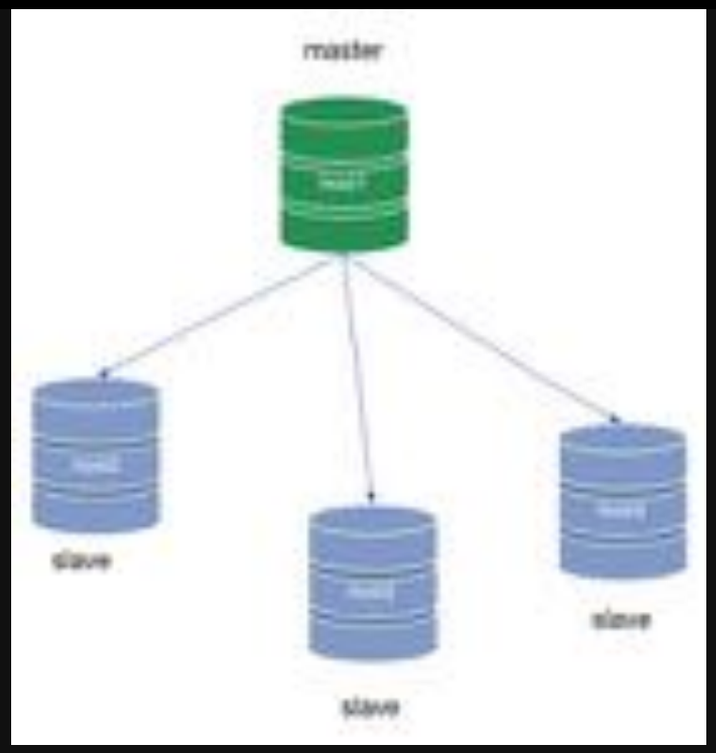
Allows data flow where the replicated data is applied only once

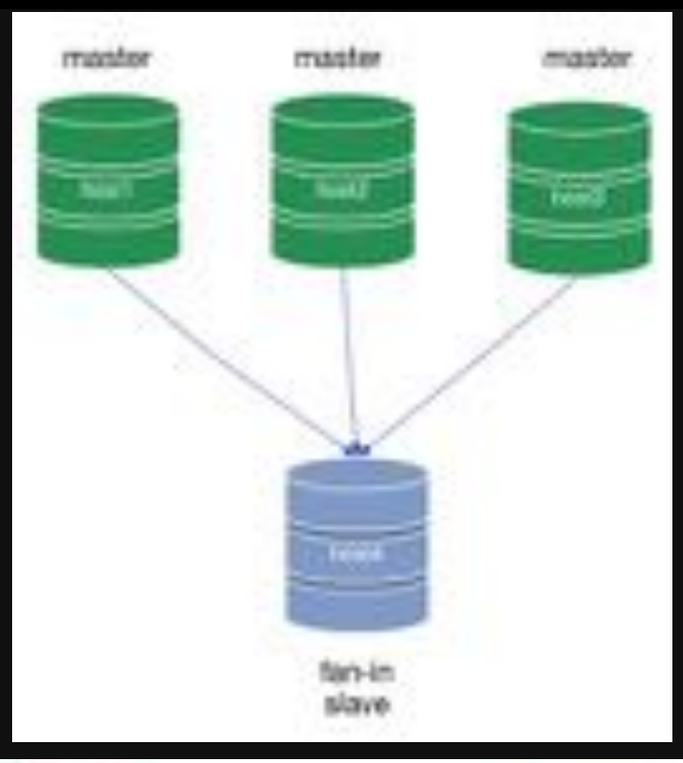


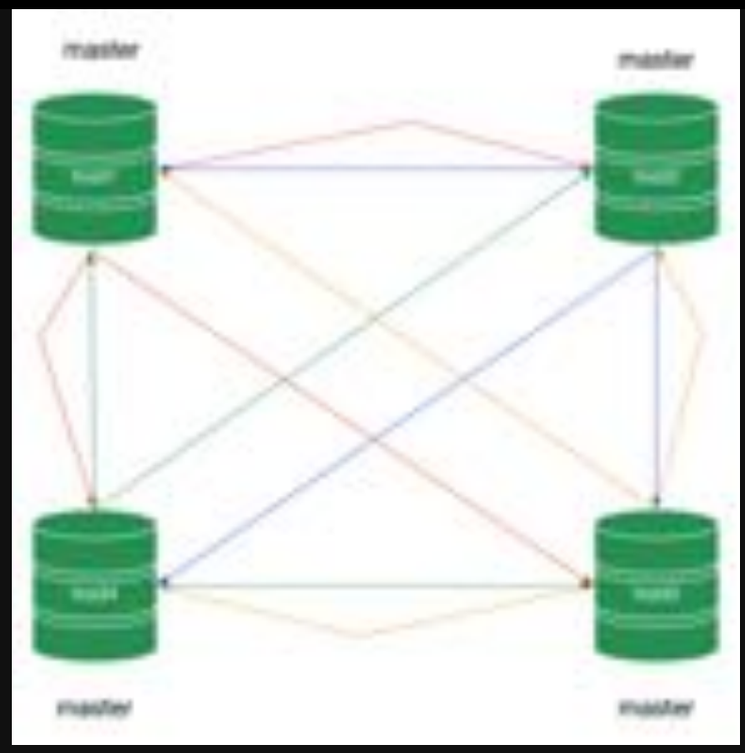
point-to-point all-masters replication

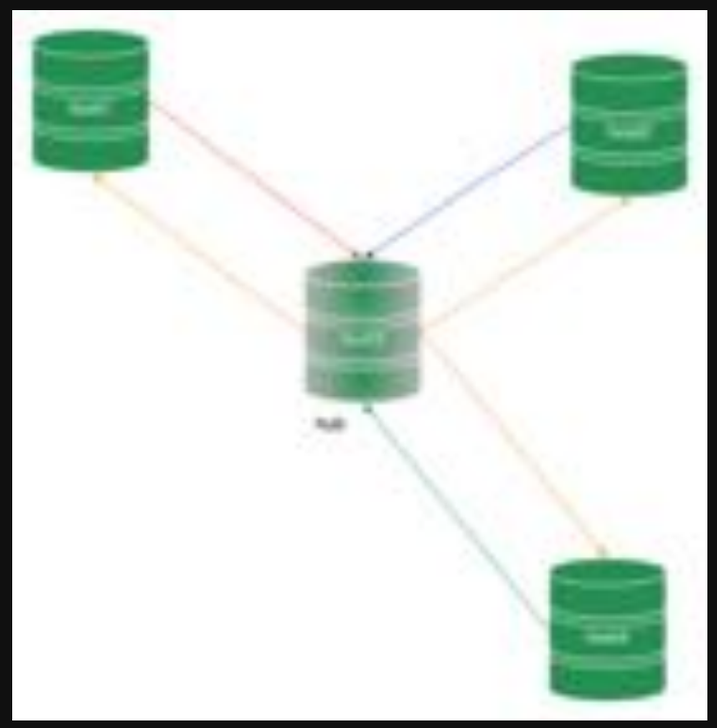
Creating replication topologies

When you can do multi-source replication, your perception of the world changes



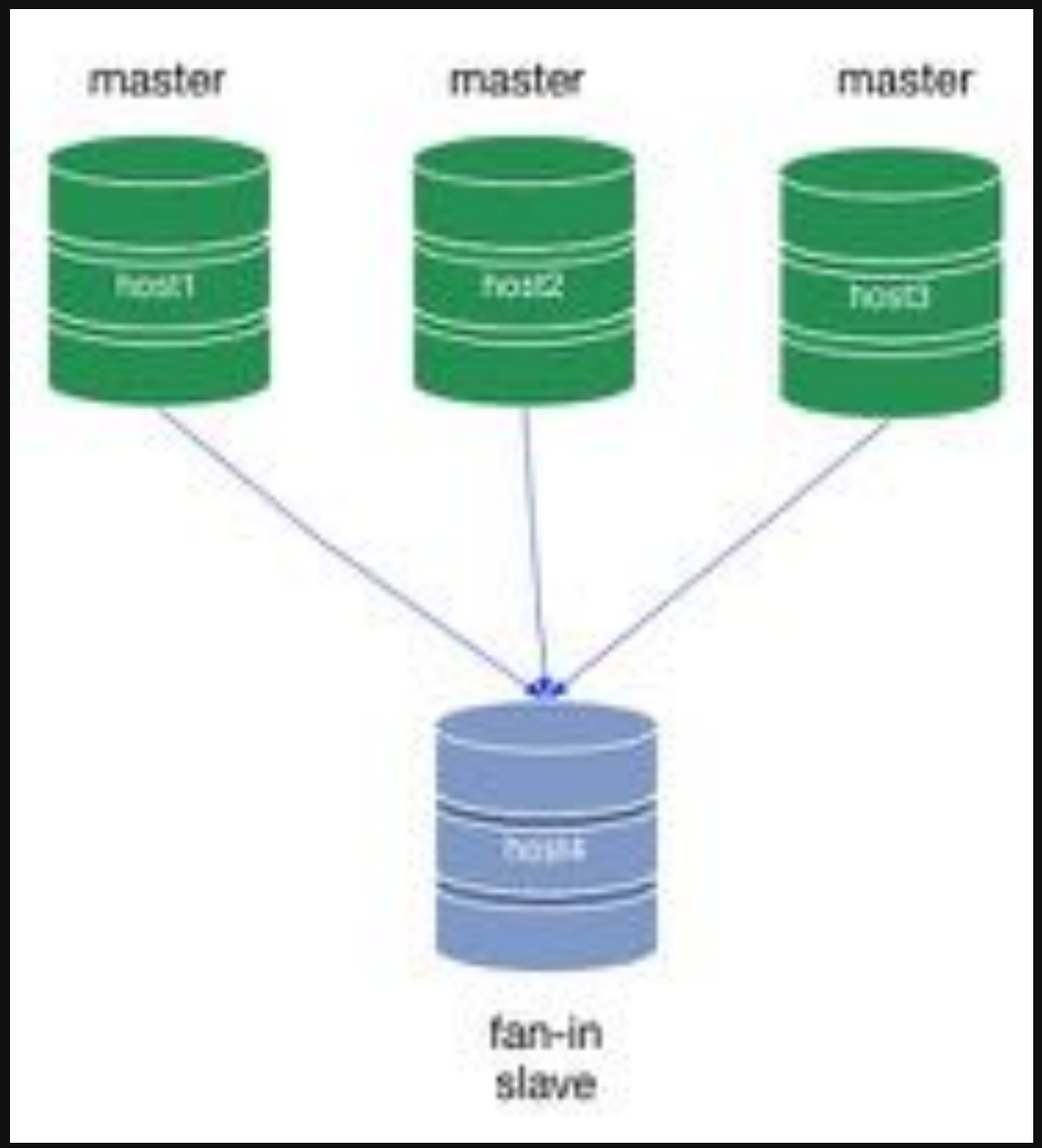






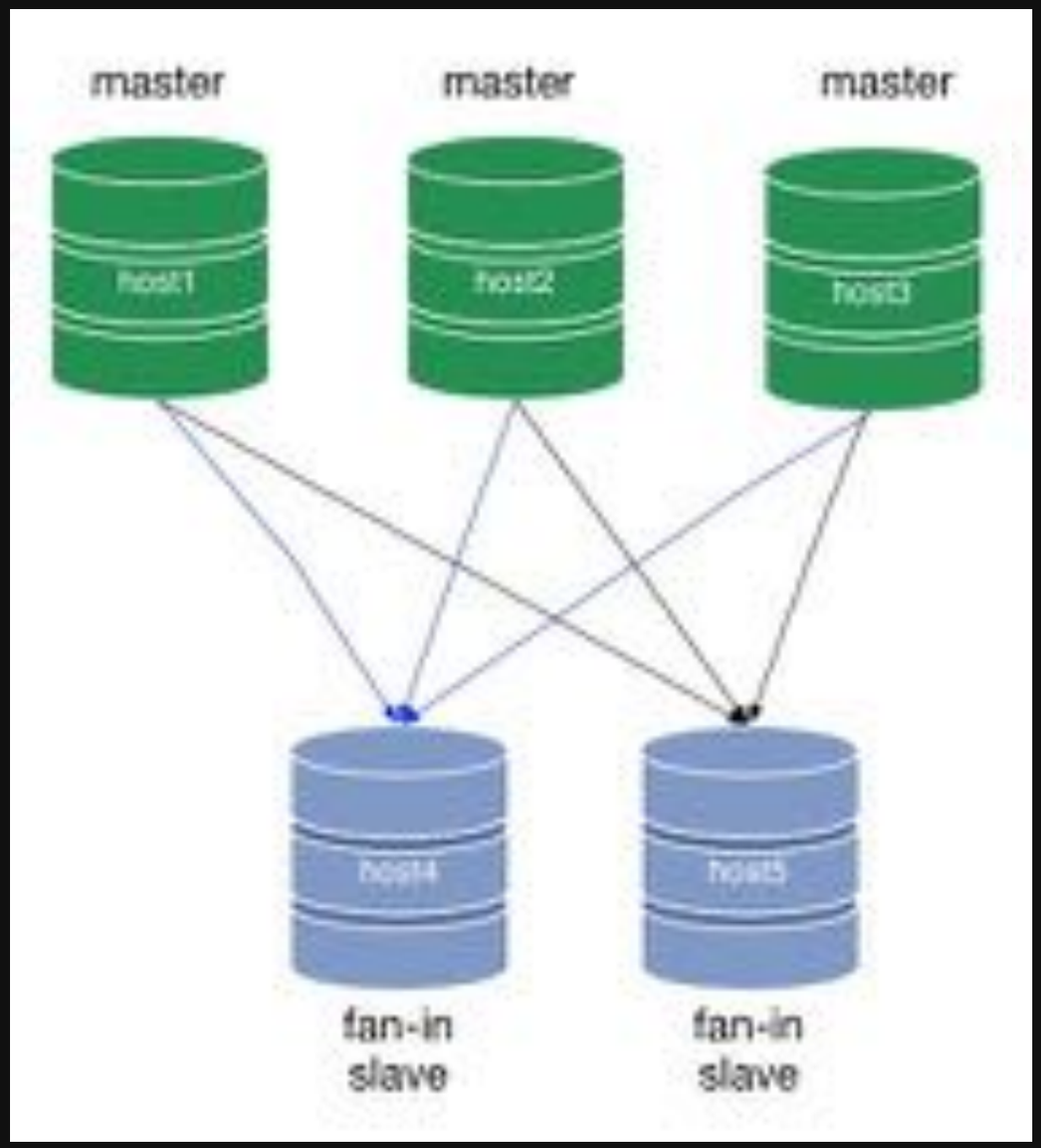
Fan-in topology

The simplest multi-source deployment can become complicated



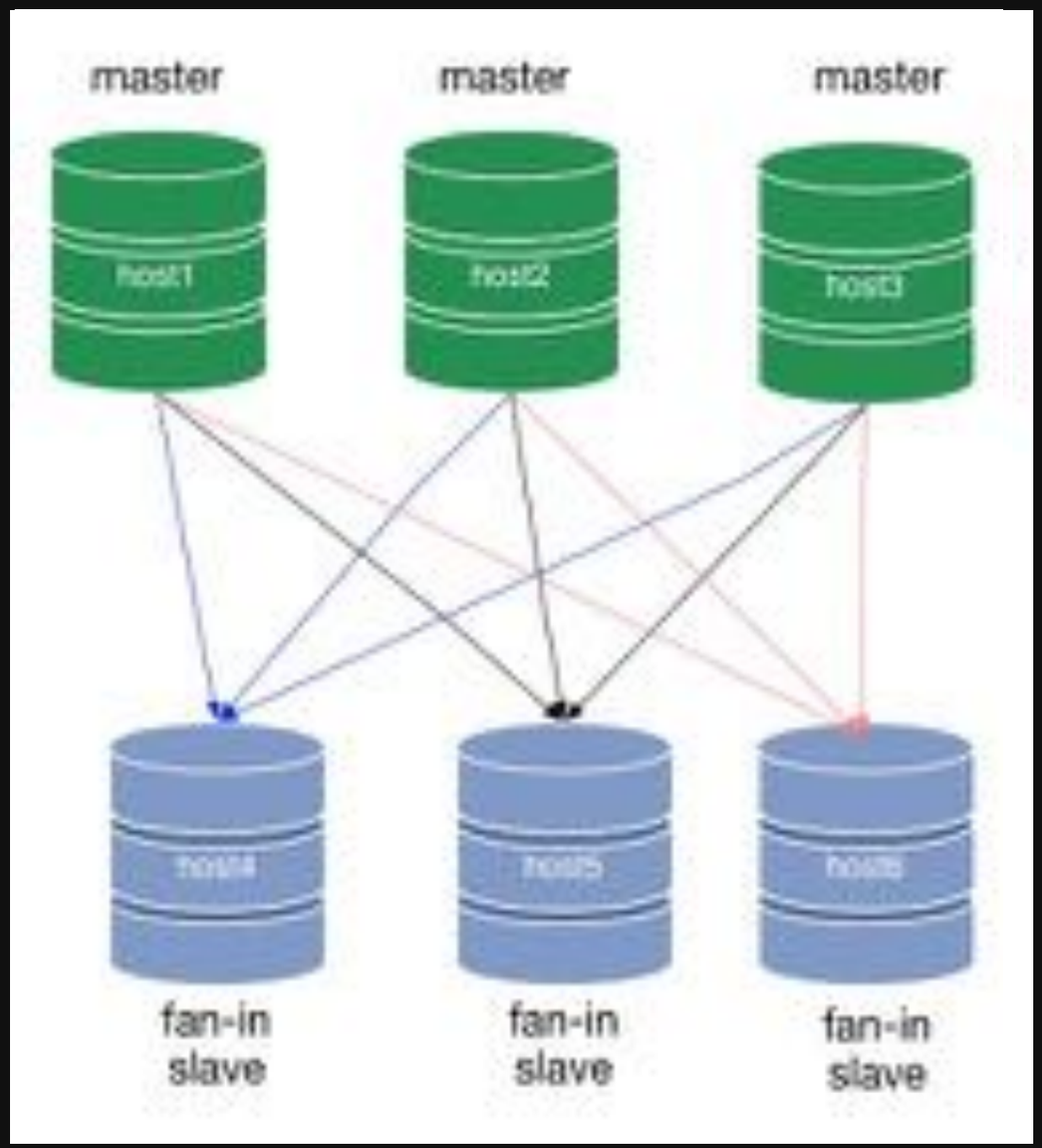
Fan-in topology

The simplest multi-source deployment can become complicated



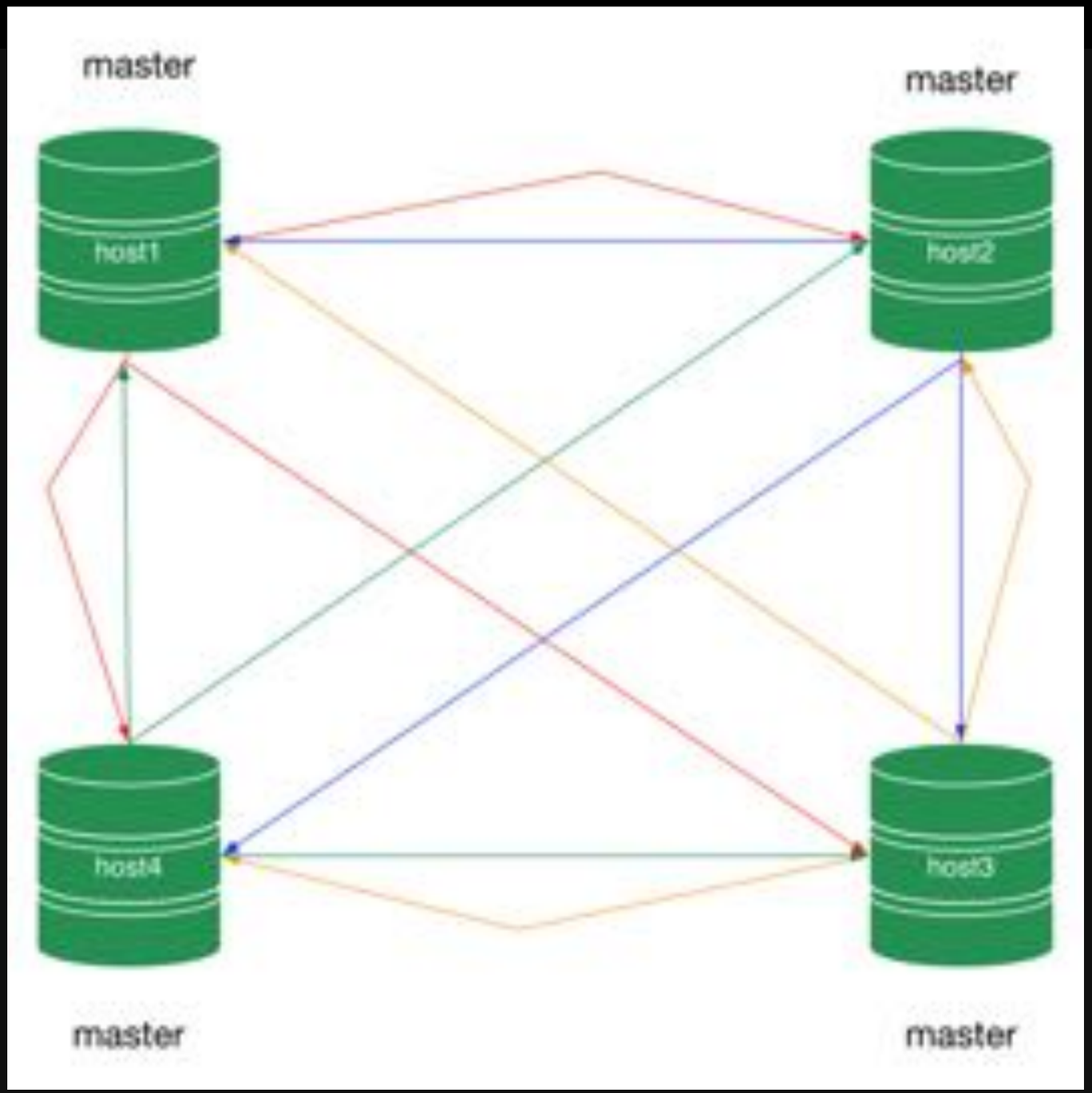
Fan-in topology

The simplest multi-source deployment can become complicated



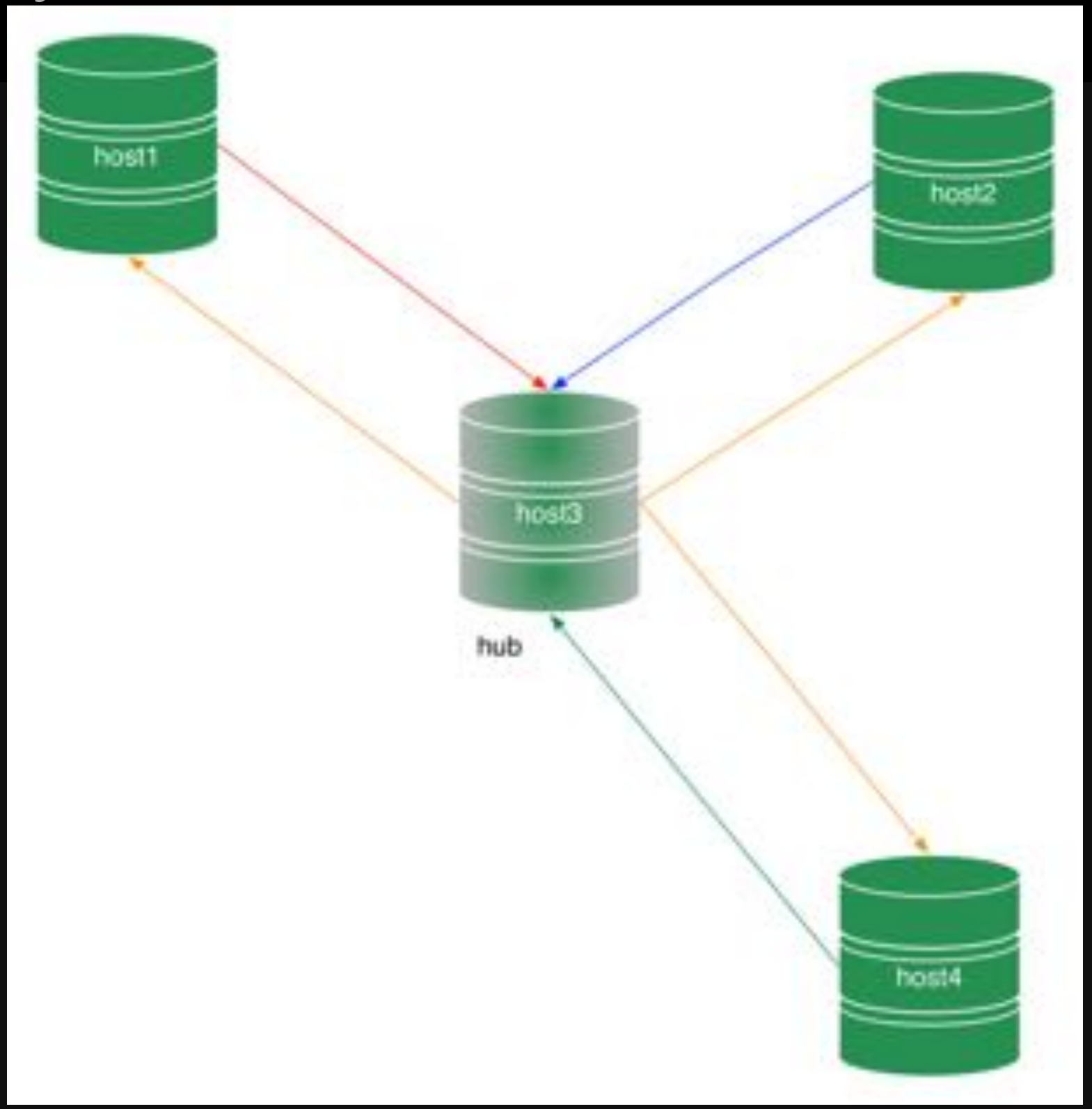
All masters point-to-point topology

The most crowded, yet most efficient replication system



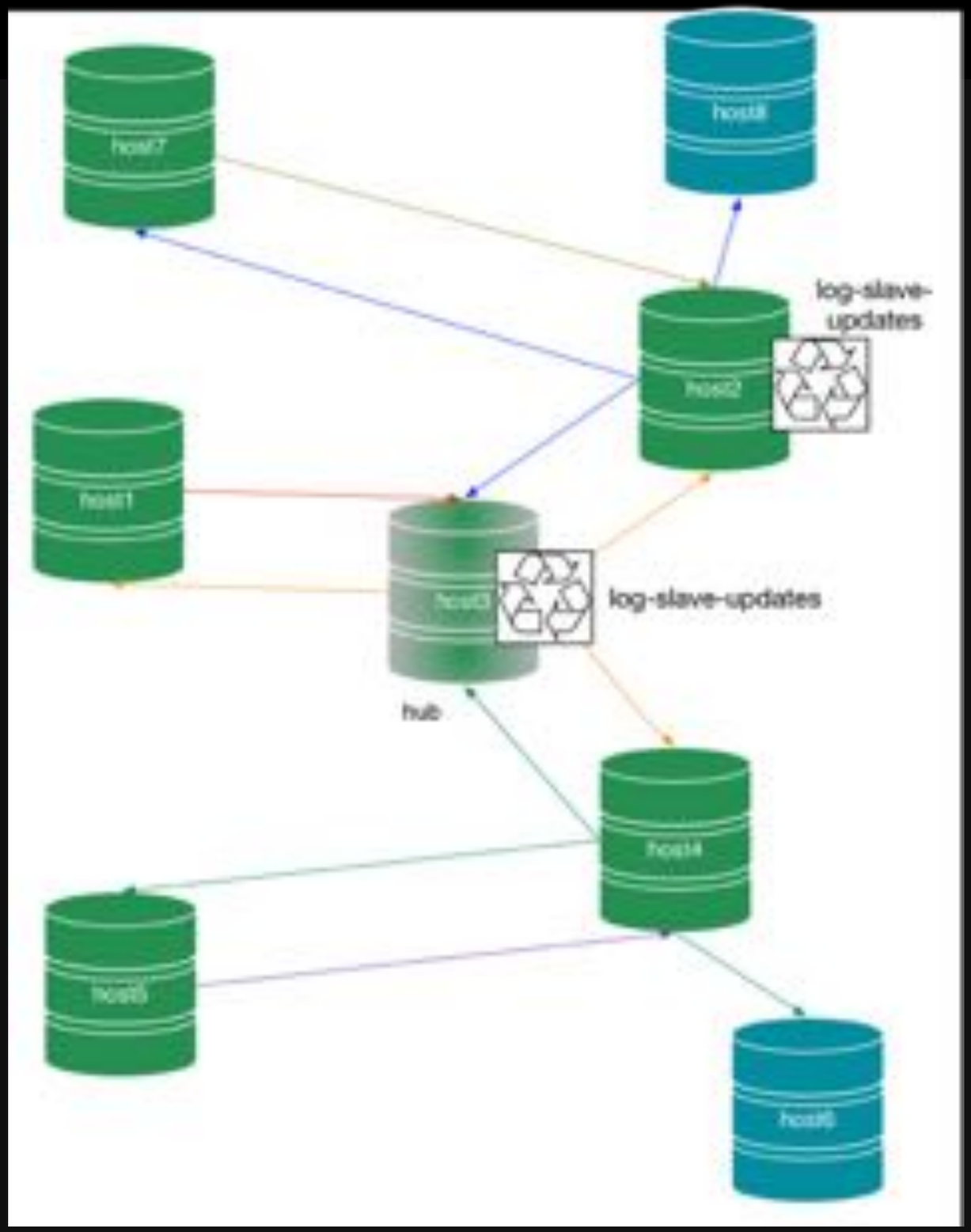
Star topology

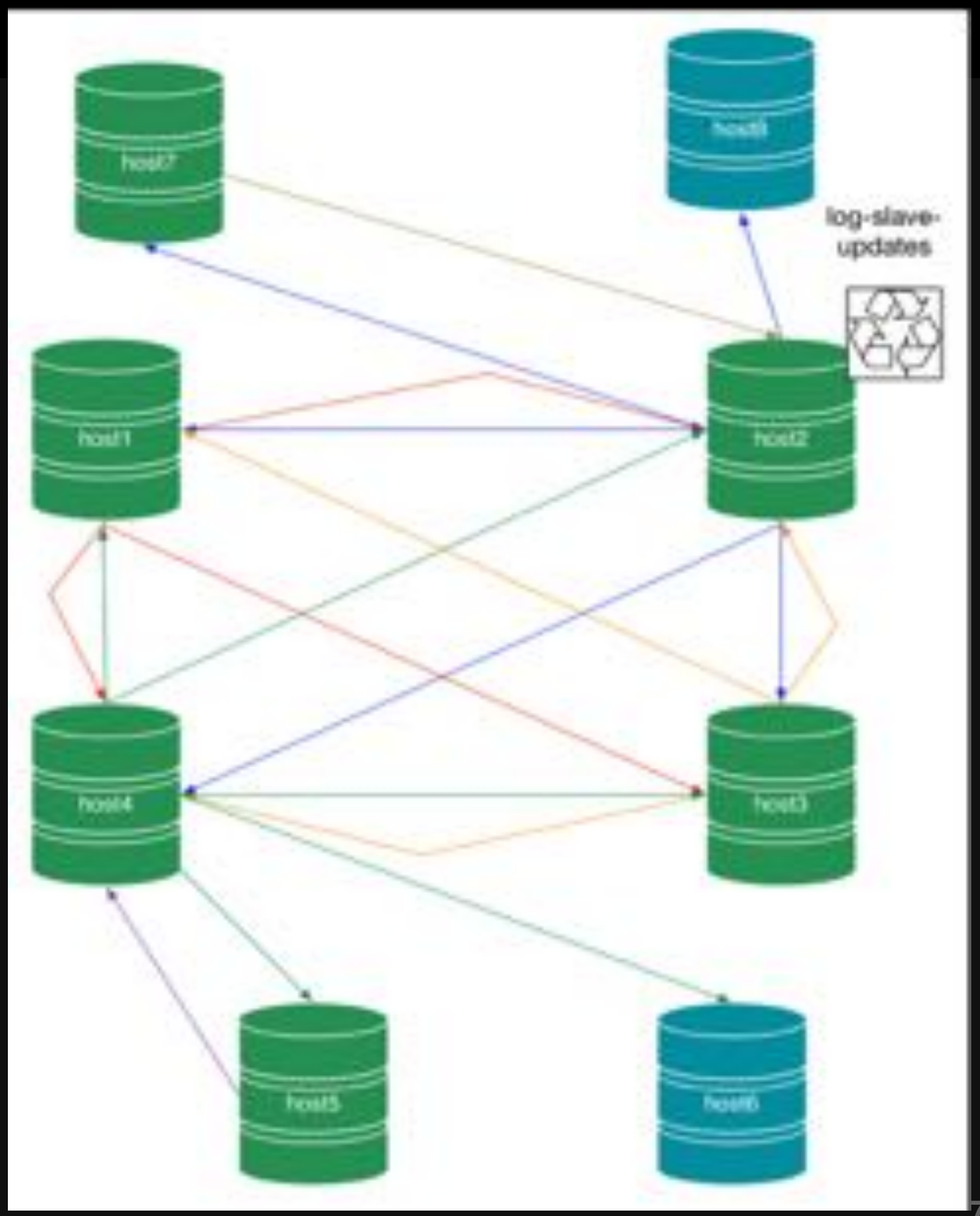
A thin deployment, with a SPOF



Hybrid topologies

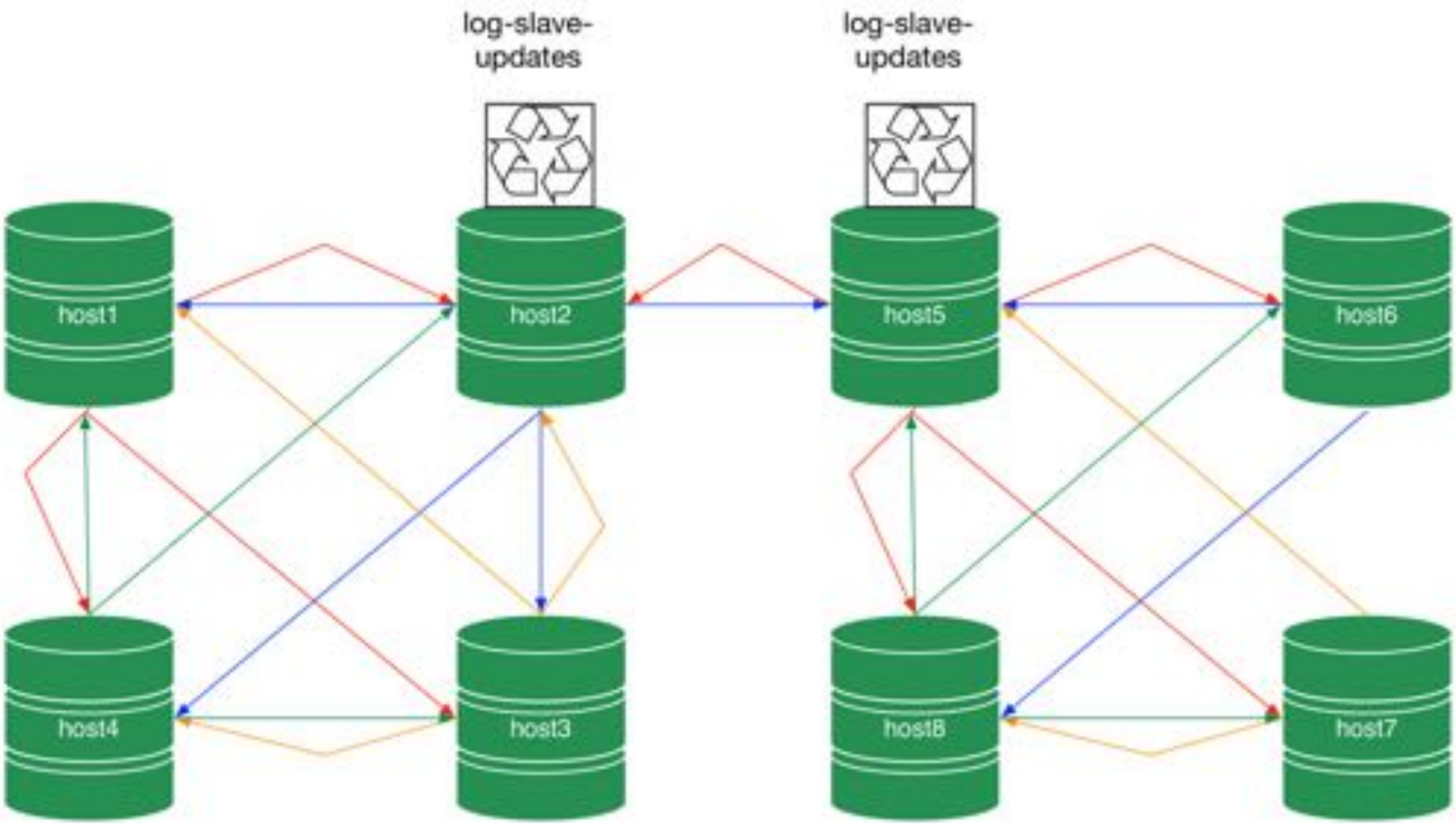
You can enhance star or point-to-point topologies





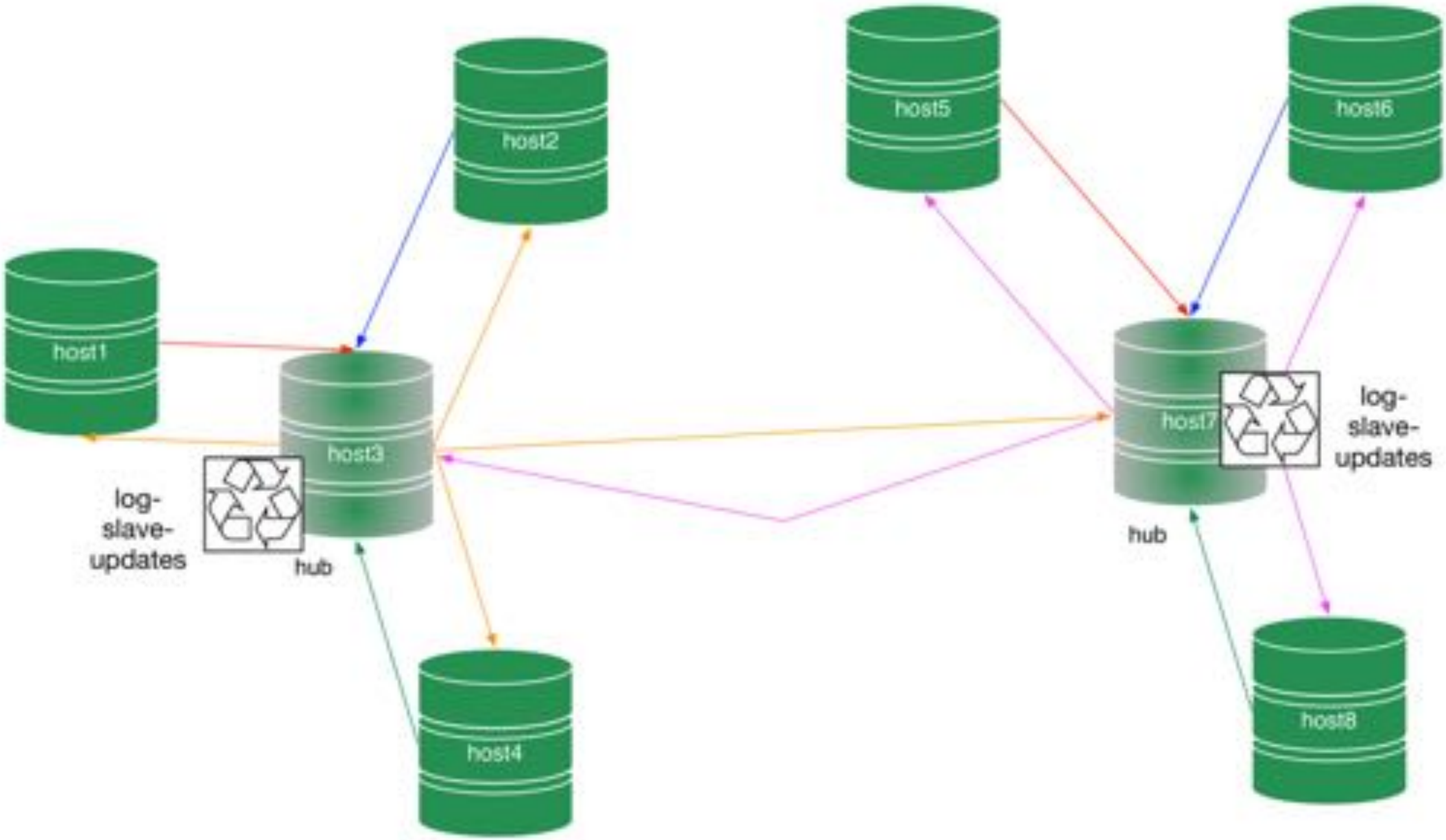
Hybrid topologies

You can combine two point-to-point topologies ...



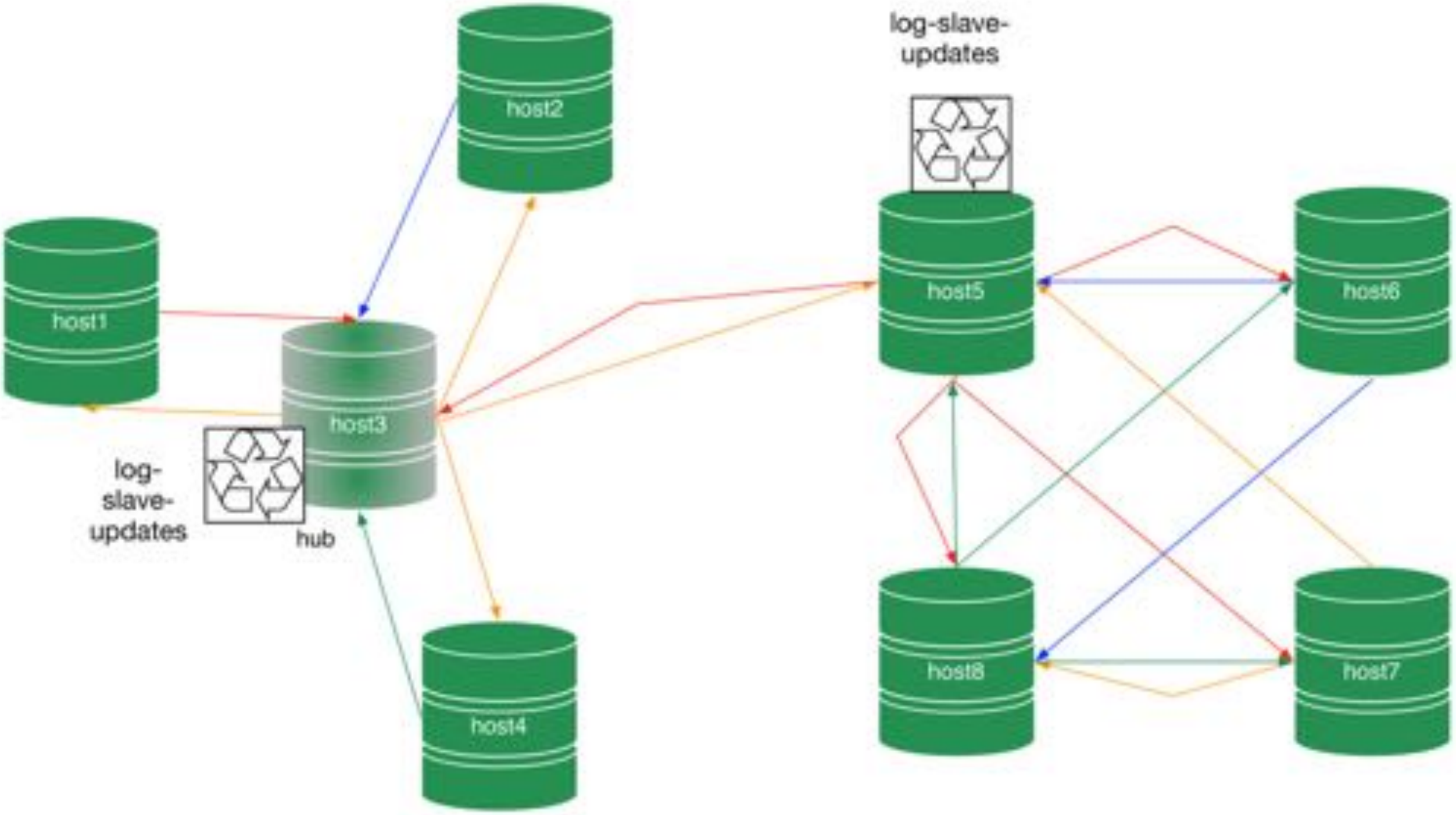
Hybrid topologies

... or two star topologies ...



Hybrid topologies

... or a star and a point-to-point together



Multi source demo

Claim: Multi source replication

- Claimed by
  - MySQL 5.7
  - MariaDB 10.0 and 10.1

Sceptic assessment: Multi source replication

- Both:
  - (+) Yes. You can run multi-source replication;
  - (+) SHOW SLAVE STATUS with many rows;
  - (+) Monitoring tables with many rows
  - (-) Mixed info about data created and received

Sceptic assessment: Multi source replication

CAN DO MUCH BETTER!

- Both:
  - (+) Yes. You can run multi-source replication;
  - (+) SHOW SLAVE STATUS with many rows;
  - (+) Monitoring tables with many rows
  - (-) Mixed info about data created and received

MySQL multi-source issues

- (-) Same issues for single stream, but worsened by multiple channels
- (+) SHOW SLAVE STATUS has a separate item for each channel.
- (-) GTID info is repeated as a group for every channel
- (-) show master status mixes up info about the data created and received

MariaDB multi-source issues

- (-) Same issues for single stream, but worsened by multiple channels
- (-) Syntax is different from MySQL
- (+) SHOW ALL SLAVES STATUS has a separate item for each channel.
- (-) GTID info is repeated as a group for every channel
- (-) GTID info in SHOW SLAVE STATUS include data created in the server.

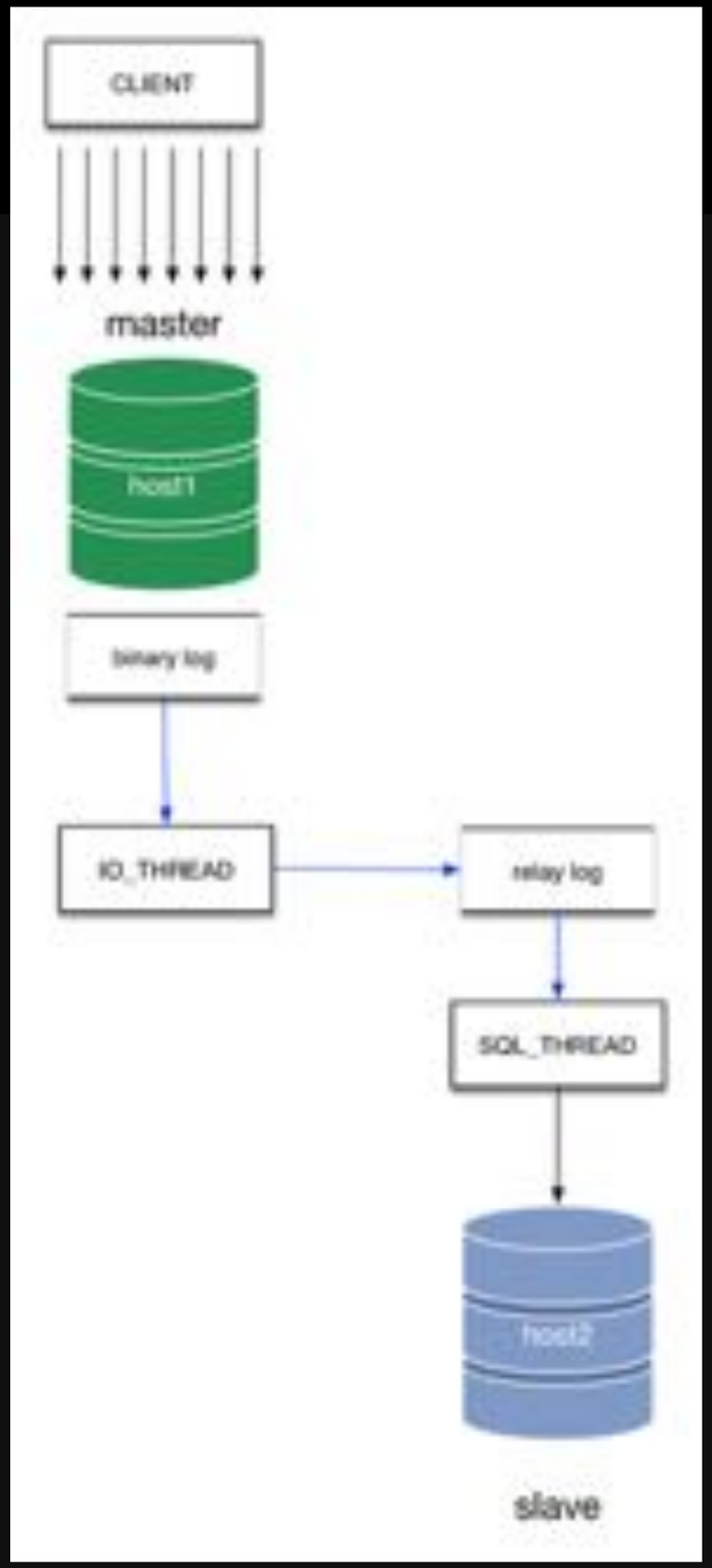
Parallel replication

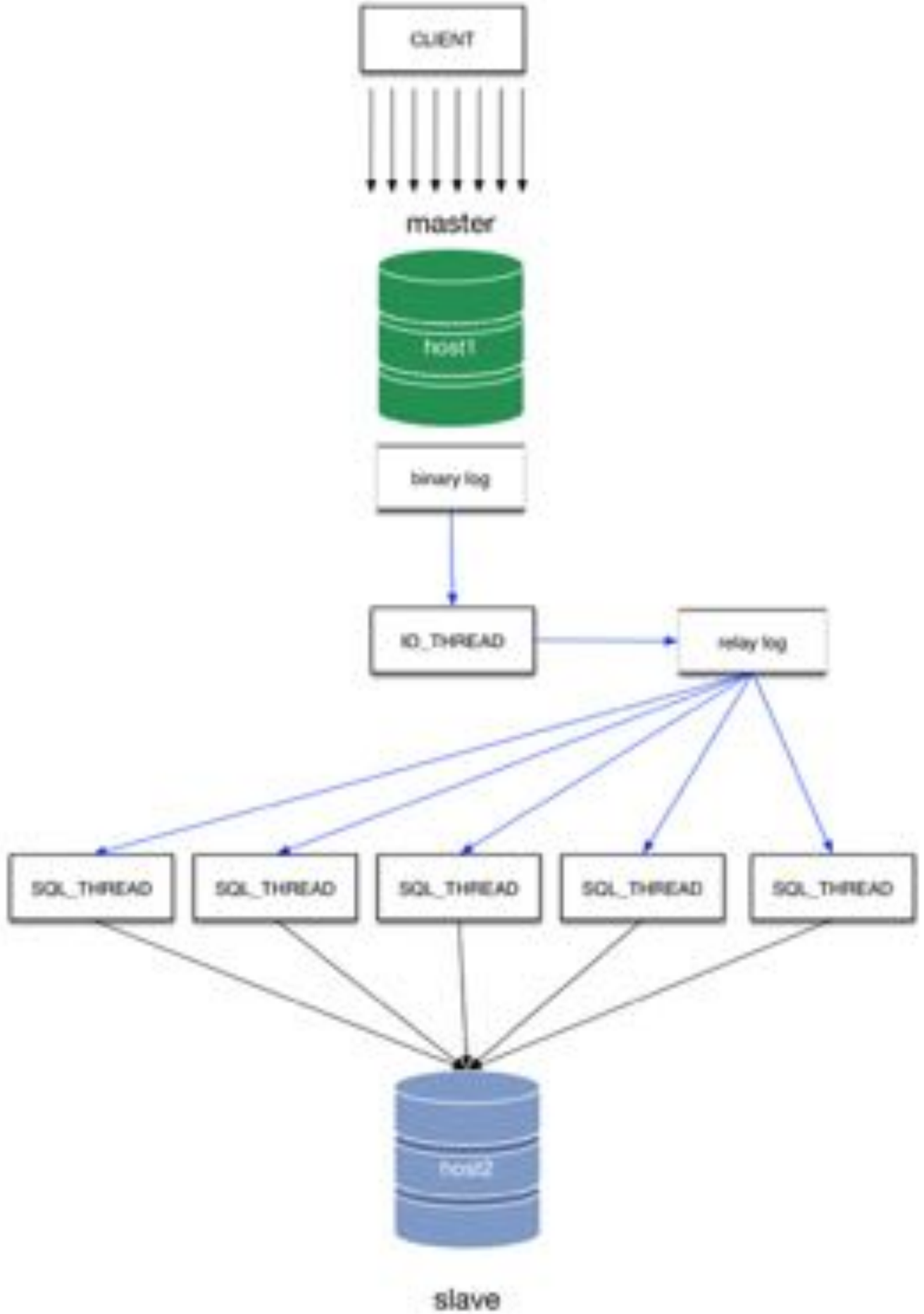
Parallel apply

When the slave lags, using parallel threads may speed up things

- It’s the ability of executing binary log events in parallel.
- Implemented in Tungsten Replication (2011, schema based), MySQL 5.6 (2012, schema based), MariaDB 10 (2013, boundless), MySQL 5.7 (2013, boundless)

Single vs parallel

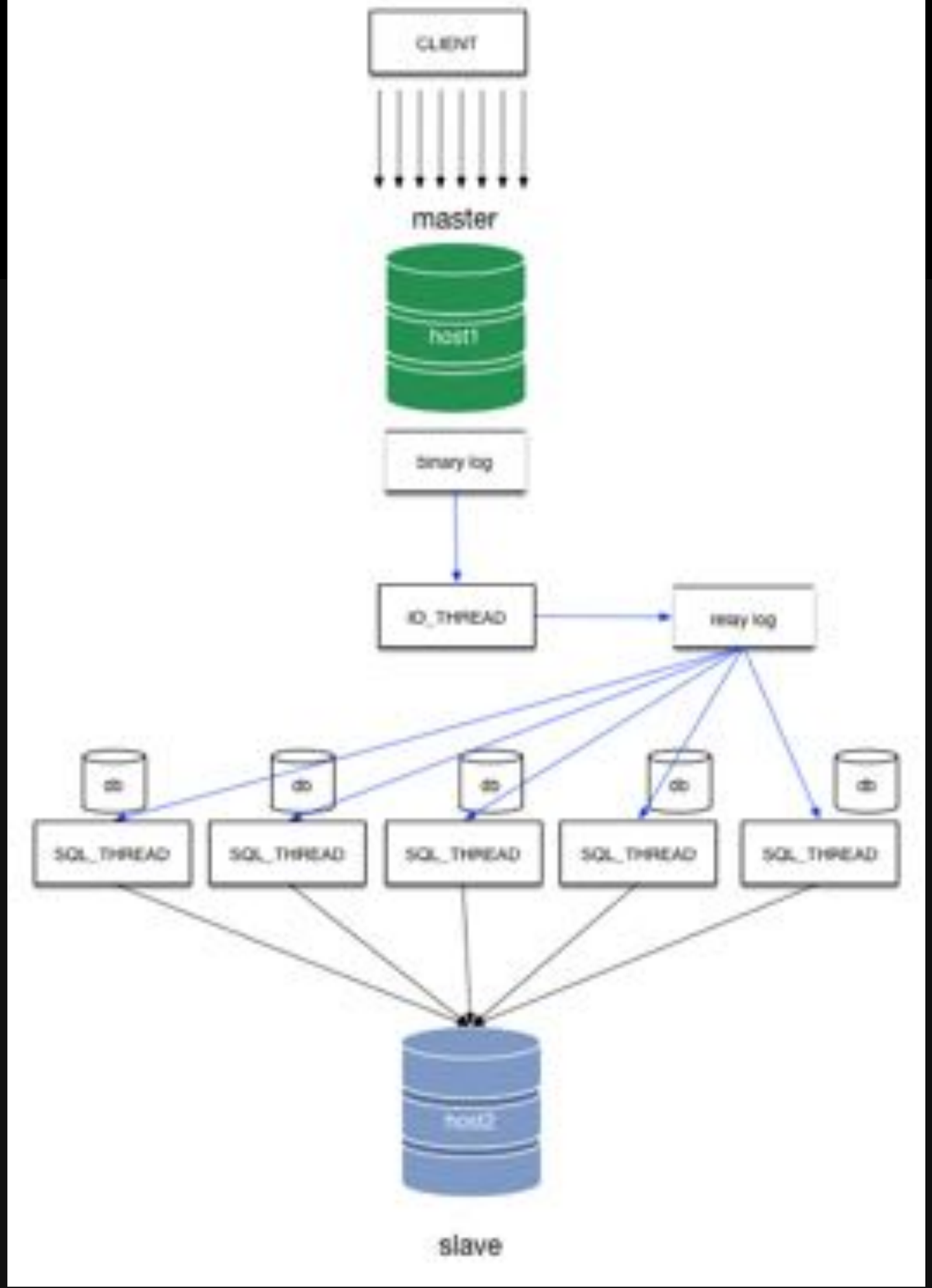




Implementation (1) Tungsten Replicator

The granddaddy of parallel replication, happily deployed in production for years

- Based on schema boundaries.
- No risk of deadlocks.
- Can be shared by criteria other than database, but only during provisioning.
- Fully integrated in the instrumentation;
- Provides extra information for monitoring and troubleshooting


CLIENT
master
host1
binary log
IO_THREAD
relay log
db
db
db
db
db
SQL_THREAD
SQL_THREAD
SQL_THREAD
SQL_THREAD
SQL_THREAD
host2
slave

Implementation (2) MySQL 5.6

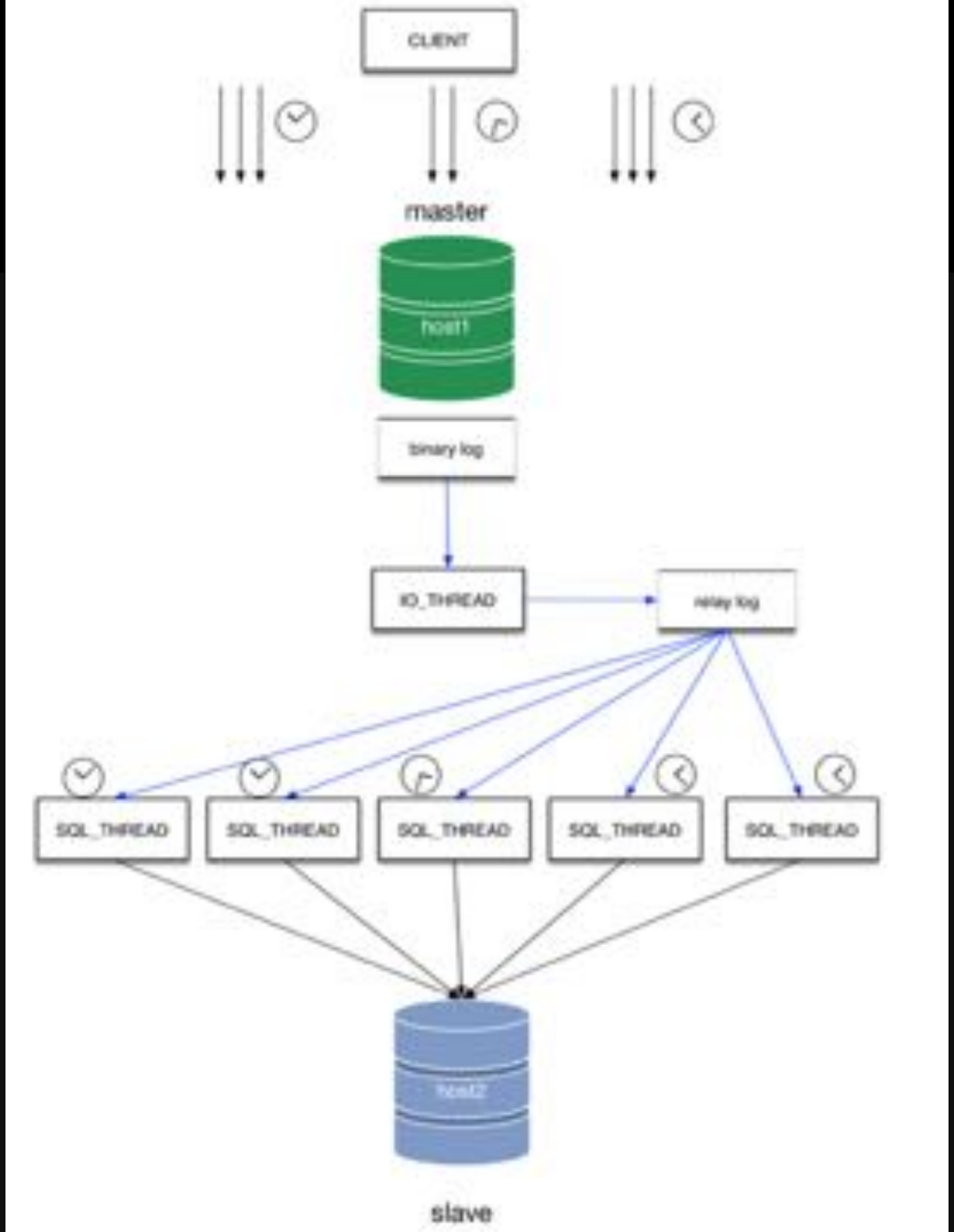
The first integrated solution for parallel replication

- Schema based, same as Tungsten.
- Requires both master and slave of the same version;
- No integration with GTID;
- No extra instrumentation.

Implementation (3) MySQL 5.7

Breaking the schema barriers

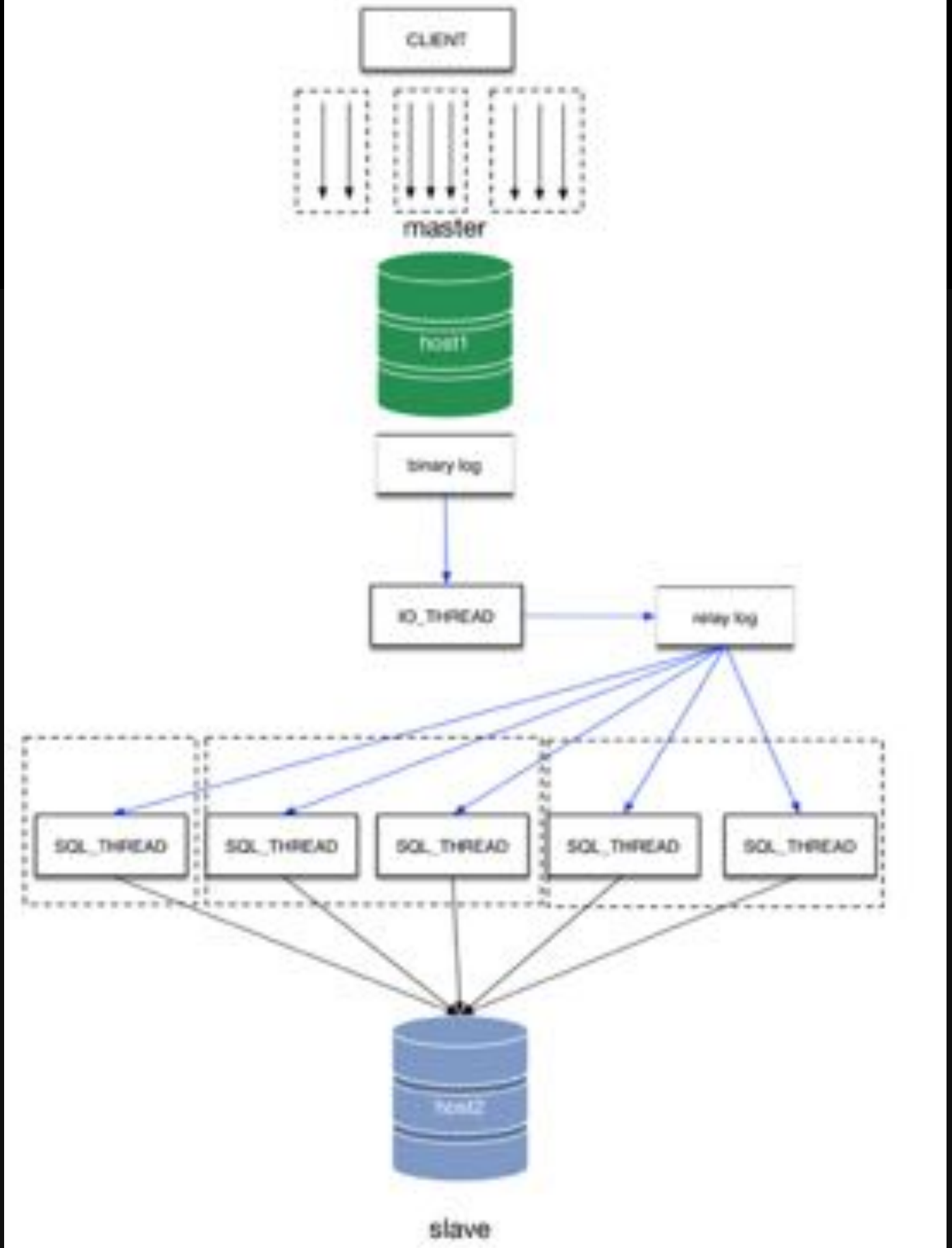
- Not schema based. Parallelism is defined by extra metadata from the master (logical clock).
- Requires both master and slave of the same version;
- Uses monitoring tables in performance schema
- Limited troubleshooting info;
- With multi-source, it’s all or nothing


CLIENT
master
host1
binary log
IO_THREAD
relay log
SQL_THREAD
SQL_THREAD
SQL_THREAD
SQL_THREAD
SQL_THREAD
host2
slave

Implementation (4) MariaDB 10

The latest contender

- Not schema based. Uses information from the coordinator to define how to parallelise;
- Integrated with GTID;
- Little instrumentation for troubleshooting.
- You can choose to which channel to apply (set default_master_connection='x').


CLIENT
master
host1
binary log
IO_THREAD
relay log
SQL_THREAD
SQL_THREAD
SQL_THREAD
SQL_THREAD
SQL_THREAD
slave

New development in MariaDB 10.1

A new algorithm for parallel replication

- Optimistic parallelisation
- Does not require preparation in the master
- Just released

Parallel replication expectations

Looking for performance, sometimes it's deceiving

- Performance depends on data distribution.
- Same data can have different performance on various methods.
- Slave resources and tuning affect reliability.

Parallel replication demo

Claim: parallel replication

- Claimed by
  - MySQL 5.6 & 5.7
  - MariaDB 10.0 and 10.1

Sceptic assessment: parallel replication

- Both:
  - (+) Yes. You can improve performance with parallel replication;
  - (-) There is LITTLE support for monitoring;
- MySQL 5.7
  - Some improvement in monitoring. Better info on failure
- MariaDB 10.x
  - Terrible instrumentation: like driving in the dark

Sceptic assessment: parallel replication

NEEDS BETTER METADATA!

- Both:
  - (+) Yes. You can improve performance with parallel replication;
  - (-) There is LITTLE support for monitoring;
- MySQL 5.7
  - Some improvement in monitoring. Better info on failure
- MariaDB 10.x
  - Terrible instrumentation: like driving in the dark

Supporting material and software

http://bit.ly/my-rep-samples

(or check 'datacharmer' on GitHub)

Useful links

- GTID in MySQL
- Performance_schema tables for replication
- GTID in MariaDB
- Multi Source in MySQL
- Multi Source in MariaDB
- Parallel Replication in MariaDB

Q&A