facebook

How Facebook Got Consistency with MySQL in the Cloud

Sam Dunster

Production Engineer

Apache

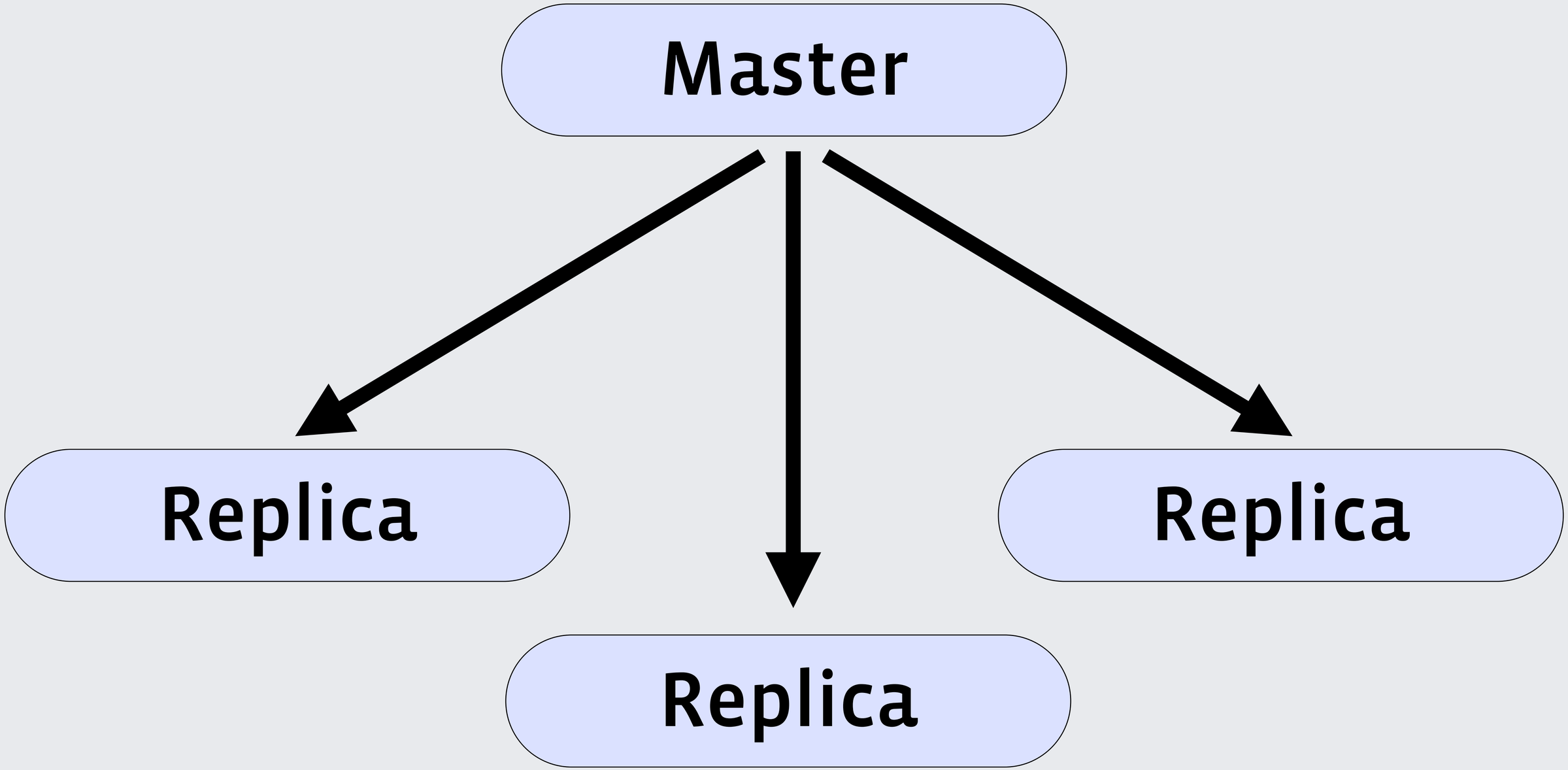
MySQL™

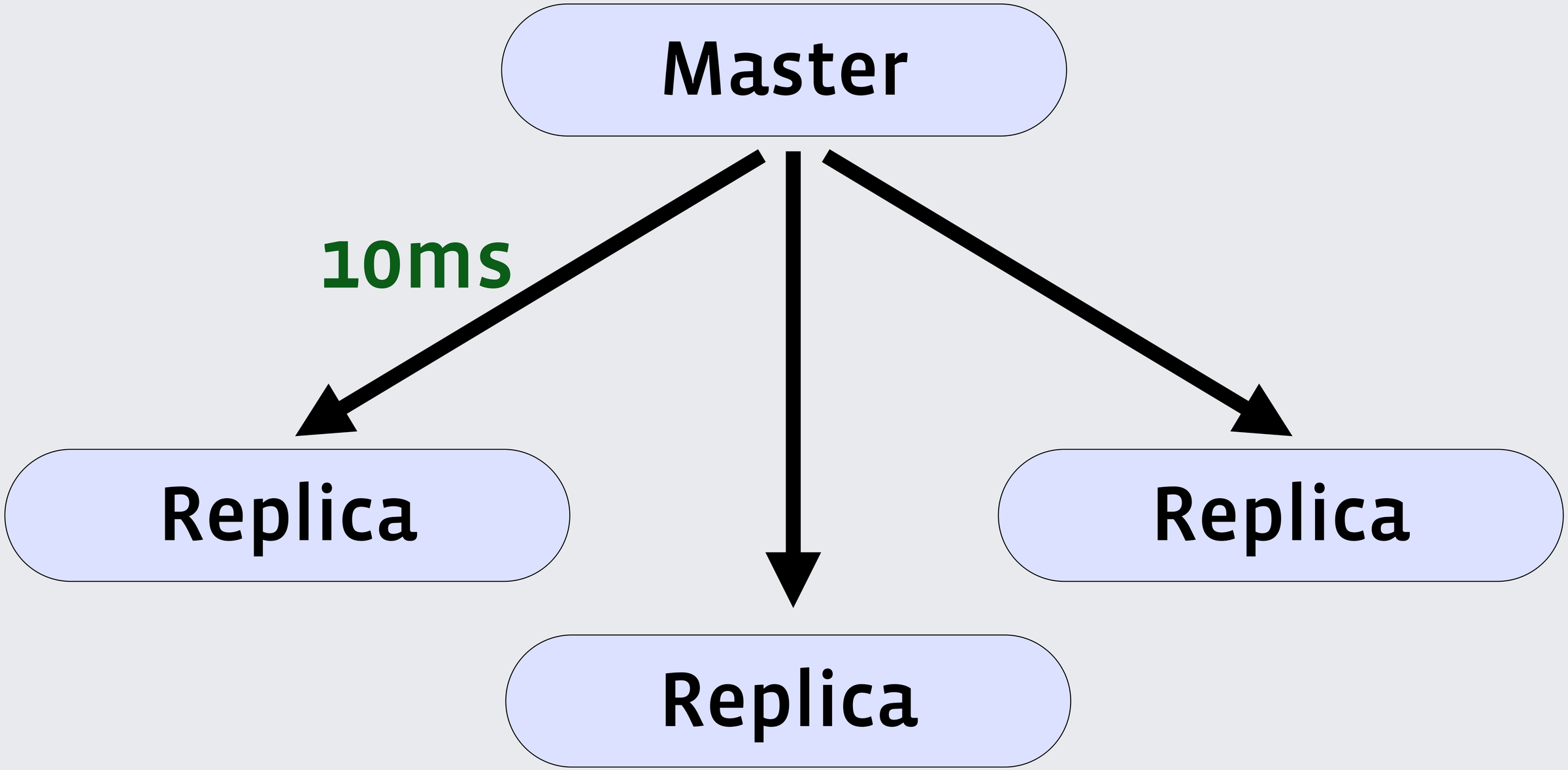
php

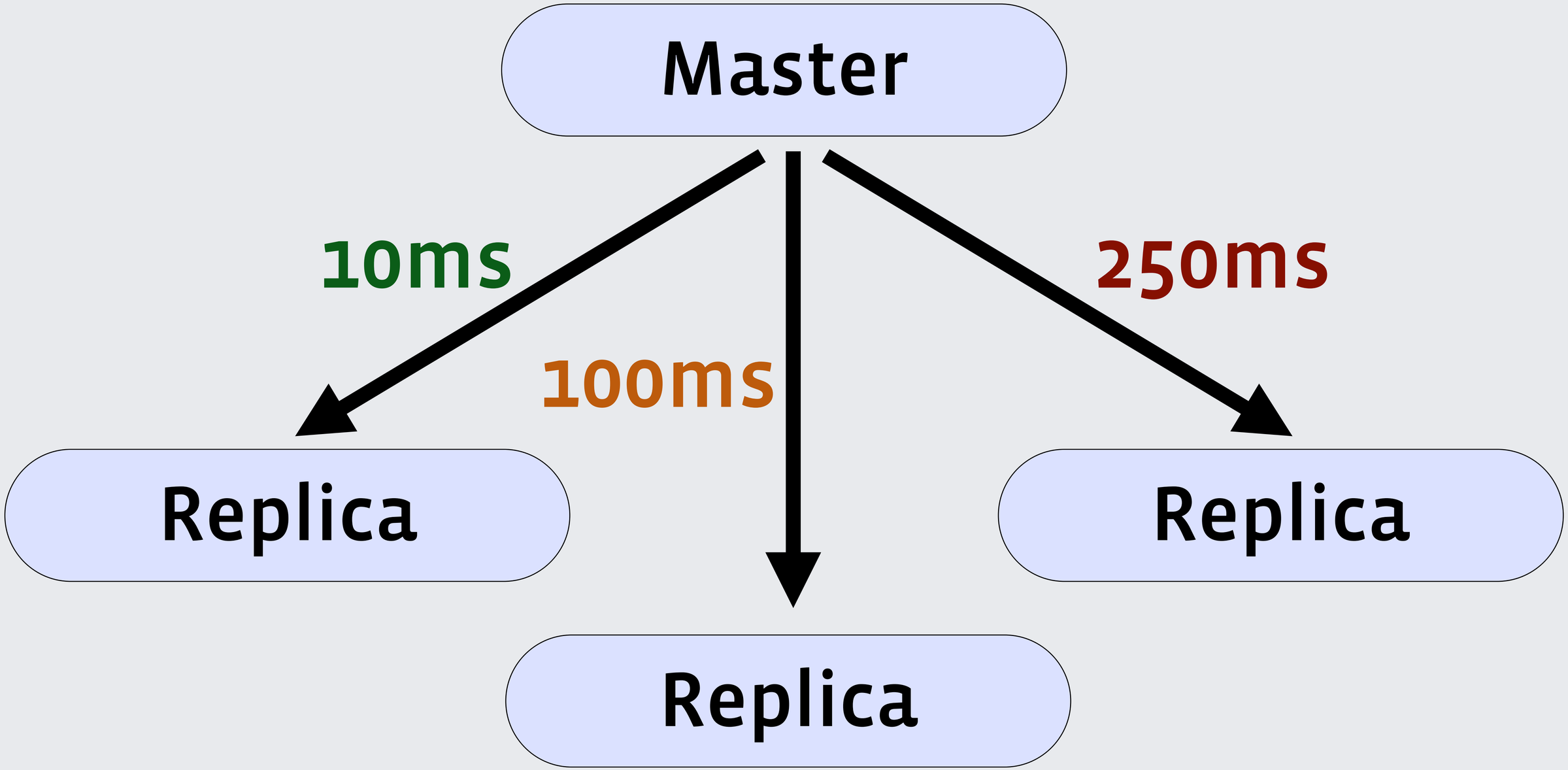
Consistency

Replication

Replication for High Availability


Master
Replica
Replica
Replica

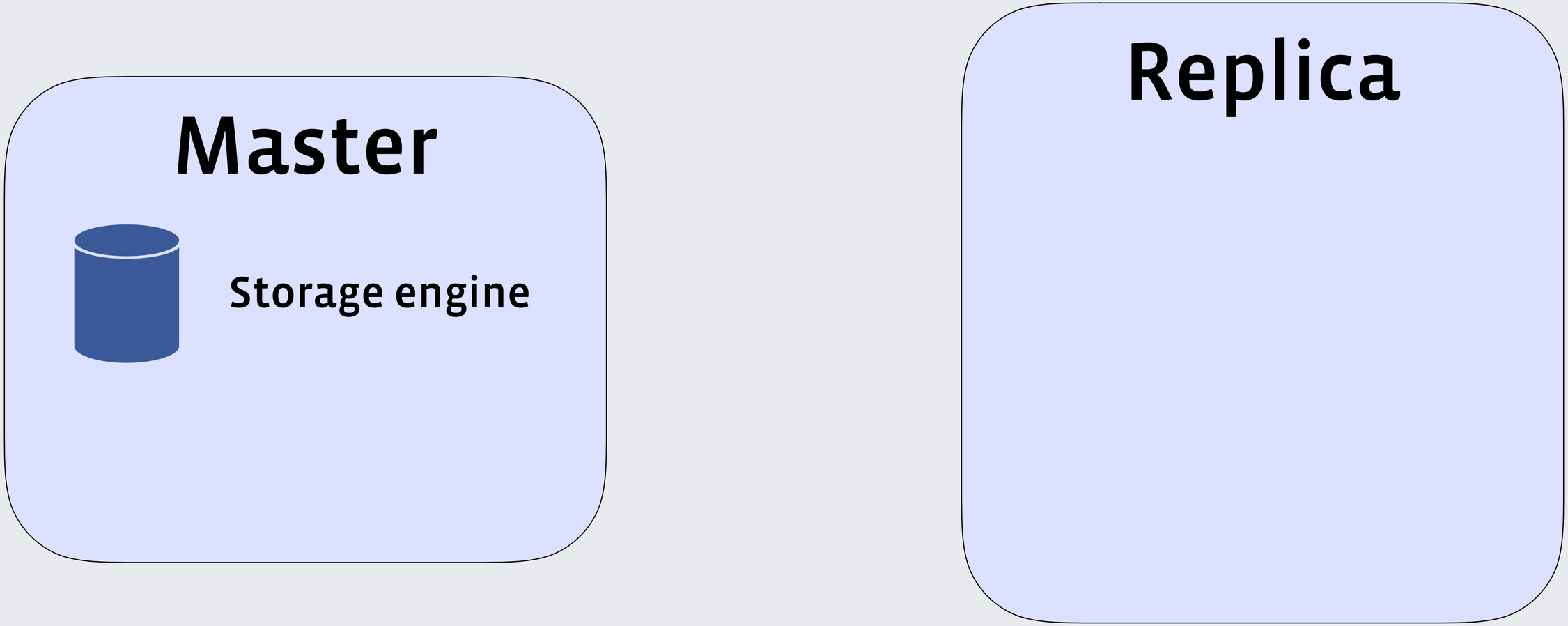

Master
10ms
Replica
Replica
Replica

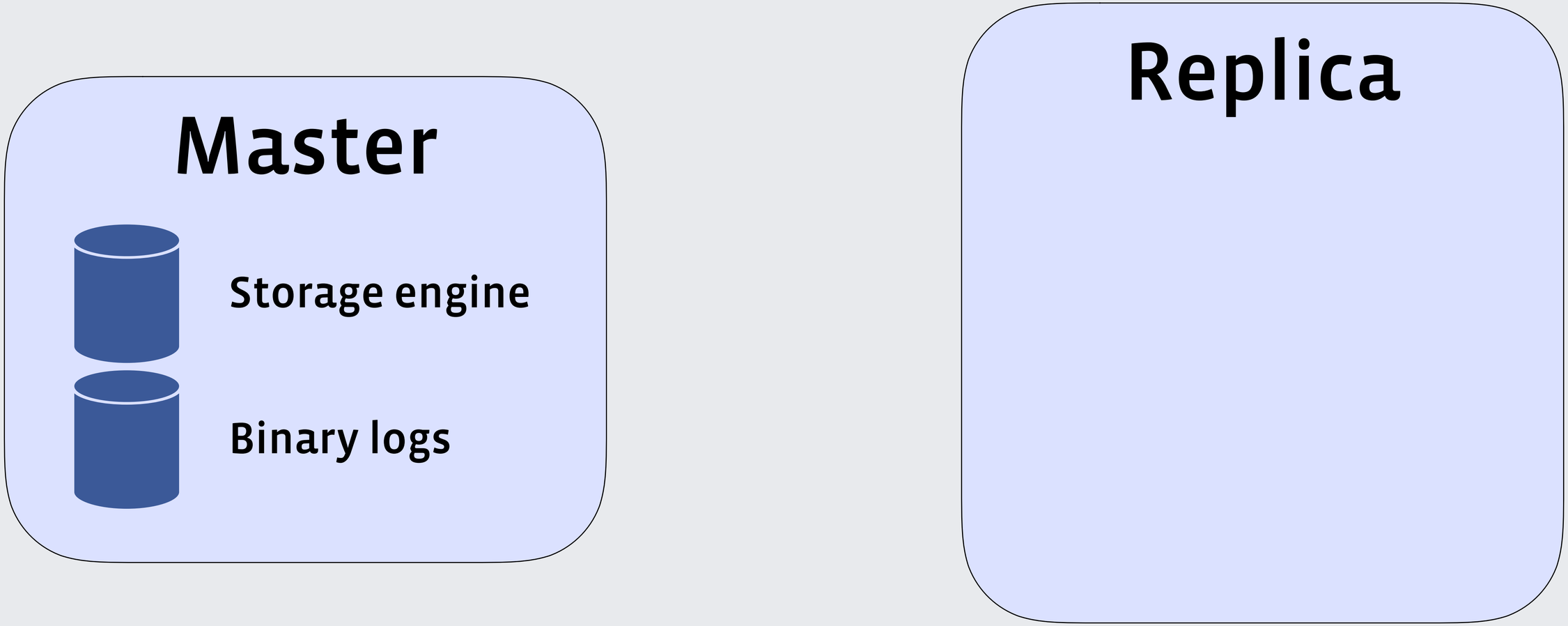

Master
10ms
100ms
250ms
Replica
Replica
Replica

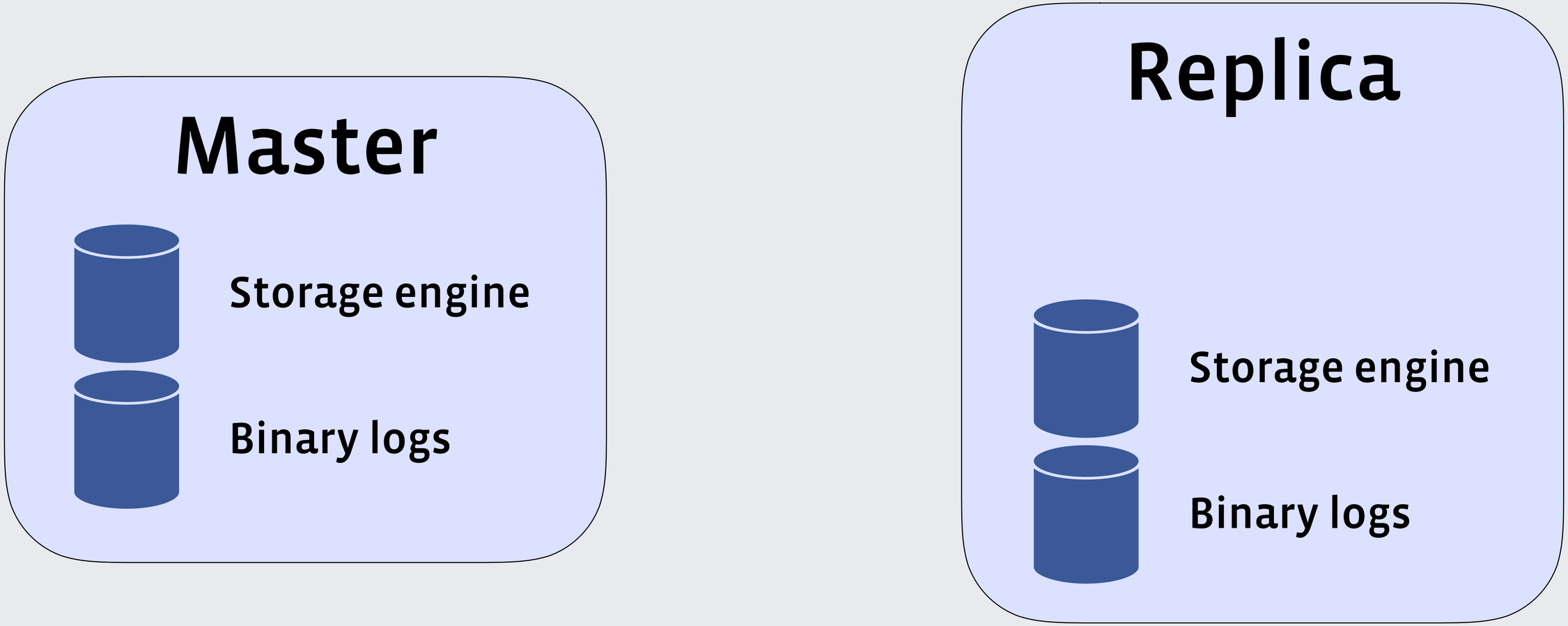
Asynchronous Replication

Master

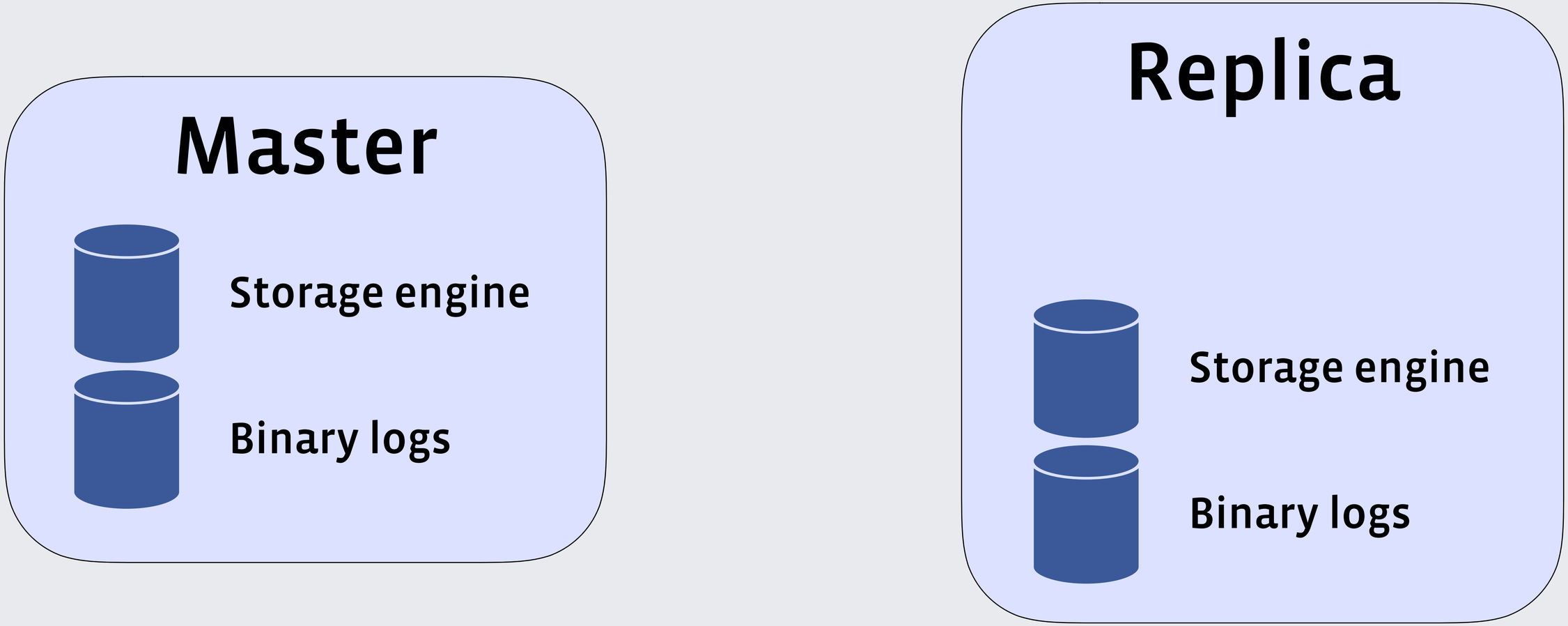
Replica


Master
Storage engine
Replica

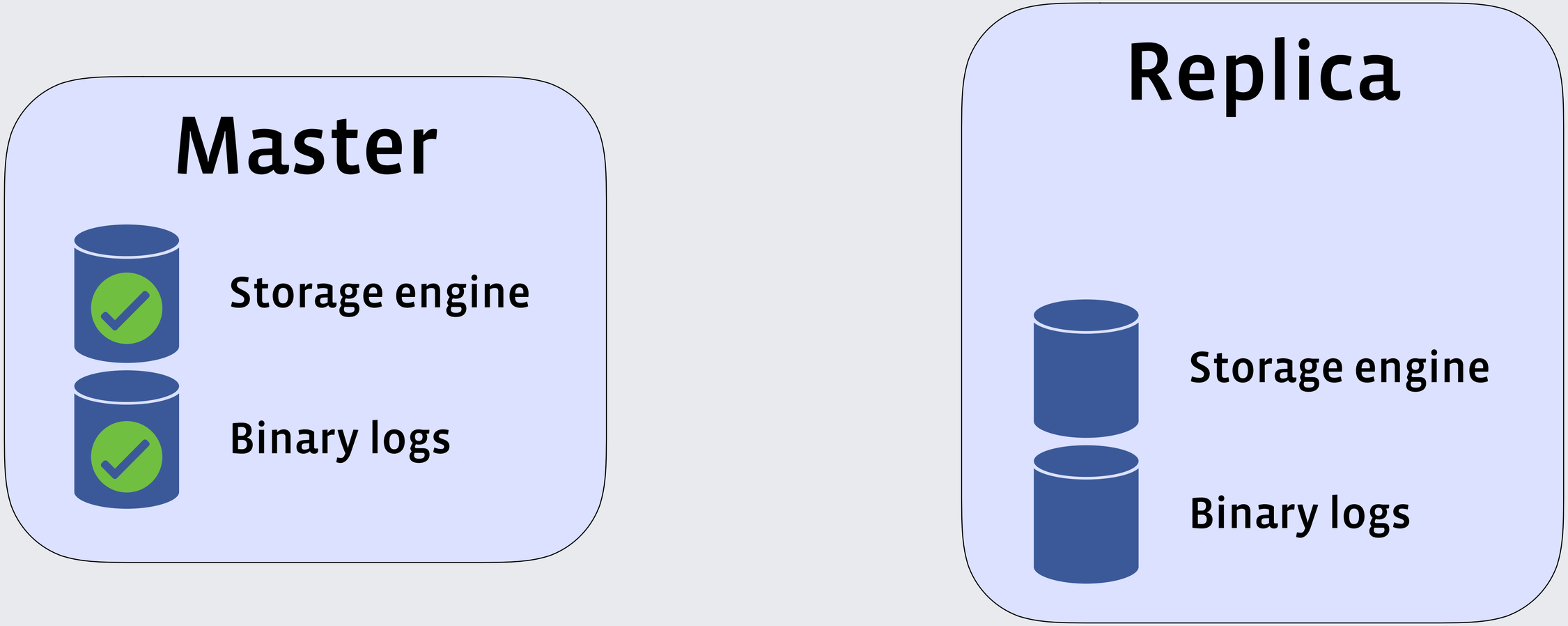

Master
Storage engine
Binary logs
Replica


Master
Storage engine
Binary logs
Replica
Storage engine
Binary logs

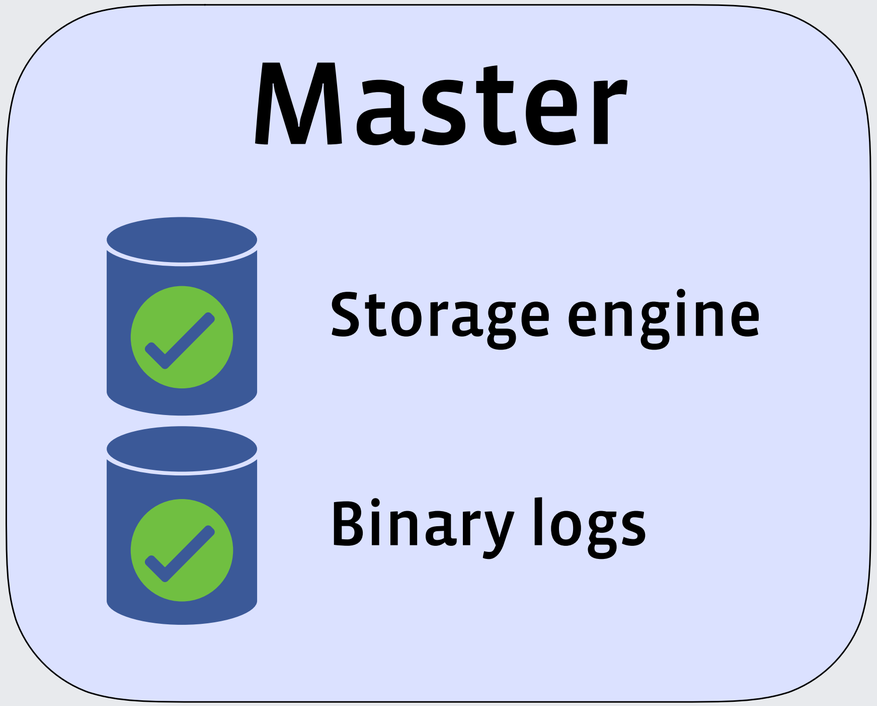
```
UPDATE table_a SET foo = "bar";
```

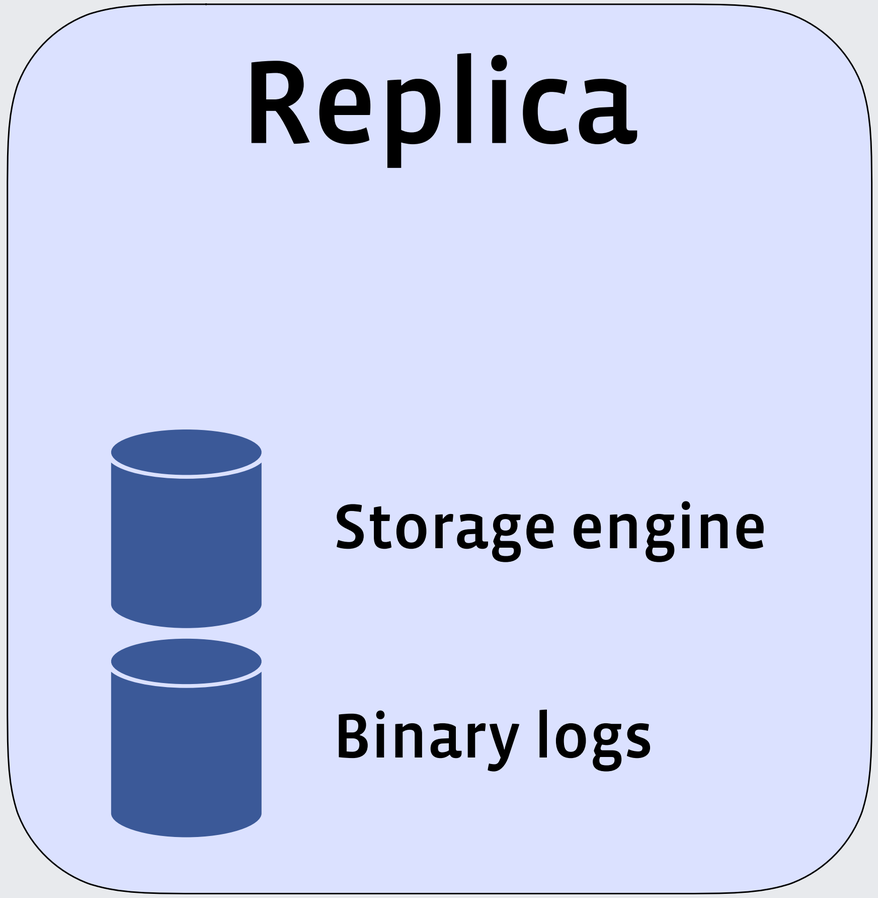


```
UPDATE table_a SET foo = "bar";
```

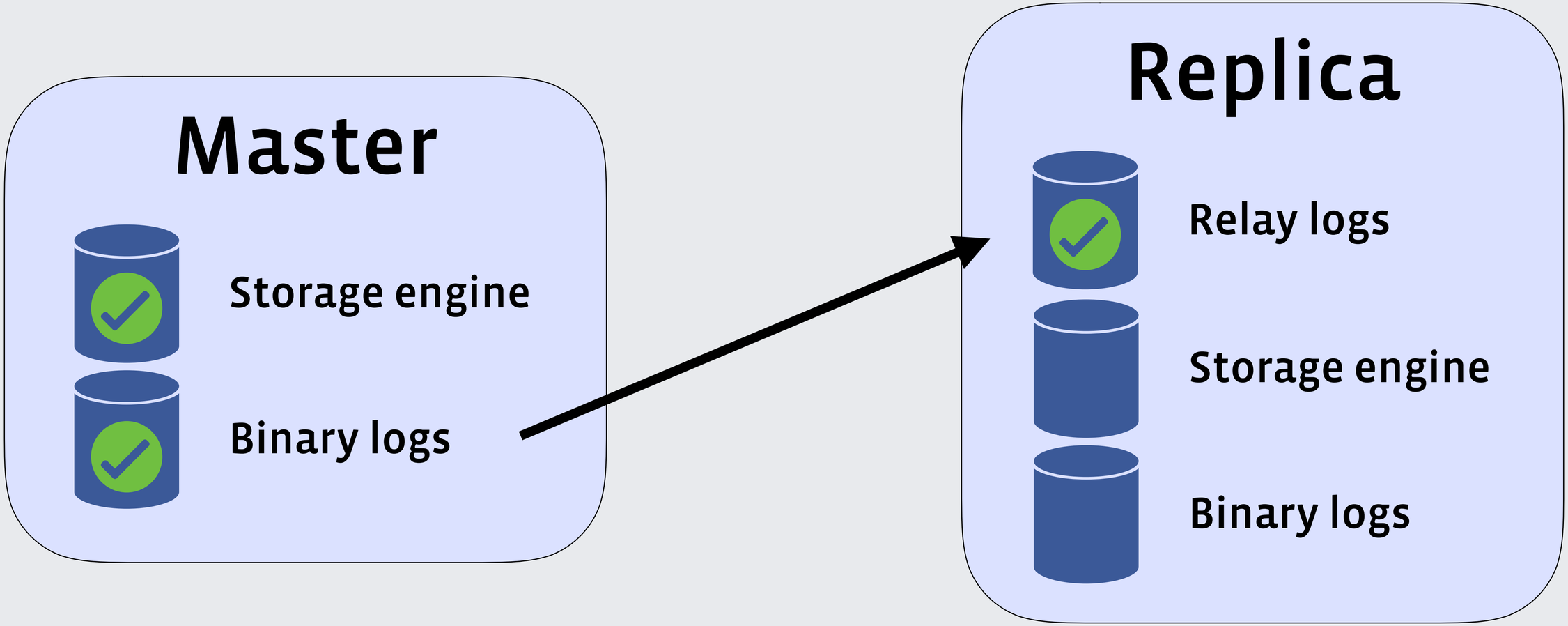


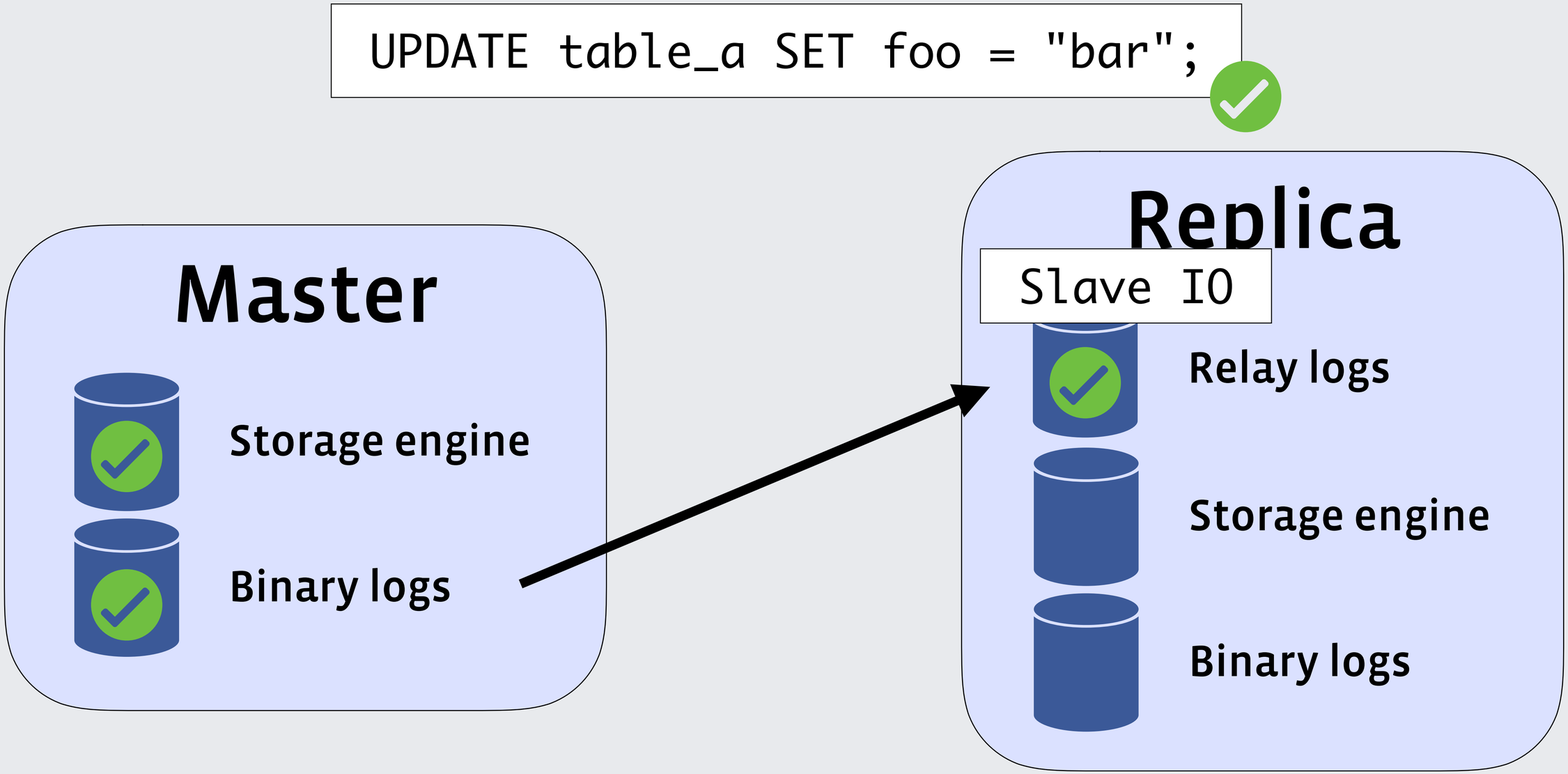
```
UPDATE table_a SET foo = "bar";
```

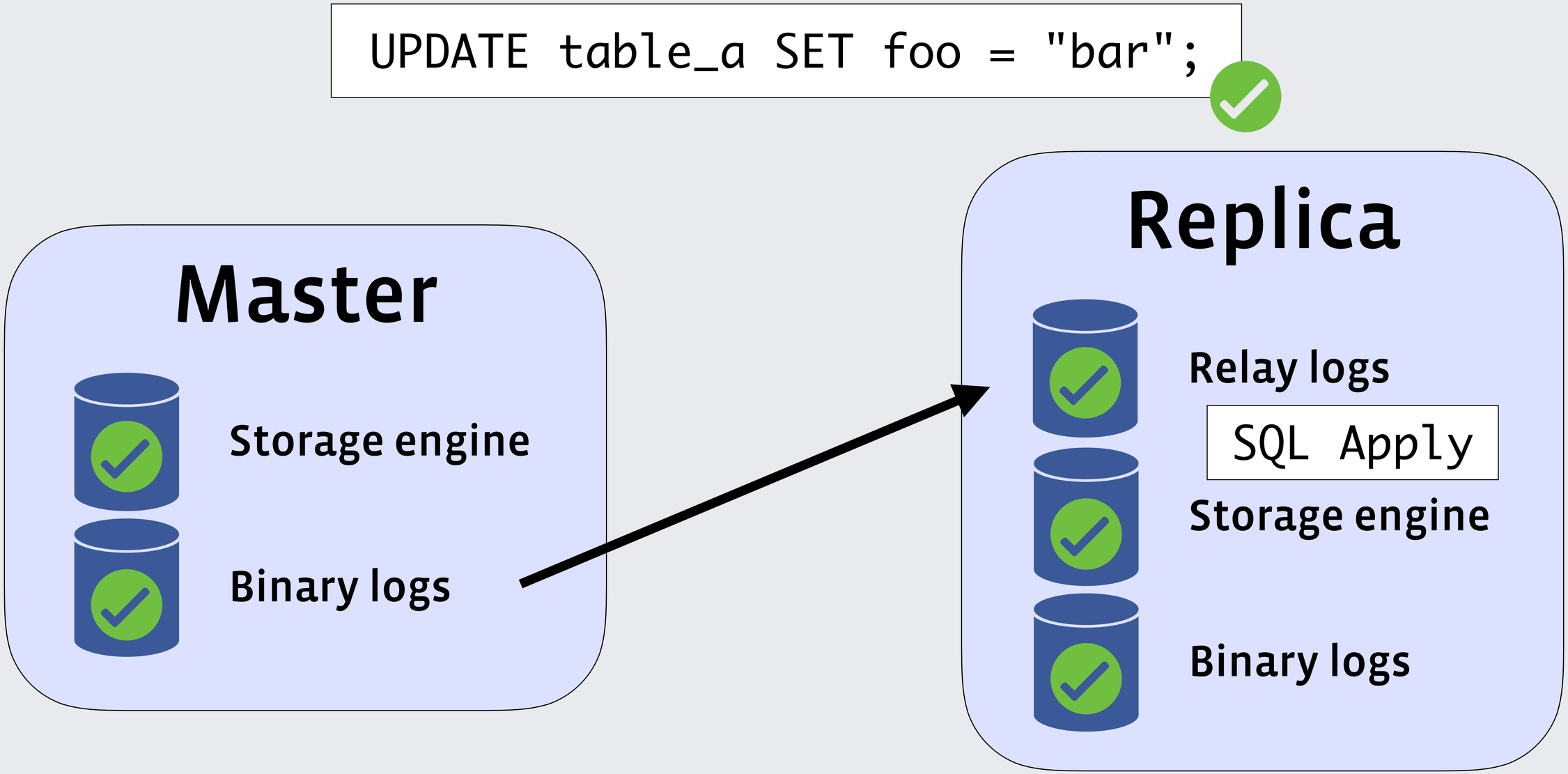




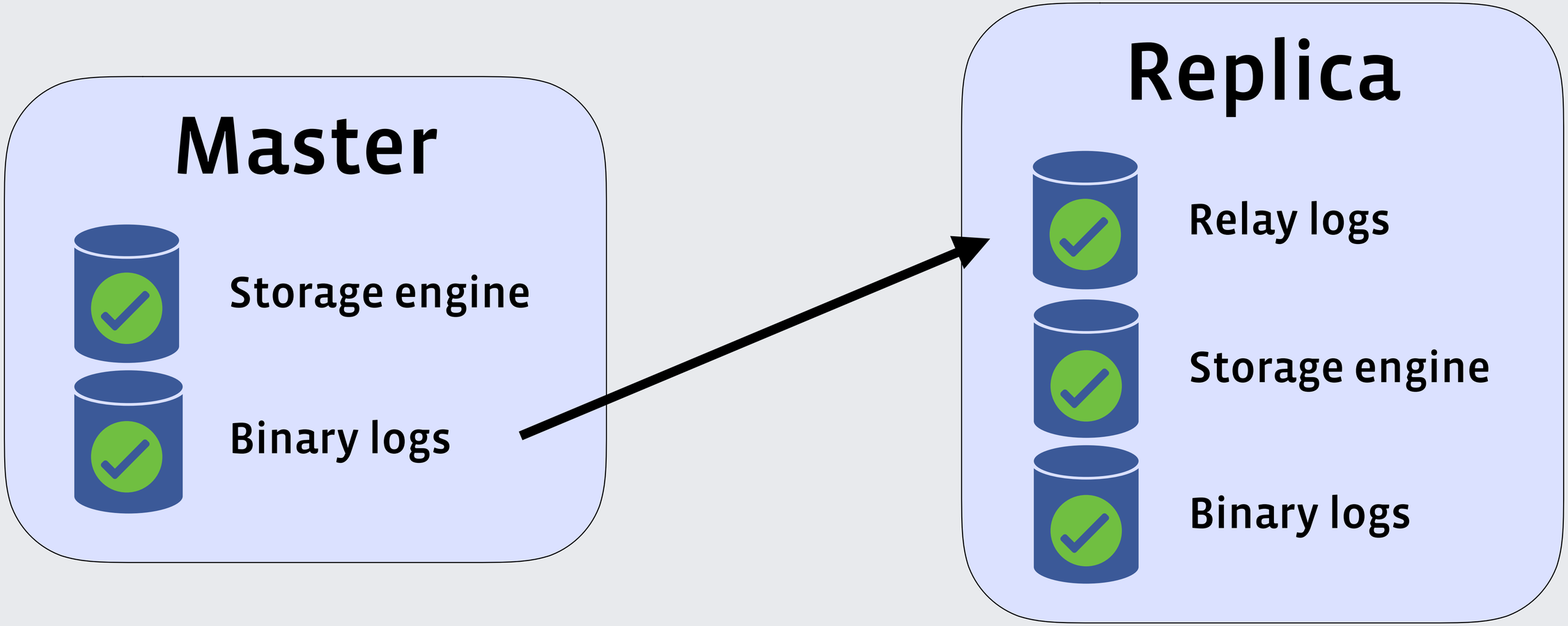
```
UPDATE table_a SET foo = "bar";
```




UPDATE table_a SET foo = "bar";
Master
Storage engine
Binary logs
Replica
Slave IO
Relay logs
Storage engine
Binary logs

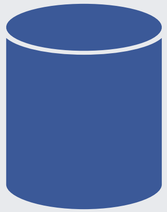

UPDATE table_a SET foo = "bar";
Master
Storage engine
Binary logs
Replica
Relay logs
SQL Apply
Storage engine
Binary logs

```
UPDATE table_a SET foo = "bar";
```



read/write

read (delayed)



Binary logs

Position-based

```
mysql > SHOW MASTER STATUS;
+--------------------+----------+
| File               | Position |
+--------------------+----------+
| mysql-bin.000003 | 73       |
+--------------------+----------+
```

GTID

Global transaction ID

3E11FA47-71CA-11E1-9E33-C80AA9429562:23

source_id **transaction_id**

GTID Set

Show me which transactions you have executed

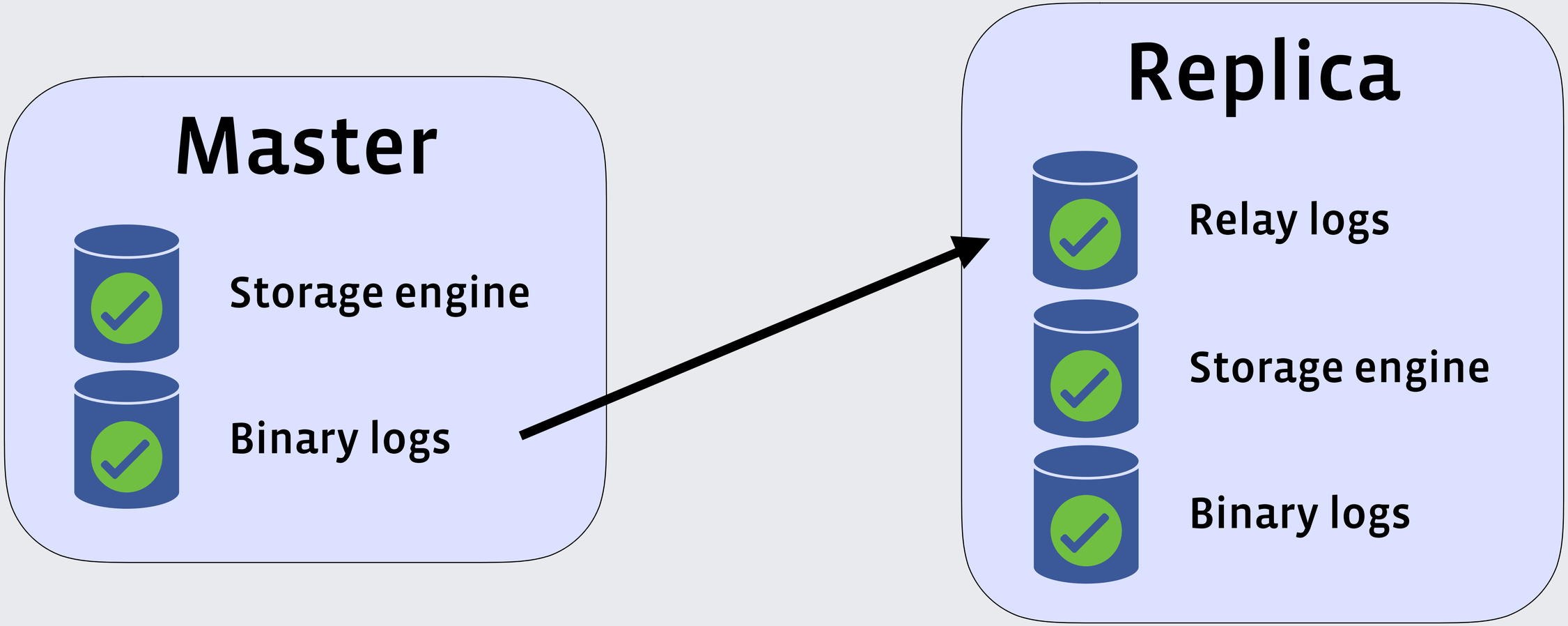
2174B383-5441-11E8-B90A-C80AA9429562:1-3,
24DA167-0C0C-11E8-8442-00059A3C7B00:1-19

GTID-based Auto positioning

```
mysql> CHANGE MASTER TO
    >     MASTER_HOST = host,
    >     MASTER_PORT = port,
    >     MASTER_USER = user,
    >     MASTER_PASSWORD = password,
    >     MASTER_AUTO_POSITION = 1;
```

Glob tra on ID

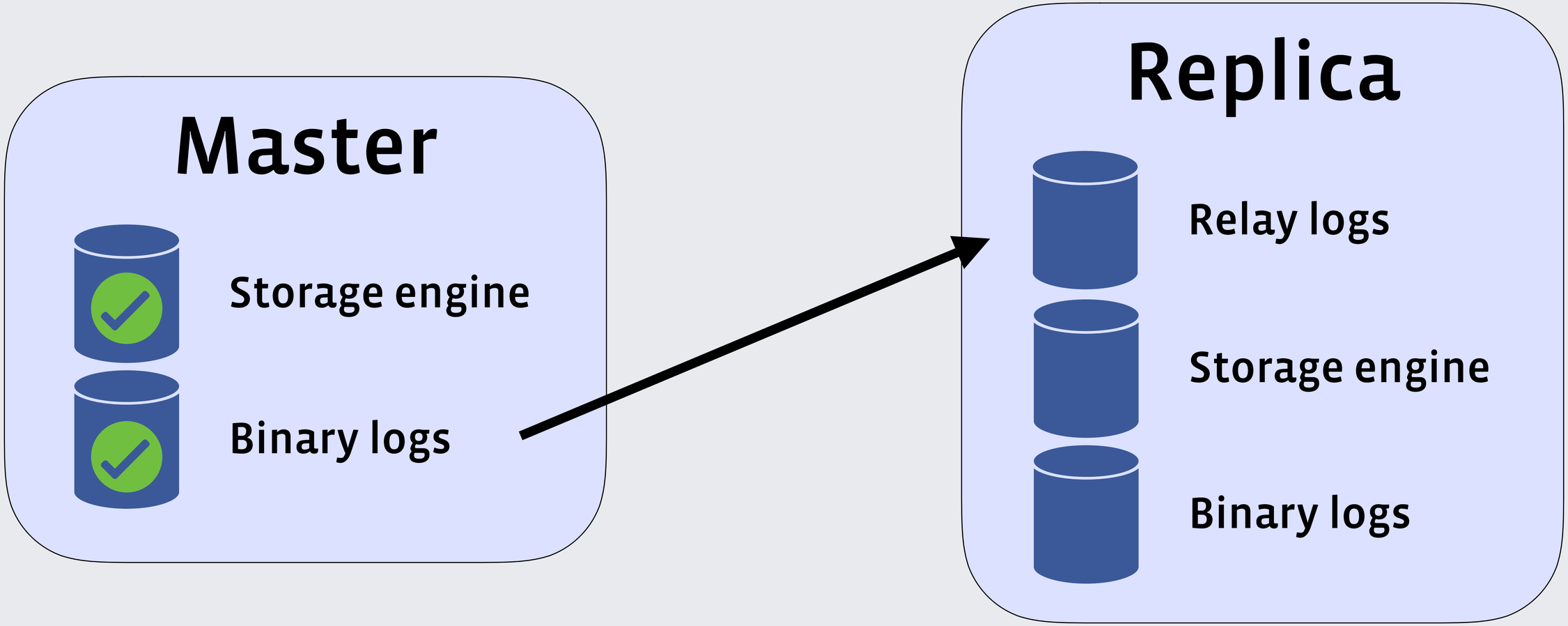
```
UPDATE table_a SET foo = "bar";
```



read/write

read (delayed)

```
UPDATE table_a SET foo = "bar";
```



read/write

read (delayed)

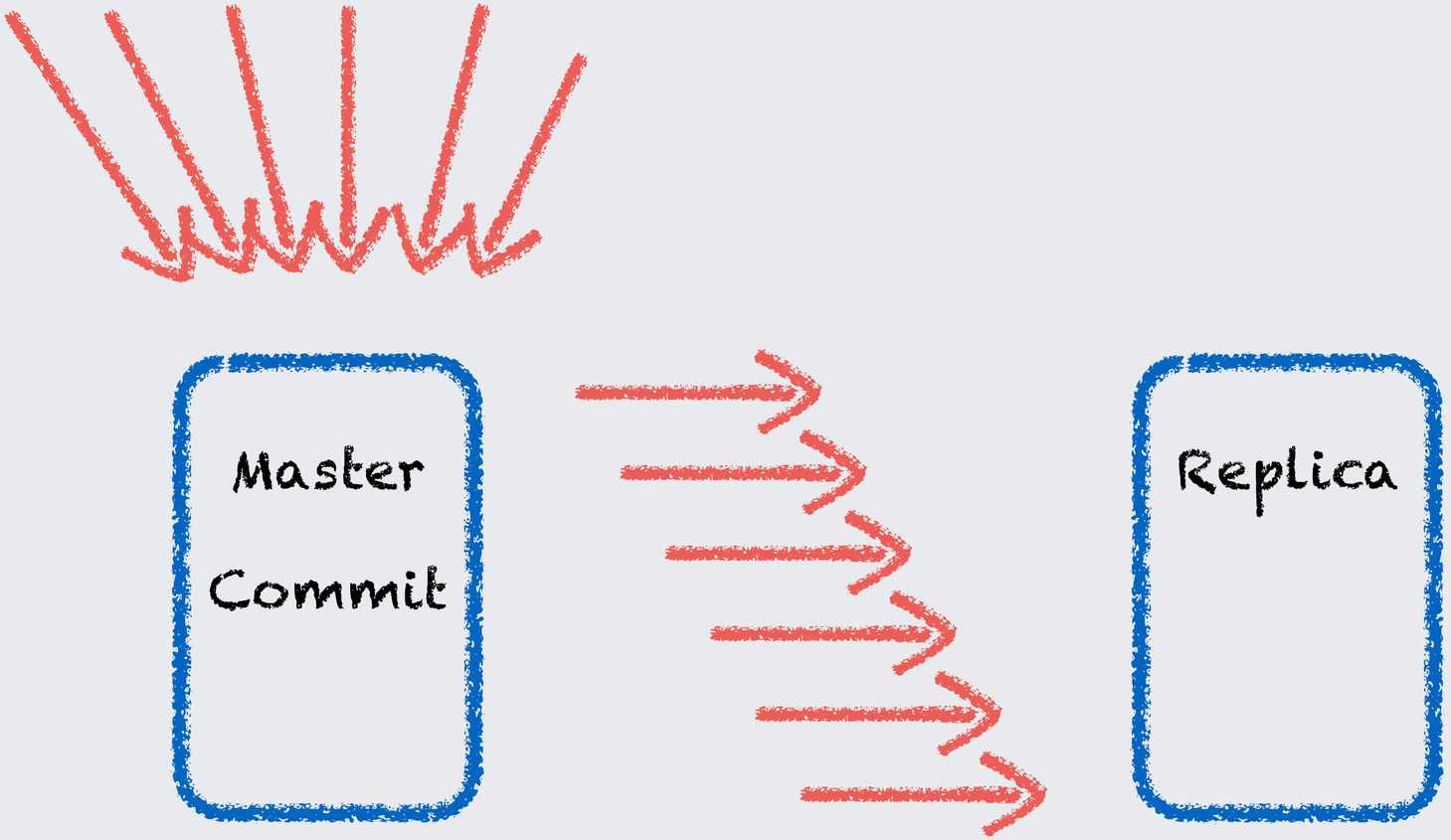
Promotion

Live Master Promotion

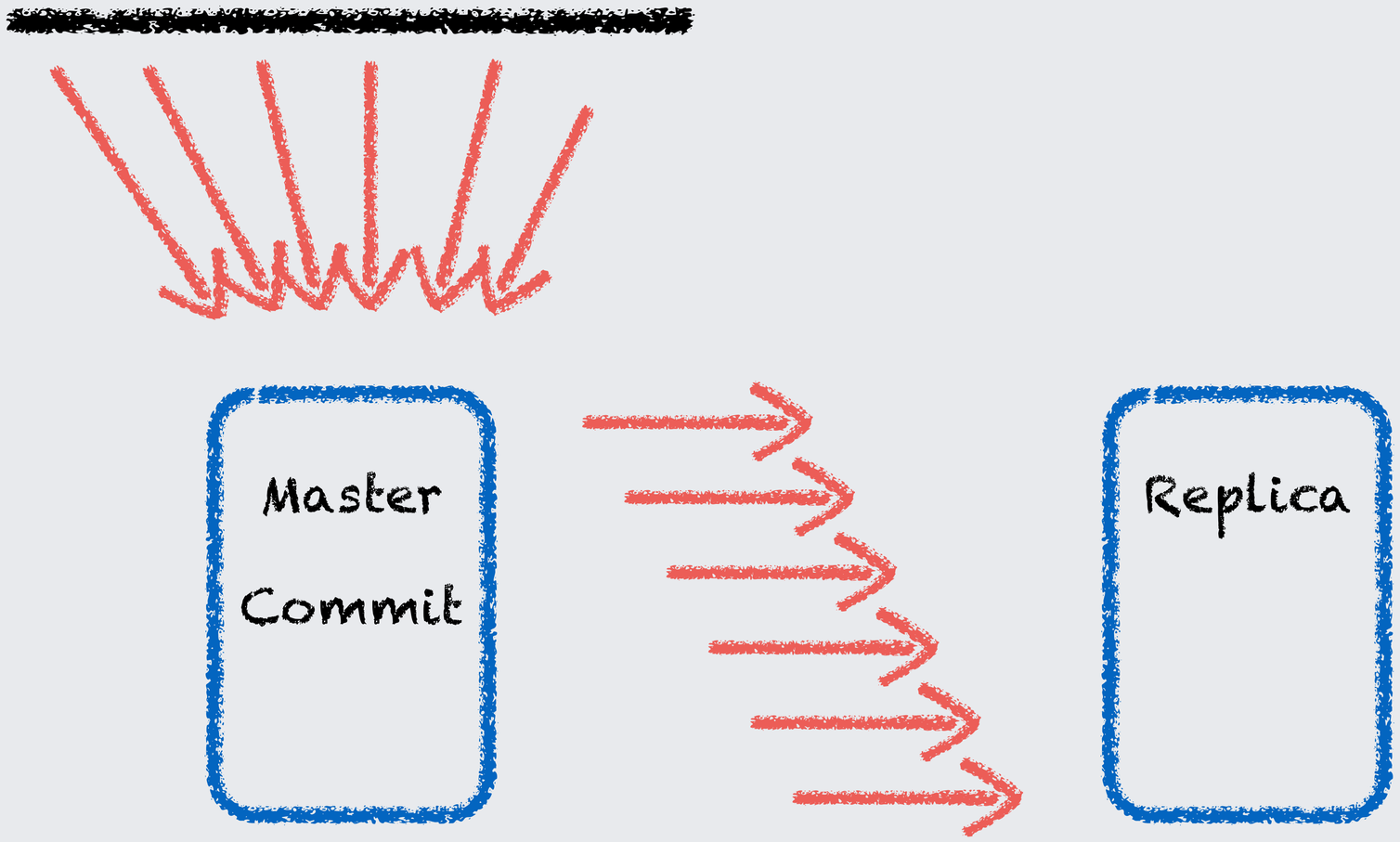
Live Master Promotion

Dead Master Promotion

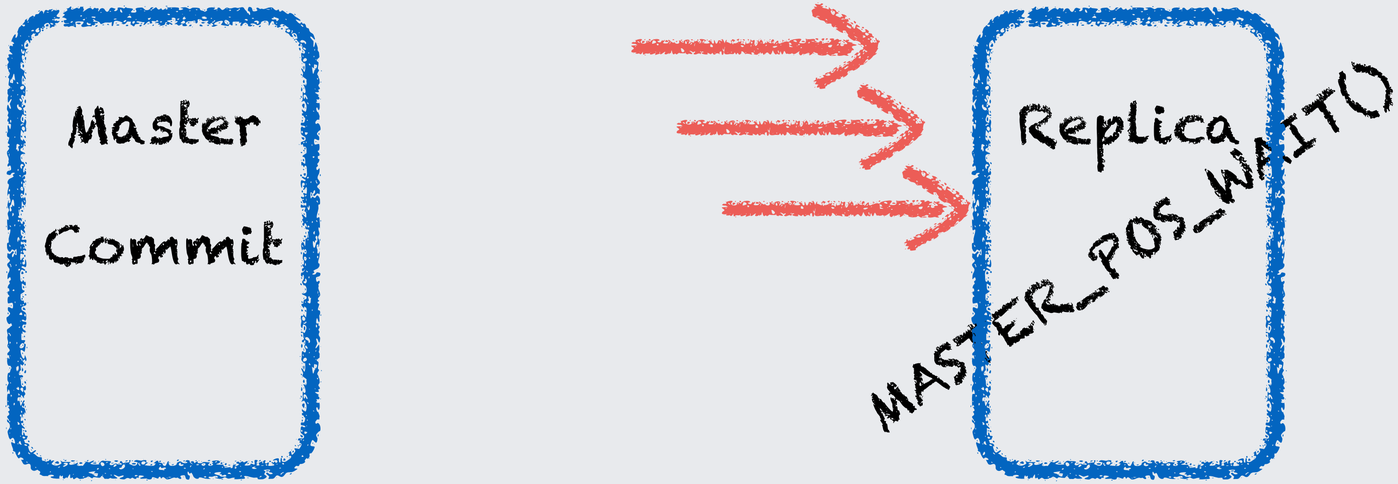
Live Master Promotion



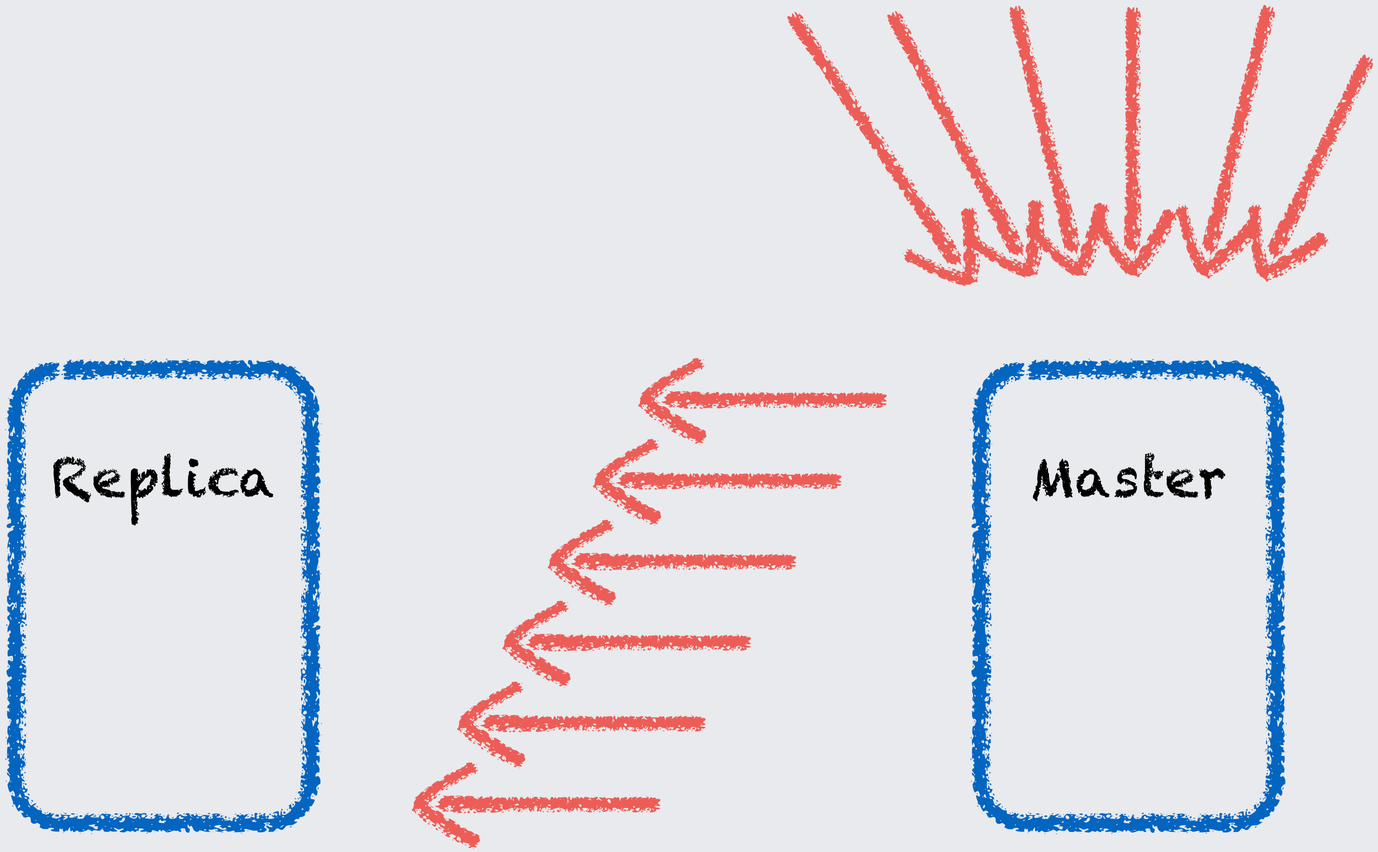
Live Master Promotion



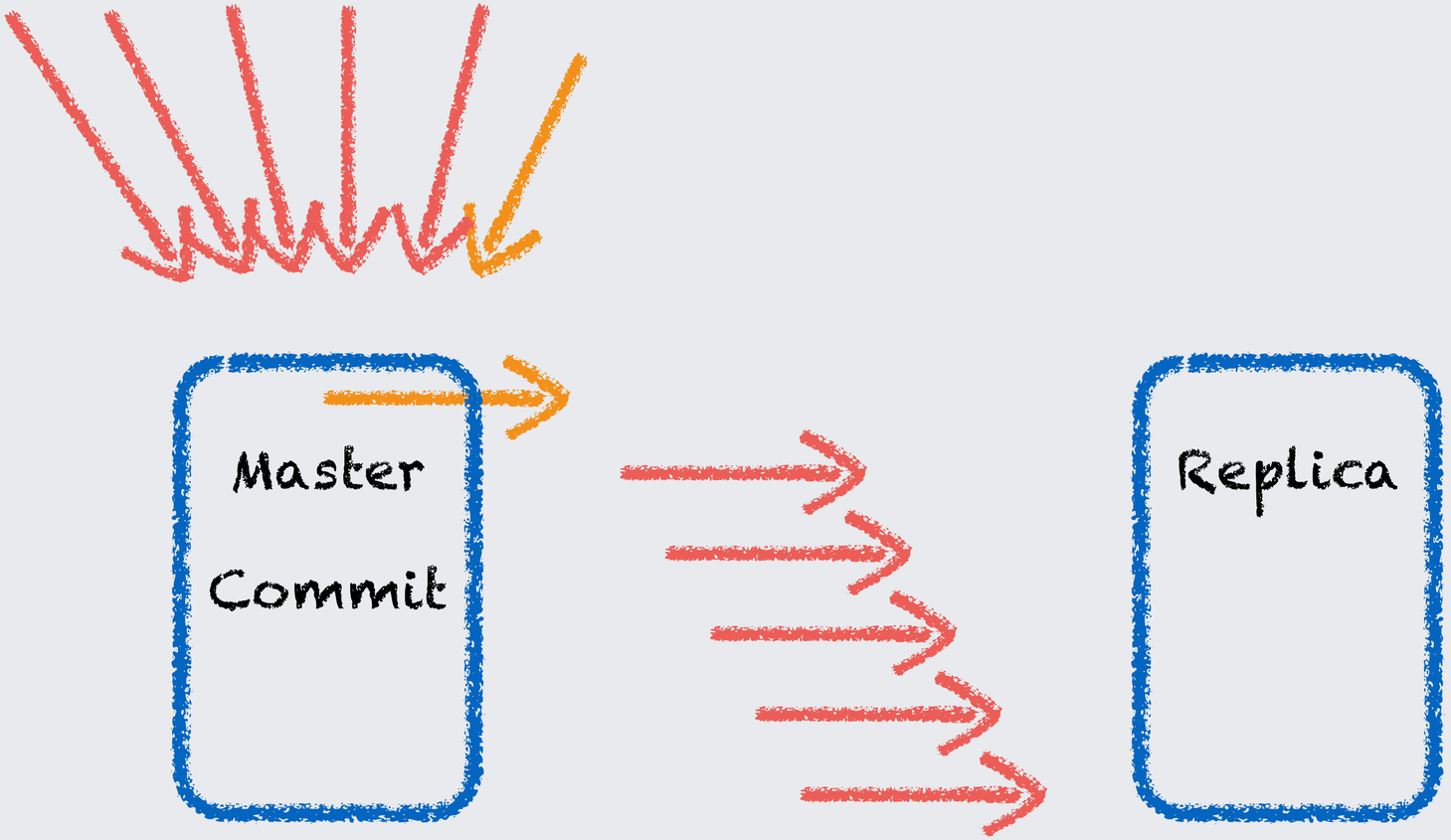
Live Master Promotion



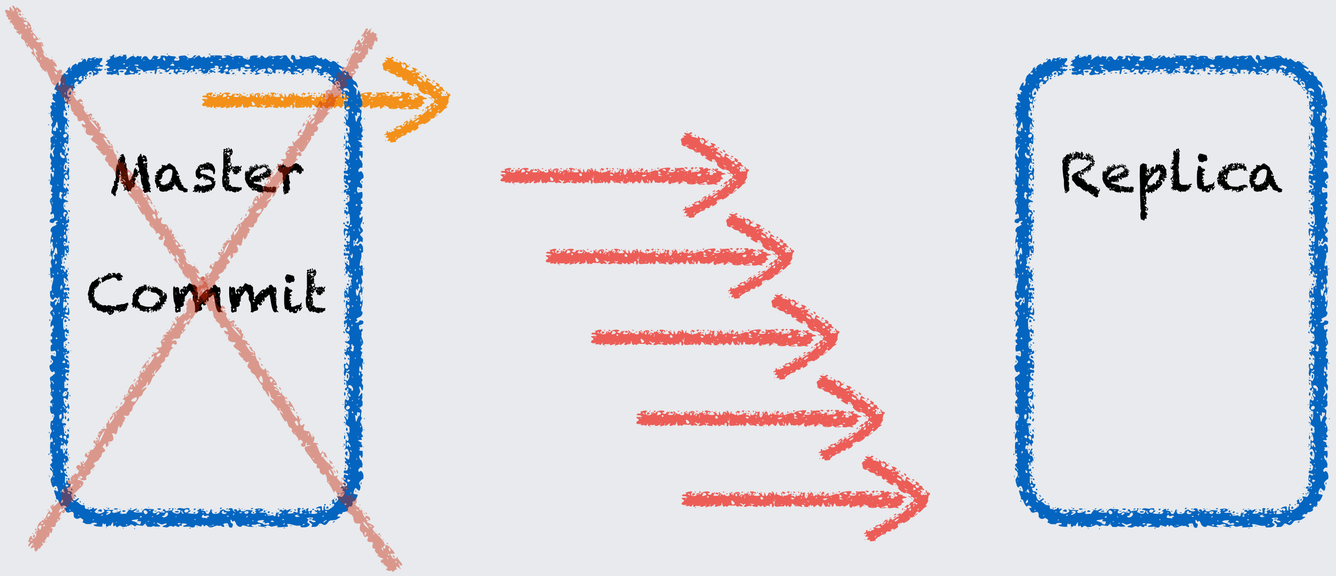
Live Master Promotion



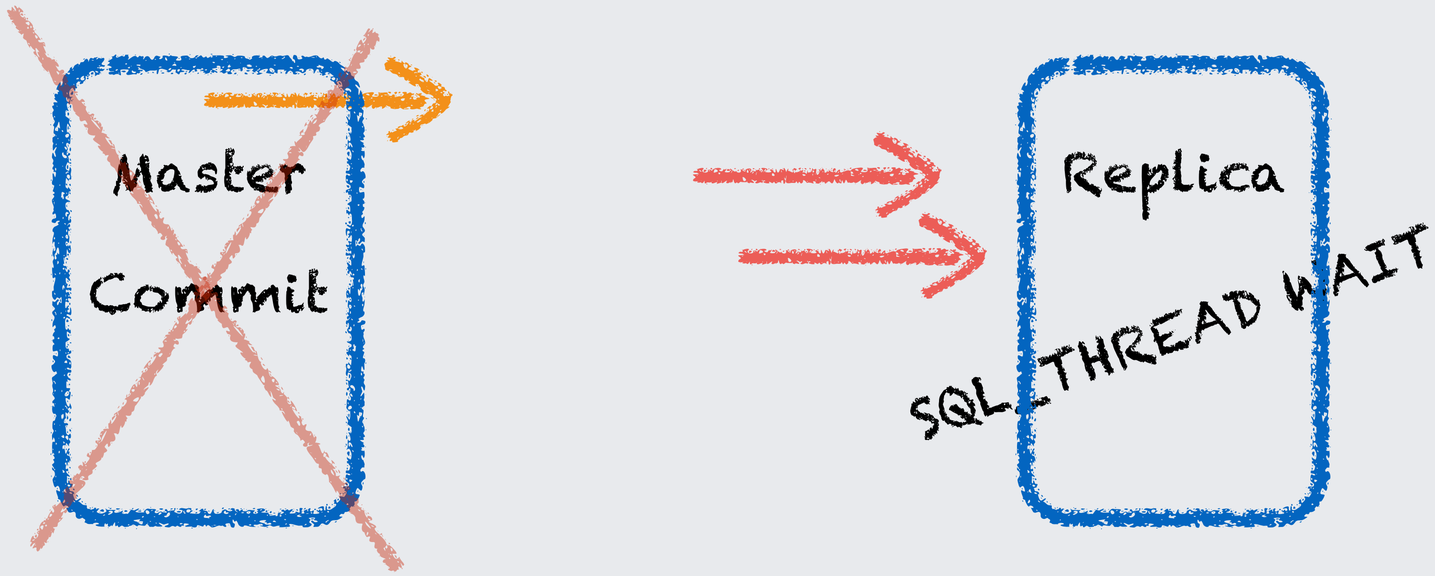
Dead Master Promotion



Dead Master Promotion

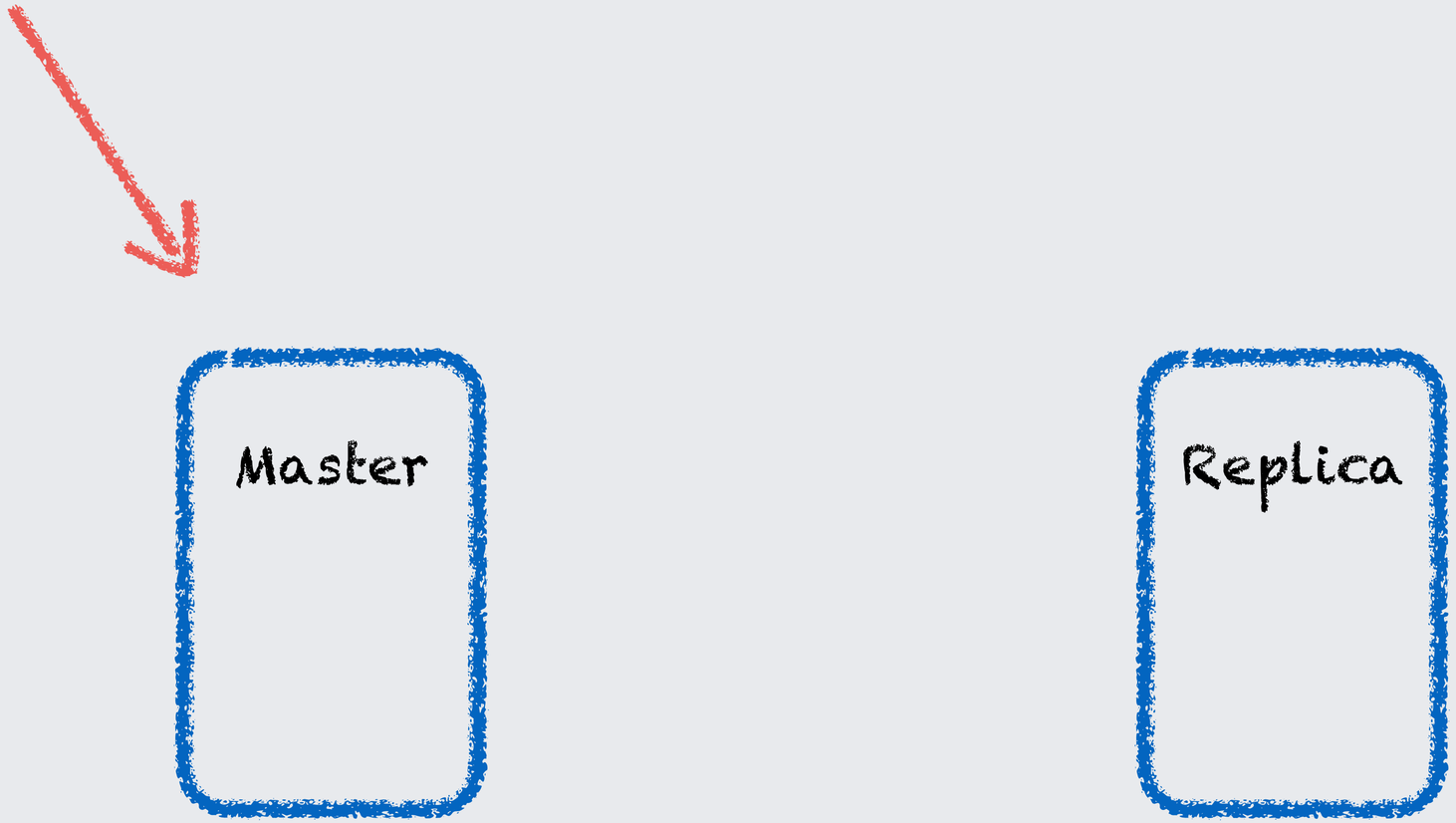


Dead Master Promotion

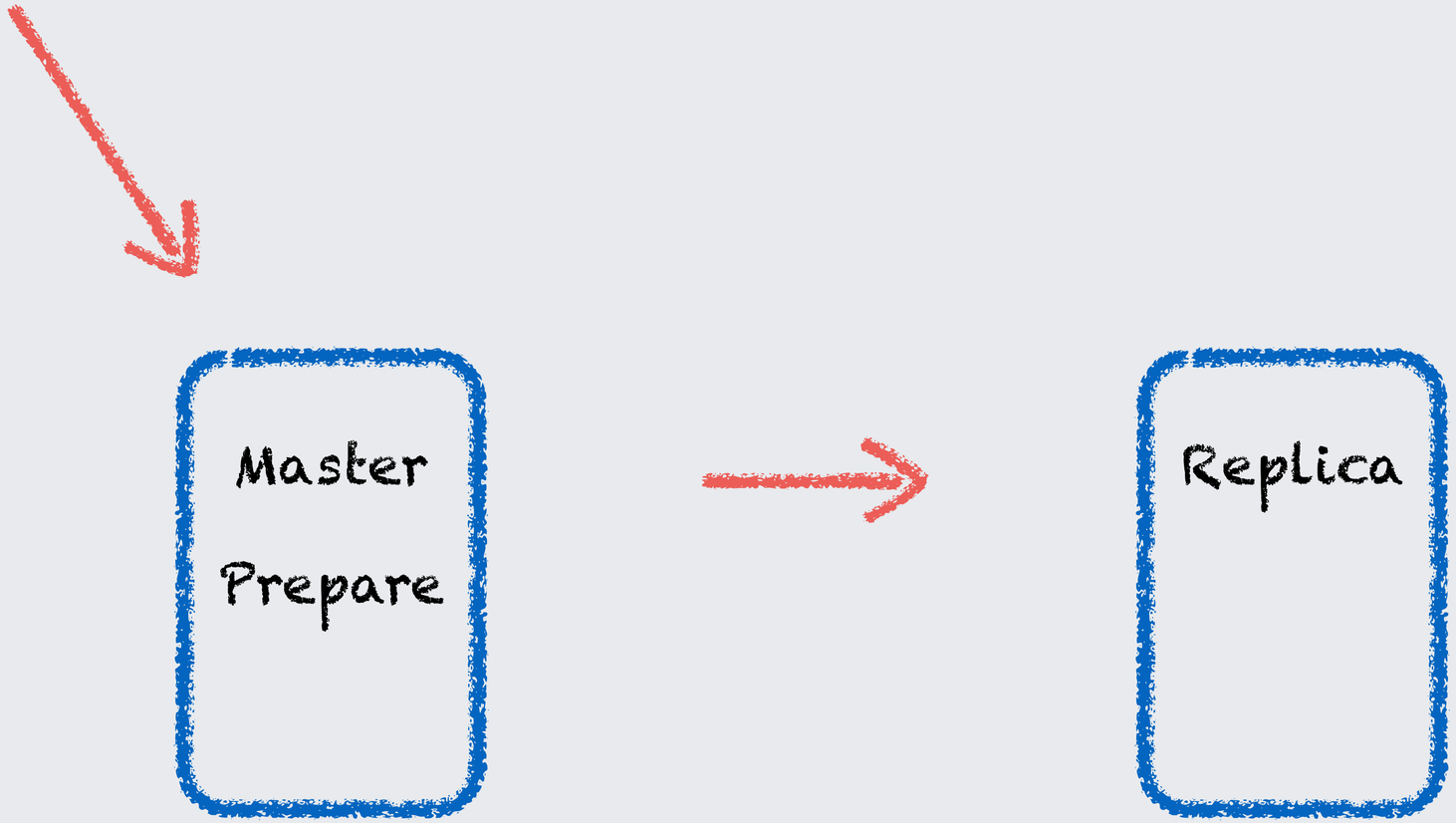


Synchronous Replication

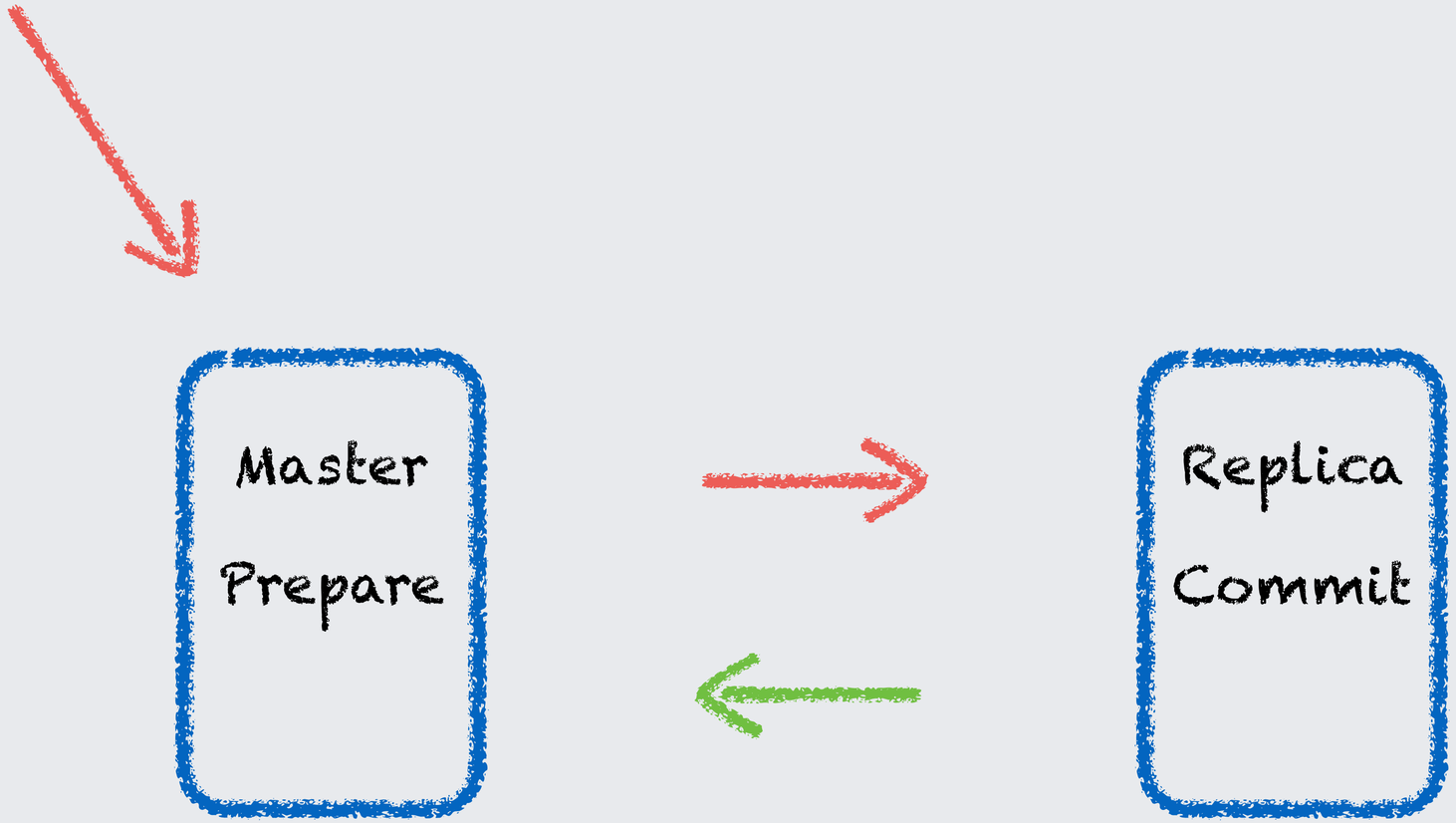
Synchronous Replication



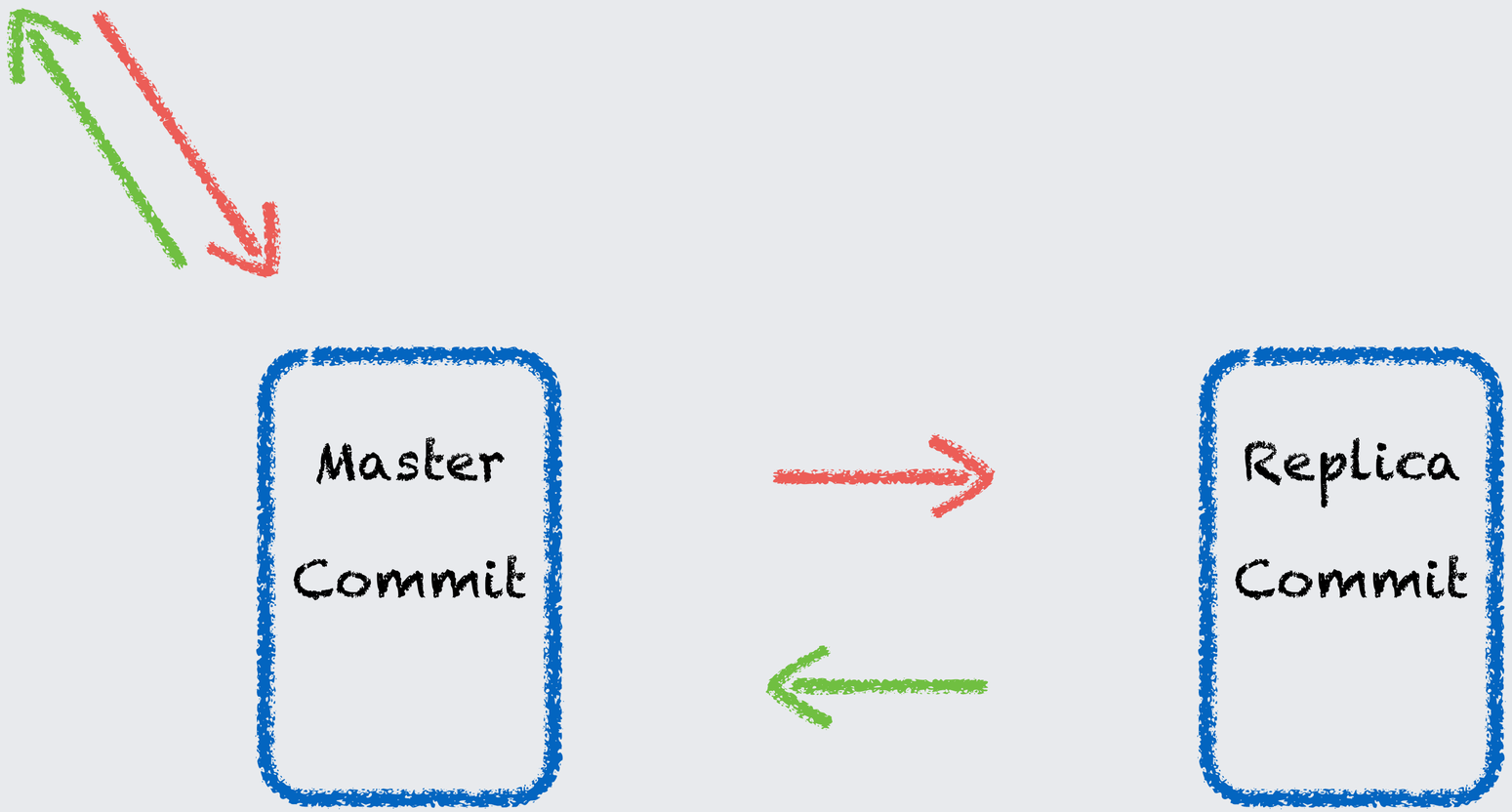
Synchronous Replication



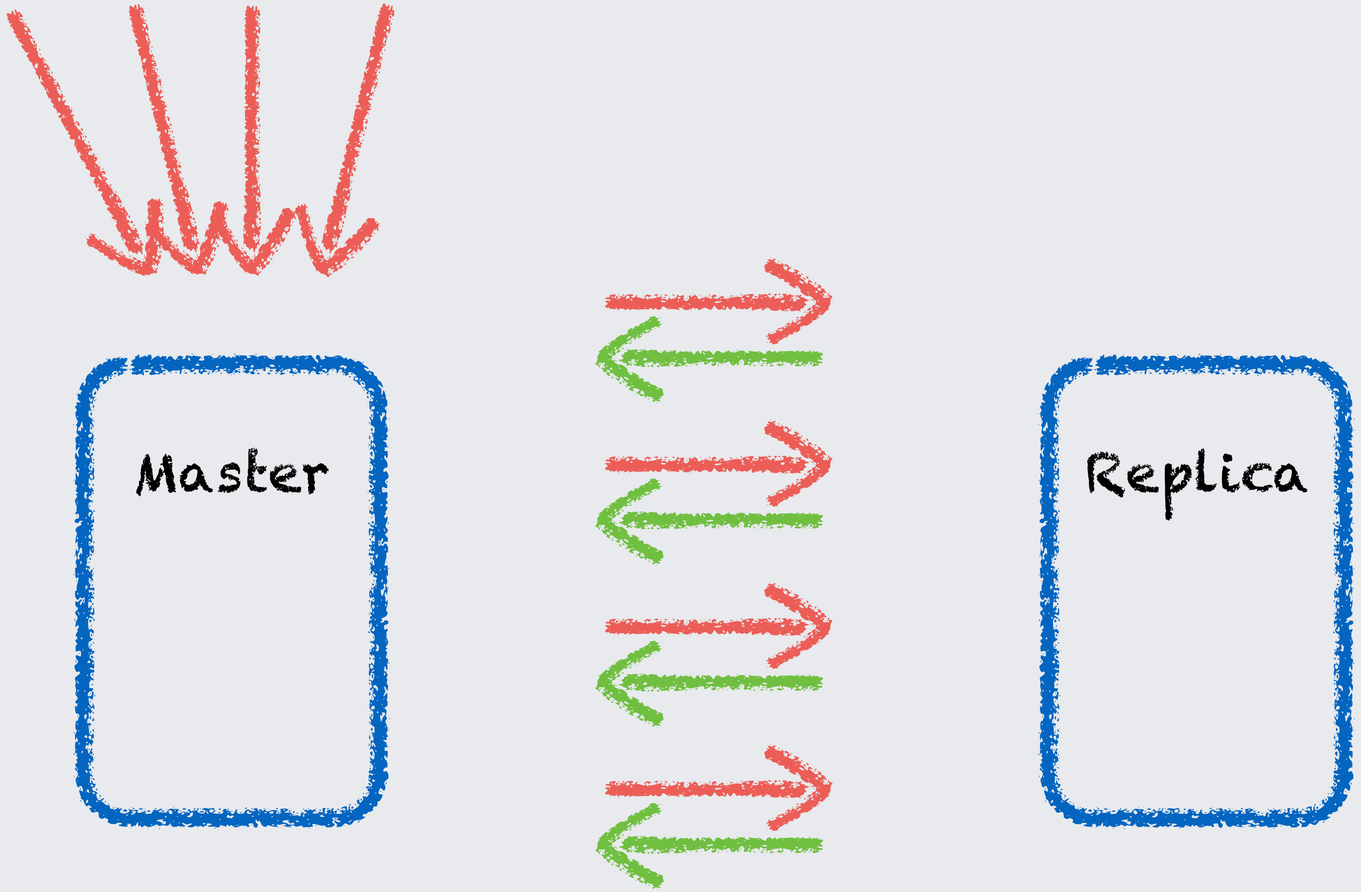
Synchronous Replication



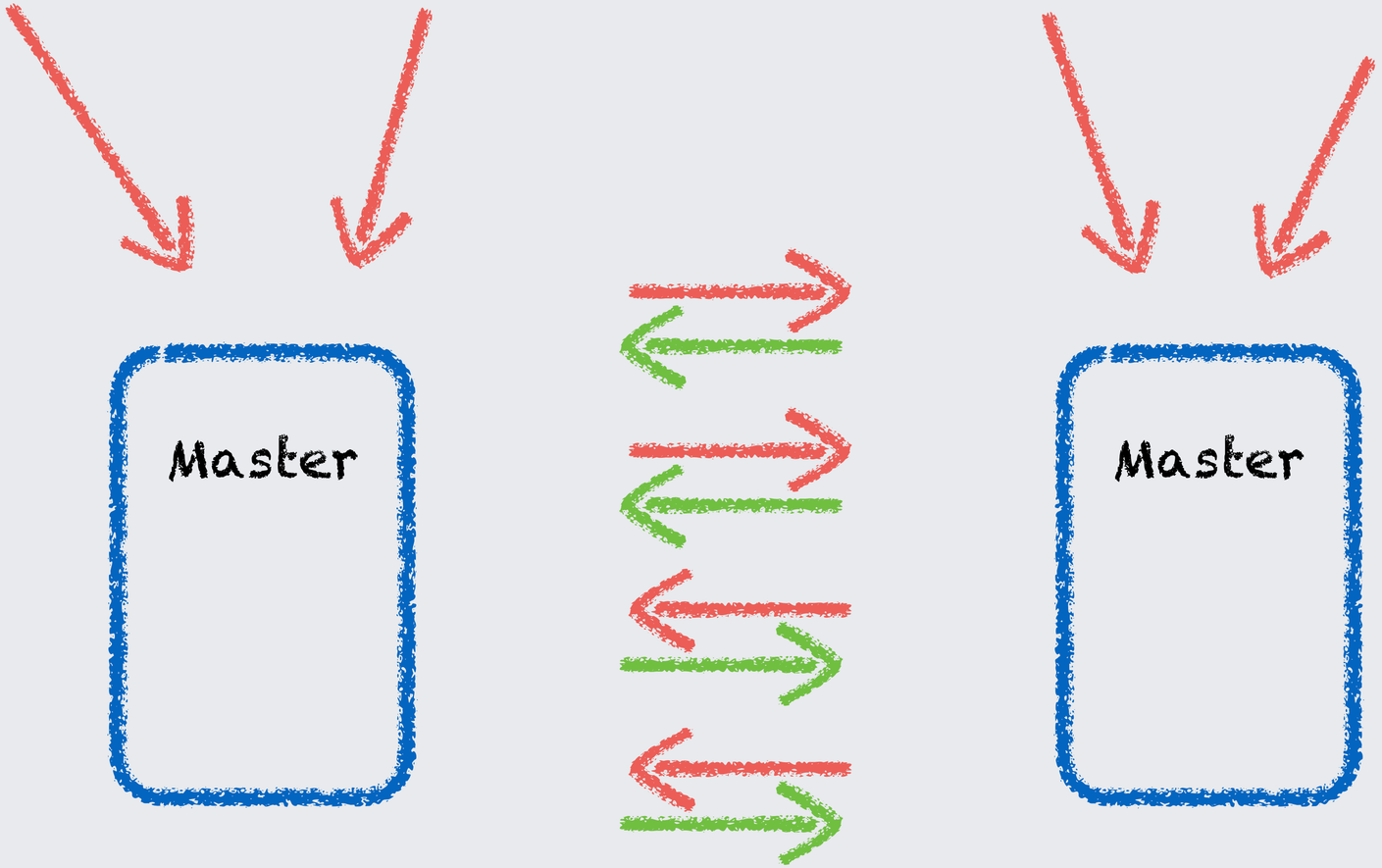
Synchronous Replication



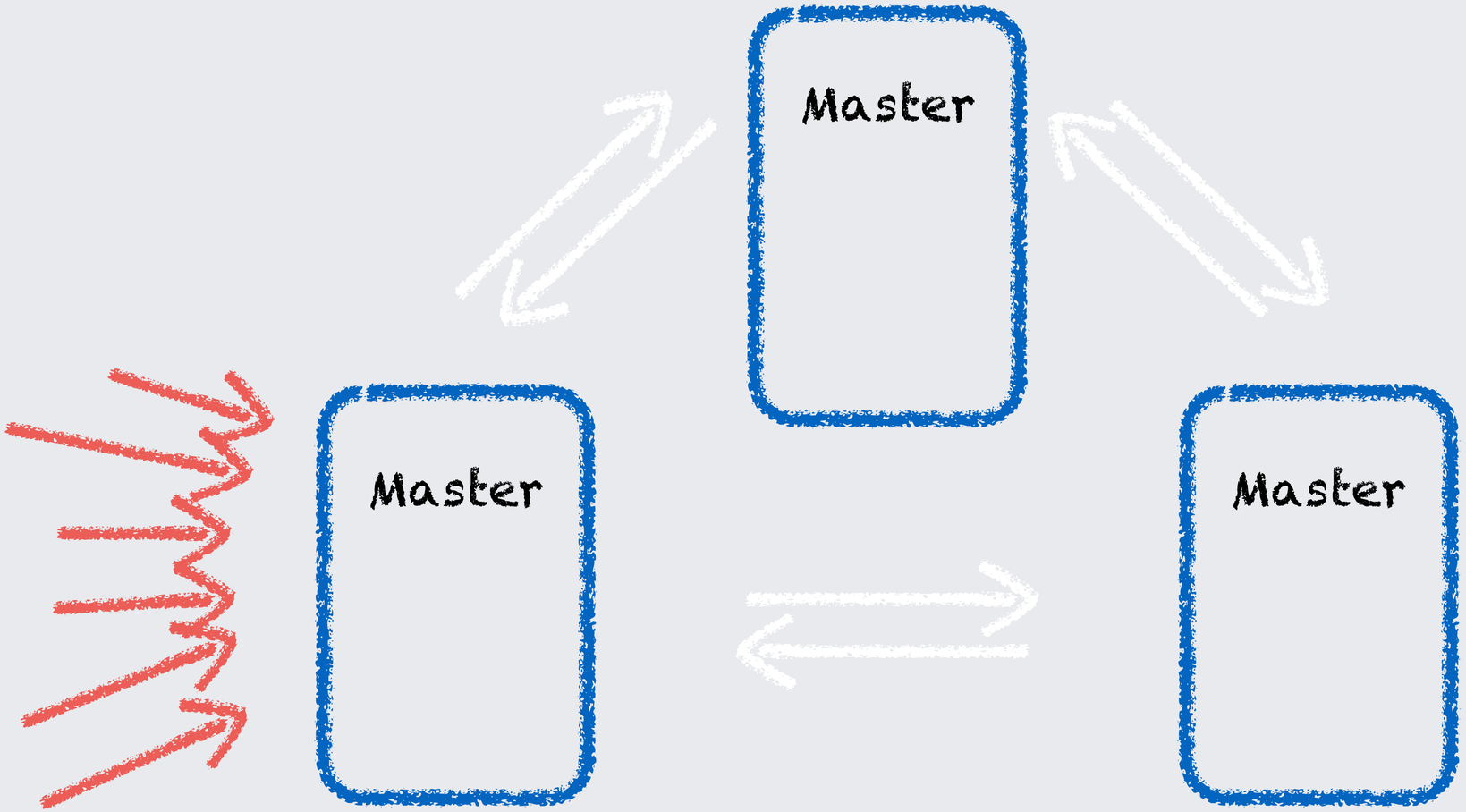
Live Master Promotion



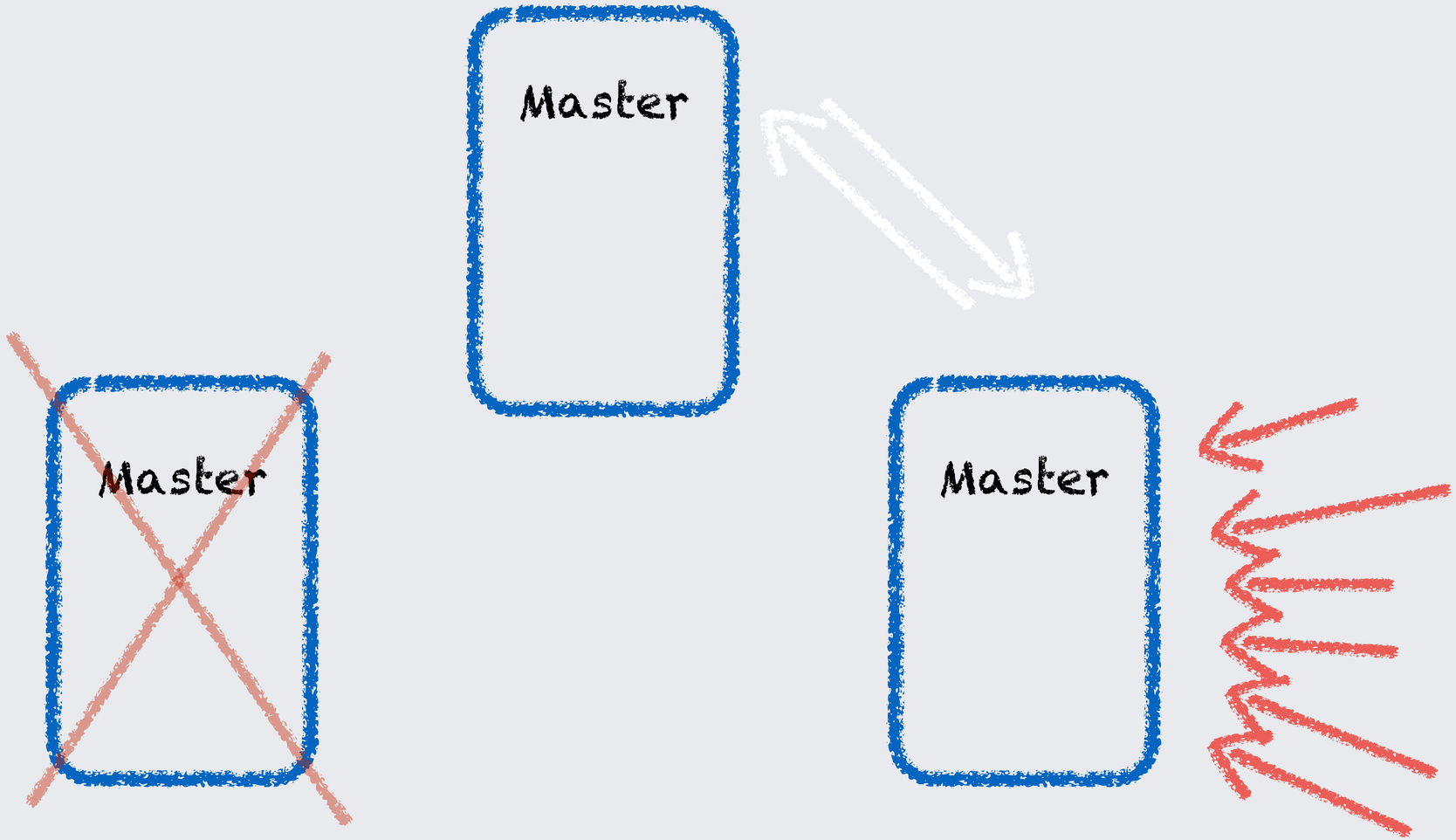
Live Master Promotion



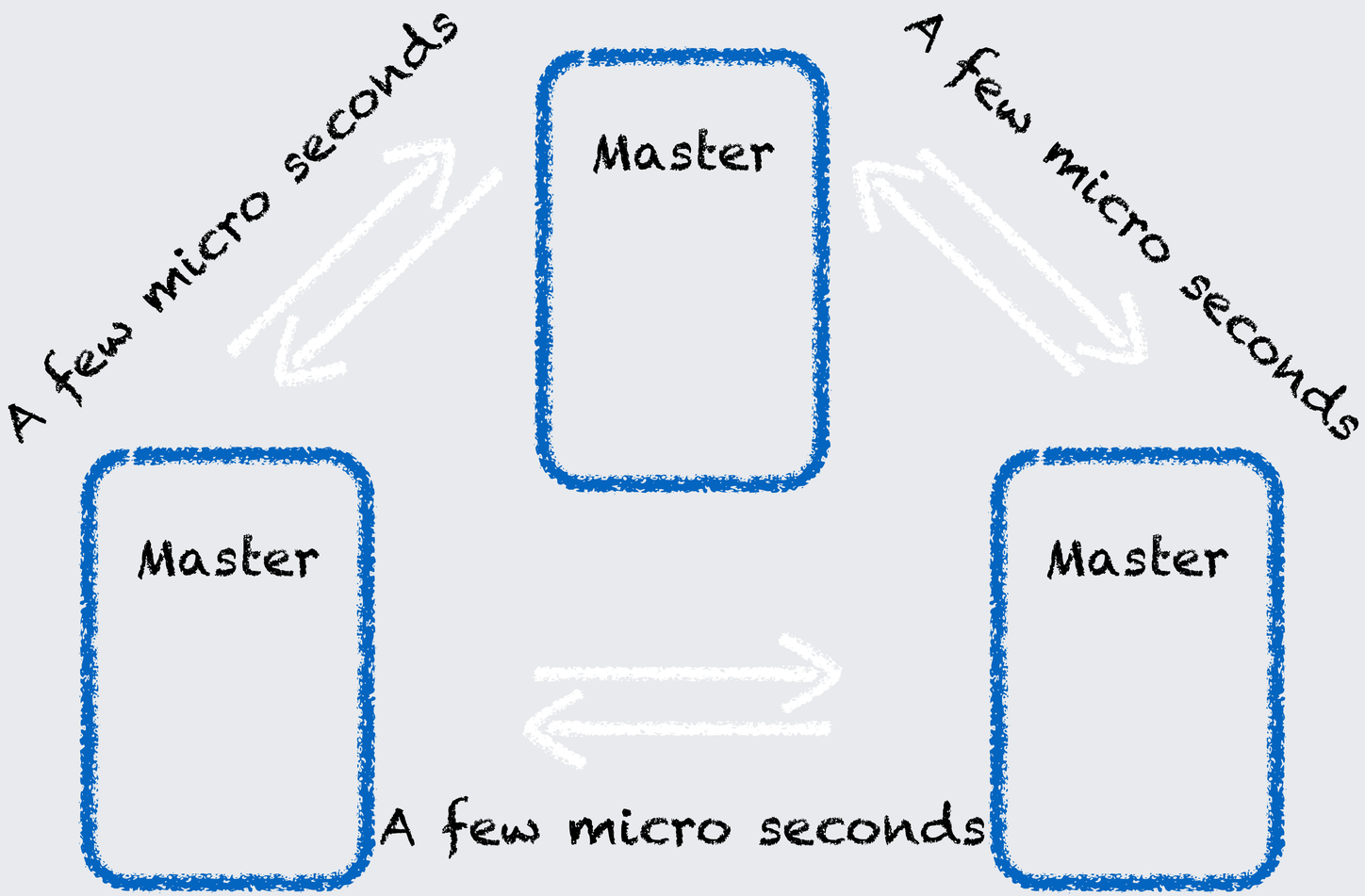
Dead Master Promotion



Dead Master Promotion

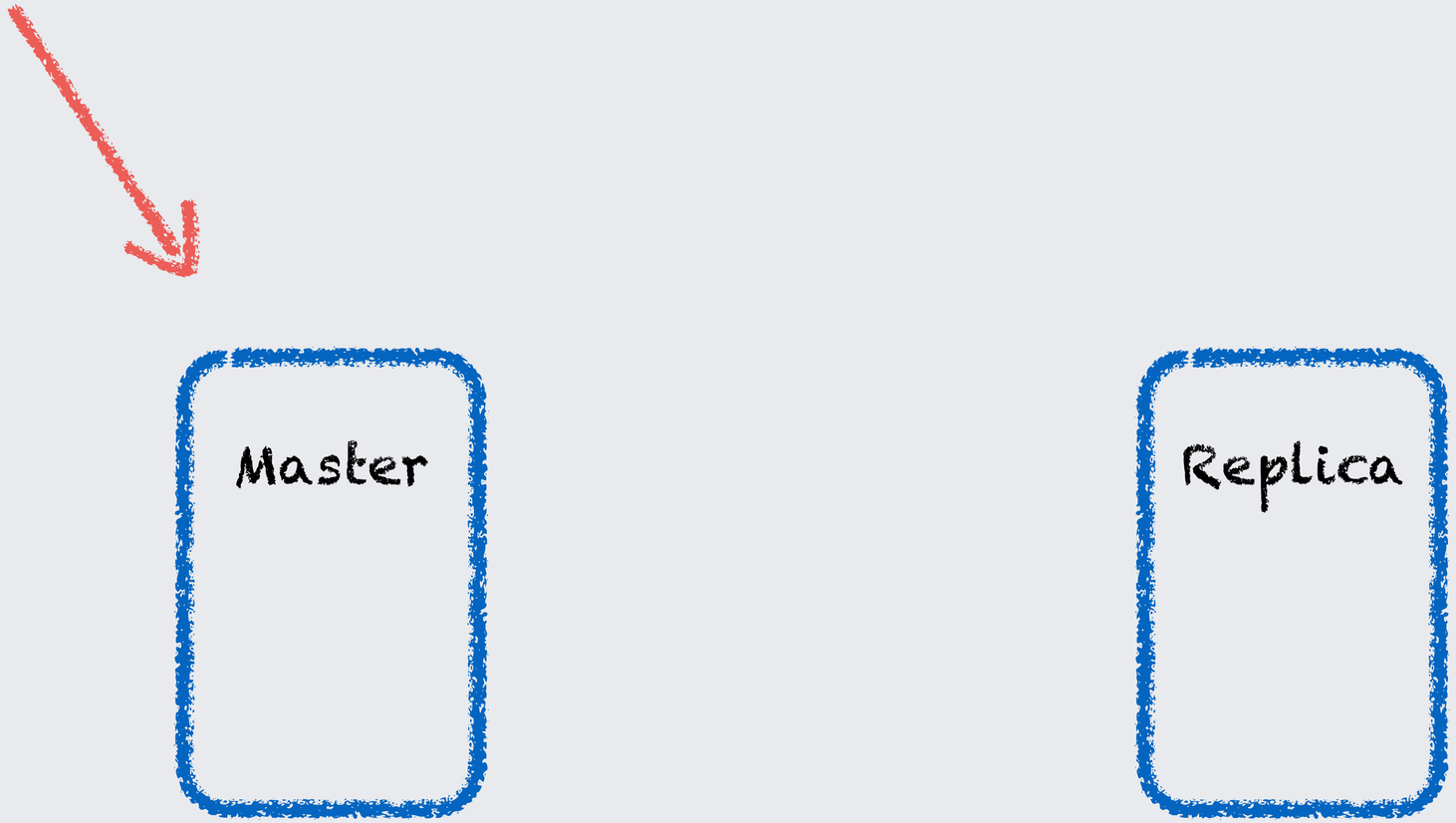


Synchronous Constraints

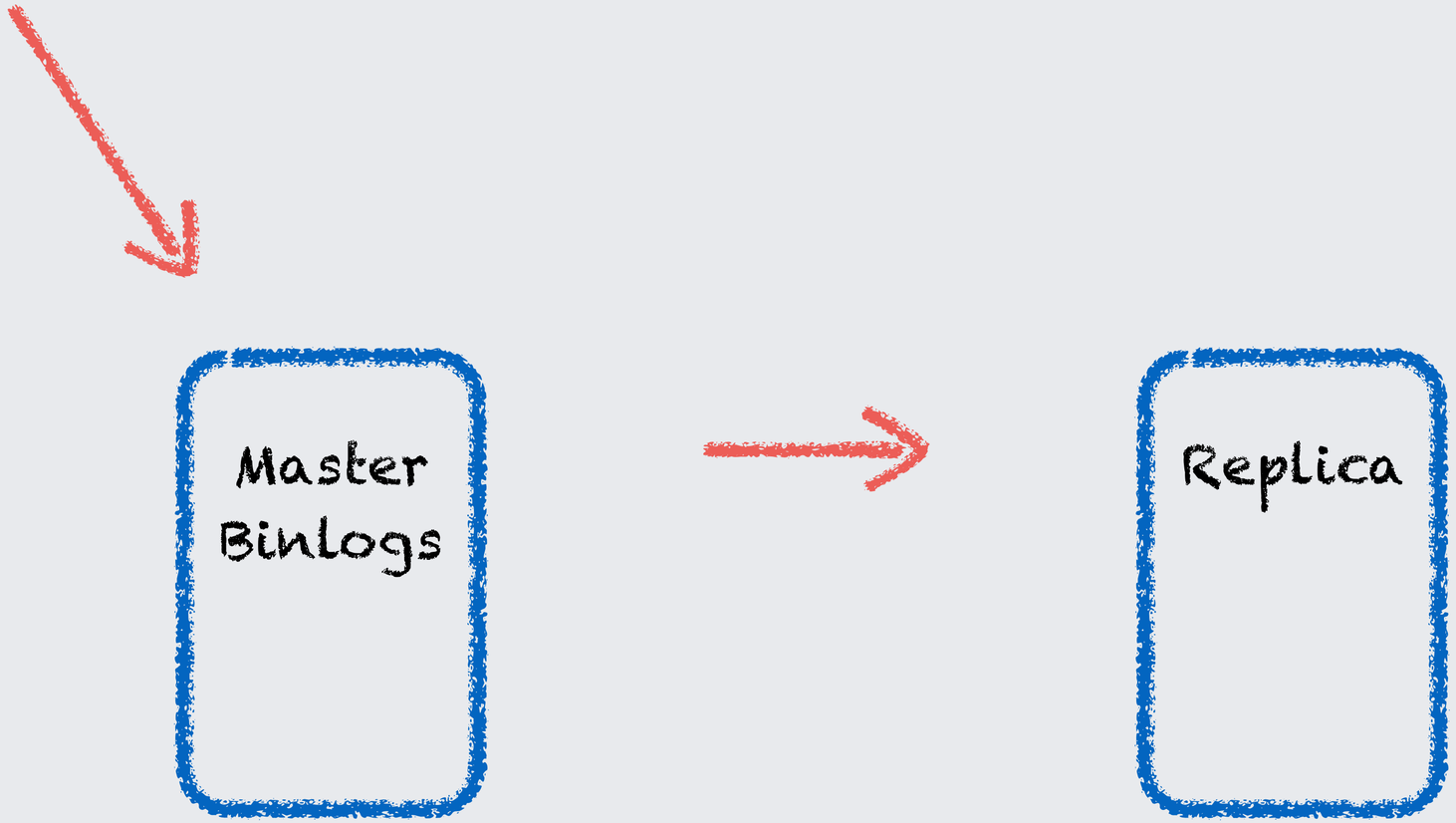


Semi-Synchronous Replication

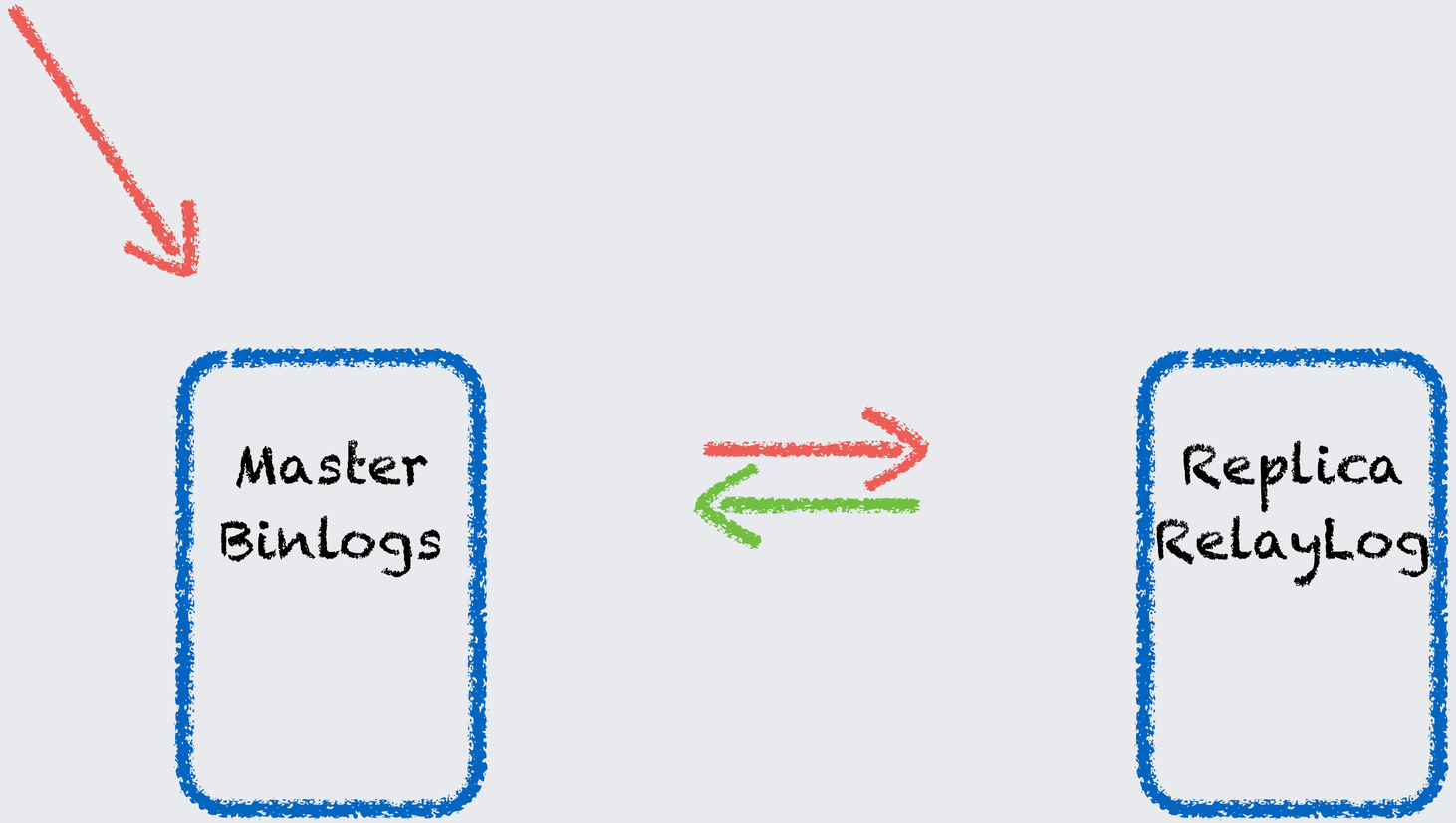
Semi-Synchronous Replication



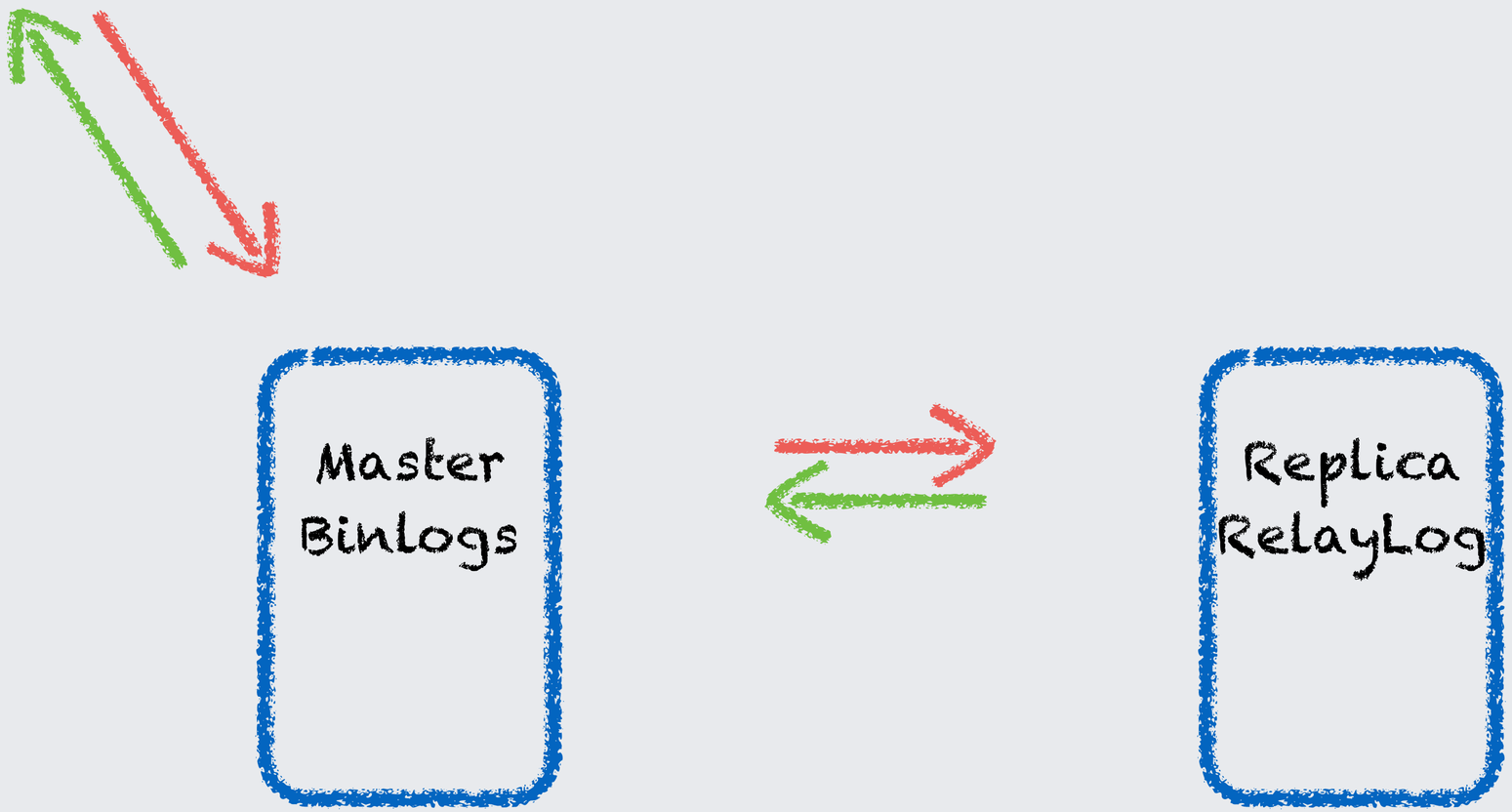
Semi-Synchronous Replication



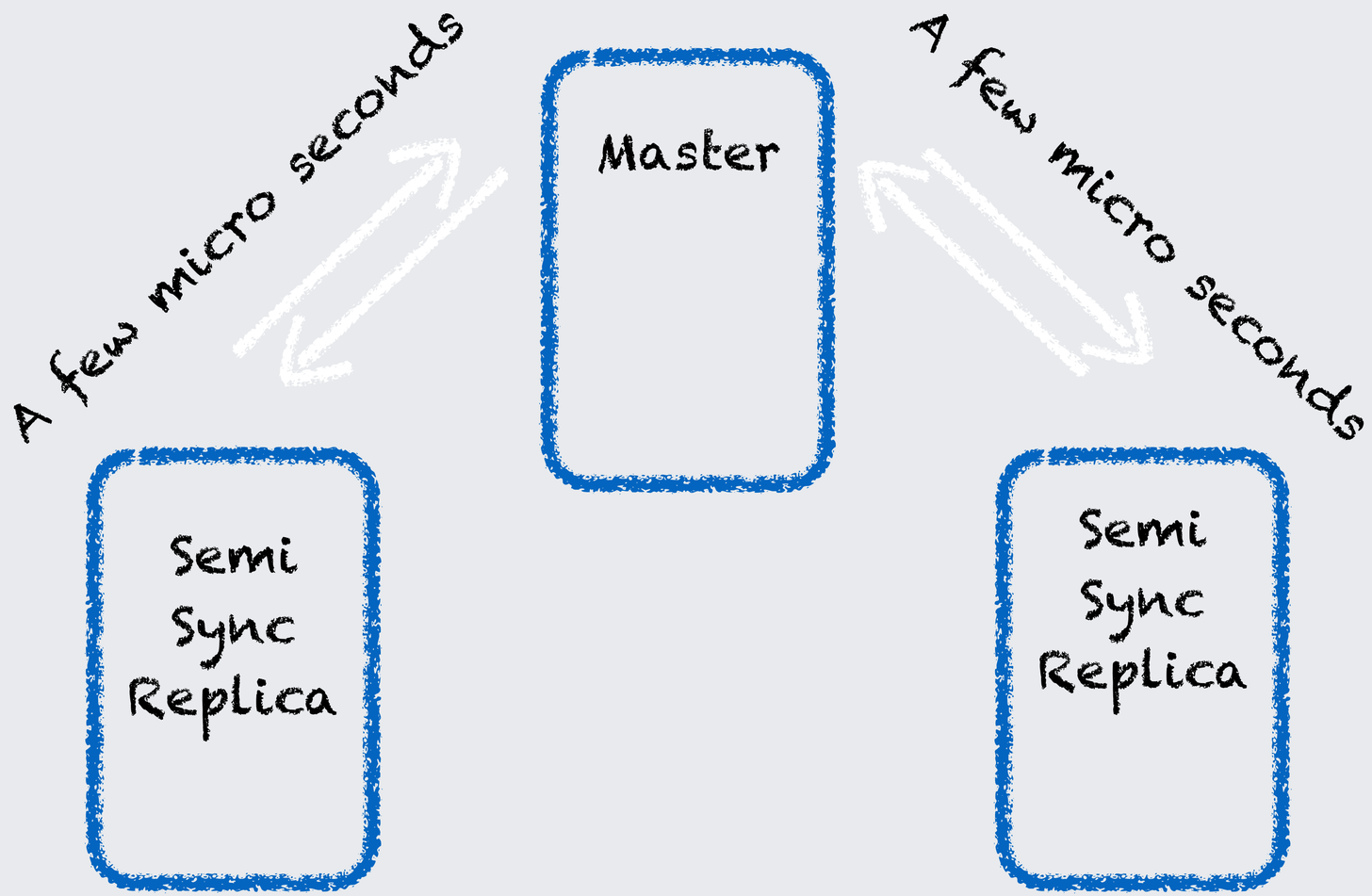
Semi-Synchronous Replication



Semi-Synchronous Replication



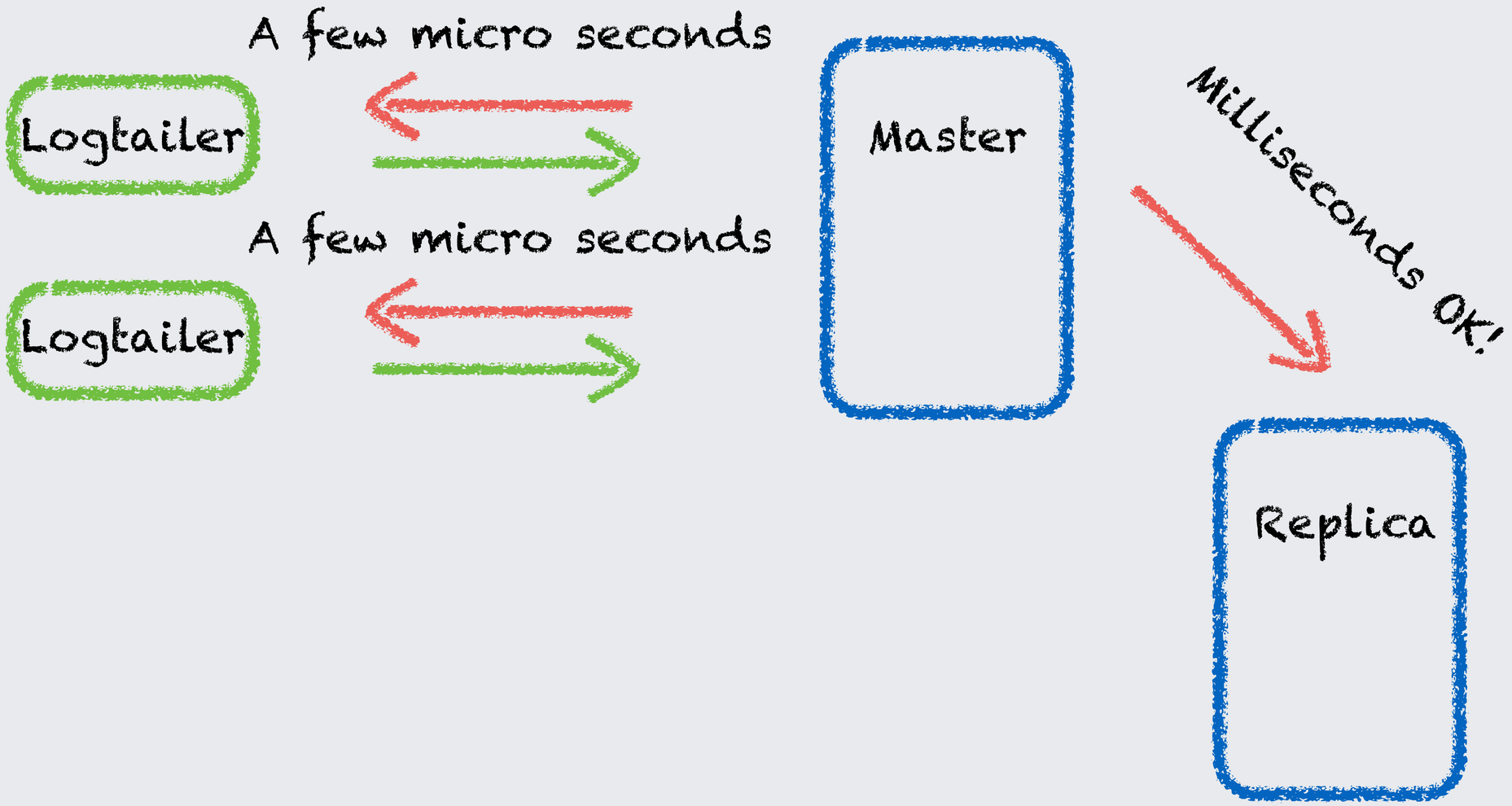
Semi-Synchronous Constraints



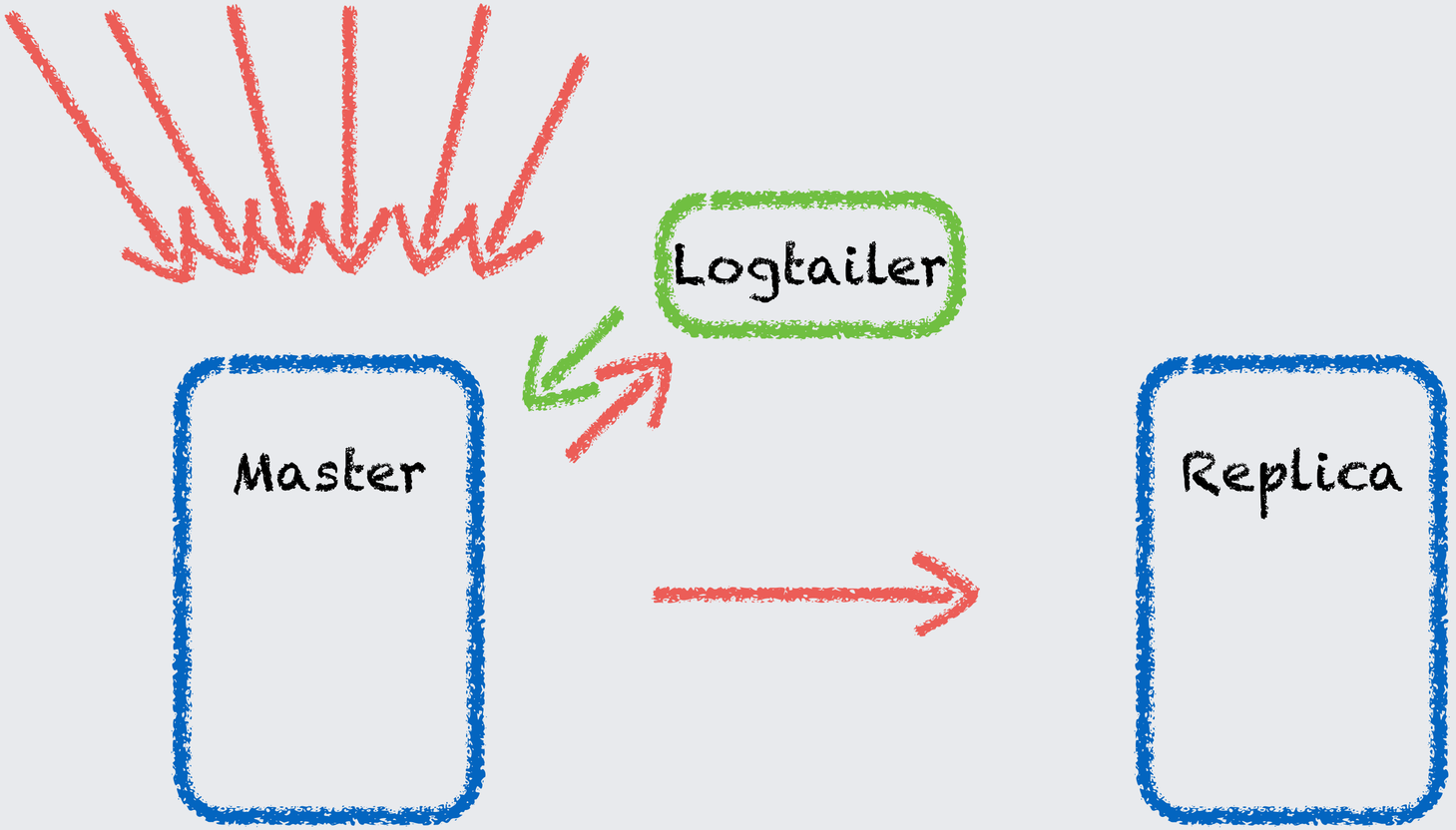
Semi-Synchronous Replication

Semi-Synchronous mysqlbinlog

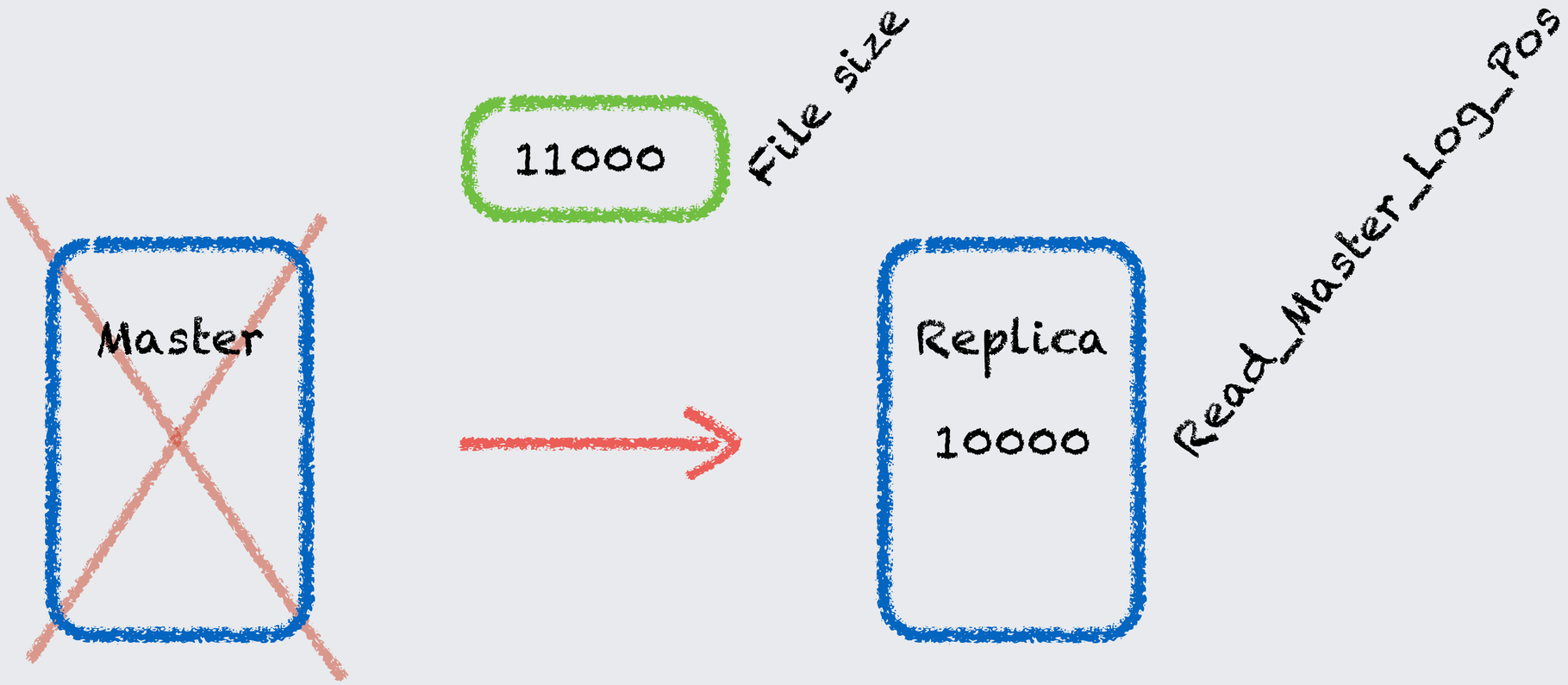
Semi-Synchronous Constraints



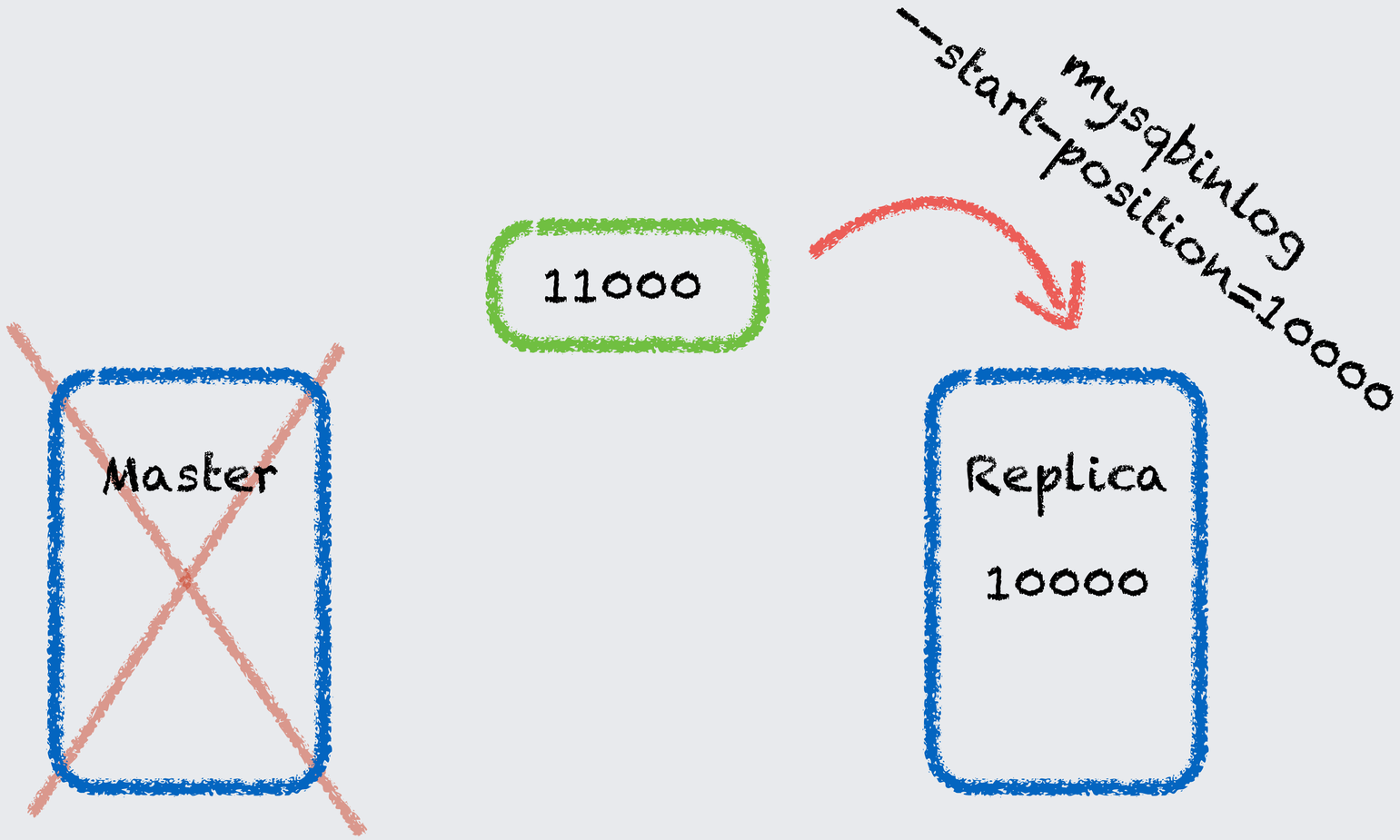
Dead Master Failover



Dead Master Failover



Dead Master Failover



Lossless semi-sync

Lossless Semi-Sync

=> Commit;
Binlog Prepare => No-op
InnoDB Prepare => Written to InnoDB for recovery
Binlog Commit => Written to binlog
InnoDB Commit => Visible from other clients
<= OK;

Lossless Semi-Sync

=> Commit;
Binlog Prepare => No-op
InnoDB Prepare => Written to InnoDB for recovery
Binlog Commit => Written to binlog
InnoDB Commit => Visible from other clients
<= OK;

=> Commit;
Binlog Prepare
InnoDB Prepare
Binlog Commit
InnoDB Commit
Wait for Semi-Sync Ack
<= OK;

Lossless Semi-Sync

=> Commit;
Binlog Prepare => No-op
InnoDB Prepare => Written to InnoDB for recovery
Binlog Commit => Written to binlog
InnoDB Commit => Visible from other clients
<= OK;

=> Commit;
Binlog Prepare
InnoDB Prepare
Binlog Commit
InnoDB Commit

Crash!

Wait for Semi-Sync Ack
<= OK;

Lossless Semi-Sync

(MySQL 5.7)

=> Commit;
Binlog Prepare => No-op
InnoDB Prepare => Written to InnoDB for recovery
Binlog Commit => Written to binlog
InnoDB Commit => Visible from other clients
<= OK;

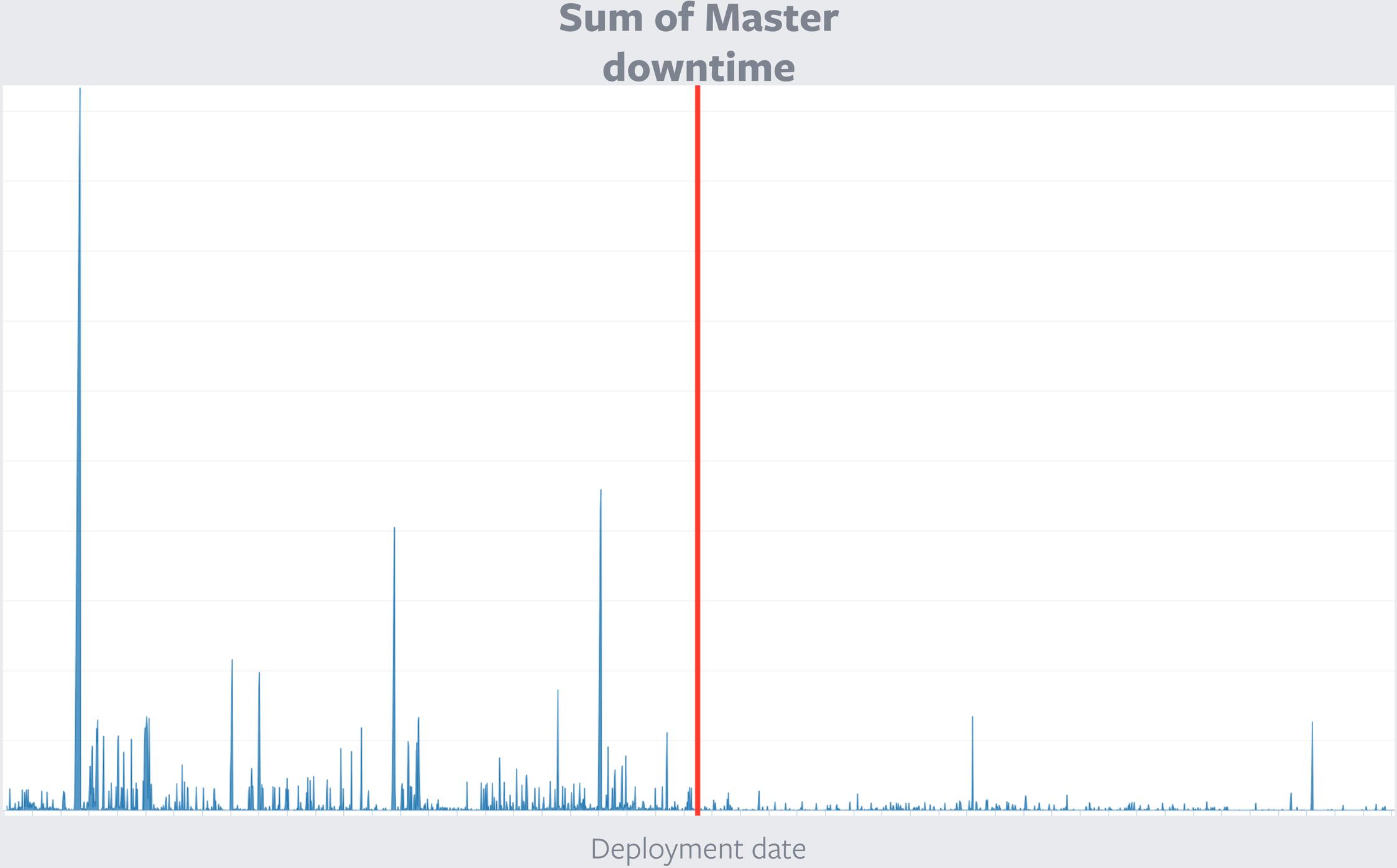
=> Commit;
Binlog Prepare
InnoDB Prepare
Binlog Commit
InnoDB Commit
Wait for Semi-Sync Ack
<= OK;

Crash!

=> Commit;
InnoDB Prepare
Binlog Prepare
Binlog Commit
Wait for Semi-Sync Ack
InnoDB Commit
<= OK;

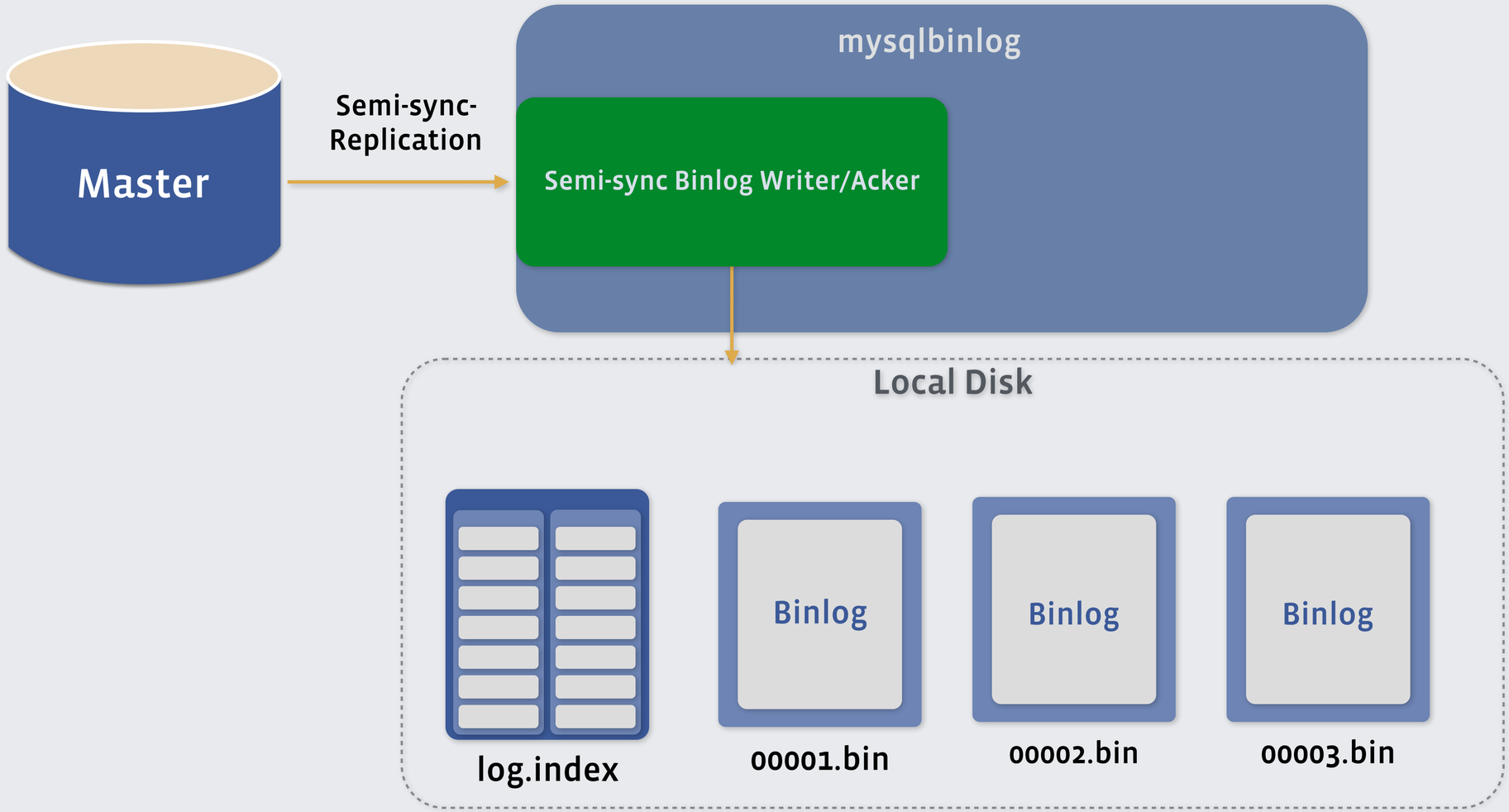
Rollout

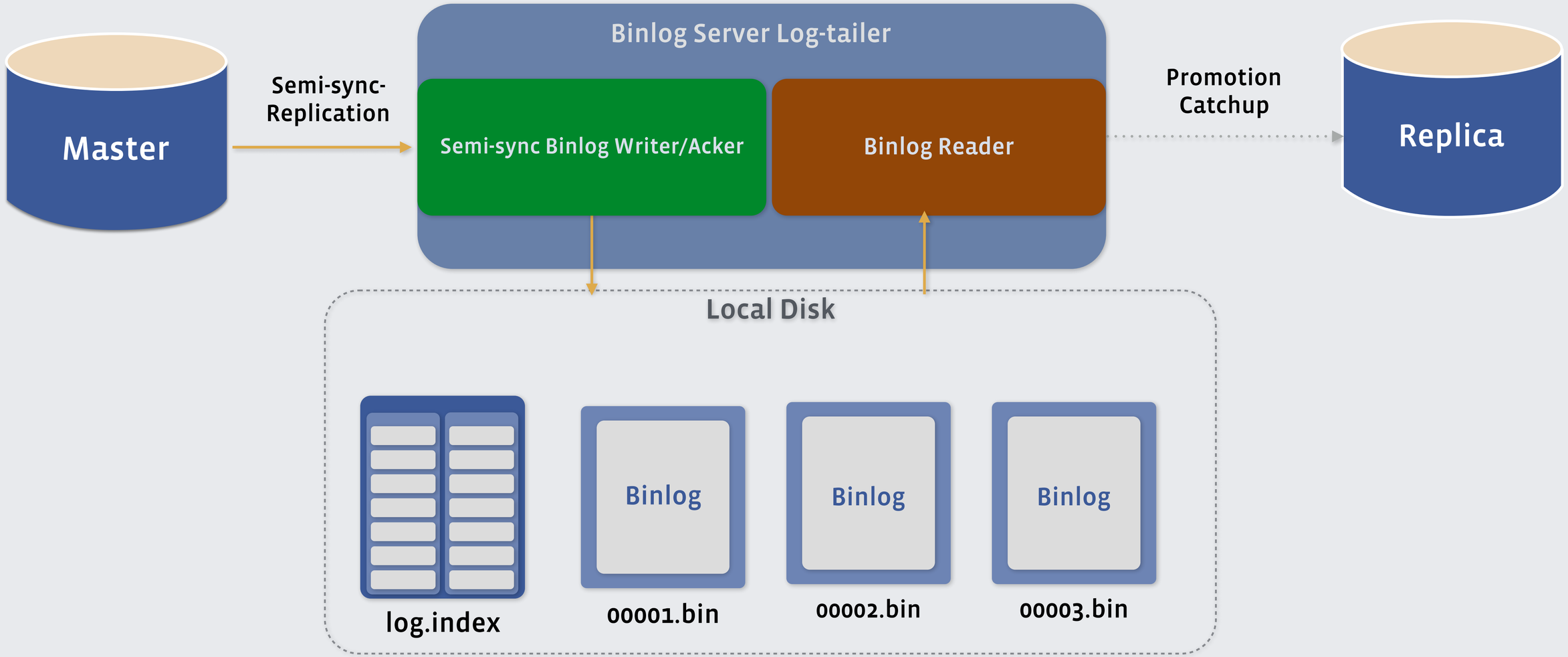
https://www.youtube.com/watch?v=sl-PACHY2zE

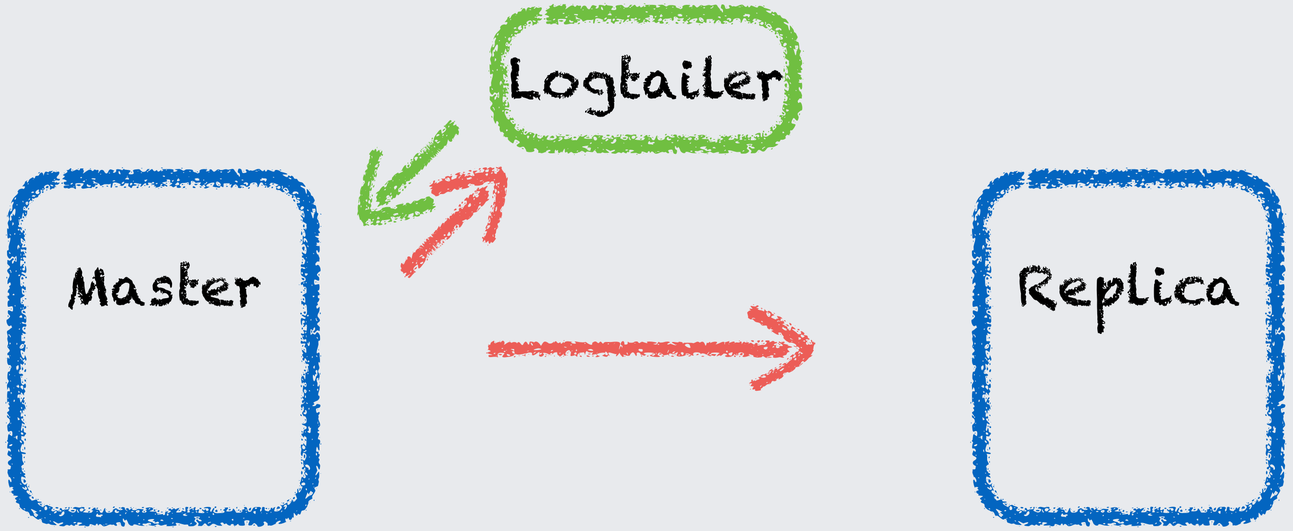

Sum of Master downtime
Deployment date

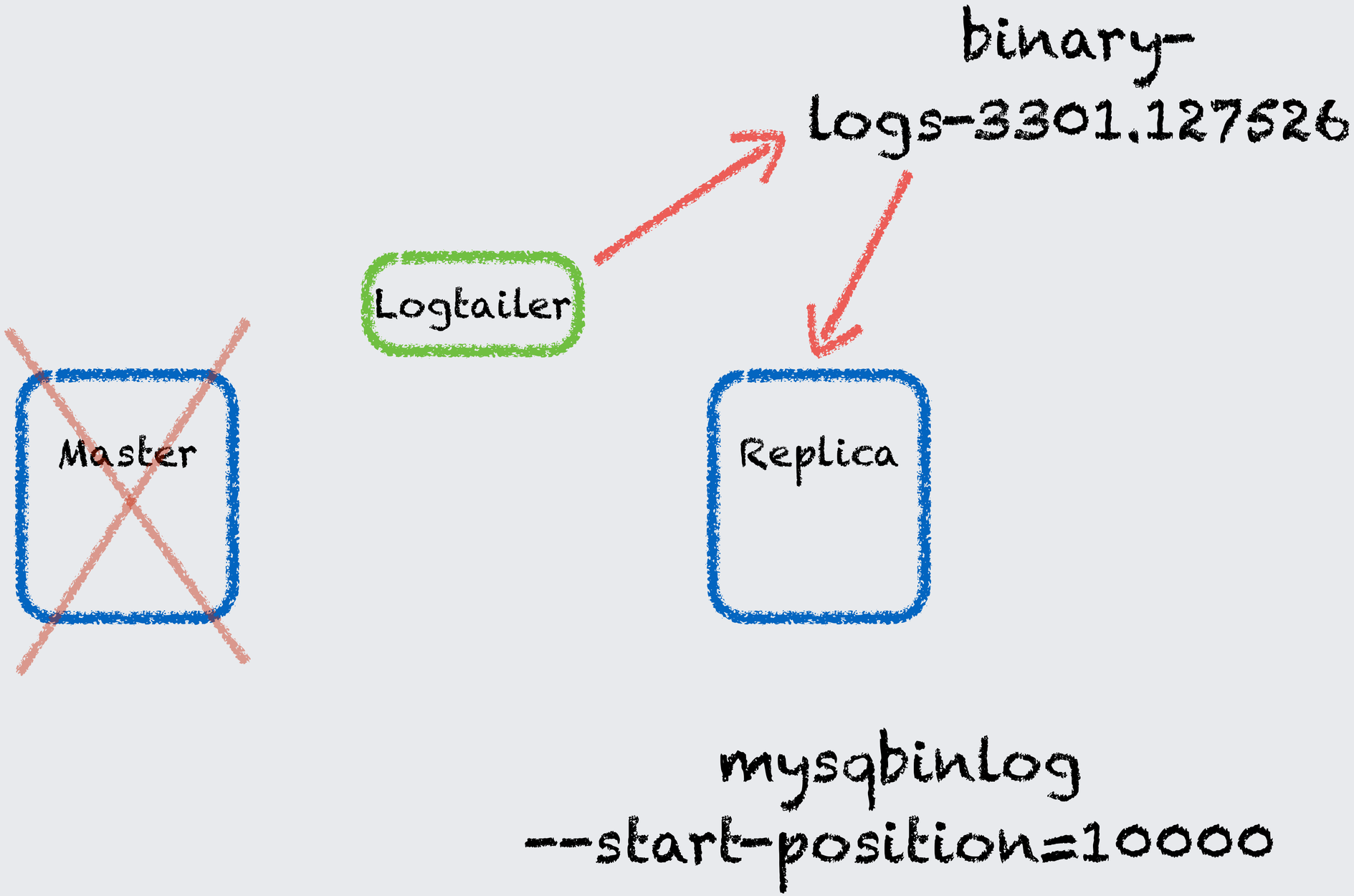
mysqlbinlog + semi-sync patches

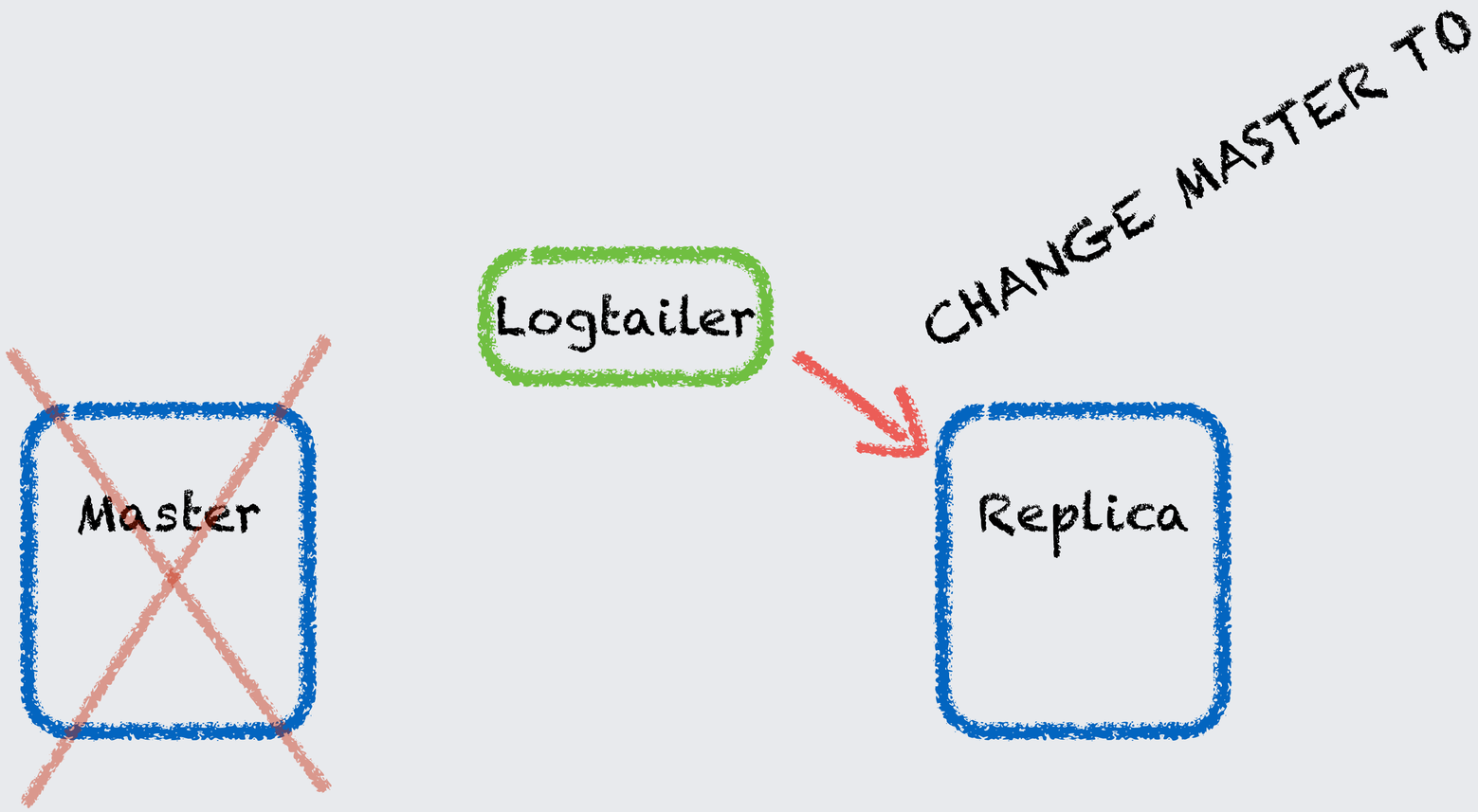
Binlog Server


Master
Semi-sync-
Replication
mysqlbinlog
Semi-sync Binlog Writer/Acker
Local Disk
log.index
Binlog
00001.bin
Binlog
00002.bin
Binlog
00003.bin


Master
Semi-sync-
Replication
Binlog Server Log-tailer
Semi-sync Binlog Writer/Acker
Binlog Reader
Promotion
Catchup
Replica
Local Disk
log.index
Binlog
00001.bin
Binlog
00002.bin
Binlog
00003.bin

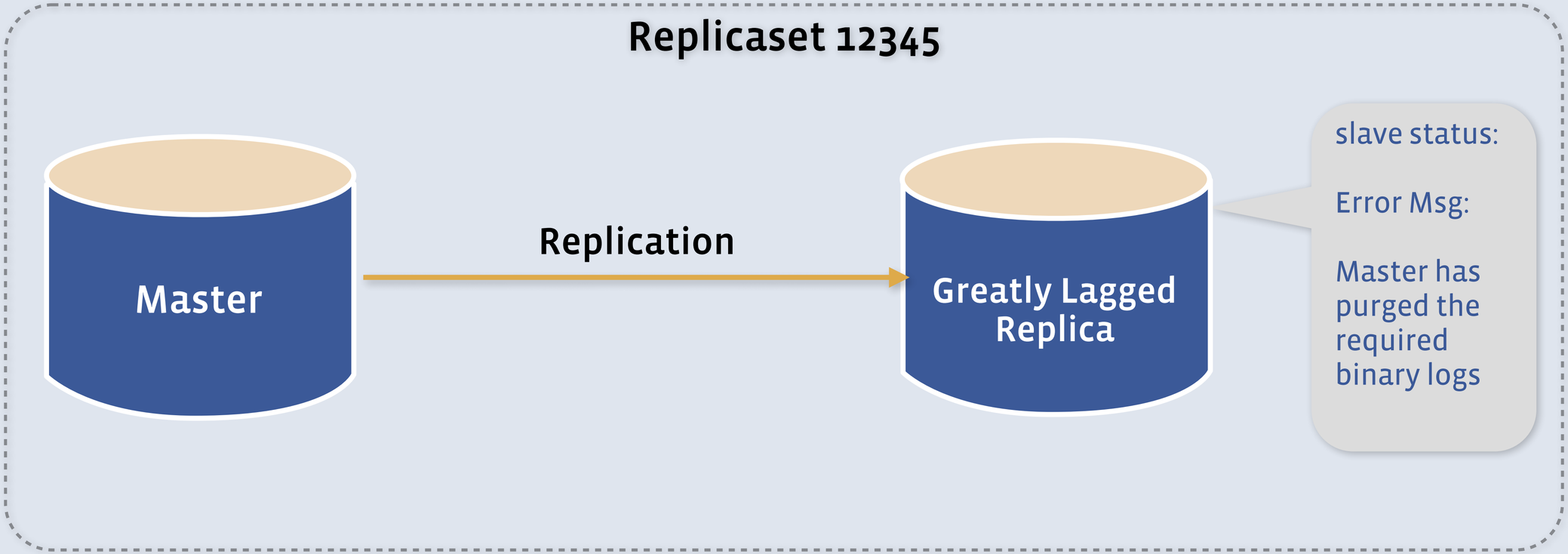

Logtailer
Master
Replica


binary-
logs-3301.127526
Logtailer
Master
Replica
mysqlbinlog
--start-position=10000


Logtailer
CHANGE MASTER TO
Master
Replica

Binlog Server ++

Lagged replicas?
Error 1236


Replicaset 12345
Master
Replication
Greatly Lagged Replica
slave status:
Error Msg:
Master has purged the required binary logs


Replicaset 12345
Binlog Reader/Sender
Binlog Locator
Binlog Server
change master to Binlog Server;
Greatly Lagged Replica

```
binlog_server> show slave status\G
*************************** 1. row ***************************
             Slave_IO_State: Waiting for master to send event
                Master_Host: HOSTNAME
                Master_Port: PORT
              Connect_Retry: 0
            Master_Log_File: binary-logs-xxxxxx.007964
        Read_Master_Log_Pos: 97115
                Binlog_File: binary-logs-xxxxxx.007964
                 Binlog_Pos: 97115
              Last_IO_Errno: 0
            Master_Server_Id: 3695980966
          Executed_Gtid_Set: ea4a5e01-b3e4-4273-a25e-88d06db8d1a5:1-902842,
b29a87bd-d60b-4455-9ab8-90d7b720f169:1-81669
            Mysql_Replicaset: REPLICA_SET_NAME
Replicaset_Tier_Version: VERSION_NUM
             Semisync_Slave: Yes
```

There’s plenty more to Binlog Server

Search for ”Binlog Server at Facebook”

MariaDB MaxScale

https://mariadb.com/resources/blog/the-binlog-server/

https://github.com/mariadb-corporation/MaxScale

Distributed systems are really hard

DBAs don’t scale as well as MySQL does

Lossless Semi-Sync

=> Commit;
Binlog Prepare => No-op
InnoDB Prepare => Written to InnoDB for recovery
Binlog Commit => Written to binlog
InnoDB Commit => Visible from other clients
<= OK;

=> Commit;
Binlog Prepare
InnoDB Prepare
Binlog Commit
InnoDB Commit
Wait for Semi-Sync Ack
<= OK;

Crash!

=> Commit;
InnoDB Prepare
Binlog Prepare
Binlog Commit
Wait for Semi-Sync Ack
InnoDB Commit
<= OK;

Lossless Semi-Sync

=> Commit;
Binlog Prepare => No-op
InnoDB Prepare => Written to InnoDB for recovery
Binlog Commit => Written to binlog
InnoDB Commit => Visible from other clients
<= OK;

=> Commit;
Binlog Prepare
InnoDB Prepare
Binlog Commit
InnoDB Commit
Wait for Semi-Sync Ack
<= OK;

Crash!

=> Commit;
InnoDB Prepare
Binlog Prepare
Binlog Commit
Wait for Semi-Sync Ack
InnoDB Commit
<= OK;

	status	semi-sync thread	async thread
transaction 1	acked	←	←

	status	semi-sync thread	async thread
transaction 1	acked	←	←
transaction 2	prepare		

	status	semi-sync thread	async thread
transaction 1	acked		
transaction 2	waiting for ack		

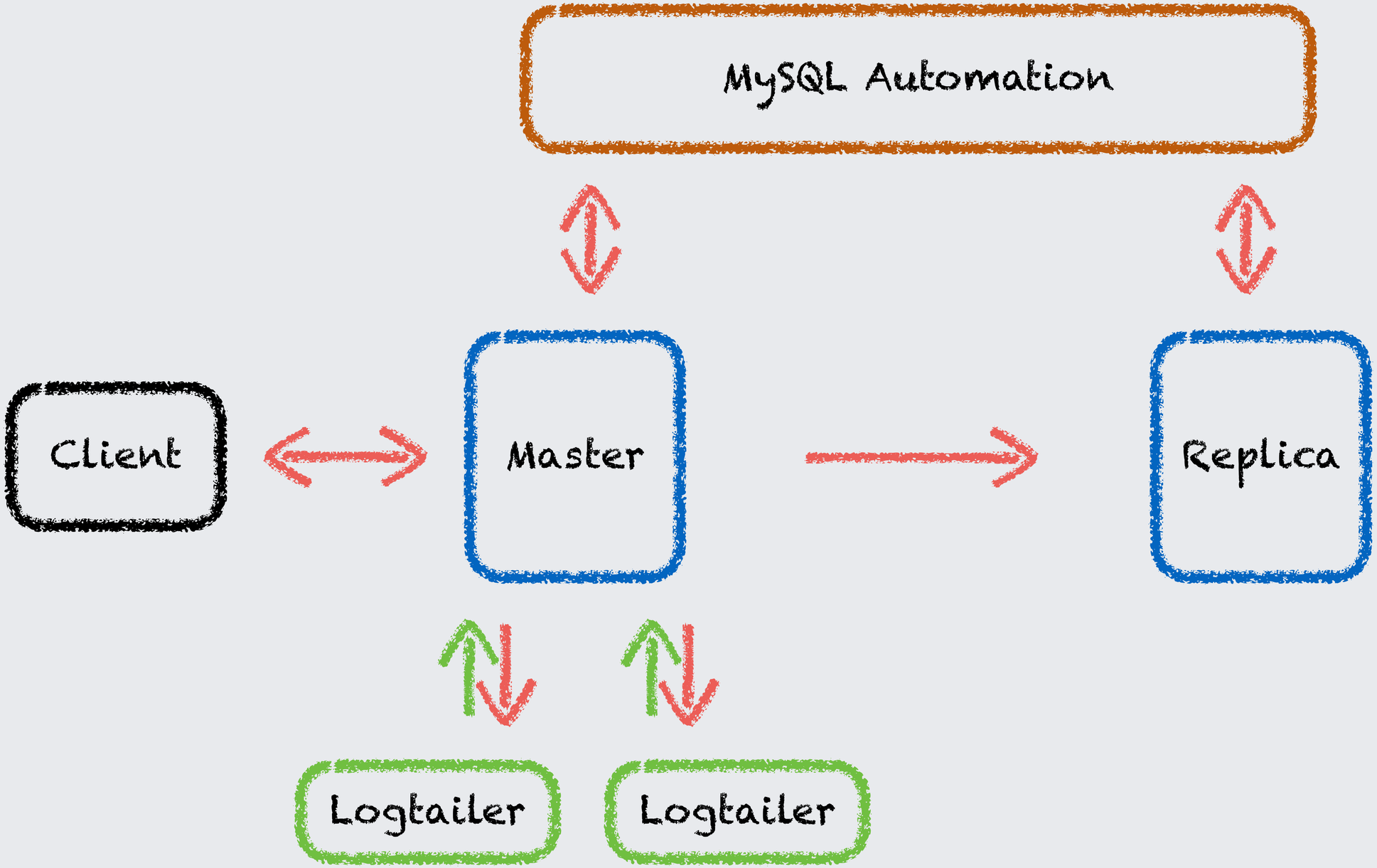
	status	semi-sync thread	async thread
transaction 1	acked		
transaction 2	acked	←	←

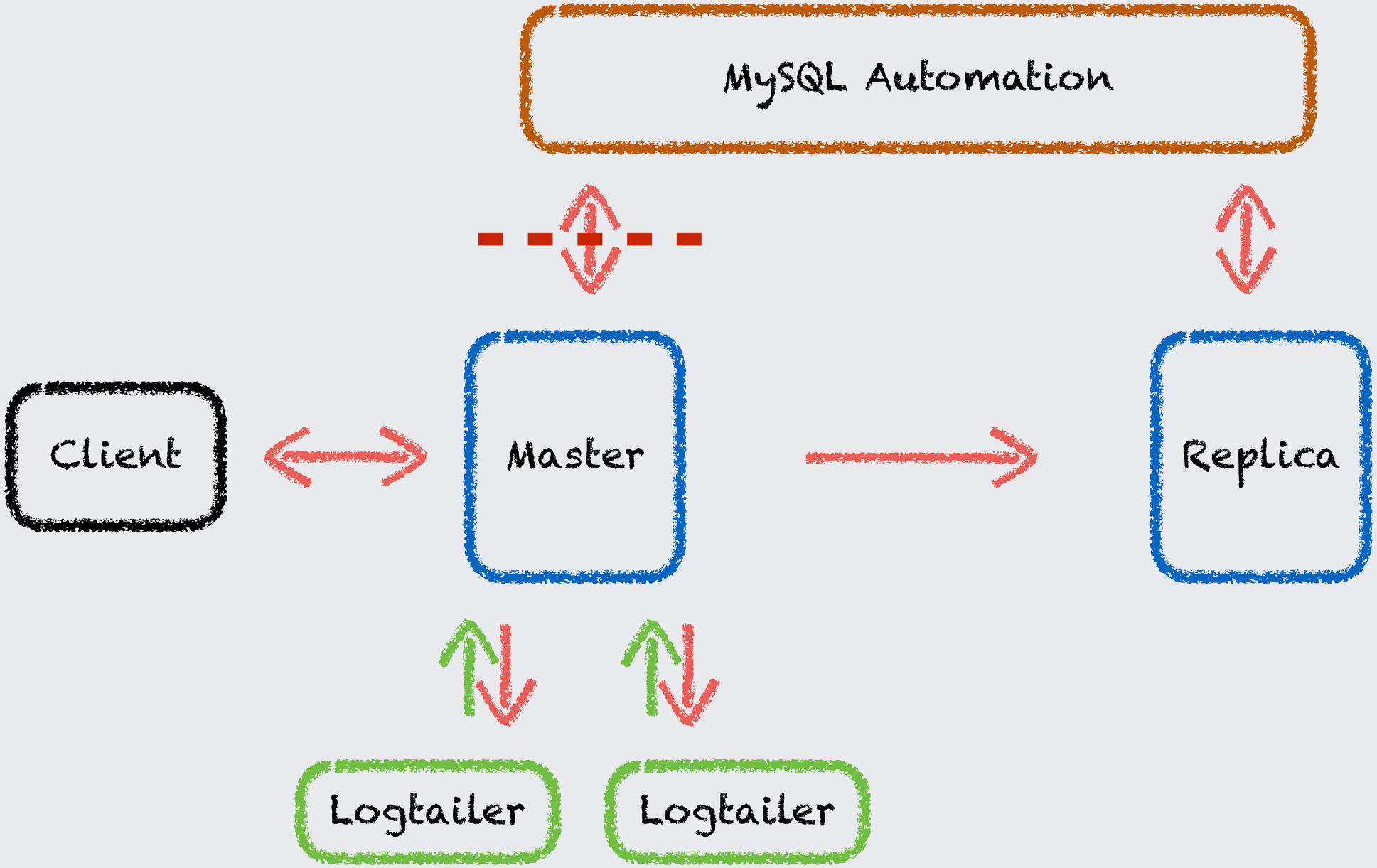
	status	semi-sync thread	async thread
transaction 1	acked		
transaction 2	acked	←	←
transaction 3	prepare		

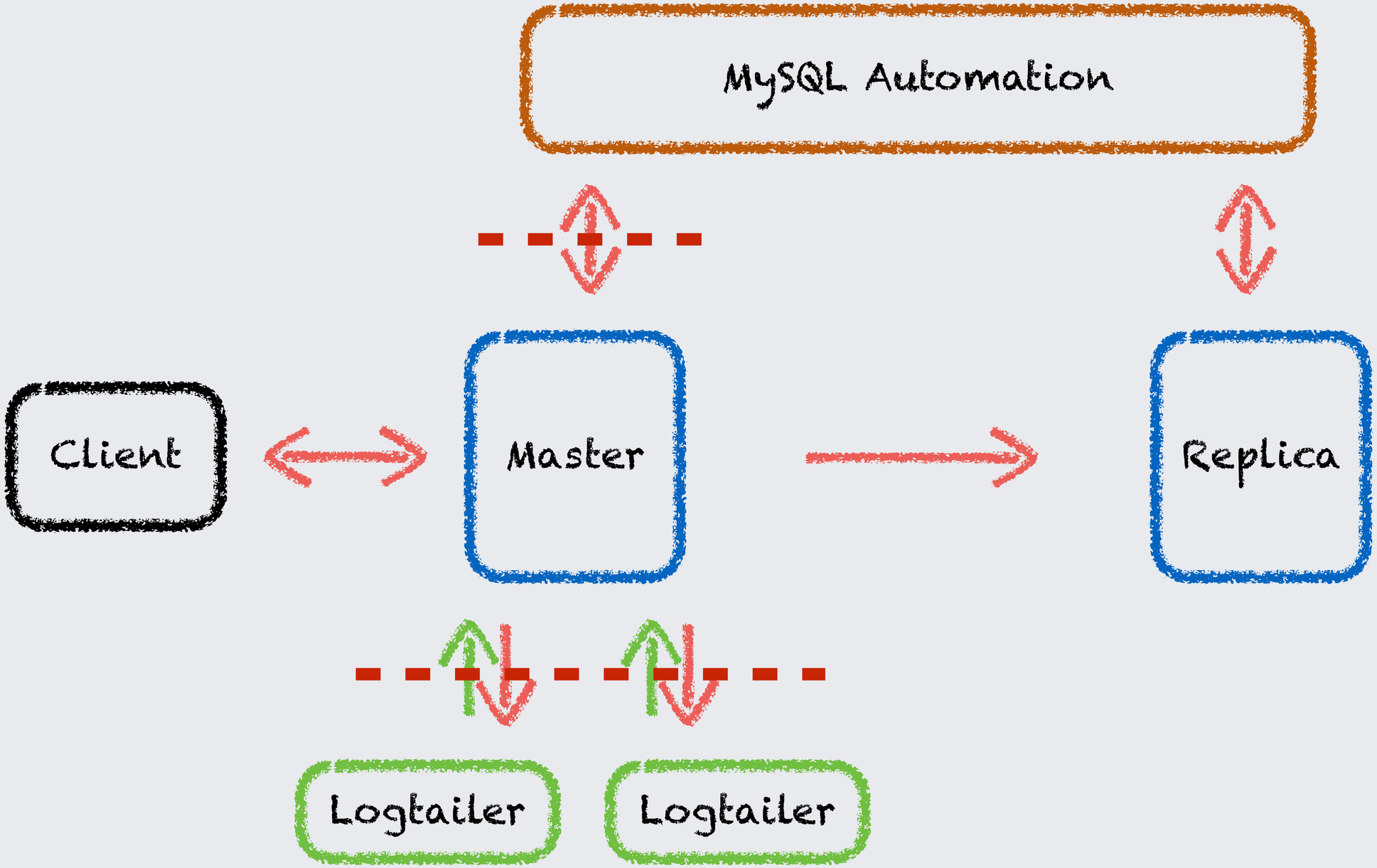
	status	semi-sync thread	async thread
transaction 1	acked		
transaction 2	acked		
transaction 3	waiting for ack		

	status	semi-sync thread	async thread
transaction 1	acked		
transaction 2	acked		
transaction 3	acked	←	←

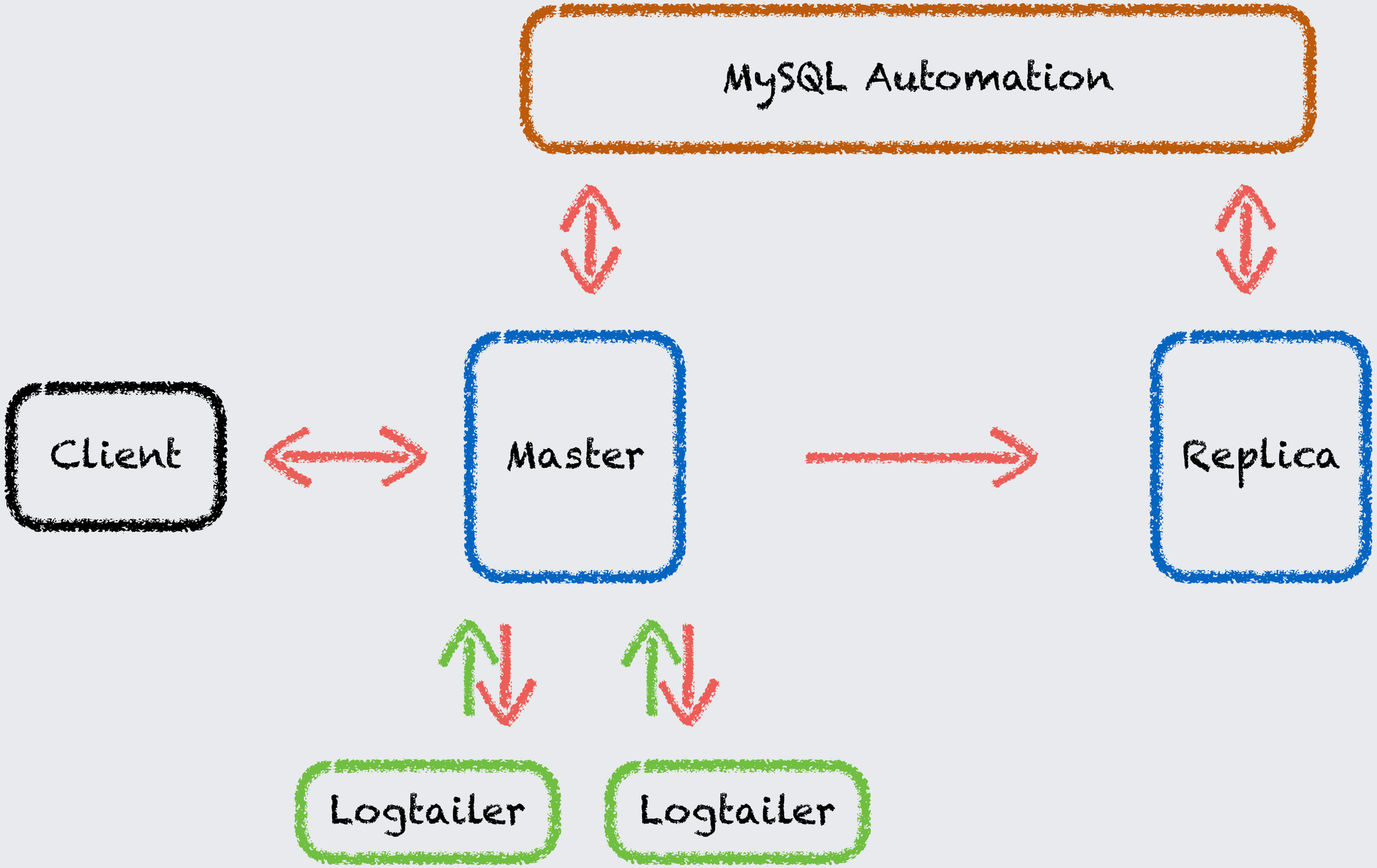
Flappy/partially-isolated master

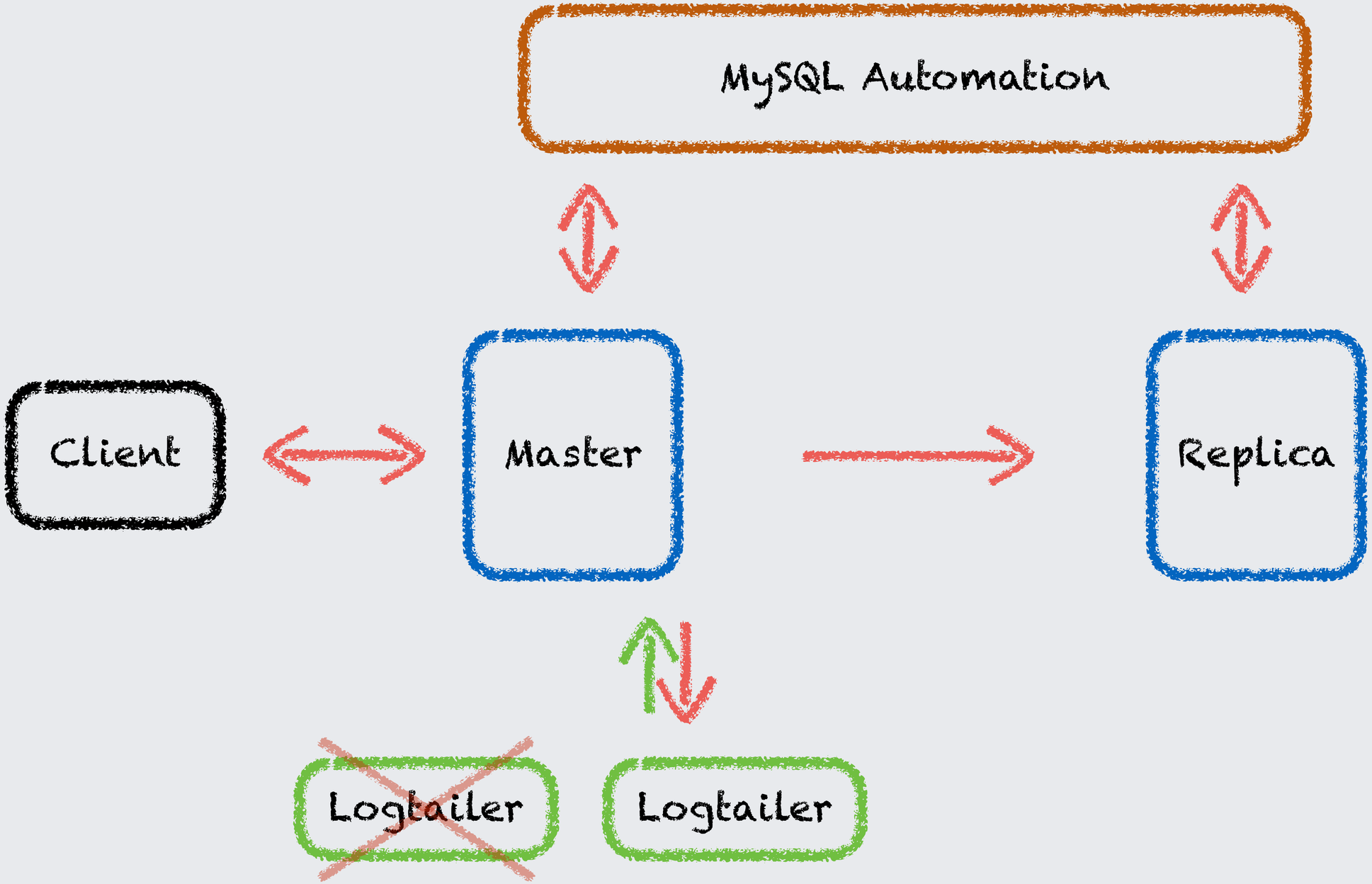

MySQL Automation
Client
Master
Replica
Logtailer
Logtailer

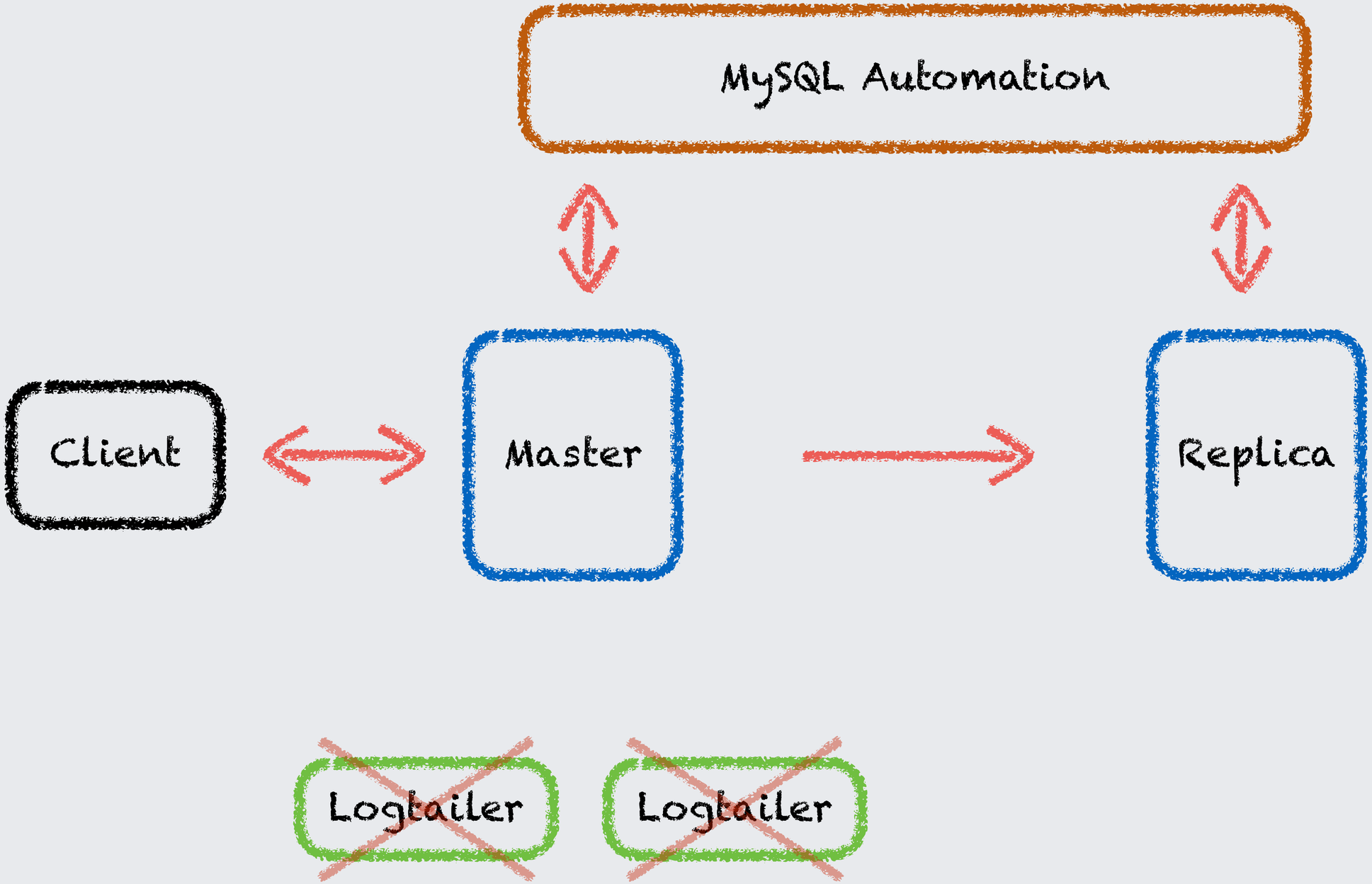

MySQL Automation
Client
Master
Replica
Logtailer
Logtailer

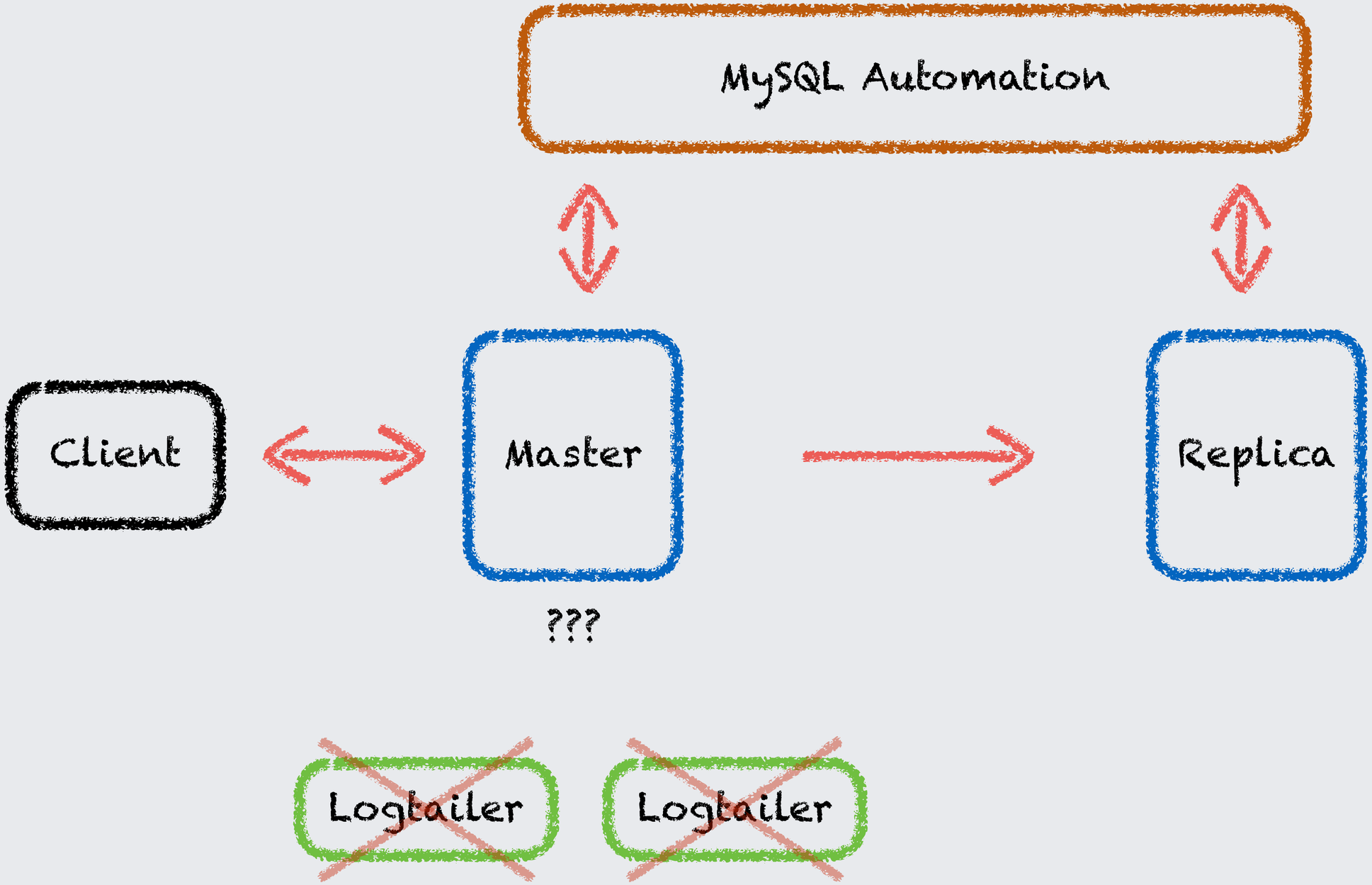

MySQL Automation
Client
Master
Replica
Logtailer
Logtailer

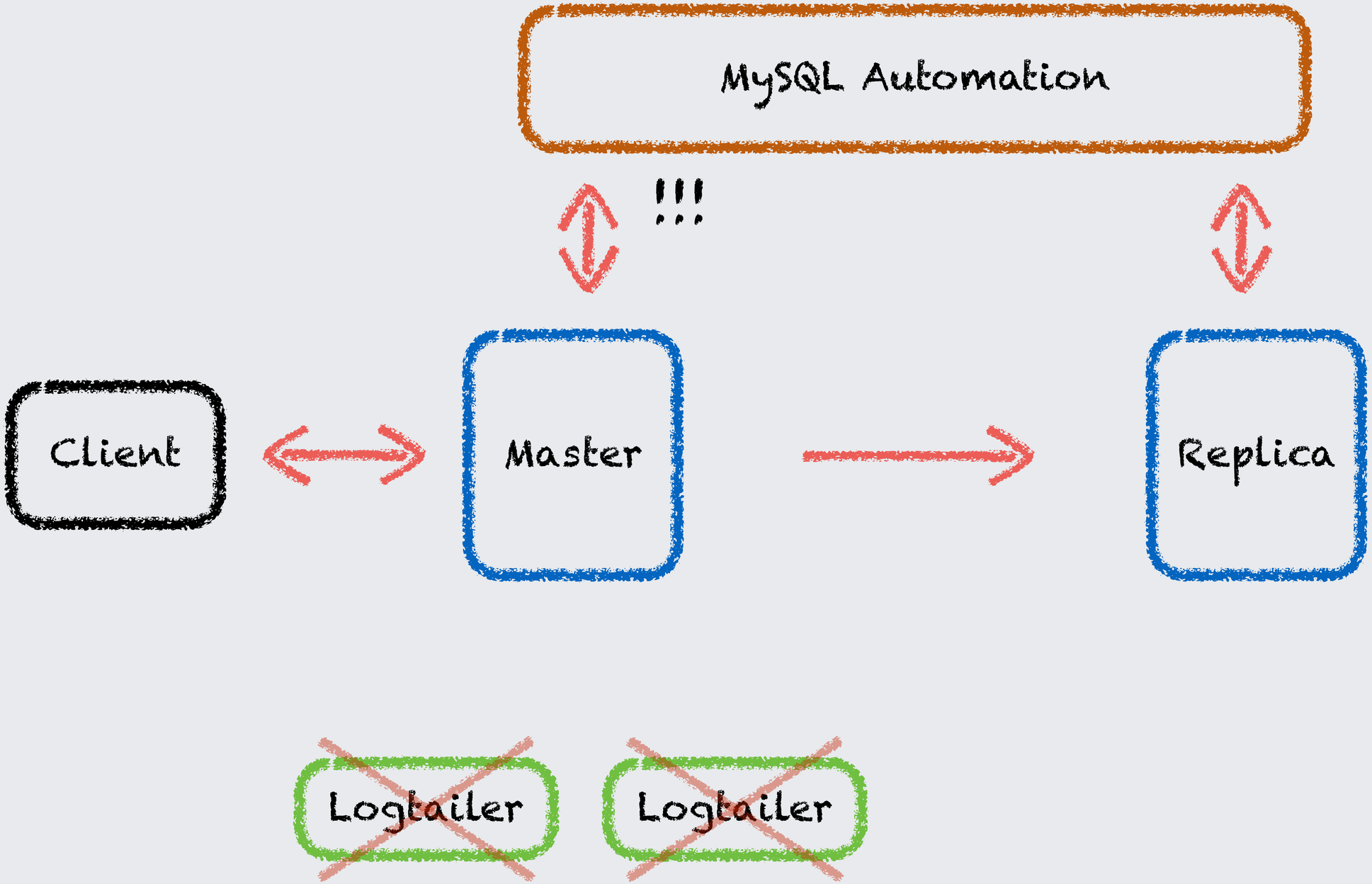
Logtailer failures

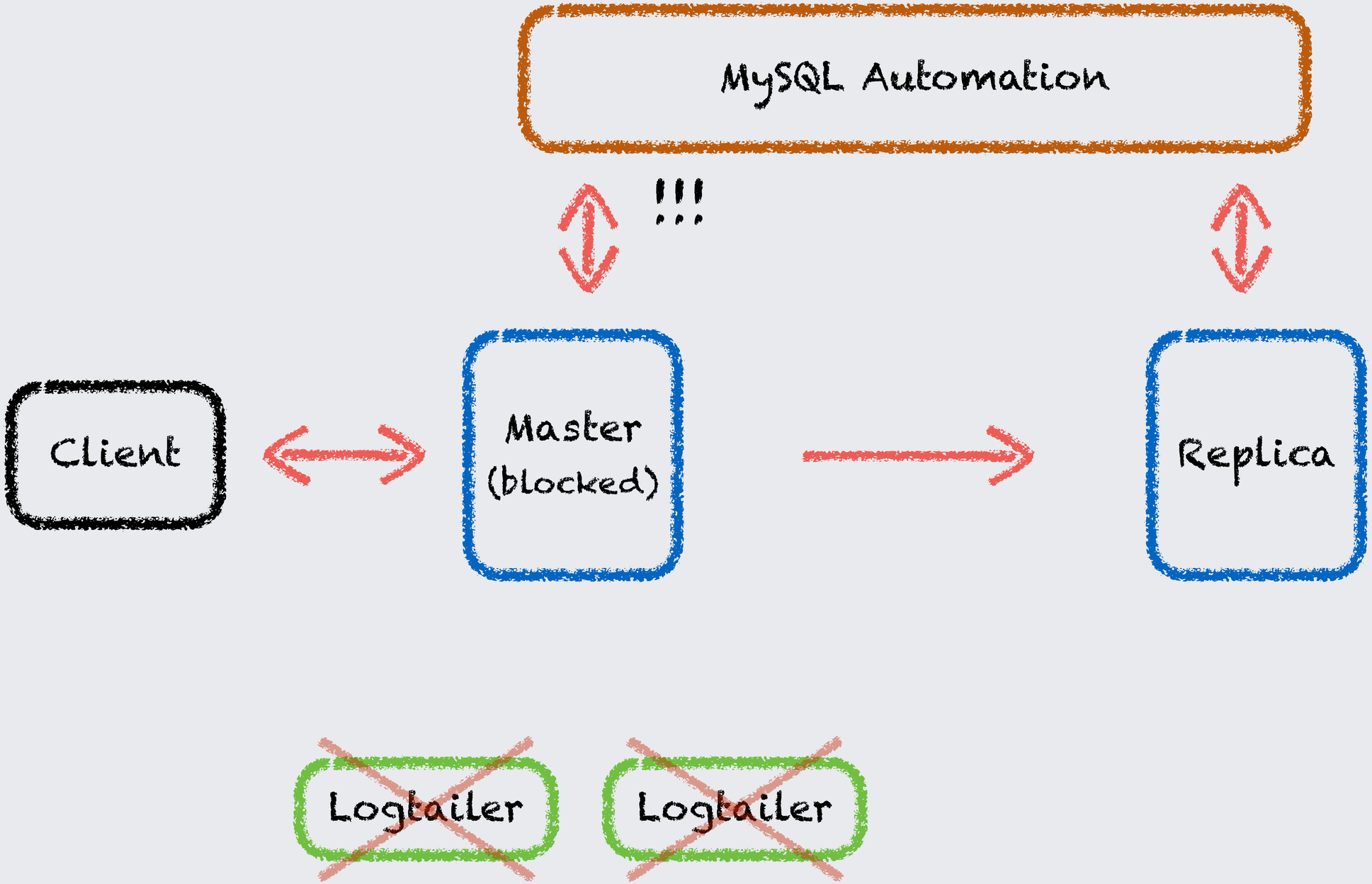

MySQL Automation
Client
Master
Replica
Logtailer
Logtailer

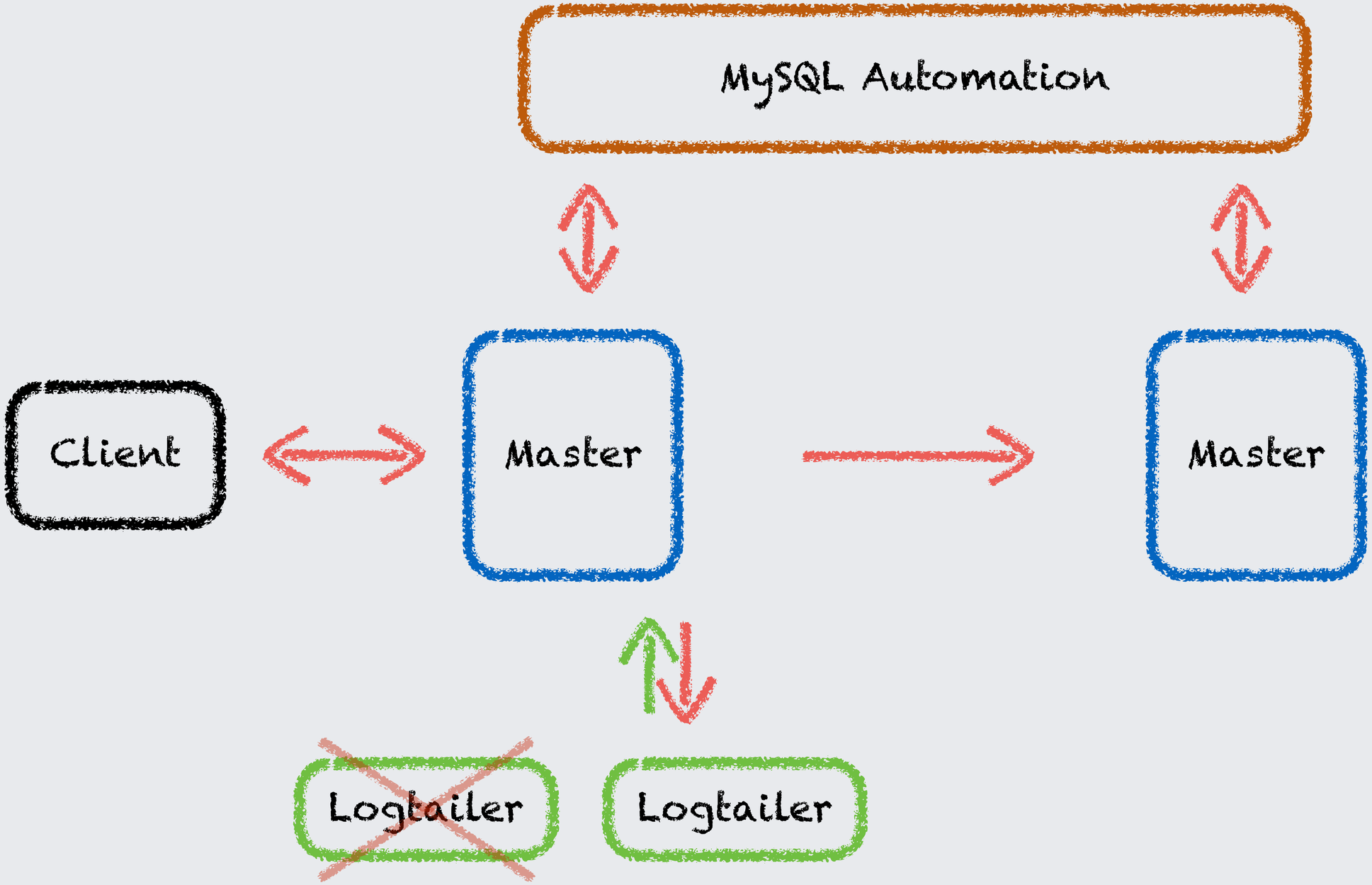

MySQL Automation
Client
Master
Replica
Logtailer
Logtailer

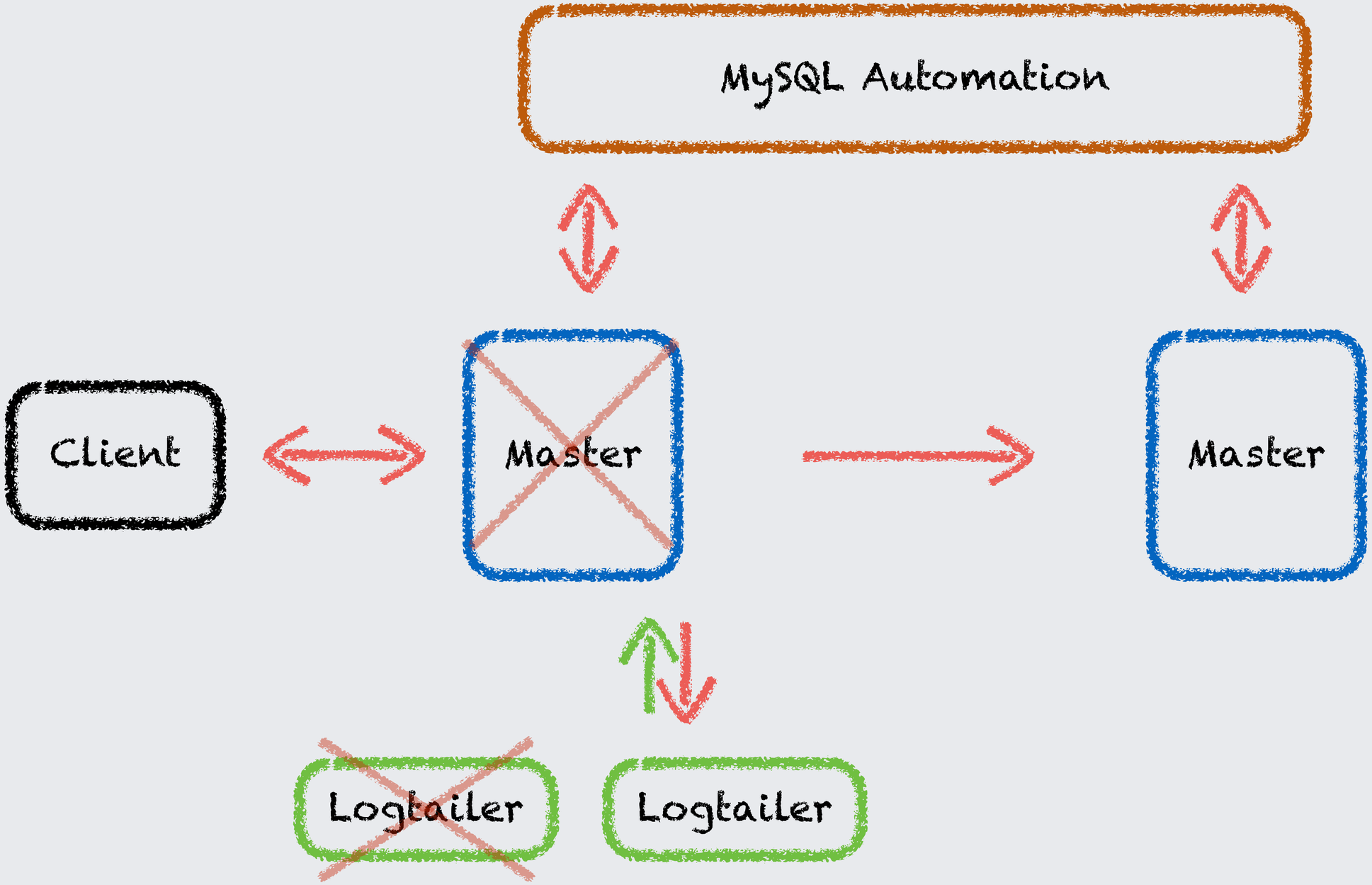

MySQL Automation
Client
Master
Replica
Logtailer
Logtailer

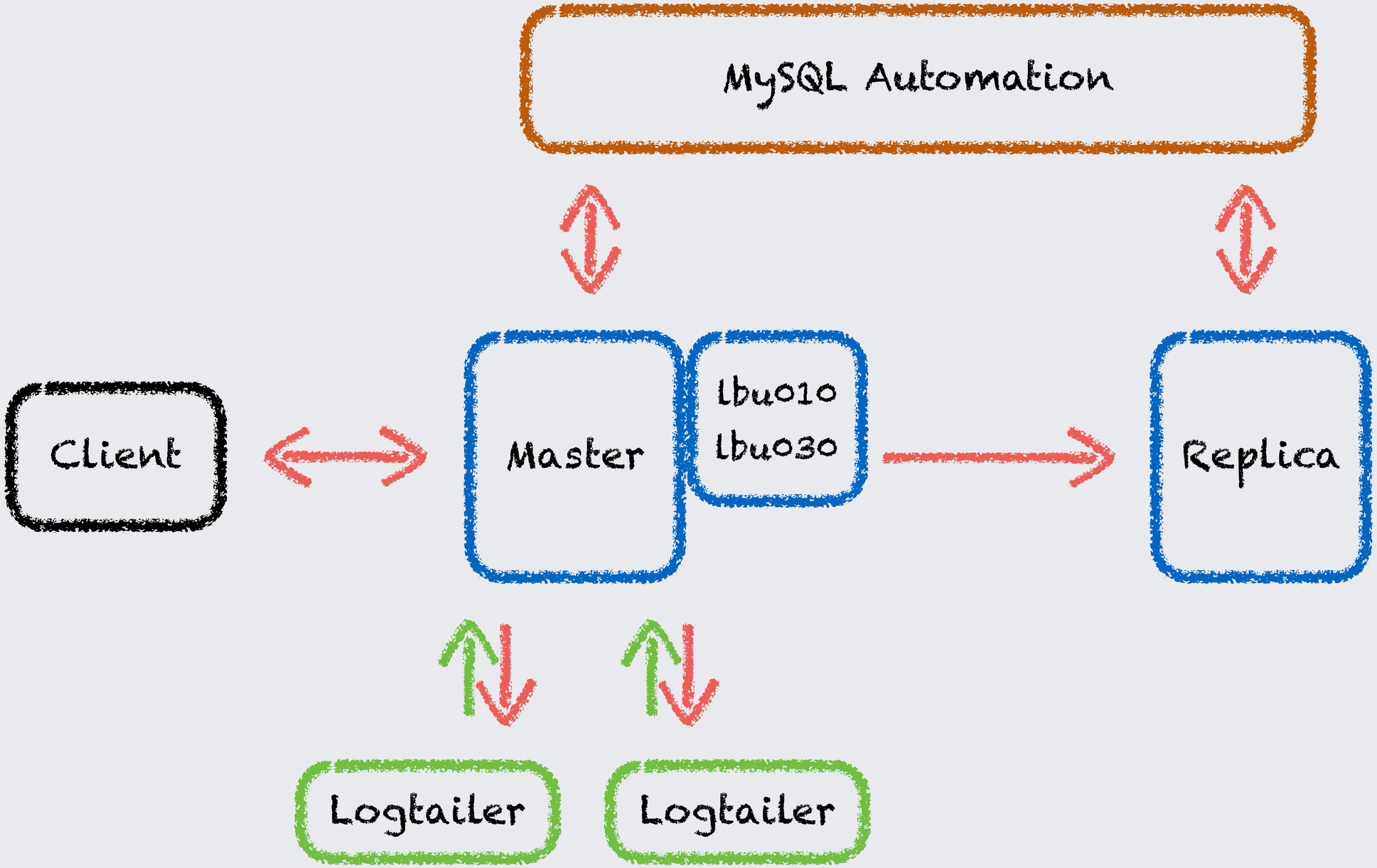

MySQL Automation
Client
Master
Replica
???
Logtailer
Logtailer

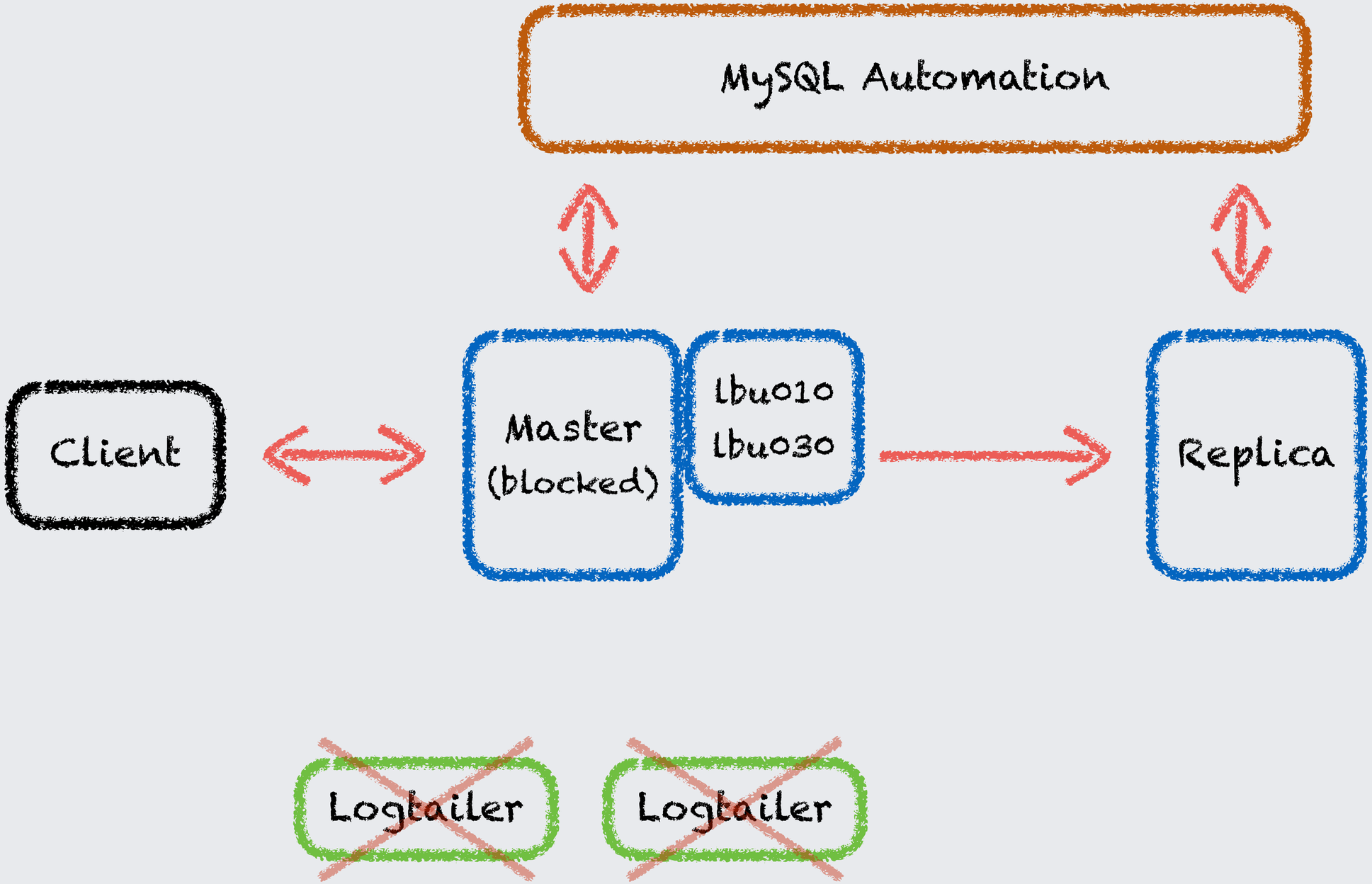

MySQL Automation
!!!
Client
Master
Replica
Logtailer
Logtailer

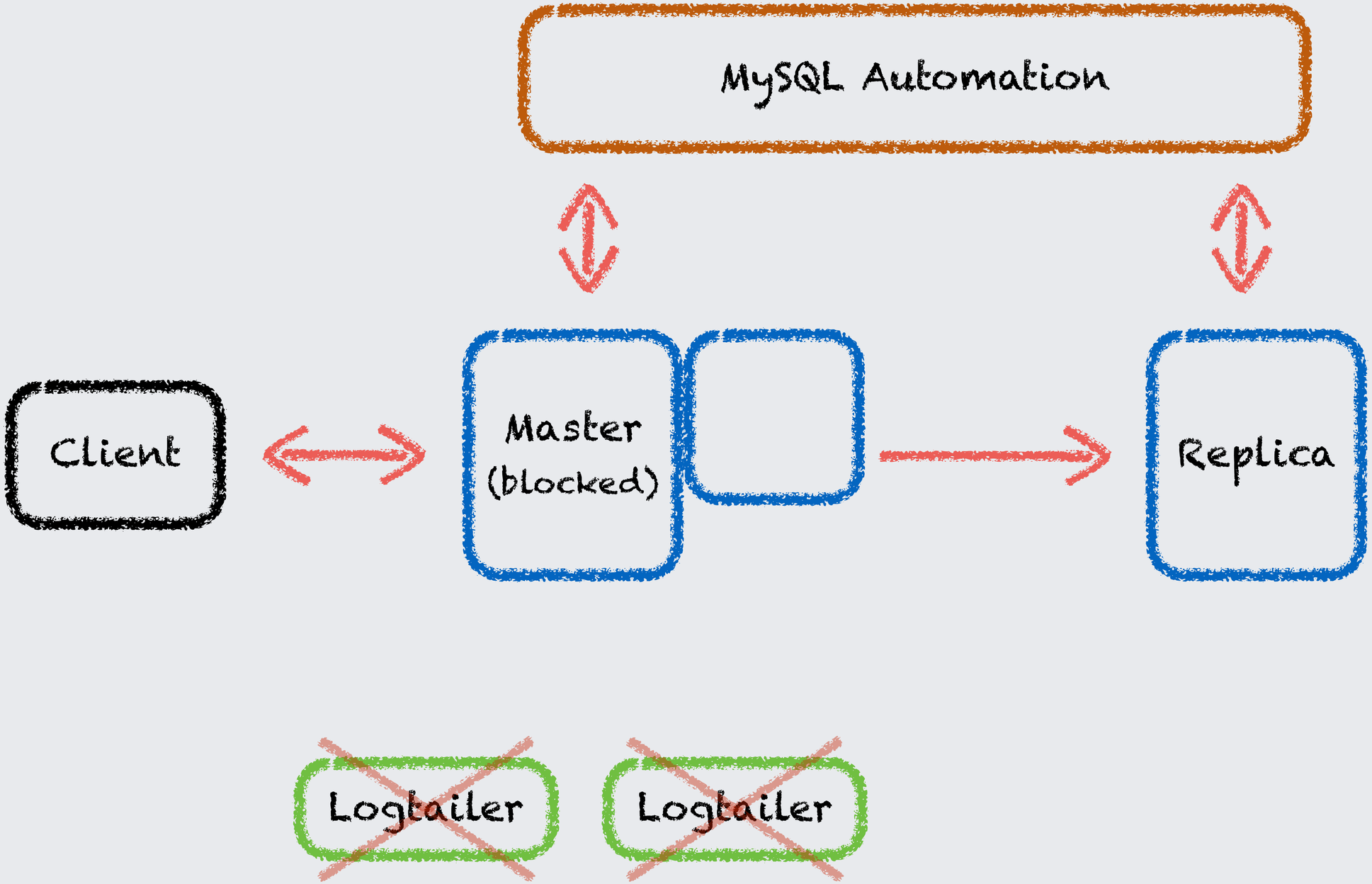

MySQL Automation
!!!
Client
Master
(blocked)
Replica
Logtailer
Logtailer


MySQL Automation
Client
Master
Master
Logtailer
Logtailer


MySQL Automation
Client
Master
Master
Logtailer
Logtailer


MySQL Automation
Client
Master
lbu010
lbu030
Replica
Logtailer
Logtailer


MySQL Automation
Client
Master
(blocked)
lbu010
lbu030
Replica
Logtailer
Logtailer


MySQL Automation
Client
Master
(blocked)
Replica
Logtailer
Logtailer

These situations were very rare

Everything open source*

https://github.com/facebook/mysql-5.6/

* except Facebook's Binlog Server

MariaDB MaxScale

https://mariadb.com/resources/blog/the-binlog-server/

https://github.com/mariadb-corporation/MaxScale

Sam Dunster

facebook

Come chat at the Facebook booth right after this!