Latency, Conflicts and Consistency:

MySQL Group Replication vs. Galera Cluster

severalnines

Housekeeping items

- Standard presentation format
  - Speaker intro
  - Group poll
  - Agenda
  - Topic
  - Q&A
- Recording will be sent to attendees

Your expert presenter

Who: Ashraf Sharif
What: Support Engineering
Team Lead
Where: Severalnines
Why: 15+ YoE as DBA for MySQL, MariaDB, Galera, MongoDB, Redis

```
<img src="image.jpg" alt="Image"
width="300" height="300">
```

Poll time!

Let's Take Your Pulse

AGENDA

1. Multi-Primary Replication
2. How it Works?
3. Trade Offs
4. Example Architecture
5. Use Cases
6. Best Practices
7. Q & A

Multi-Primary Replication

MySQL/MariaDB HA Options - 2025

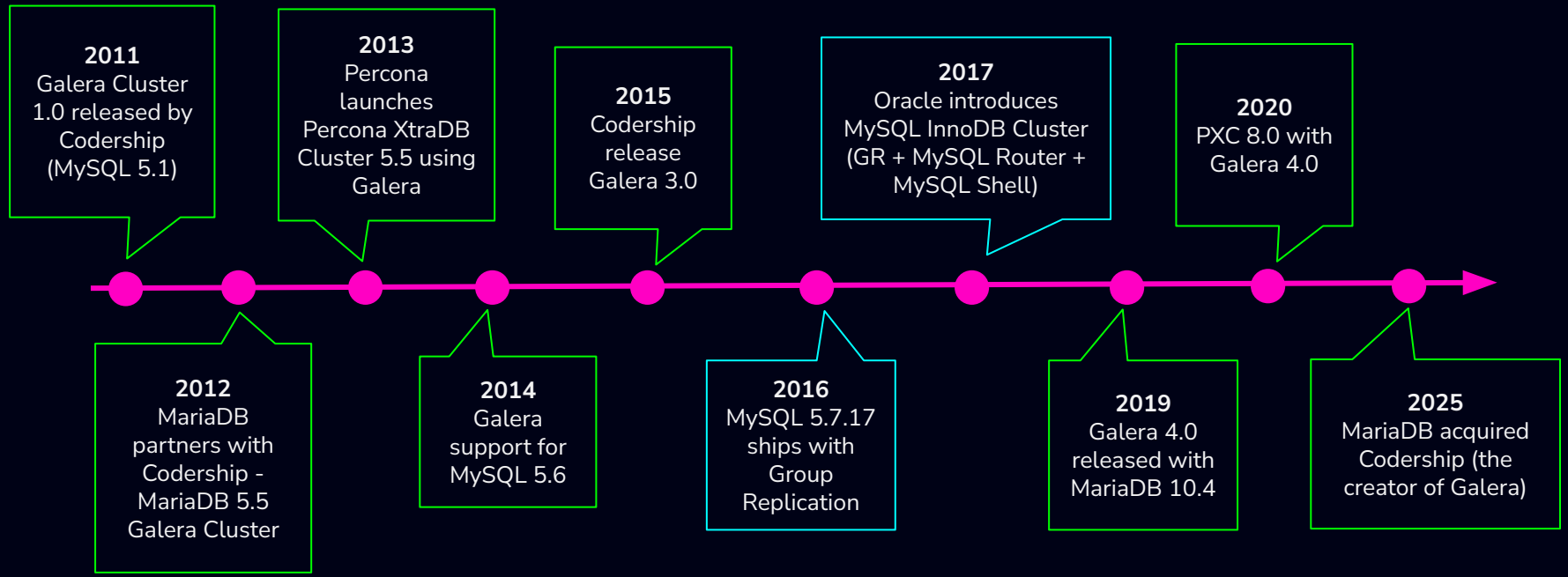
Topology	Replication		Oracle	Percona	MariaDB
Single-primary	Asynchronous/ Semi-synchronous		• MySQL Community Server • MySQL Enterprise Server	• Percona Server for MySQL • Percona Server for MySQL Pro	• MariaDB Server • MariaDB Enterprise Server
Multi-primary	Virtually synchronous	MySQL Group Replication	• MySQL InnoDB Cluster • MySQL InnoDB ClusterSet • MySQL InnoDB ReplicaSet		
		Galera Cluster		• Percona XtraDB Cluster • Percona XtraDB Cluster Pro	• MariaDB Cluster • MariaDB Enterprise Cluster
	Fully synchronous		MySQL Cluster (NDB)		

Overview

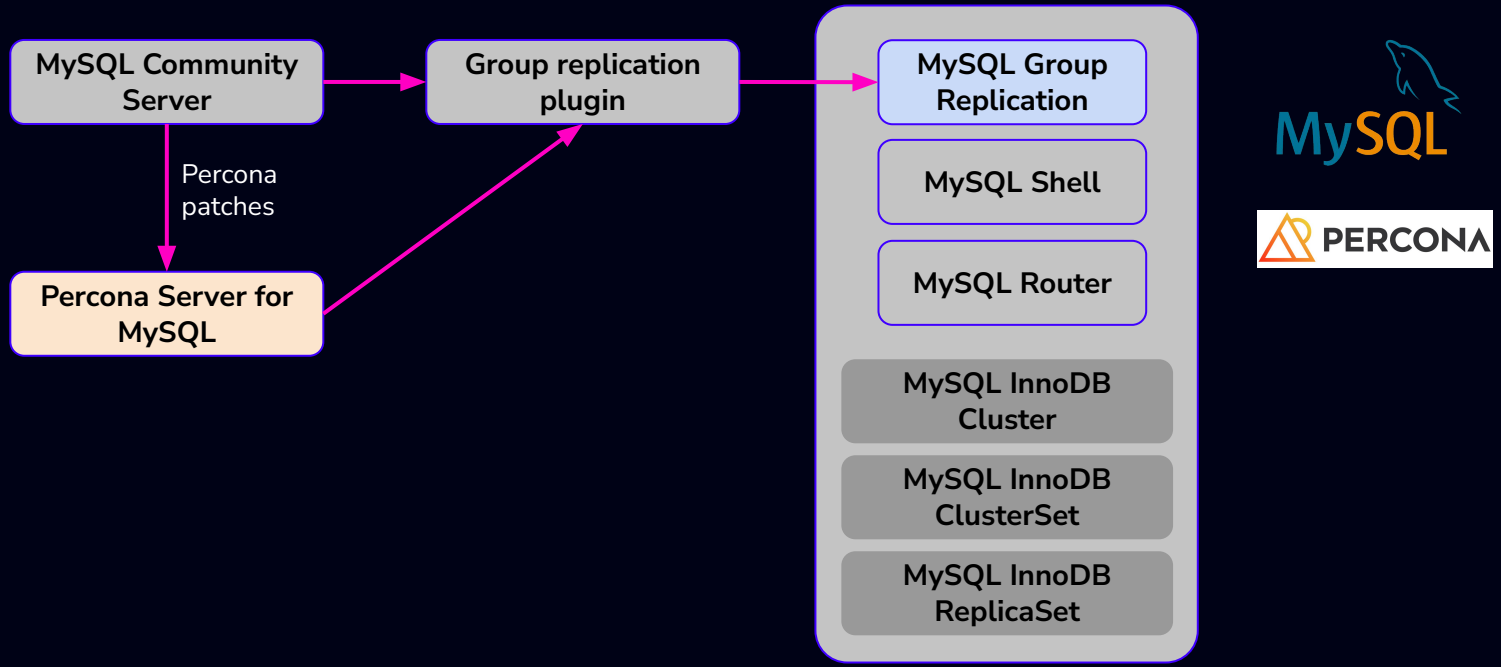
- Virtually synchronous updates on any member in a group of MySQL servers
- Transaction ordering and broadcasting
- Failure detection
- Conflict handling
- Automatic membership control and provisioning

- Current providers:
  - MySQL Group Replication (MGR) - built by Oracle
  - Galera Replication (Galera) - built by Codership (acquired by MariaDB)
- Current products:
  - MGR: MySQL InnoDB Cluster, MySQL InnoDB ClusterSet, MySQL InnoDB ReplicaSet
  - Galera: Percona XtraDB Cluster, Percona XtraDB Cluster Pro, MariaDB Cluster, MariaDB Enterprise Cluster, Galera Cluster for MySQL (Codership)

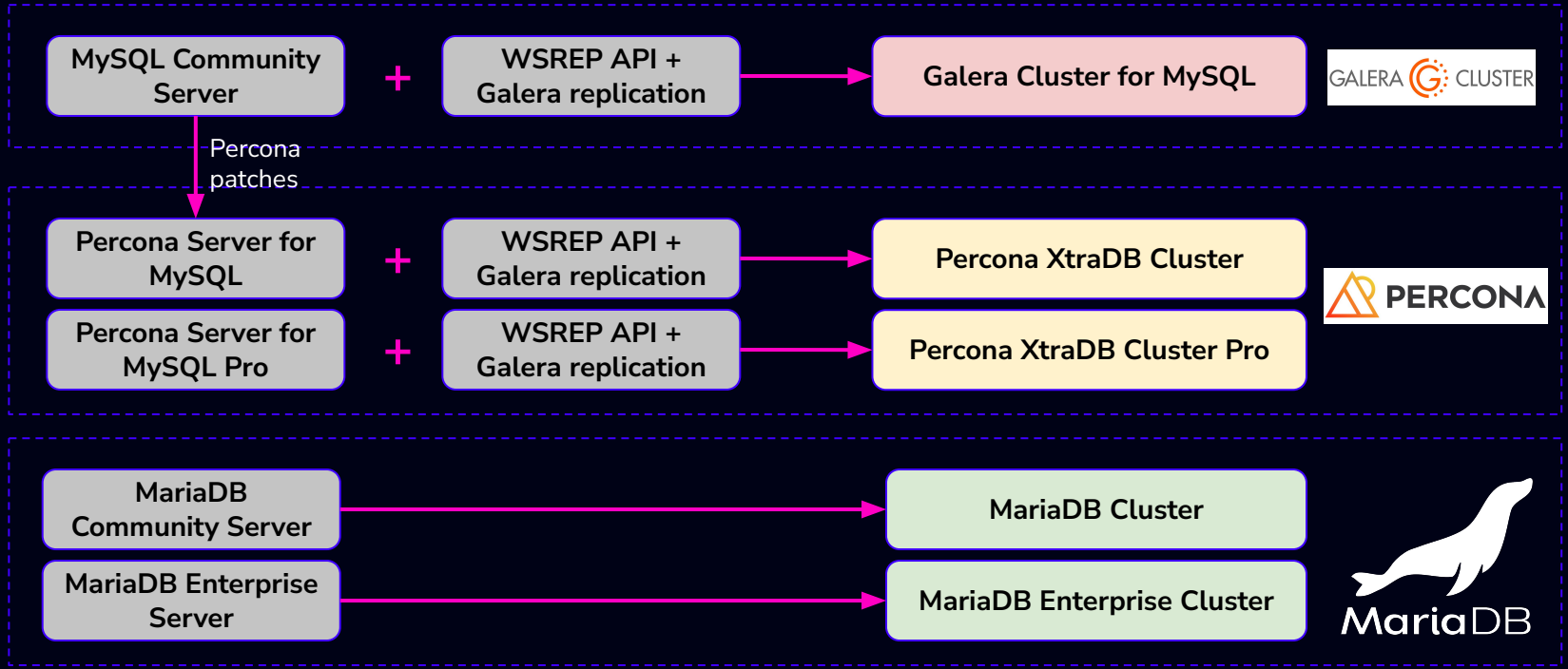
Evolution


2011
Galera Cluster 1.0 released by Codership (MySQL 5.1)
2012
MariaDB partners with Codership - MariaDB 5.5 Galera Cluster
2013
Percona launches Percona XtraDB Cluster 5.5 using Galera
2014
Galera support for MySQL 5.6
2015
Codership release Galera 3.0
2016
MySQL 5.7.17 ships with Group Replication
2017
Oracle introduces MySQL InnoDB Cluster (GR + MySQL Router + MySQL Shell)
2019
Galera 4.0 released with MariaDB 10.4
2020
PXC 8.0 with Galera 4.0
2025
MariaDB acquired Codership (the creator of Galera)

MySQL Group Replication Vendor



Galera Cluster Vendor



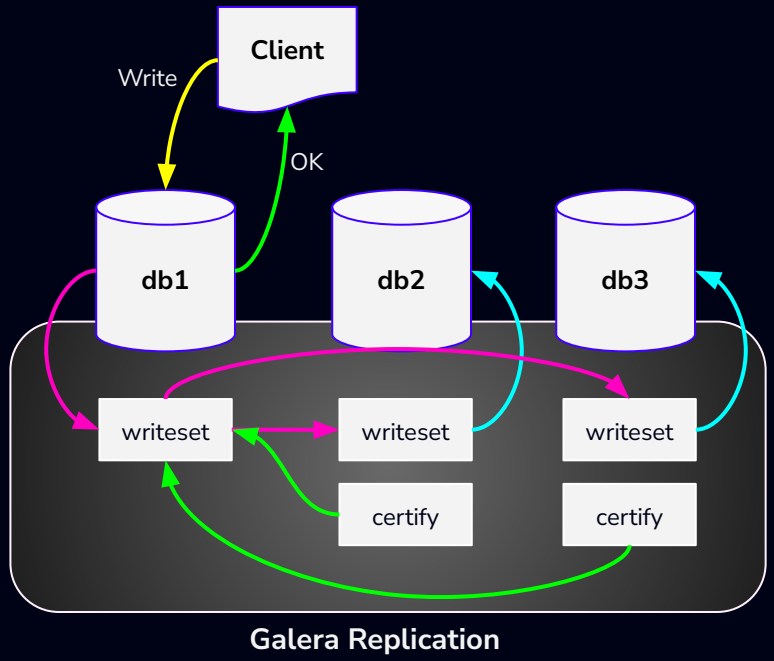
How it Works

several9s

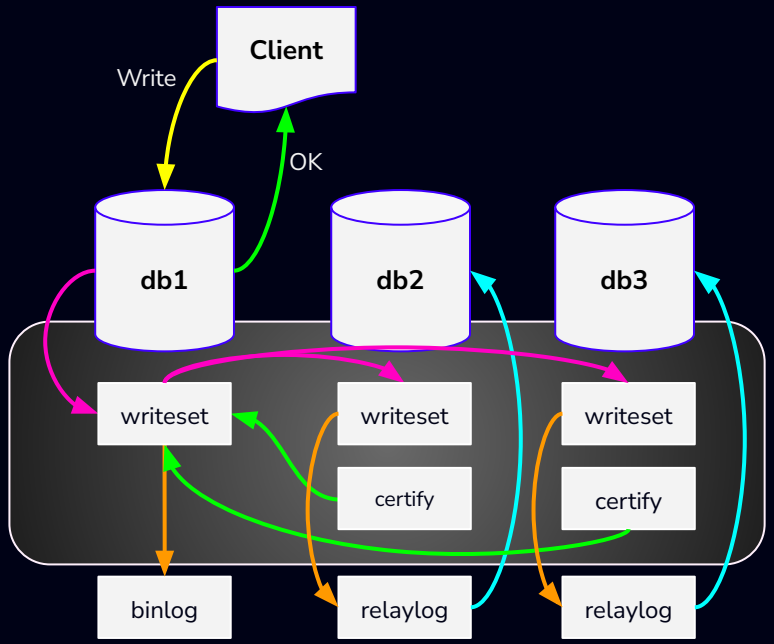
Requirements

- A transactional database engine - support COMMIT/ROLLBACK (InnoDB or XtraDB).
- A primary key on every table.
- A data synchronization tool:
  - MGR: CLONE plugin
  - Galera: mysqldump, rsync, Percona Xtrabackup, MariaDB Backup
- Group communication plugins/patch:
  - MGR: Paxos/Corosync
  - Galera: GCS + WSREP API
-
- Communication ports:
  - MGR: 3306, 33061, multicast traffic
  - Galera: 3306, 4567, 4568, 4444
- At least 3 nodes for quorum. 1 node is possible without HA. Arbitrator is also possible (Galera).
- Time sync: NTP/chrony mandatory
- Version:
  - Galera: Same MySQL and Galera major version across nodes.
  - MGR: Same MySQL major and minor version across nodes

Virtually Synchronous Replication

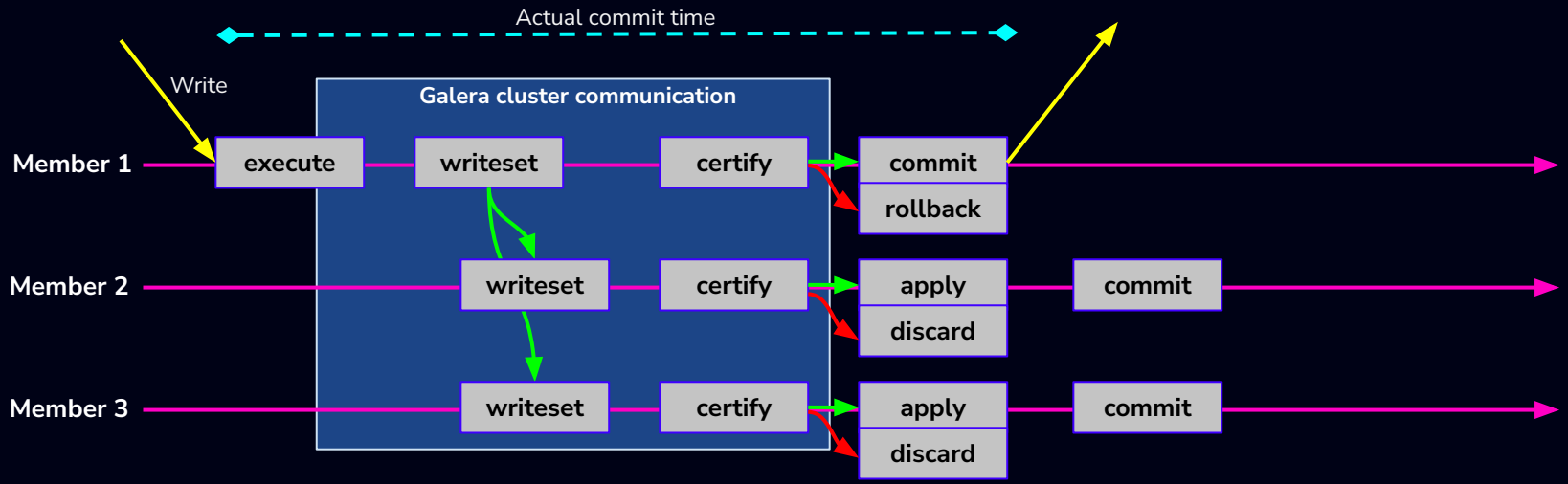


Galera Replication

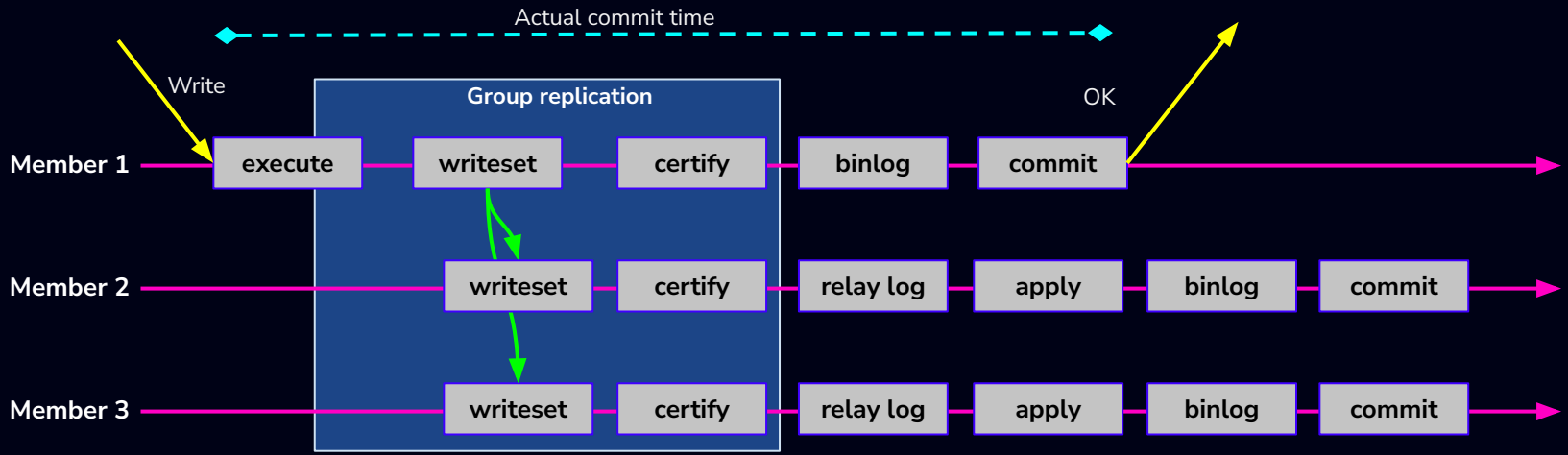


MySQL Group Replication

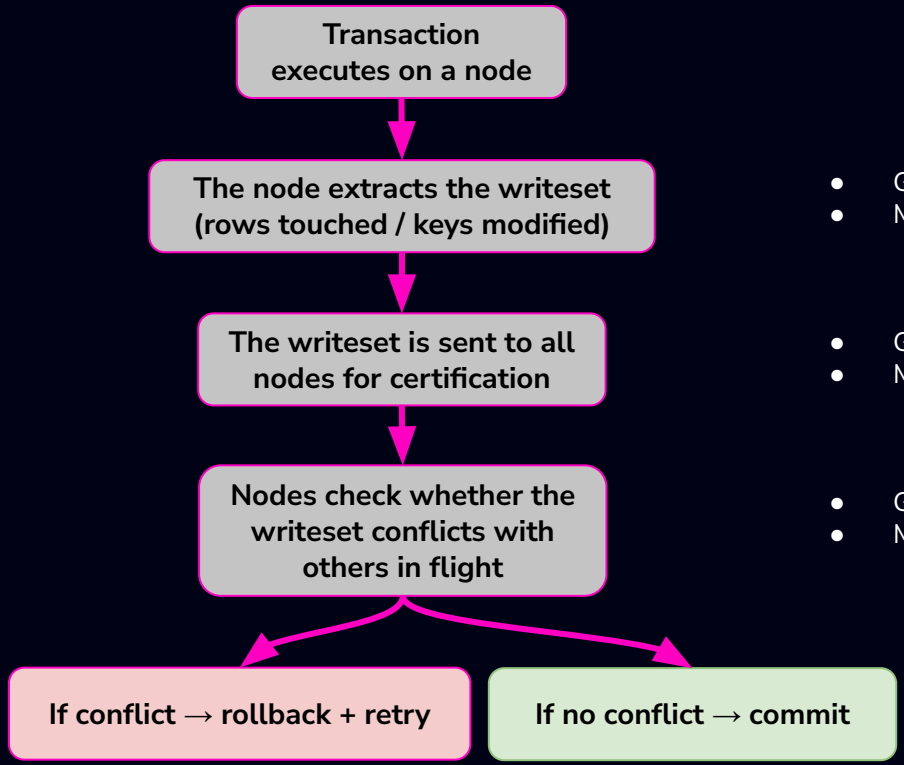
Virtually Synchronous Replication (Galera)



Virtually Synchronous Replication (MGR)



Certification



Write DMLs only (INSERT/UPDATE/DELETE/REPLACE)

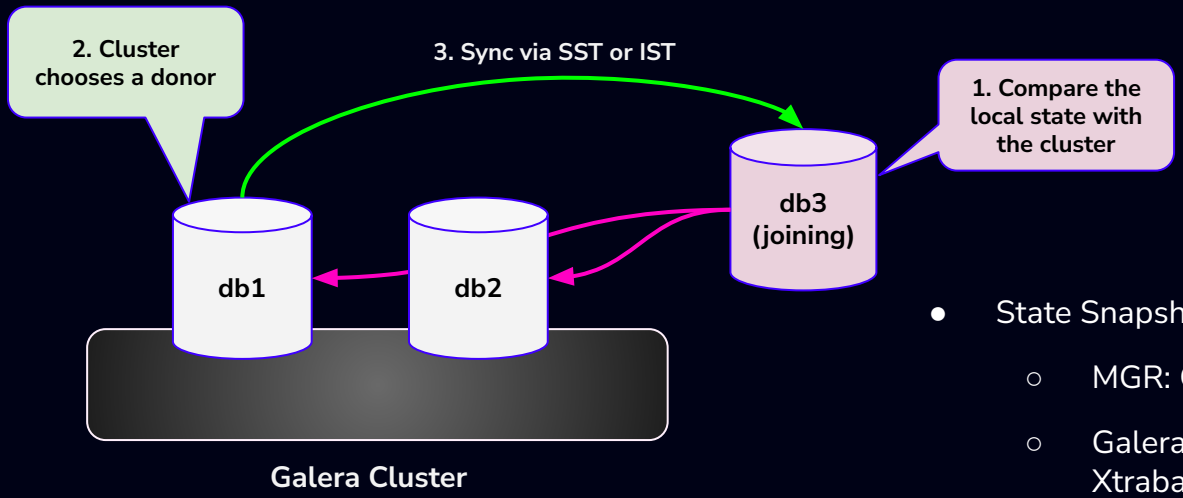
- Galera: Extract writeset at row-level
- MGR : Transaction write-set extracted from MySQL's binary log/GTID

- Galera: WSREP API + Galera plugin
- MGR : Paxos-like group communication protocol

- Galera: Key-based conflict detection
- MGR : Transaction keys conflict detection

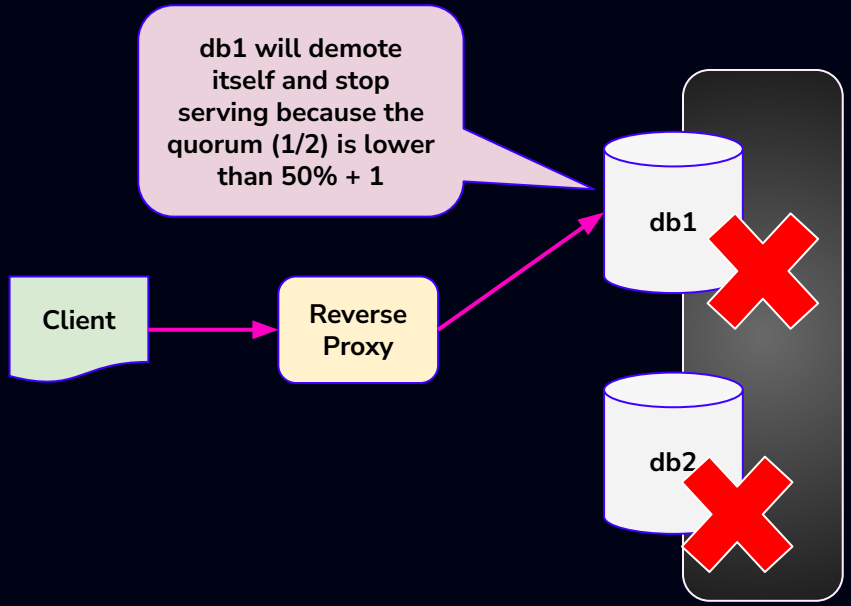
- Galera: Certified writesets queue in received queue and apply
- MGR : Certified writesets queue in relaylog and apply

Automatic Node Provisioning



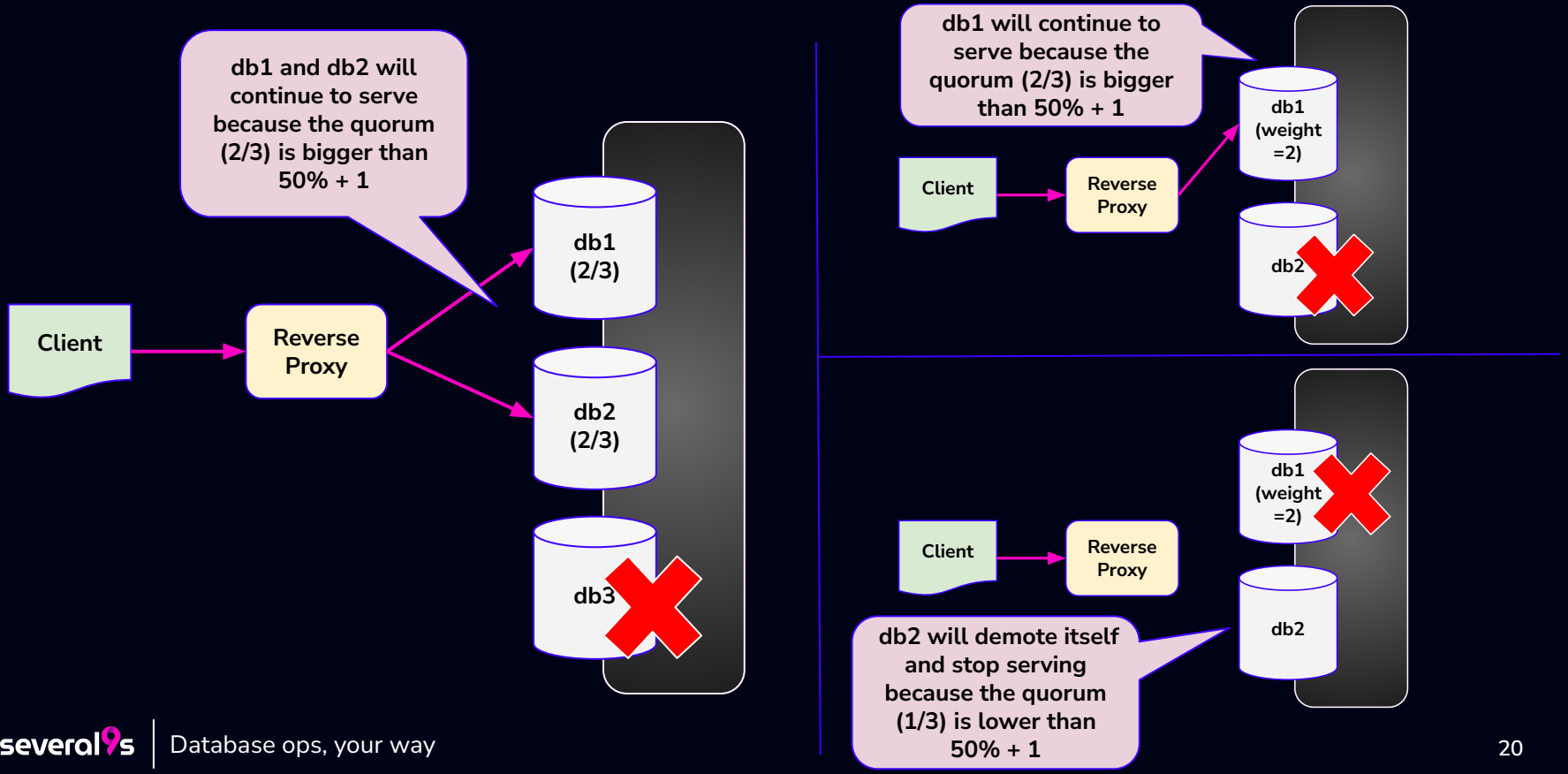
- State Snapshot Transfer (SST, a.k.a full syncing):
  - MGR: CLONE plugin
  - Galera: mysqldump, rsync, Percona Xtrabackup, MariaDB Backup
- Incremental State Transfer (IST, a.k.a delta syncing):
  - MGR: binary logs
  - Galera: Galera cache (gcache)

Quorum & Heartbeat



- Quorum is the minimum number of members required to be available (usually the majority).
- Heartbeat:
  - Galera: Virtual synchrony EVS membership protocol
  - MGR: XCom (Paxos-like) failure detection and consensus protocol
- Quorum calculation: Quorum => 50% + 1
- For 2 nodes, if 1 node goes down, quorum will be lost thus both nodes will be unavailable.
- At least 3 nodes to tolerate unavailability of 1 node unavailability (5 nodes for 2, 7 nodes for 3, and so on)
- Otherwise, use:
  - Weighted quorum (1 node = 2 votes)
  - Arbitrator node (a vote-only node)

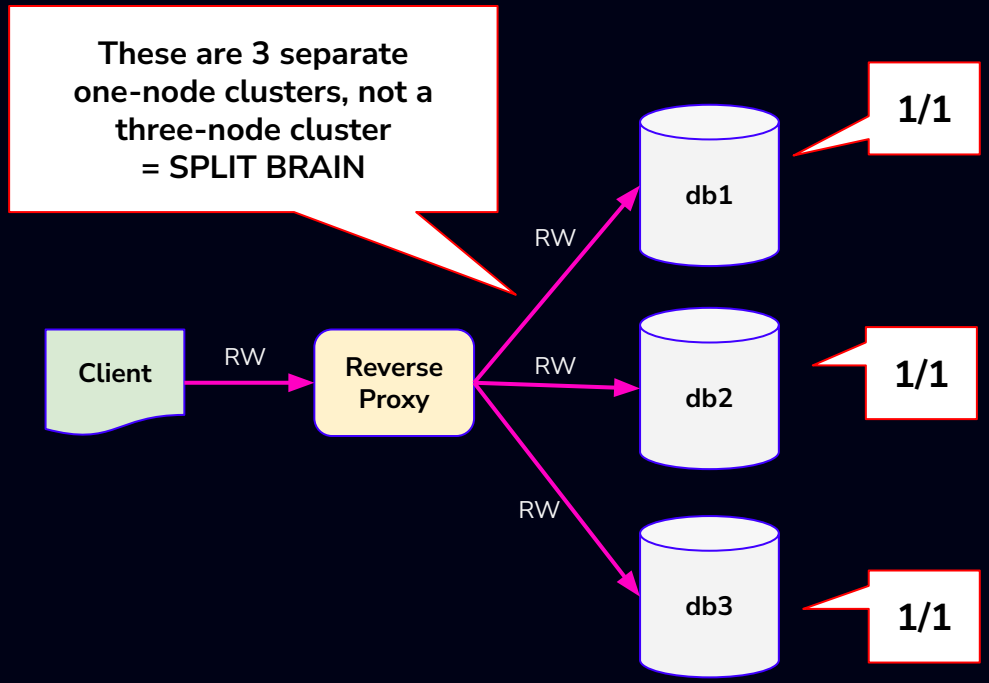
Quorum & Heartbeat



Split Brain

- Split-brain is the state when two sites are partitioned, cannot determine the quorum and both remain available:
  - Result: Data divergent. 2 different versions of data.
  - Impact: Pretty hard to rollback once happens, possibility of data loss.
- Although Galera and MGR has some kind of split brain protection, they can be misconfigured to cause a split brain.
  - Accidentally bootstrap a partitioned node into a new cluster
  - Misconfigured `wsrep_cluster_address=gcomm://` or `group_replication_bootstrap_group=ON` on all nodes.

Split Brain

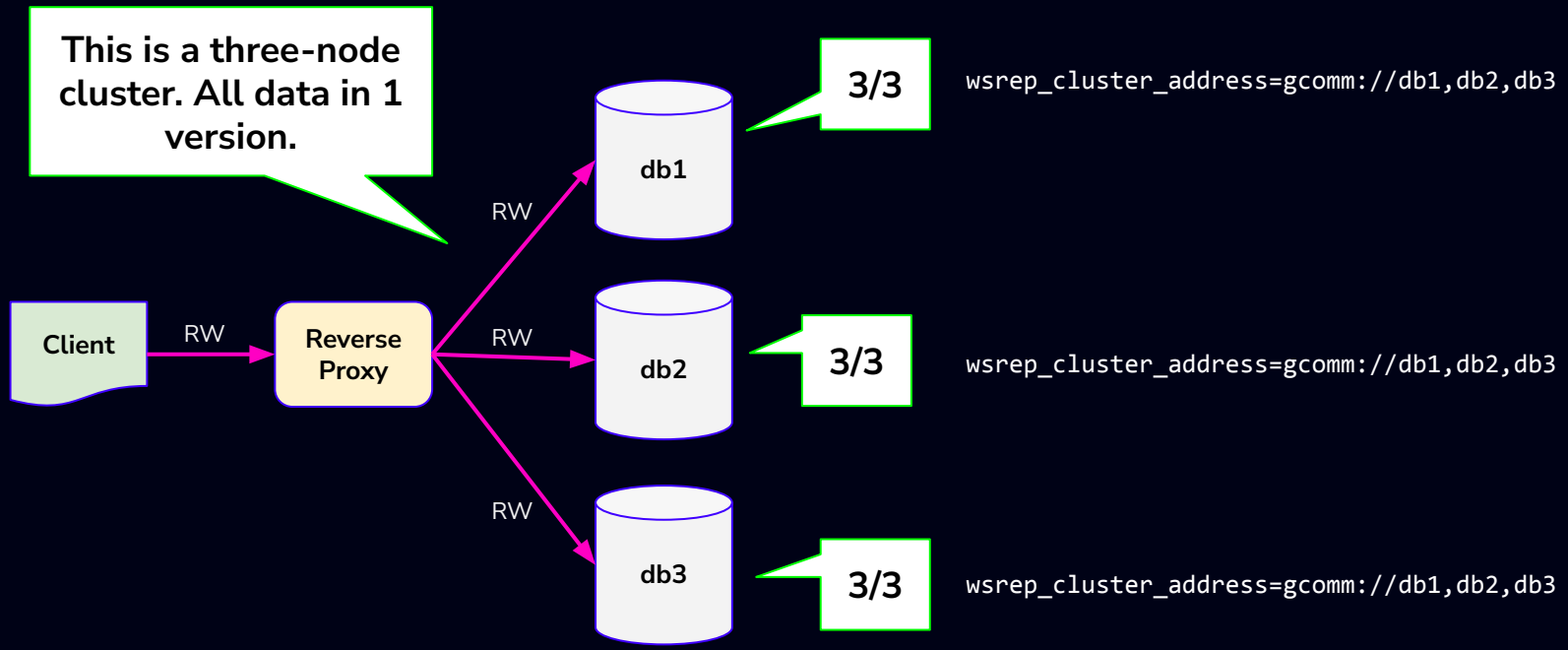


- `Galera: wsrep_cluster_address=gcomm://`
- `MGR: group_replication_bootstrap_group=ON`

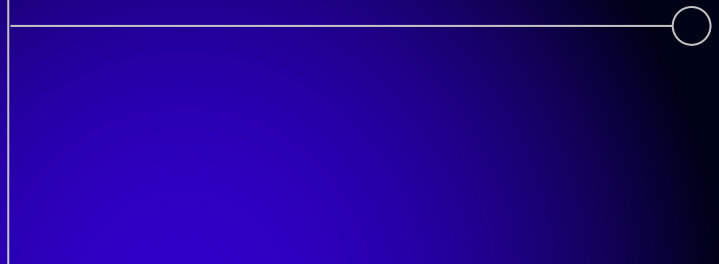
- `Galera: wsrep_cluster_address=gcomm://`
- `MGR: group_replication_bootstrap_group=ON`

- `Galera: wsrep_cluster_address=gcomm://`
- `MGR: group_replication_bootstrap_group=ON`

Split Brain


This is a three-node cluster. All data in 1 version.
Client
RW
Reverse Proxy
RW
RW
RW
db1
db2
db3
3/3
3/3
3/3
wsrep_cluster_address=gcomm://db1,db2,db3
wsrep_cluster_address=gcomm://db1,db2,db3
wsrep_cluster_address=gcomm://db1,db2,db3

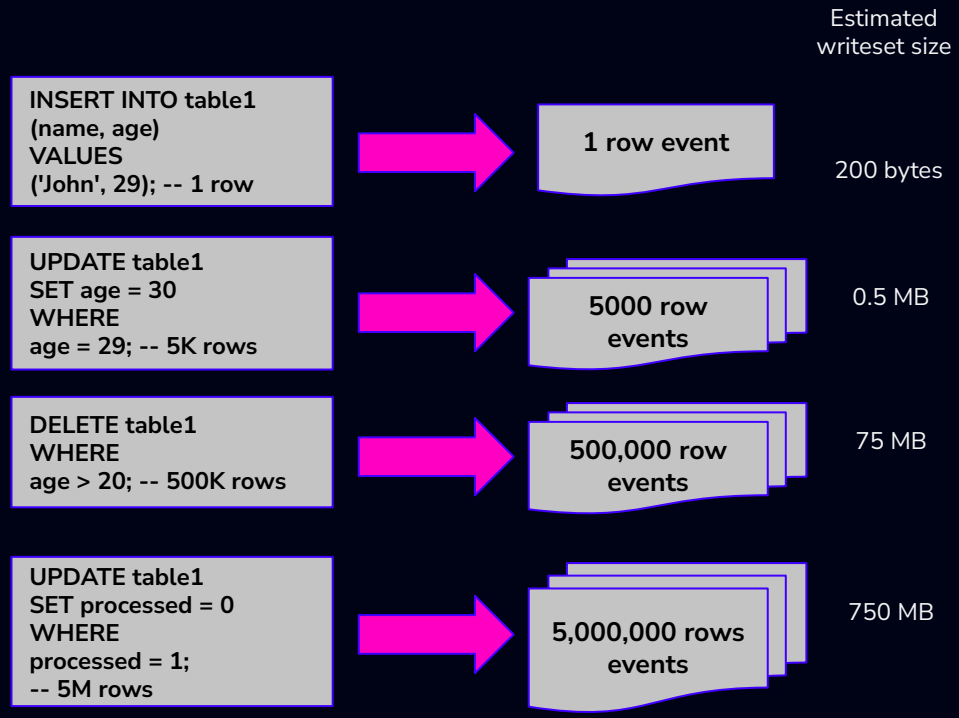
Trade Offs



several9s

Replication Payload

- A writeset of 50+ MB is huge. This can cause:
  - Replication stalls - Big writesets block certification and stall writes cluster-wide.
  - Flow control triggering (wsrep_flow_control_paused) - Nodes will stop applying until they catch up.
  - Potential transaction failure - If it exceeds wsrep_max_ws_size (default 1 GB)
- The bigger the transaction size, the higher risk it will be conflicting with other transactions that come from another primary.



Replication Payload

- Chunk up a big transaction, into a smaller transaction.
- Conflicting transactions waste server resources, plus cause a huge rollback to the originator node.
    - A ROLLBACK operation in MySQL is way slower and less optimized than a COMMIT operation.
- For huge deletes, consider using **pt-archiver** from the Percona Toolkit – a low-impact, forward-only job to nibble old data out of the table without impacting OLTP queries much.

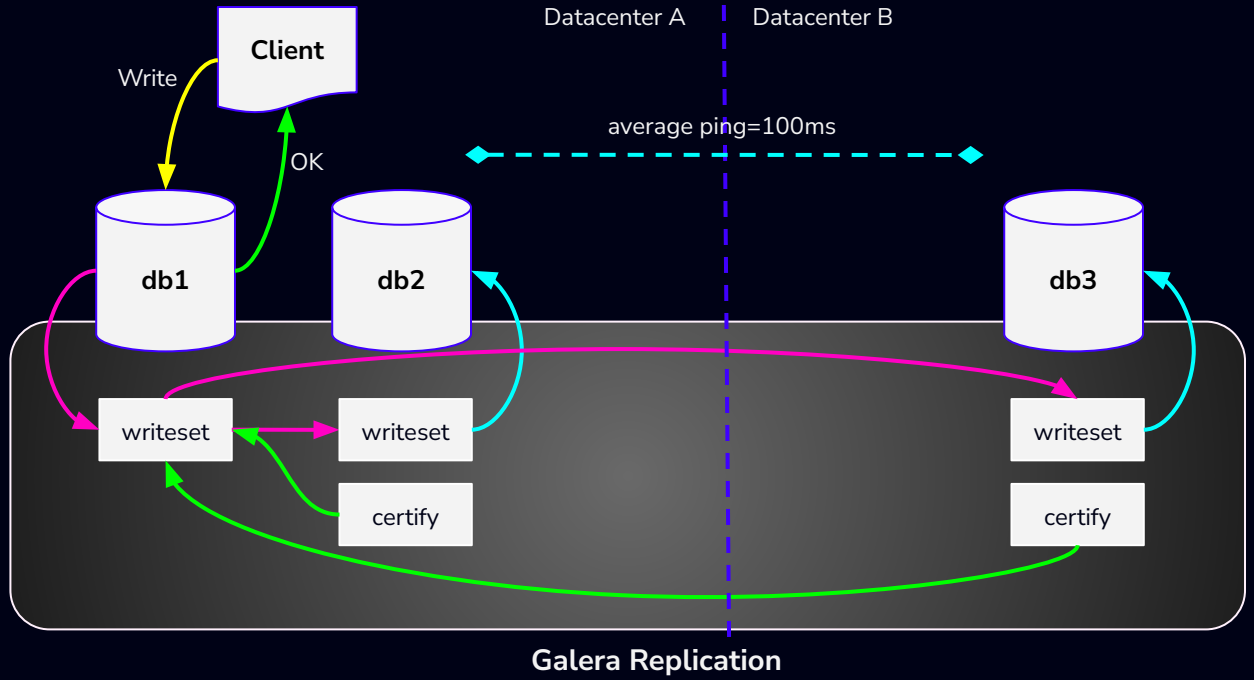
```
# A 5 million rows table. Don't do this:

mysql> UPDATE mydb.settings SET success = 1;


# Instead do this. Limit 10K rows per transaction and loop for 500x

(bash)$ for i in {1..500}; do
     mysql -uuser -p'mypassword' -e \
           "UPDATE mydb.settings \
           SET success = 1 \
           WHERE success != 1 \
           LIMIT 10000";
     sleep 2;
done
```

Round Trip Time



"A given row can't be modified more than once per RTT"

- 100ms = 0.1 seconds
- 1/0.1 = 10
- Maximum writeset replication performance = 10 writesets/second

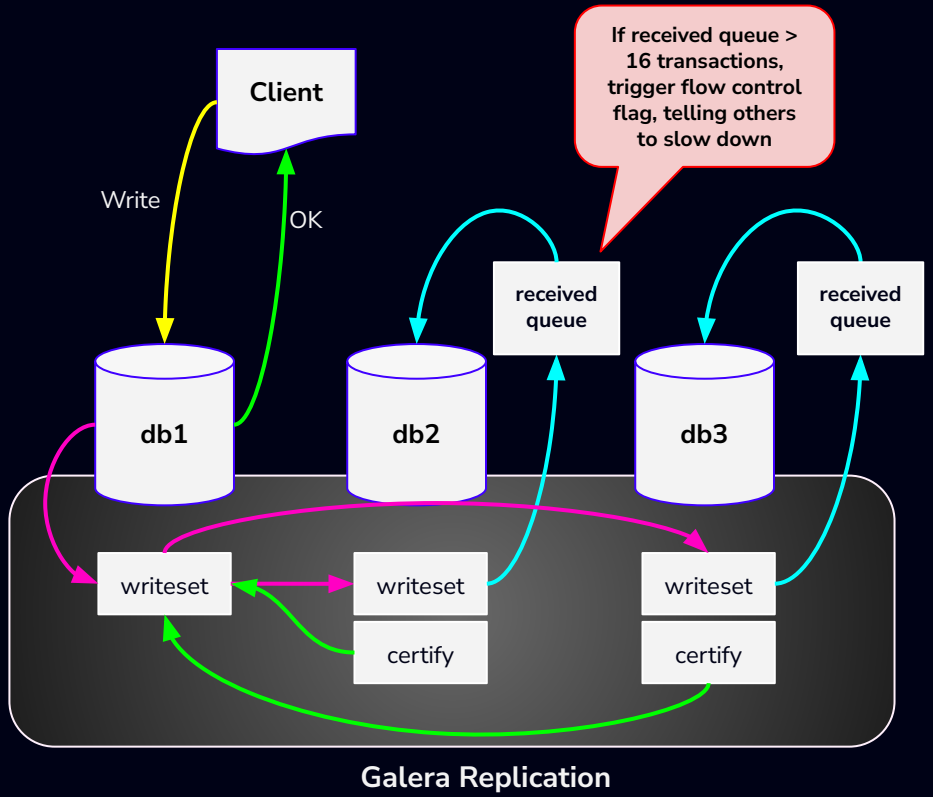
Round Trip Time

```
64 bytes from 192.168.73.13: icmp_seq=202 ttl=64 time=0.069 ms
64 bytes from 192.168.73.13: icmp_seq=203 ttl=64 time=0.069 ms
64 bytes from 192.168.73.13: icmp_seq=204 ttl=64 time=0.047 ms
^C
--- 192.168.73.13 ping statistics ---
204 packets transmitted, 204 received, 0% packet loss, time 207910ms
rtt min/avg/max/mdev = 0.034/0.057/0.104/0.011 ms
```

RTT	Value (ms)	Value (s)	Calculation	Estimated writeset replication performance (tps)
Maximum	0.104	0.000104	1 / 0.000104	9,615
Minimum	0.034	0.000034	1 / 0.000034	29,411
Average	0.057	0.000057	1 / 0.000057	17,543

Flow Control

- Flow Control allows a node to pause and resume replication according to its needs (throttling).
    - This prevents any node from lagging too far behind the others in applying transactions.
- Flow control makes replication stop, and therefore makes writes (which are synchronous) stop, on all nodes until flow control is relaxed.
- For Galera, the `fc_limit` defaults to 16 writesets. For MGR, 25% of `group_replication_flow_control_certifier_threshold=25000`
- Recommendation:
    - Uniform hardware specification on all nodes.
    - Connect an asynchronous replica for heavy read-only workloads like backup or analytics.



Causality

- High causality means enforcing order, consistency and "happens-before" guarantees between client writes and reads across the cluster.
- In short = **enforce data to synchronize, even though it could stall the cluster!**
- Useful for critical-reads (read-after-write semantics)
- You do not need to run all transactions with the same specific consistency level, especially if only some transactions actually require it

Causality Level	MySQL Group Replication	Galera Cluster
Variable name	group_replication_consistency	wsrep_sync_wait
Scope	Global, Session	Session
Supported values	• EVENTUAL • BEFORE_ON_PRIMARY_FAILOVER (default) • BEFORE • AFTER • BEFORE_AND_AFTER (strongest)	• 0 - Disabled (default) • 1 - READ (SELECT and BEGIN/START TRANSACTION). • 2 - UPDATE and DELETE; • 3 - READ, UPDATE and DELETE; • 4 - INSERT and REPLACE; • 5 - READ, INSERT and REPLACE; • 6 - UPDATE, DELETE, INSERT and REPLACE; • 7 - READ, UPDATE, DELETE, INSERT and REPLACE; • 8 - SHOW • 9 - READ and SHOW • 10 - UPDATE, DELETE and SHOW • 11 - READ, UPDATE, DELETE and SHOW • 12 - INSERT, REPLACE and SHOW • 13 - READ, INSERT, REPLACE and SHOW • 14 - UPDATE, DELETE, INSERT, REPLACE and SHOW • 15 - READ, UPDATE, DELETE, INSERT, REPLACE and SHOW (strongest)

Causality

MySQL Group Replication

```
-- client 1
> SET @@SESSION.group_replication_consistency= 'BEFORE_AND_AFTER';
> UPDATE accounts SET balance=balance+100 WHERE id=1;
> exit

-- client 2
> SET @@SESSION.group_replication_consistency= 'BEFORE_AND_AFTER';
> SELECT balance FROM accounts WHERE id=1;
> exit
```

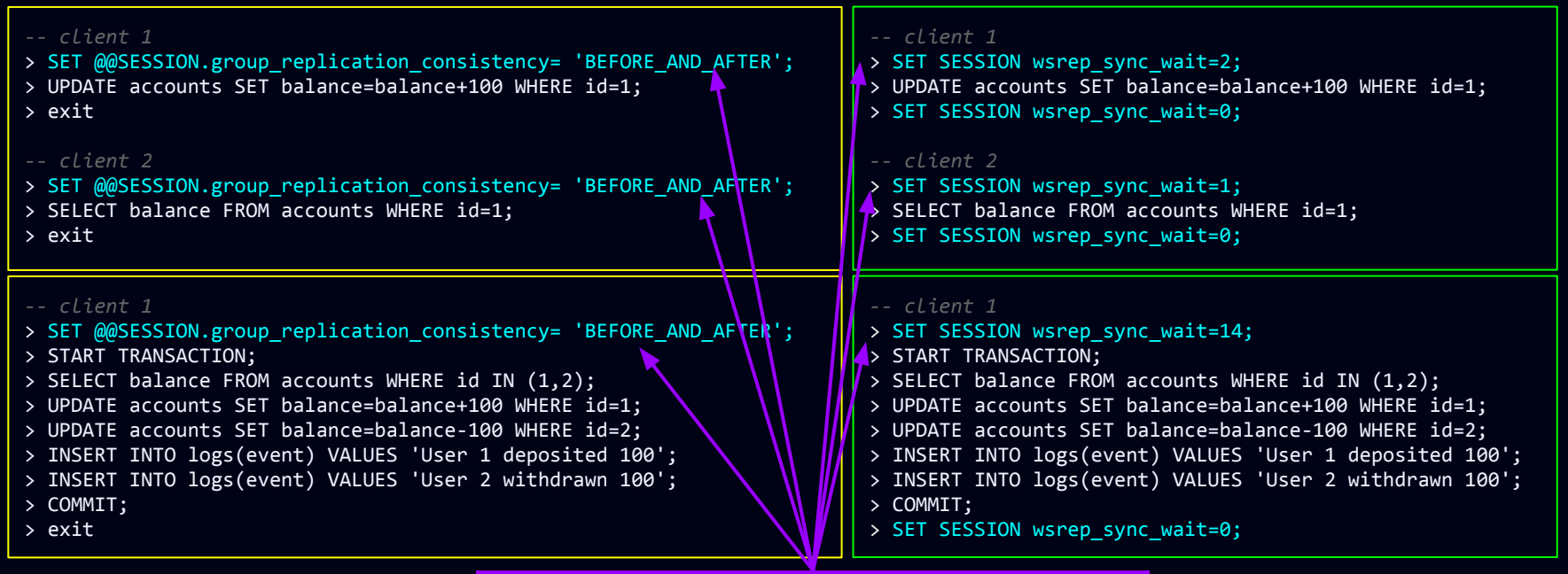
```
-- client 1
> SET @@SESSION.group_replication_consistency= 'BEFORE_AND_AFTER';
> START TRANSACTION;
> SELECT balance FROM accounts WHERE id IN (1,2);
> UPDATE accounts SET balance=balance+100 WHERE id=1;
> UPDATE accounts SET balance=balance-100 WHERE id=2;
> INSERT INTO logs(event) VALUES 'User 1 deposited 100';
> INSERT INTO logs(event) VALUES 'User 2 withdrawn 100';
> COMMIT;
> exit
```

Galera Cluster

```
-- client 1
> SET SESSION wsrep_sync_wait=2;
> UPDATE accounts SET balance=balance+100 WHERE id=1;
> SET SESSION wsrep_sync_wait=0;

-- client 2
> SET SESSION wsrep_sync_wait=1;
> SELECT balance FROM accounts WHERE id=1;
> SET SESSION wsrep_sync_wait=0;
```

```
-- client 1
> SET SESSION wsrep_sync_wait=14;
> START TRANSACTION;
> SELECT balance FROM accounts WHERE id IN (1,2);
> UPDATE accounts SET balance=balance+100 WHERE id=1;
> UPDATE accounts SET balance=balance-100 WHERE id=2;
> INSERT INTO logs(event) VALUES 'User 1 deposited 100';
> INSERT INTO logs(event) VALUES 'User 2 withdrawn 100';
> COMMIT;
> SET SESSION wsrep_sync_wait=0;
```

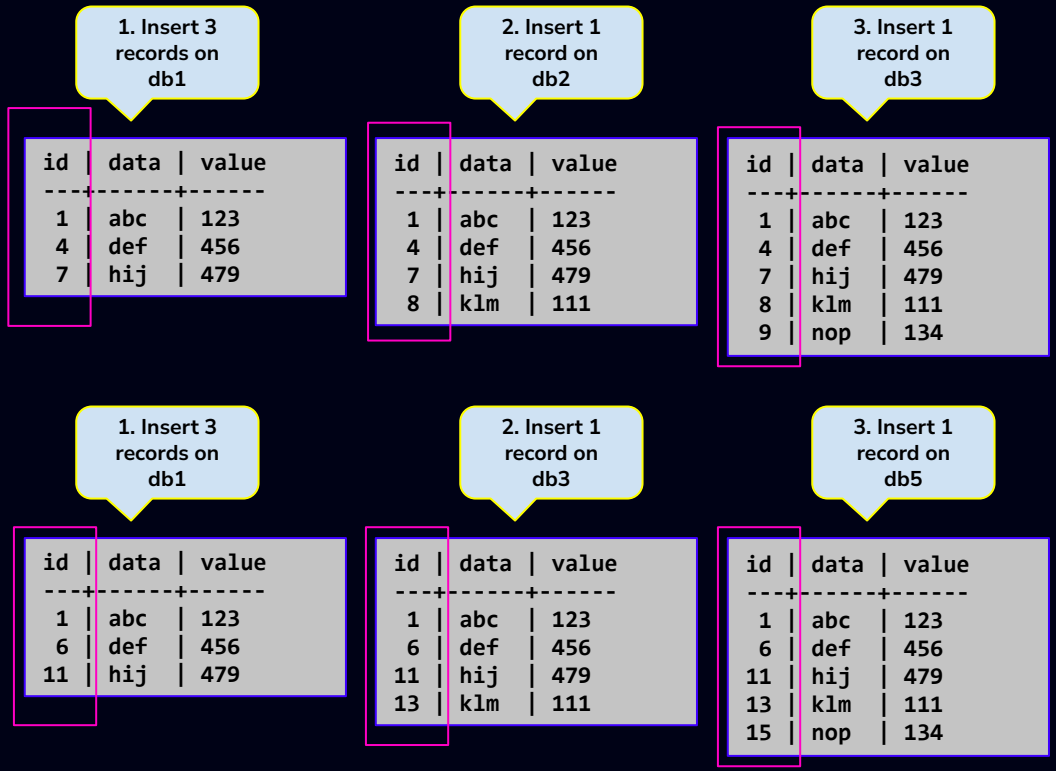


Wait for all nodes to sync first. Other writes will be blocked.
This can result in higher latency for the next writesets.

Non-Sequential Auto Increment

To avoid primary key (PK) collisions for multi-primary replication:

- Galera:
  - Uses auto-increment offset algorithm: `auto_increment_increment`, `auto_increment_offset`
- MGR:
  - Uses a slightly different approaches: `group_replication_auto_increment_increment=7`
  - `AUTO_INCREMENT` is sequential in single-primary mode.



Online Schema Upgrade (DDL)

Cluster type	OSU method	Example steps	Remarks
Galera Cluster	Total Order Isolation (TOI)	`> ALTER TABLE ..`	• Default DDL method. • DDL is processed in the same order regarding other transactions, guaranteeing data consistency. • This will block whole cluster (pause) until the operation completes.

	Rolling Schema Upgrade (RSU)	`> SET wsrep_osu_method='RSU';`
		`> ALTER TABLE ..`
		`> exit`
		`# repeat on the next node, one node at a time`
			• DDL processing is only done locally on the node, and the user needs perform the changes manually on each node.

	Non-Blocking Operation (NBO)	`> SET wsrep_osu_method='NBO';` `> ALTER TABLE ..`	• Similar to TOI, but only perform metadata lock (not cluster-wide). • Only block the table that being altered.

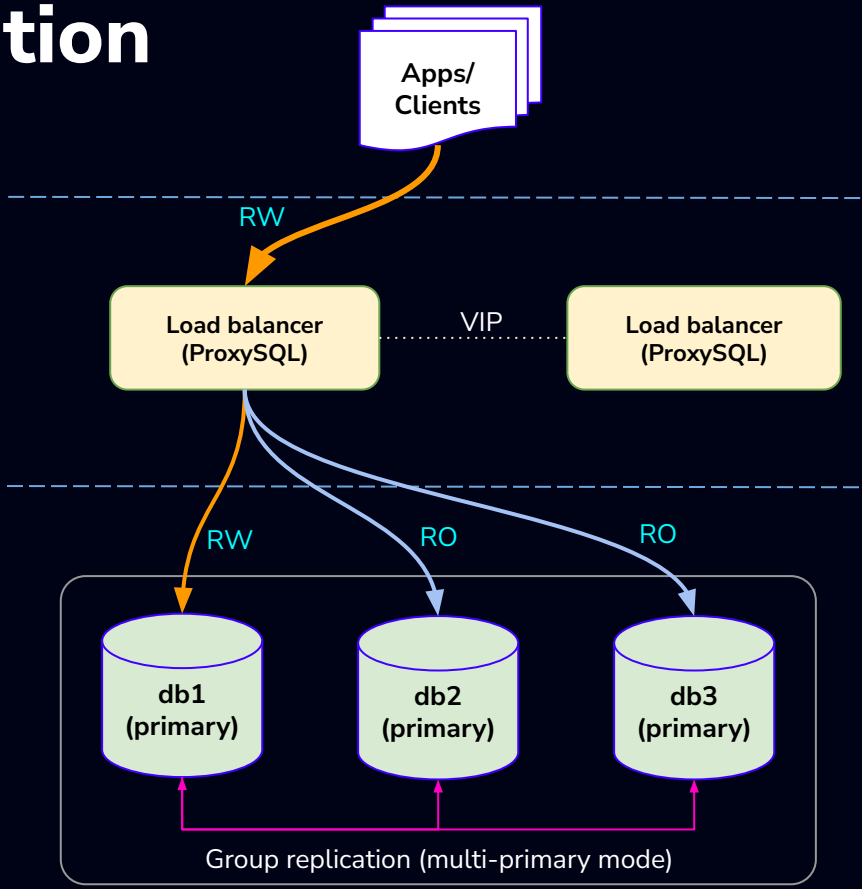
MGR has no special control over schema upgrade operation. Perform DDL only in single-primary cluster mode or ensure no DML running against the table while performing DDL operation in multi-primary cluster.

Example Architecture

several9s

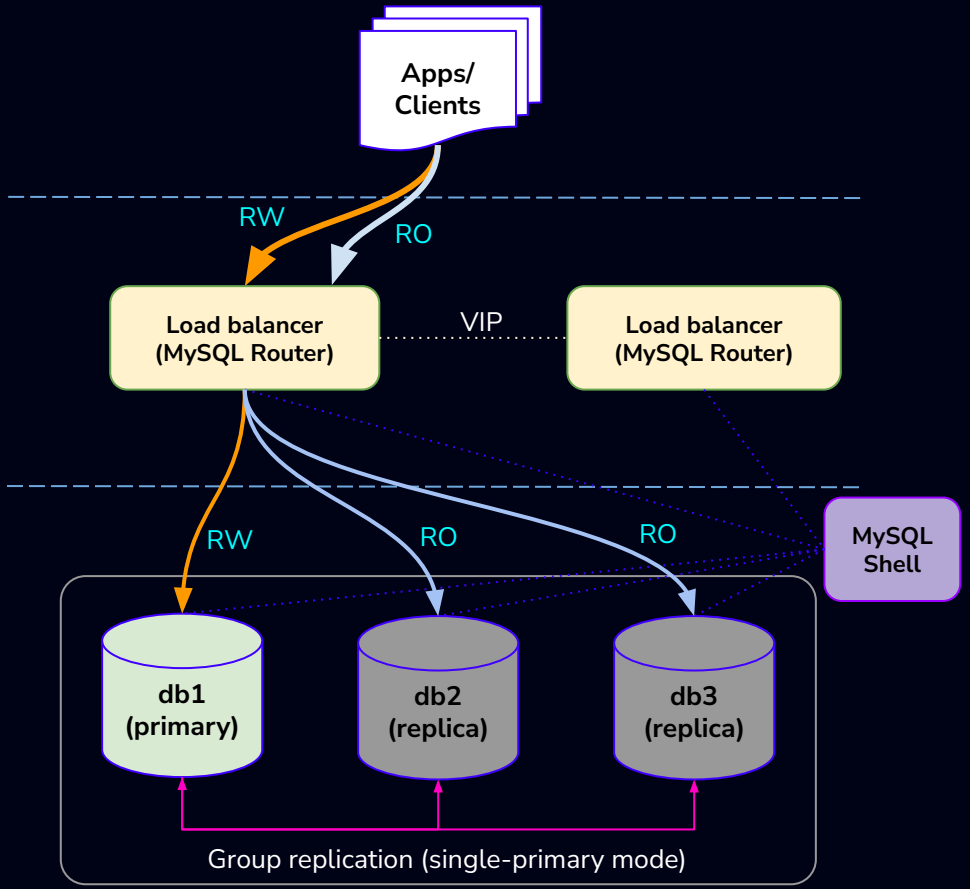
MySQL Group Replication with ProxySQL

- Apps/clients connect to the database via a load balancer (ProxySQL) - port 6033
- 2 load balancers tie with a virtual IP address (VIP) on a separate tier
- ProxySQL performs read-write splitting, forwarding writes to only one primary node. Reads will be distributed among all available nodes.

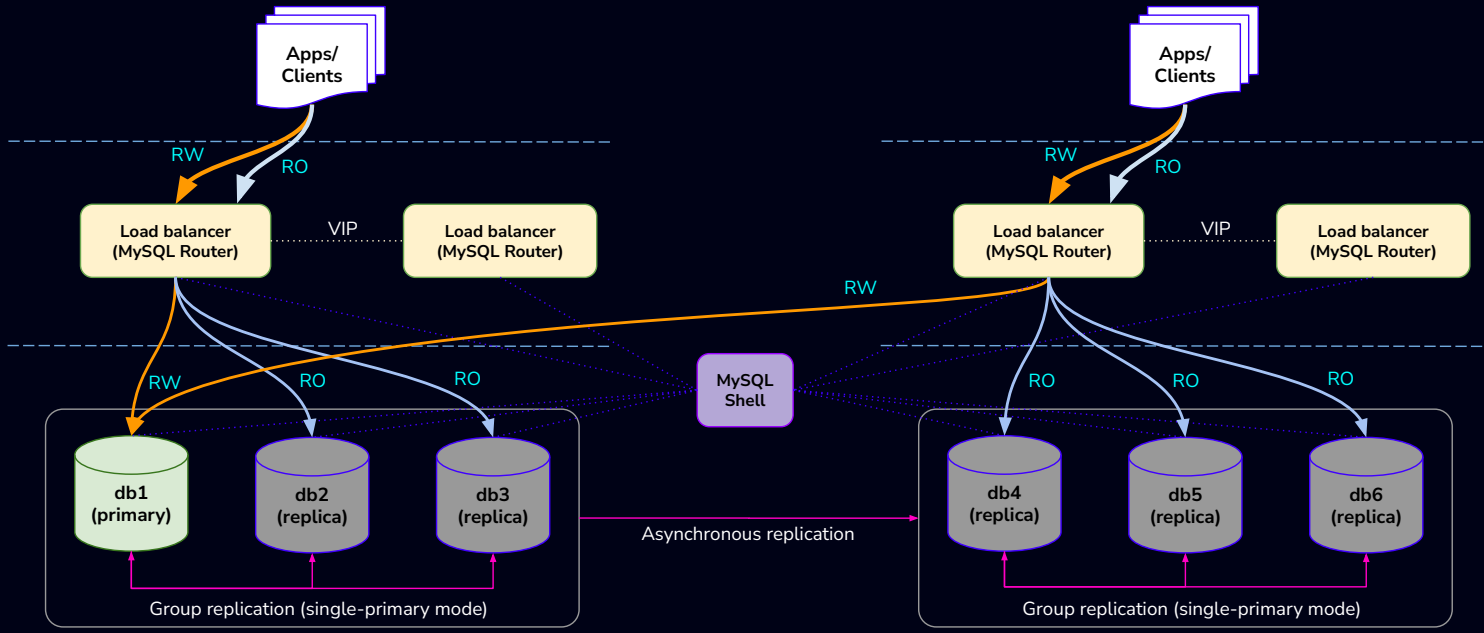


MySQL InnoDB Cluster

- Apps/clients connect to the database via a load balancer (MySQL Router):
  - RW - port 6446
  - RO - port 6447
- 2 load balancers tie with a virtual IP address (VIP) on a separate tier
- MySQL Router forwards the connections to the respective backends:
  - Port 6446 - to the primary
  - Port 6447 - to the replicas
- MySQL Shell - automates cluster creation, configuration validation, node provisioning, distributed recovery, failover coordination, metadata management, and integration with MySQL Router.

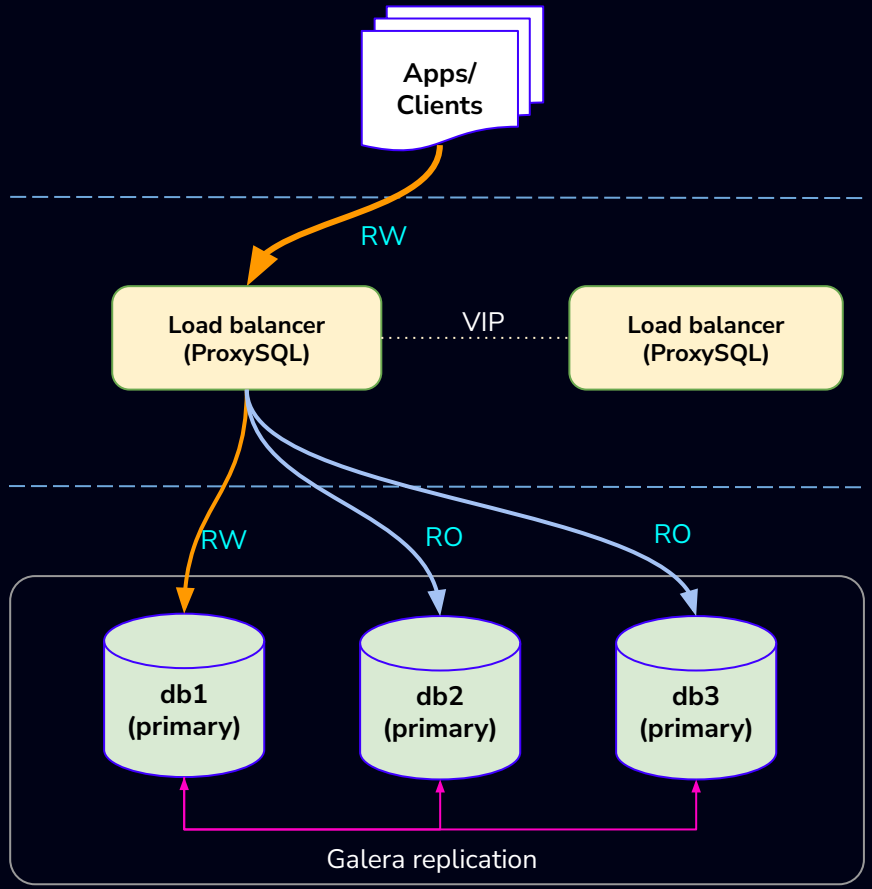


MySQL InnoDB ClusterSet



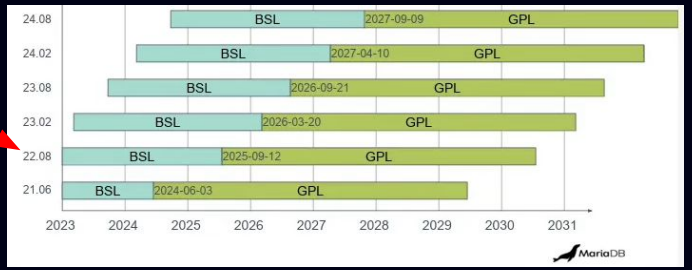
Percona XtraDB Cluster with ProxySQL

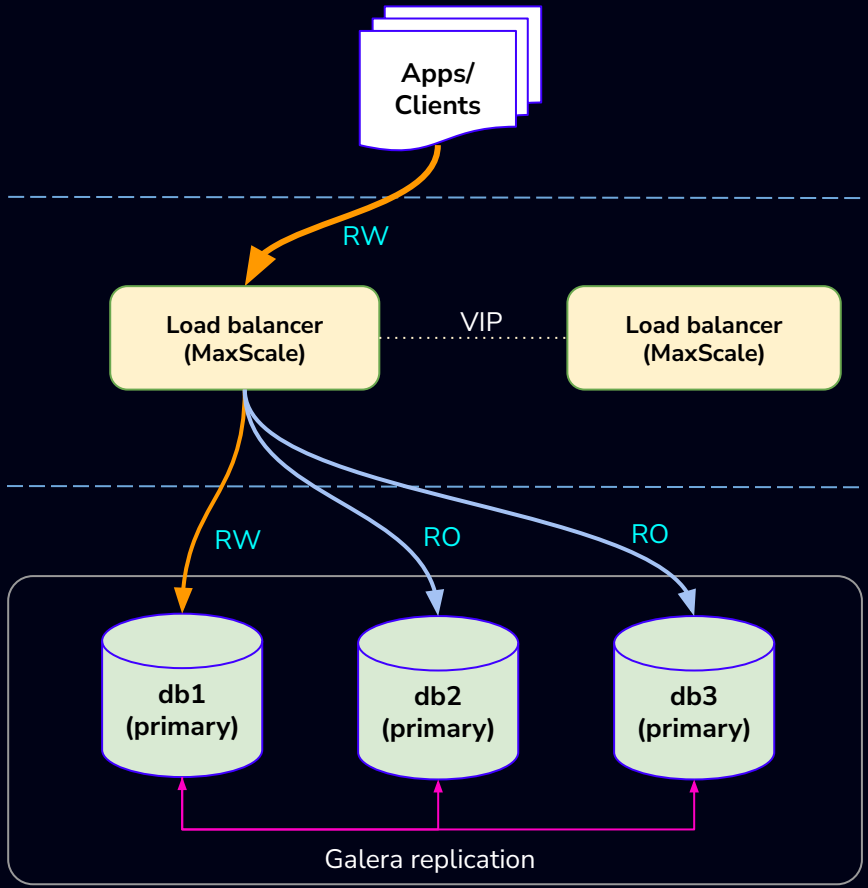
- Apps/clients connect to the database via a load balancer (ProxySQL) - port 6033
- 2 load balancers tie with a virtual IP address (VIP) on a separate tier
- ProxySQL performs read-write splitting, forwarding writes to only one primary node. Reads will be distributed among all available nodes.



MariaDB Cluster with MaxScale

- Apps/clients connect to the database via a load balancer (MaxScale) - port 4006
- 2 load balancers tie with a virtual IP address (VIP) on a separate tier
- MaxScale performs read-write splitting, forwarding writes to only one node. Reads will be distributed among all available nodes.





Use Cases

severalnines

MySQL Group Replication

- Suitable for standard online transactional processing (OLTP), with high availability requirements and small writes per transaction.
- If you need a multi-master database solution run on Windows, Solaris or MacOS (MariaDB also support Windows).
- “Official MySQL” integrated solution. If you need Oracle's brand and support for your database needs.
- If you are required to use enterprise features like Enterprise Backup (MEB), Enterprise Audit, Enterprise Firewall, Enterprise Encryption, Transparent Data Encryption, Enterprise Authentication plugins, etc.
- If you need to communicate using X protocol (33060), AdminAPI, etc.
- If you want to have WAN replication using ClusterSet (single-writer setup).
- Safety over performance. MGR is stricter which means less weird edge cases.
- Mostly read-intensive workloads.

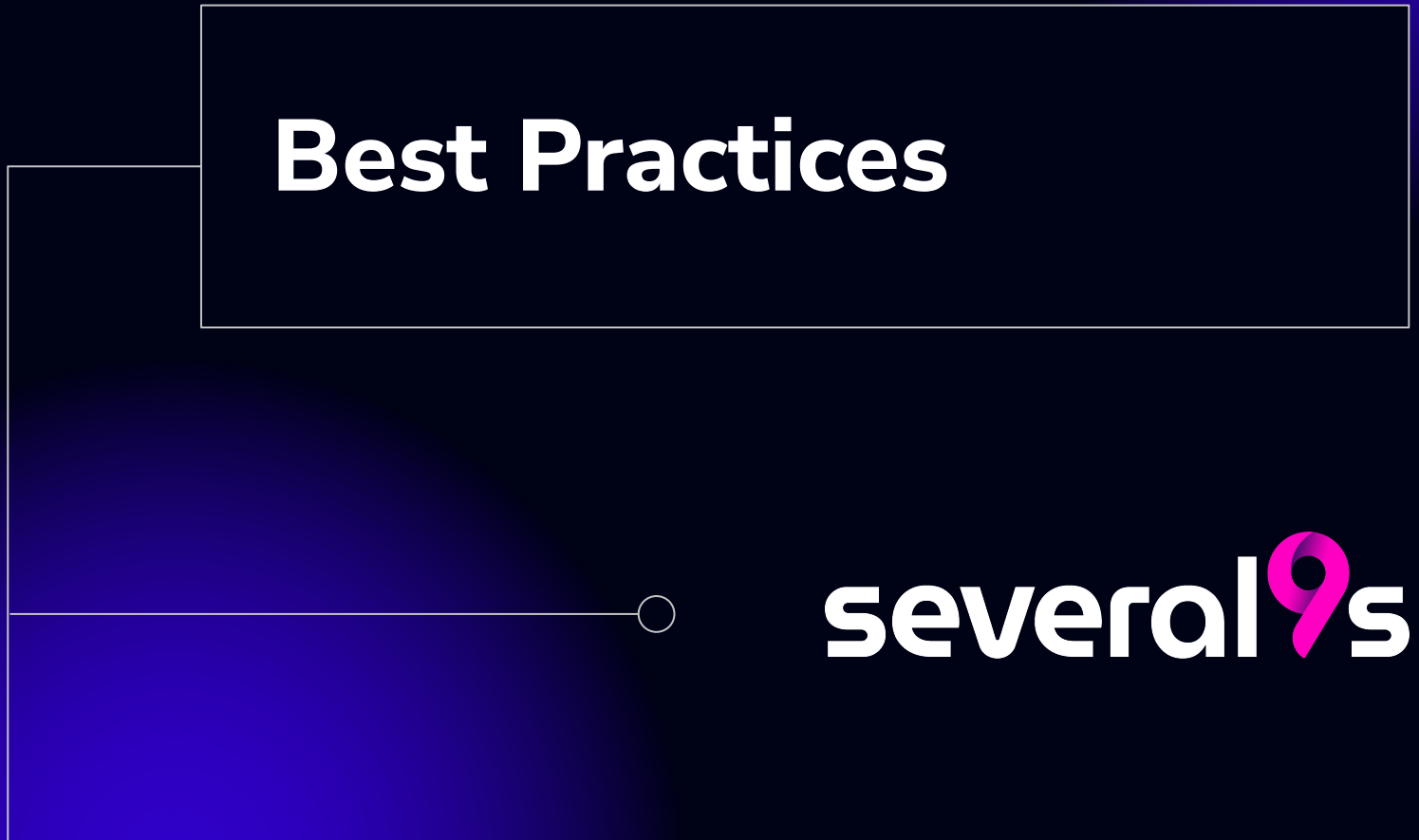
Galera Cluster

- Best for heavy online transactional processing (OLTP), with high availability requirements and small writes per transaction. E.g, e-commerce, ticketing systems, etc.
- If you want a more relaxed MySQL compatibility and highly customizable topology configuration.
- If you need to communicate using X protocol (33060), use Percona XtraDB Cluster. MariaDB Cluster only supports the classic MySQL protocol (3306).
- If you want to have multi-master WAN replication between sites. MGR only supports single-writer for InnoDB ClusterSet.
- Generally better replication performance if compared to MGR.

Comparison: MGR vs Galera Replication

Aspect	MySQL Group Replication	Galera Replication
Multi-master support	• Designed with single-primary mode as the default. • Multi-primary is "best effort", not optimized for conflict-heavy workloads.	• Built for true multi-primary. • Use deterministic global assignment via increments.
Group communication	• GCS (Paxos algorithm)	• Virtual synchrony QoS Totem Single-ring Order Protocol
InnoDB	• Use InnoDB High Priority Transaction	• Patch MySQL to kill transaction
State Transfer	• SST: CLONE plugin • IST: binary logs (it is called Automated Distributed Recovery)	• SST: mysqldump/rsync/Percona Xtrabackup/MariaDB backup • IST: Galera cache (gcache)
Monitoring	• Performance_schema	• show status like 'wsrep%';
Arbitrator	• Not supported. Must be a DB node.	• garbd (vote-only, no data)

Best Practices



Best Practices

Recommendation	Description
3 nodes + 1 read-only replica	Set up a read-only replica (asynchronous replication) for heavy read-only workloads like backups, analytics and reporting. Let the cluster focus on OLTP performance.
Use load balancer	• ProxySQL for Percona XtraDB Cluster and Group Replication • MariaDB MaxScale for MariaDB • MySQL Router (part of MySQL InnoDB Cluster) • HAProxy • Keepalived provides virtual IP address between multiple load balancers
Enable binary logging on all nodes	Galera Cluster does not have this enabled by default. Binary logging allows point-in-time recovery (PITR) and possibility to scale out with read-only replica.
<5 ms strongly recommended	The lower the network latency, the better. If higher than 5 ms, most likely asynchronous replication performs or tolerates better.
Single writer	Only one primary processing all writes, to reduce certification failure rates. More predictable performance. Reads can be scaled easily.

Common Misconceptions [1/2]

Misconception	Reality
Adding more database servers will scale up the write performance	Galera Cluster cannot scale out writes to the same degree as it scales reads, because all writes must be applied to all nodes to maintain consistency.
2 x MySQL hosts with ring replication is a multi-primary replication	2 x MySQL with ring replication is primary-to-primary replication. It does not have conflict resolution, failure detection and group membership control.
Any database table can be used in MGR/Galera.	Only InnoDB or XtraDB storage engine with a primary key for proper replication. Non-transactional storage engines like MyISAM are not supported and tables without a primary key can cause issues.
Restoring a database into the cluster is faster than into a standalone node	Restoring a database into a running cluster is significantly slower because every node must receive and process the changes to stay synchronized.
Galera Cluster uses MySQL binlogs for Galera replication	Galera Cluster does not use the standard MySQL binary log for replication; it uses its own write-set replication API. MGR uses the standard MySQL binary log.

Common Misconceptions [2/2]

Misconception	Reality
Large transactions are not an issue in Galera Cluster.	Large transactions can cause performance problems, including conflict rates, increased memory usage, and cluster freezing.
MariaDB is 100% compatible with MySQL.	They have started to diverge since MariaDB 5.5, only compatible via MySQL classic protocol (3306). MySQL has introduced a new X protocol (33060), where new features like Group Replication and MySQL Shell are built on.
If my network RTT is 100ms, the cluster can only perform 10 writes/second.	In a transaction, you can have multiple writes per transaction commit. If the transaction has 10 writes (INSERT/UPDATE/DELETE), you can now run 100 writes/s.
I can perform schema change on my cluster whenever I like.	DDL statements (ALTER/CREATE/DROP/TRUNCATE) are replicated differently in MGR/Galera. The certification process does not certify DDL. It just orders it to be executed at the same ordering on every member. DDL will lock the whole cluster (read-only, can't write until the operation completes).
All queries (DDL, DML, DCL) are replicated and certified as a writeset.	Only DML write queries (INSERT, UPDATE, DELETE, REPLACE) are replicated and certified in writesets. DDL and DCL are replicated as STATEMENT.

Summary

- Multi-primary is not a performance feature - It increases write availability, not throughput.
- Latency determines throughput.
- Conflicts shape scalability - Higher write concurrency increases certification conflicts and rollbacks.
- Consistency is strong, but conditional - Both MGR and Galera ensure consistent certified transactions, but network issues can still cause divergence or flow-control stalls.
- MGR vs Galera in practice:
    - MGR: Stricter consistency handling, tighter MySQL integration, slower in multi-primary.
    - Galera: More mature multi-primary design, lower LAN latency, widely adopted if compared to MGR.
- Choosing the right one:
    - MGR for official MySQL environments, predictable certification behavior.
    - Galera for high-concurrency, low-latency clusters.

Resources

- Online Schema Upgrade in MySQL Galera Cluster Using RSU Method
- Schema Upgrades in Galera Cluster – How to Avoid RSU Locks
- Online Schema Upgrade in MySQL Galera Cluster using TOI Method

Q&A

Thank you!