Galera Cluster

Seppo Jaakola
Codership

codership

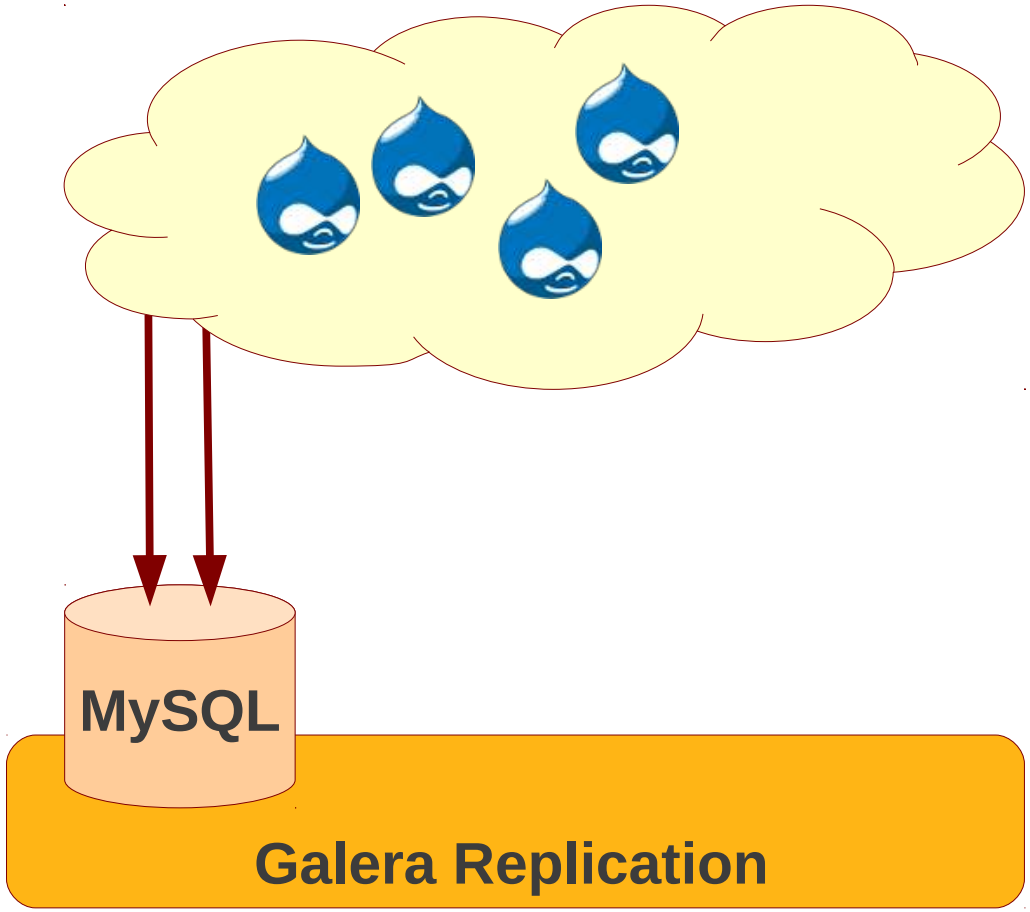
Agenda

- Galera Cluster Short Introduction
- Multi-master Conflicts
- State Transfers (SST IST)
- Backups
- Schema Upgrades
- Galera Project

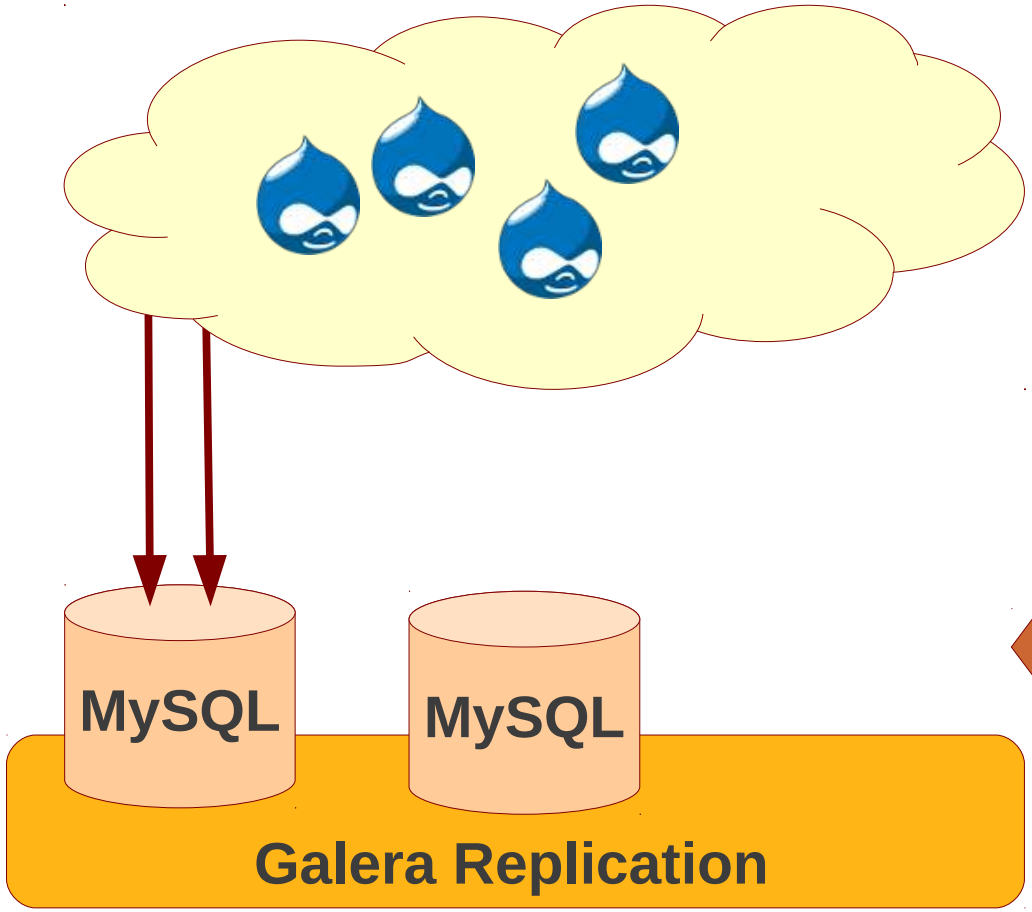
Galera Cluster

codership

Multi-Master Replication

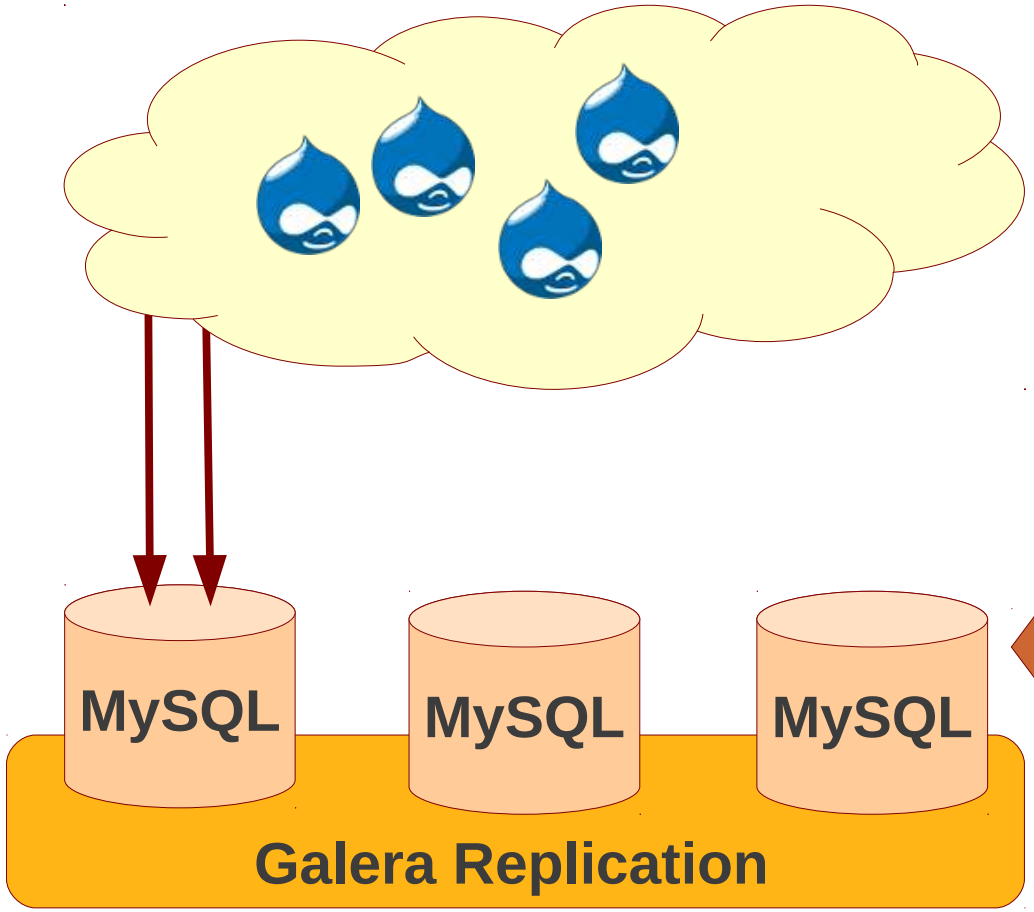


Multi-Master Replication



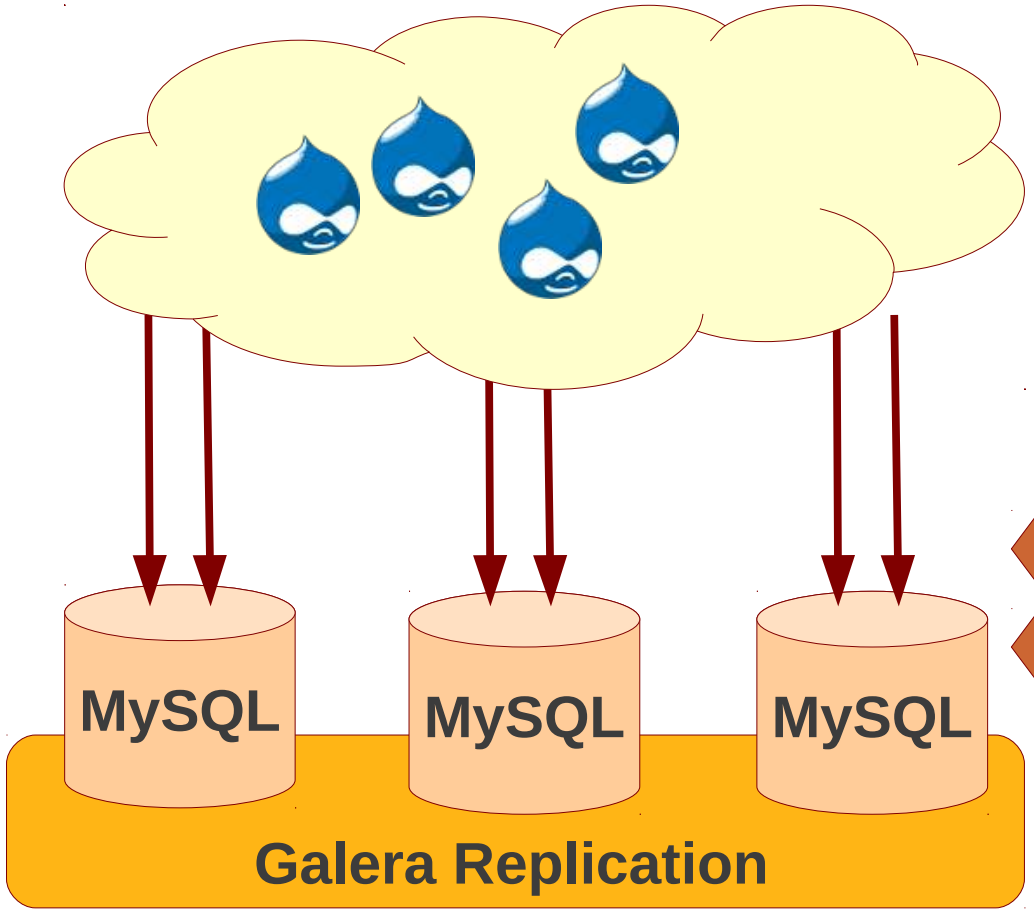
There can be several nodes

Multi-Master Replication



There can be several nodes

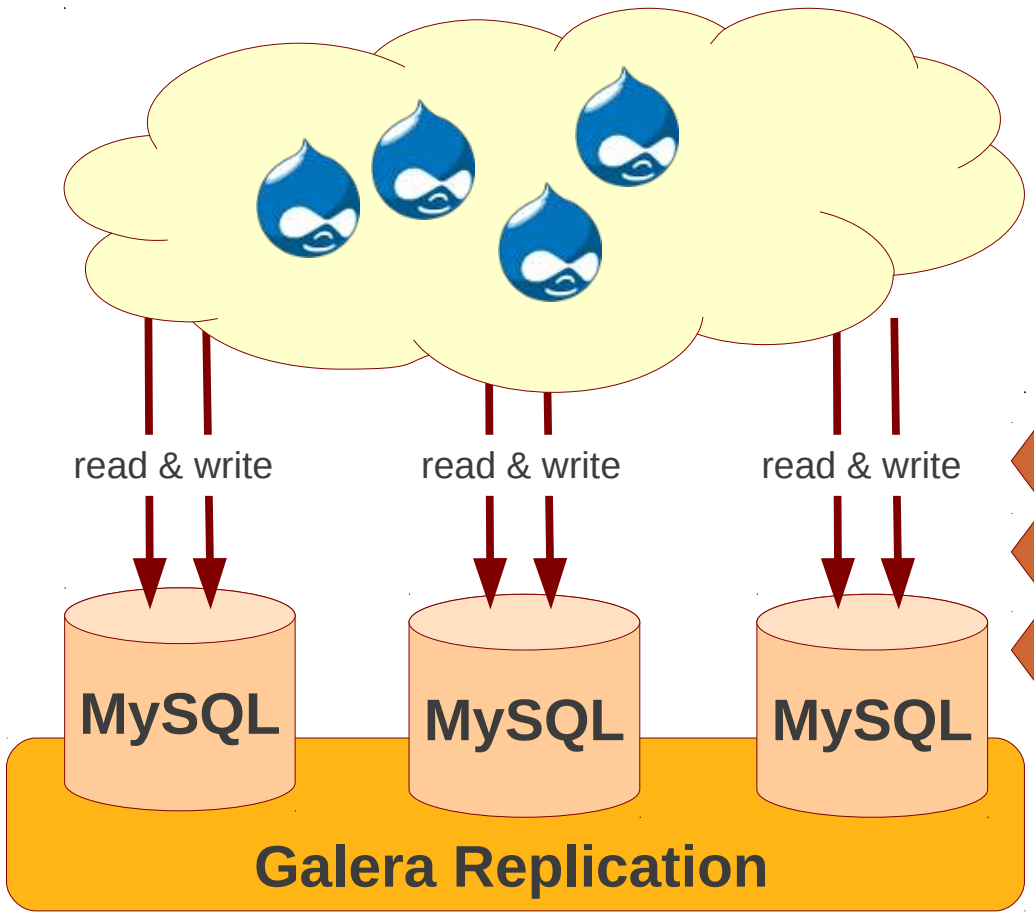
Multi-Master Replication



Client can connect to any node

There can be several nodes

Multi-Master Replication

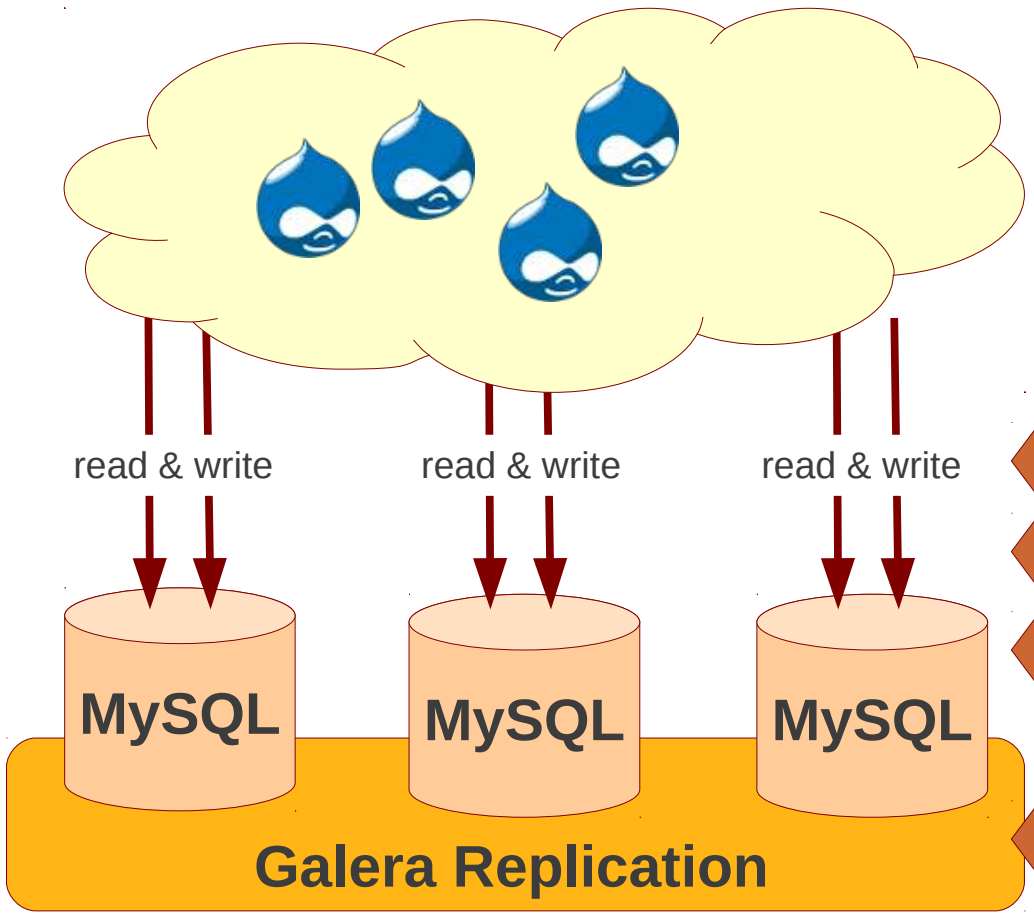


Read & write access to any node

Client can connect to any node

There can be several nodes

Multi-Master Replication



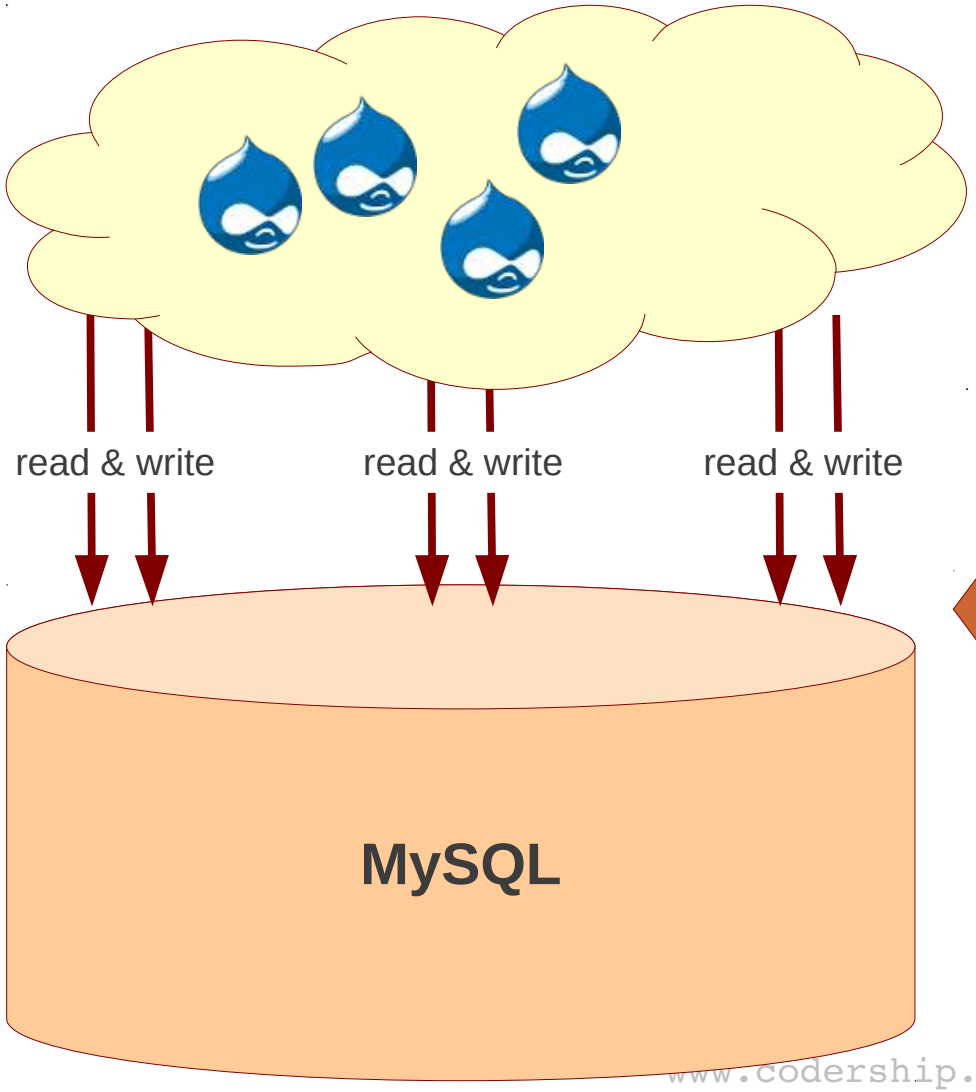
Read & write access to any node

Client can connect to any node

There can be several nodes

Replication is synchronous

Multi-Master Replication

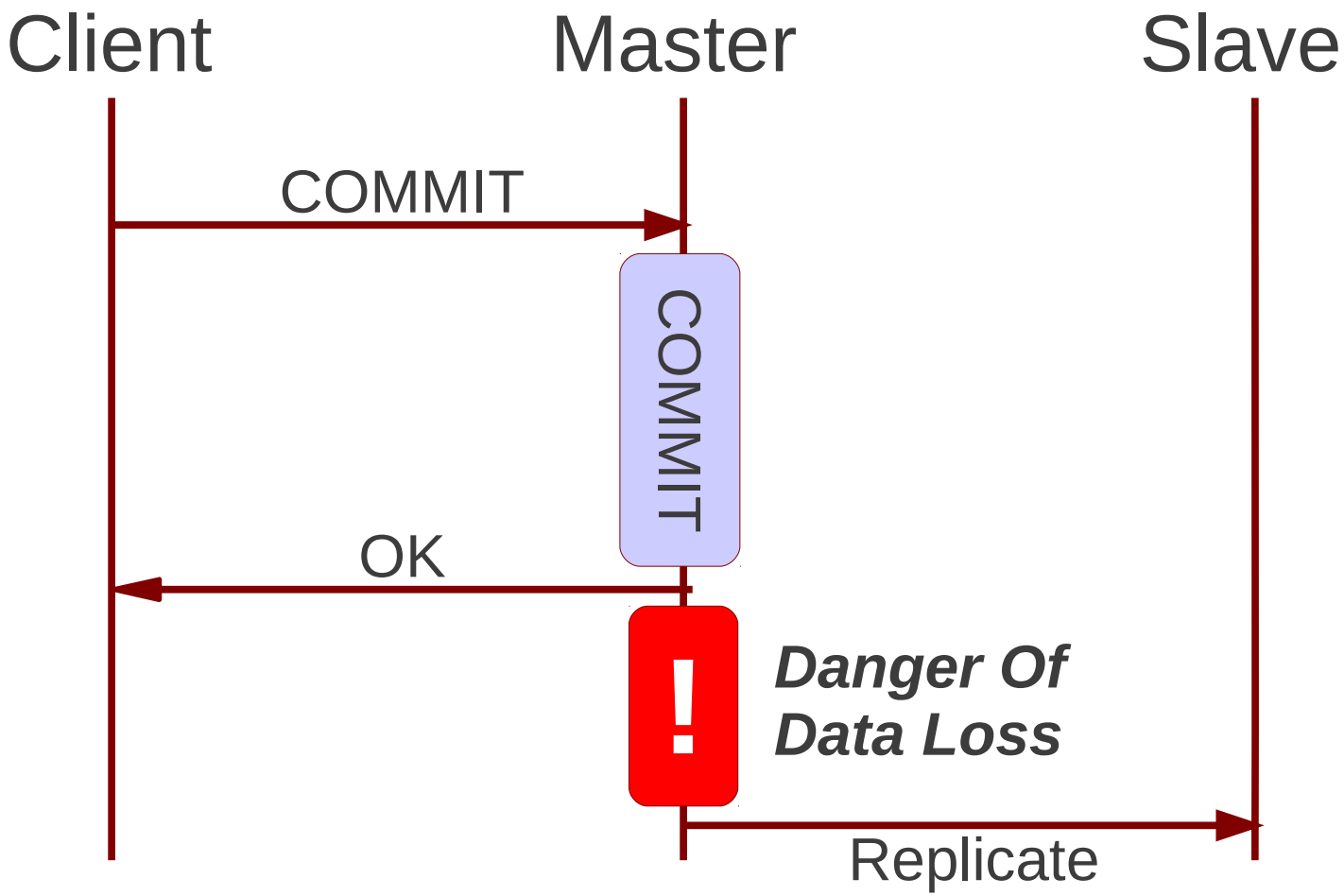


Multi-master cluster looks like one big database with multiple entry points

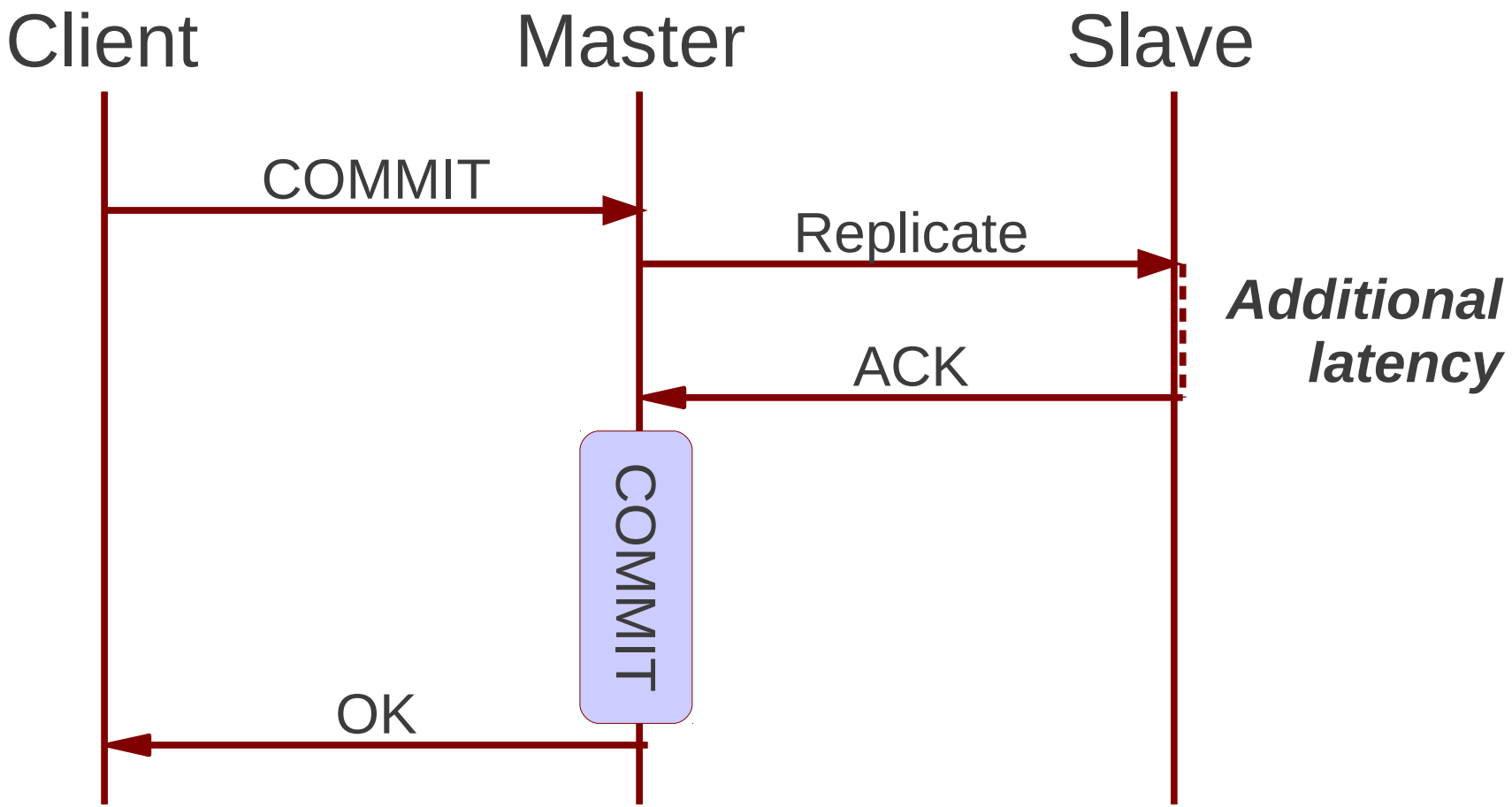
Galera Cluster

- Synchronous multi-master cluster
- For MySQL/InnoDB
- 3 or more nodes needed for HA
- Automatic node provisioning
- Works in LAN / WAN / Cloud

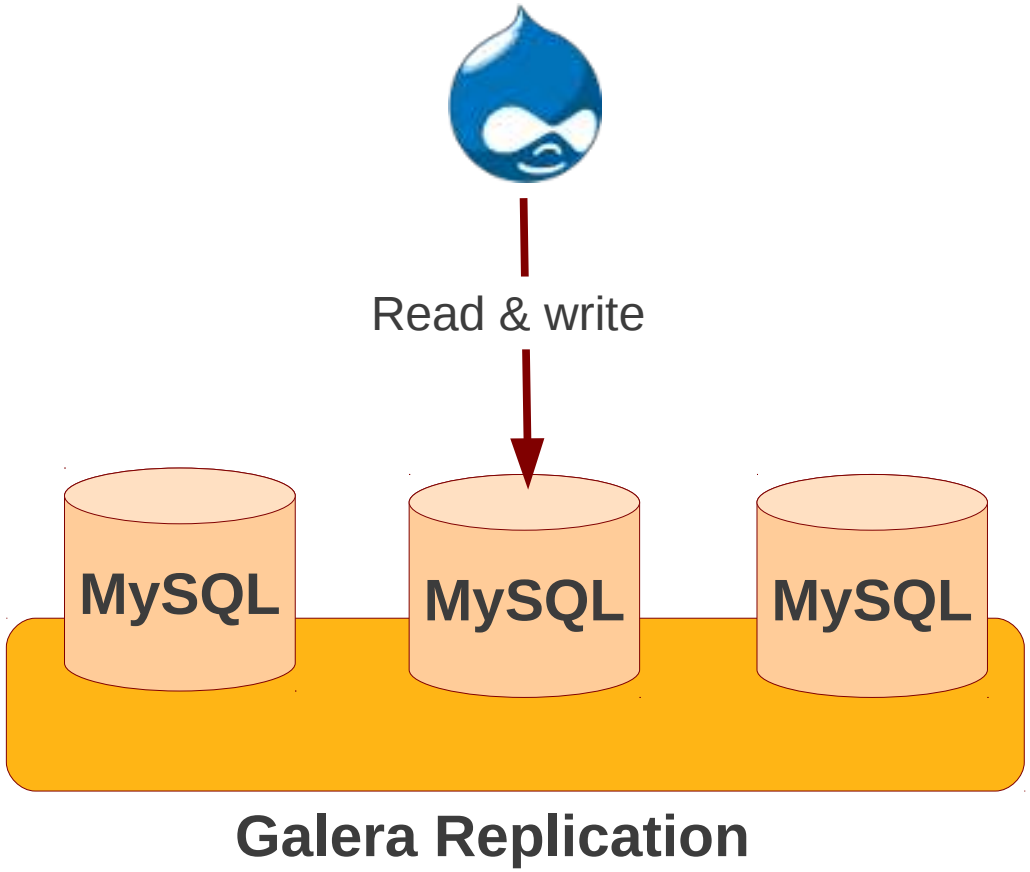
What Is Asynchronous Replication?



What Is Synchronous Replication?

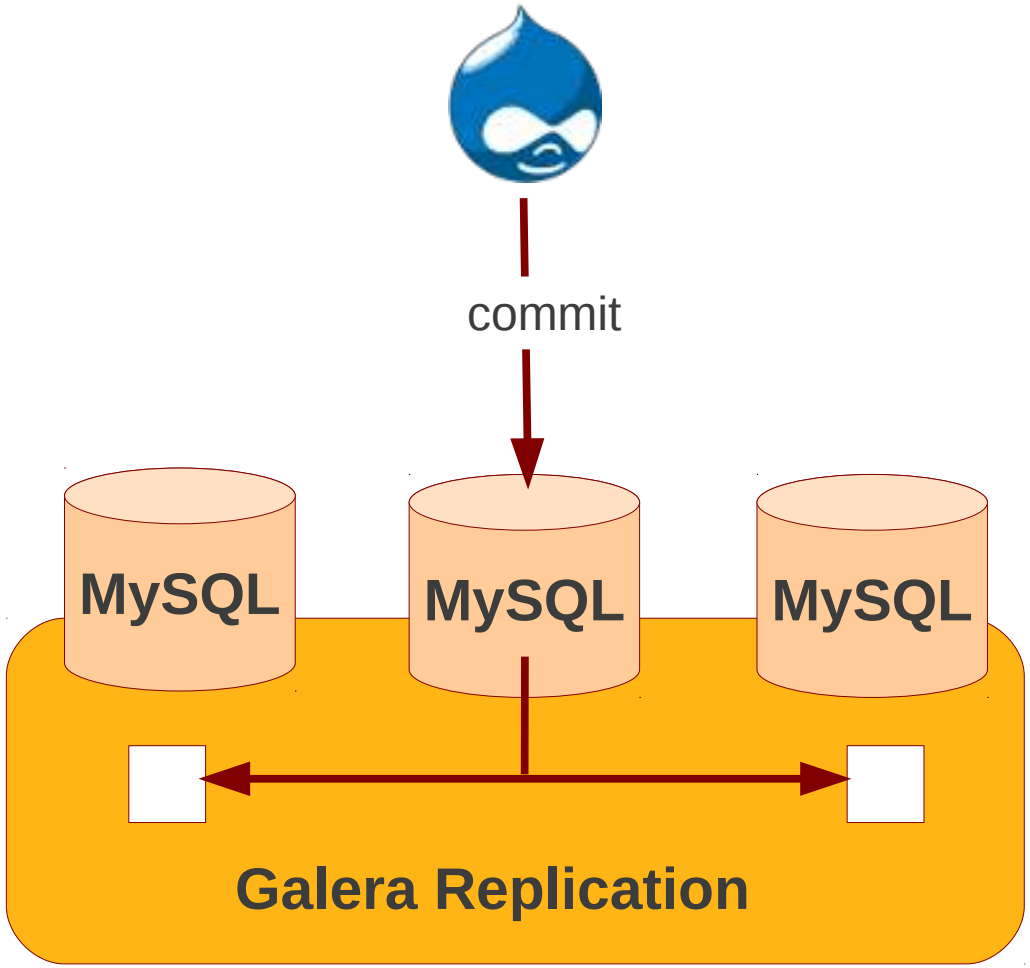


Synchronous Replication



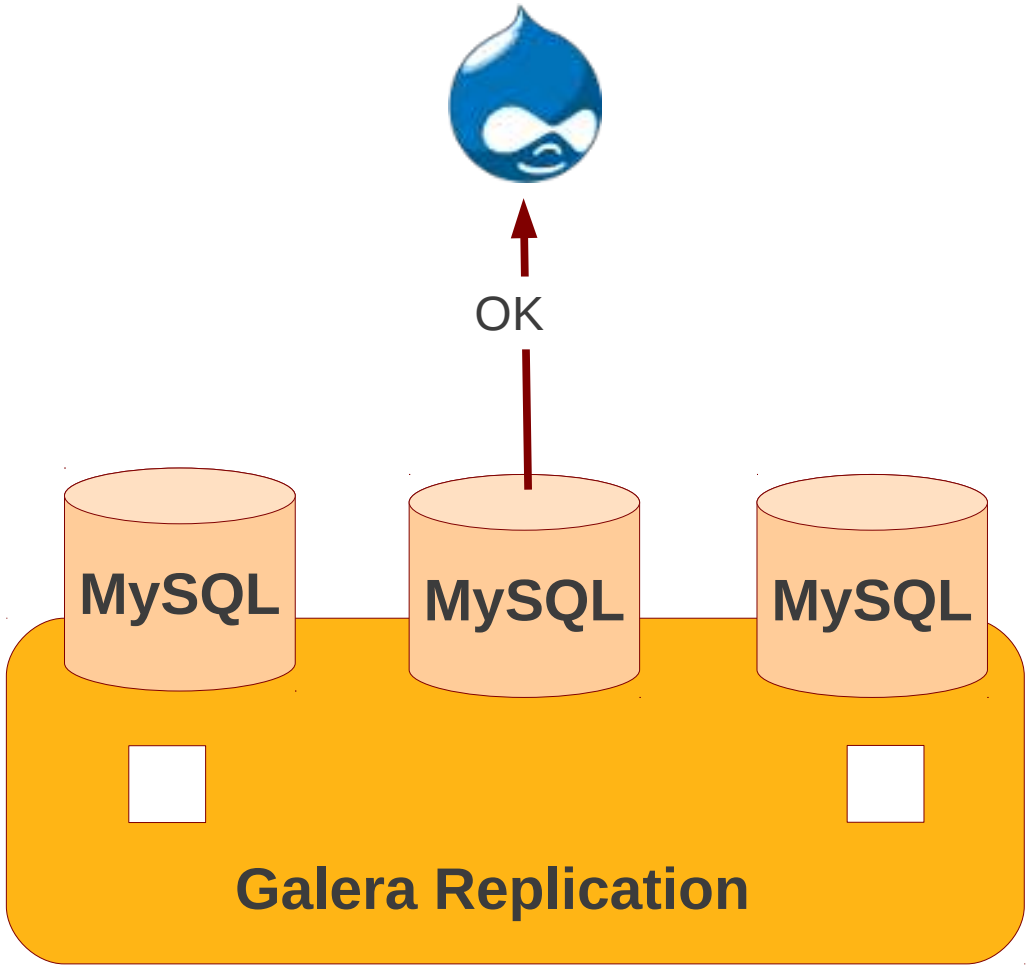
Transaction is processed locally up to commit time

Synchronous Replication



Transaction is replicated to whole cluster

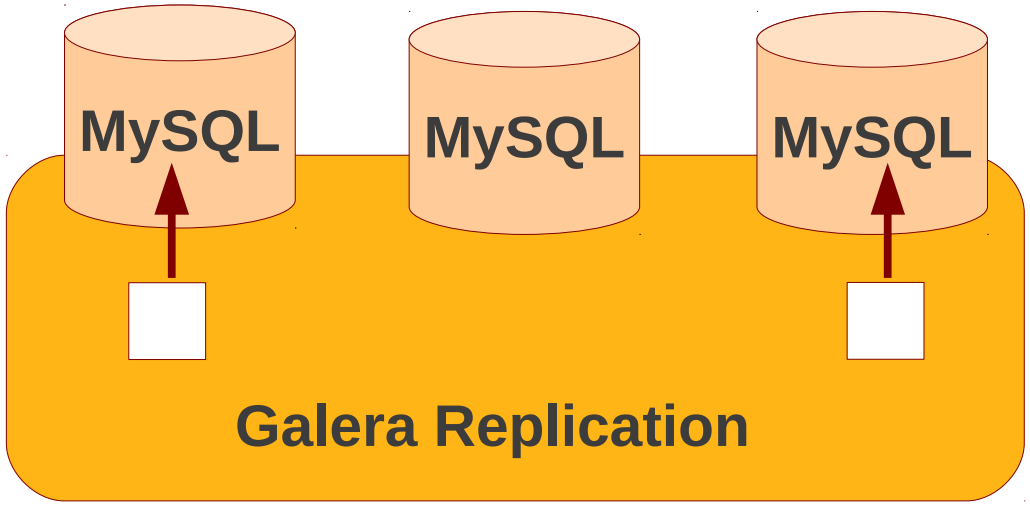
Synchronous Replication



Client gets OK status

Synchronous Replication

Transaction is applied in slaves



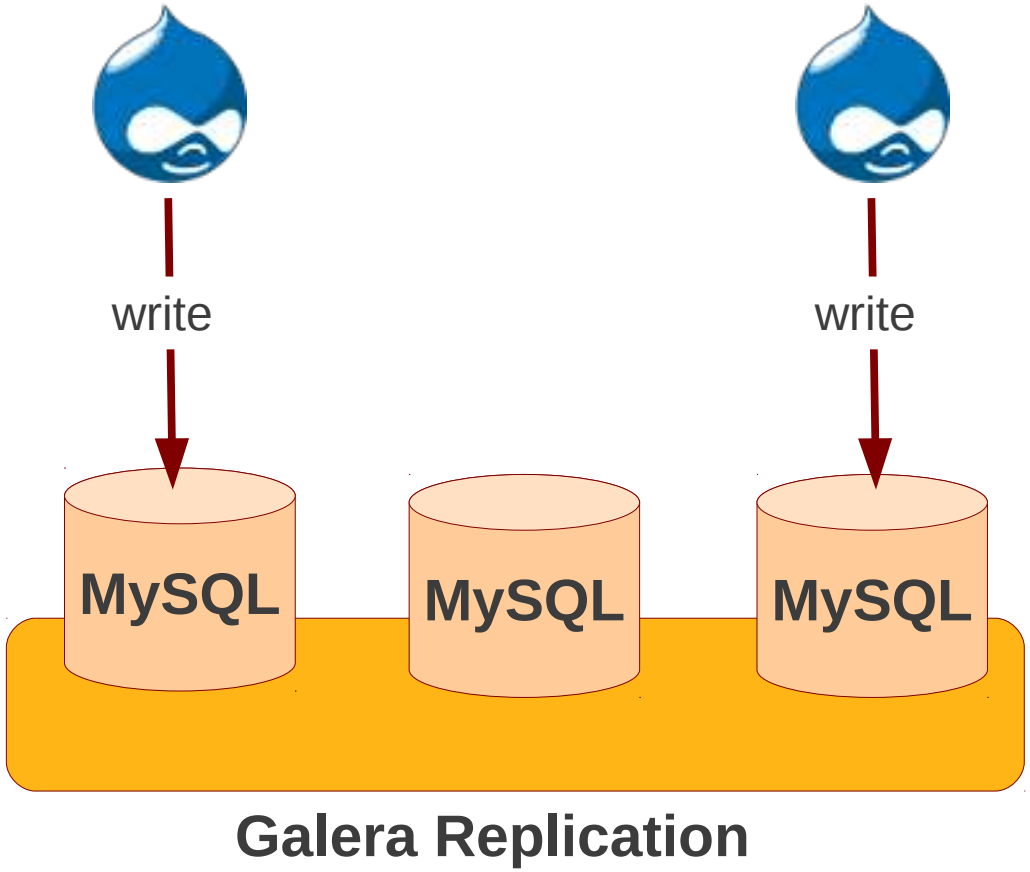
Galera Cluster

Each node is full representative of the cluster

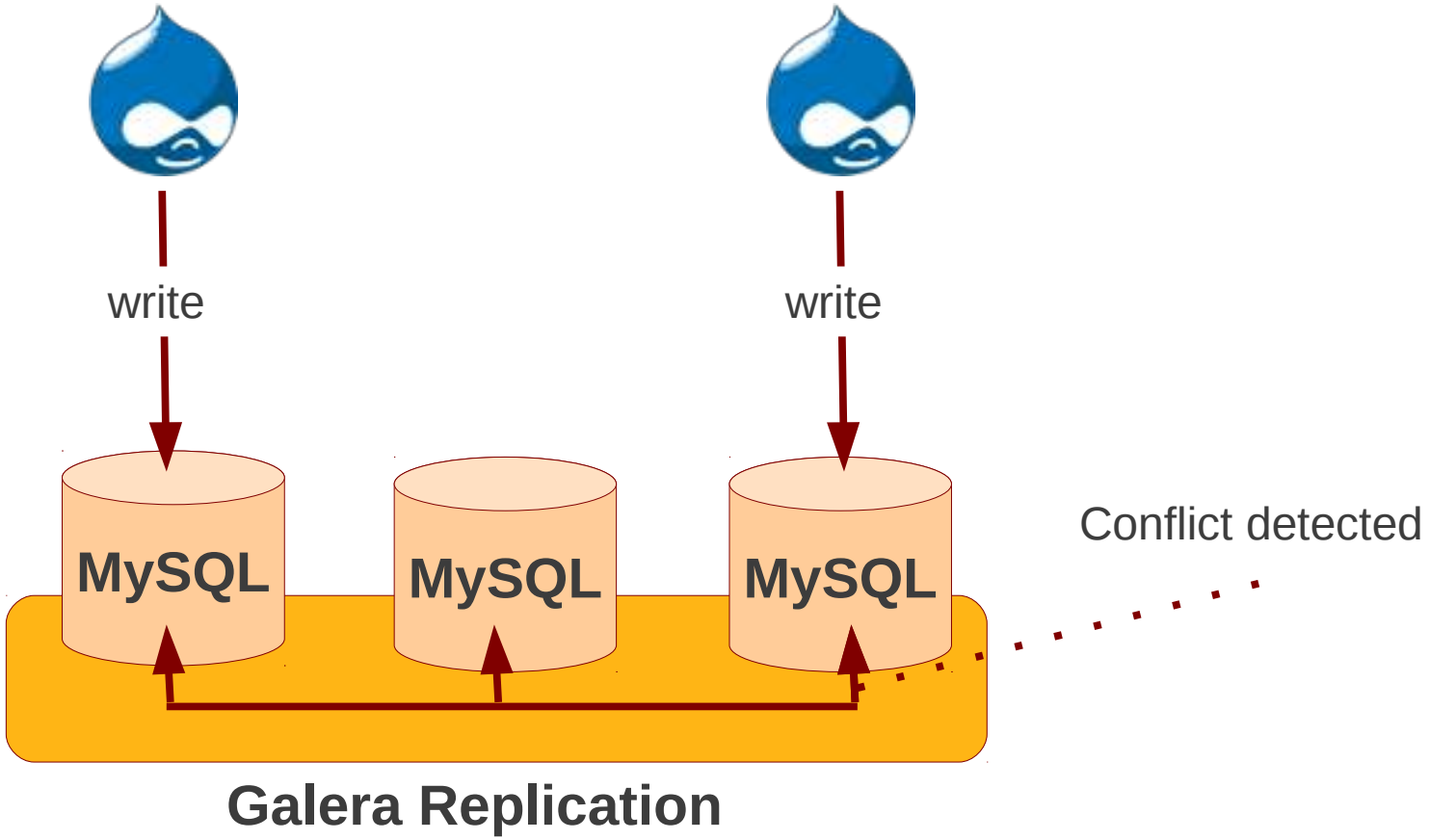
- No single point of failure
- Synchronous...
  - No Data Loss
  - No Slave Lag
  - No Master Failover
- 99.99% transparent
  - InnoDB look & feel

Dealing with Multi-Master Conflicts

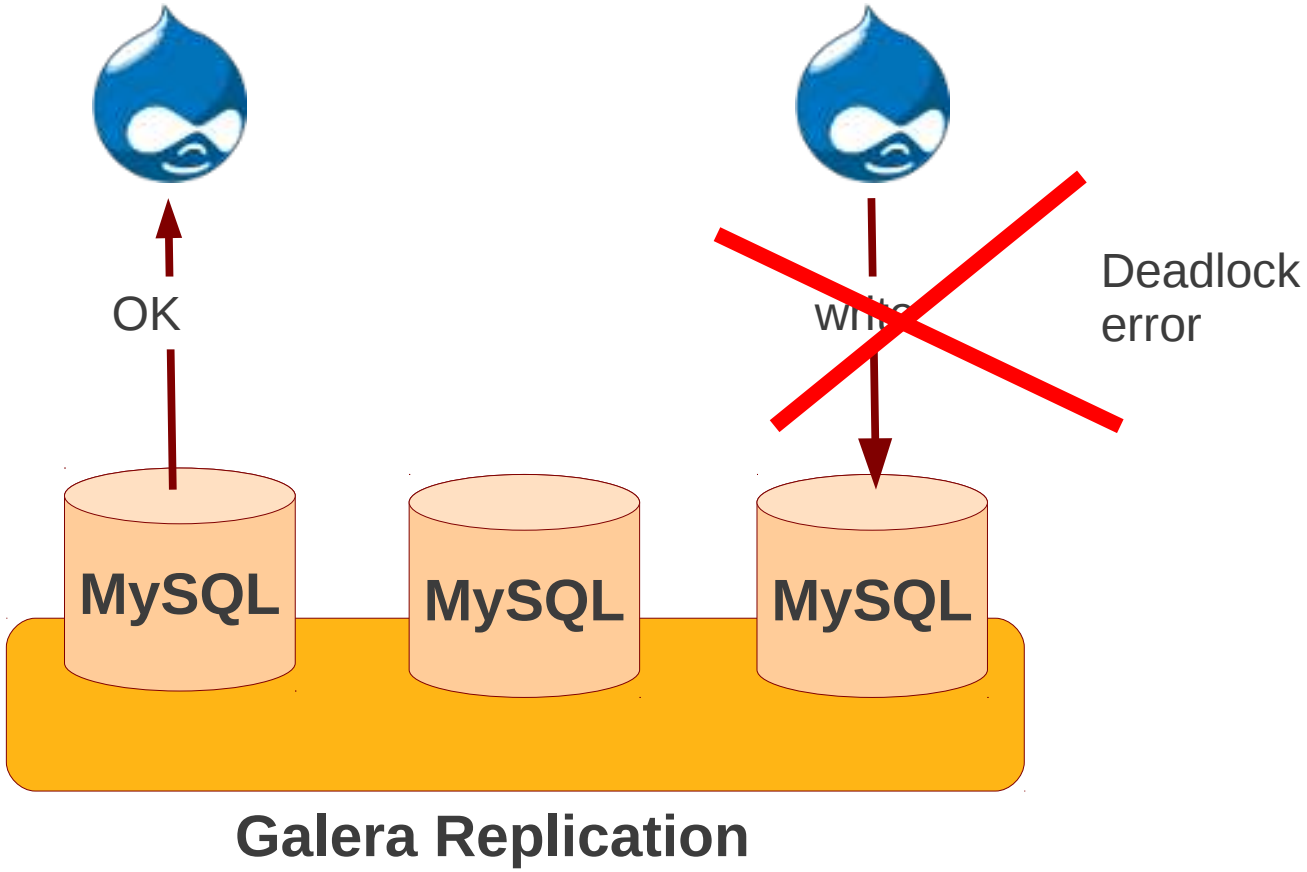
Multi-master Conflicts



Multi-master Conflicts



Multi-master Conflicts



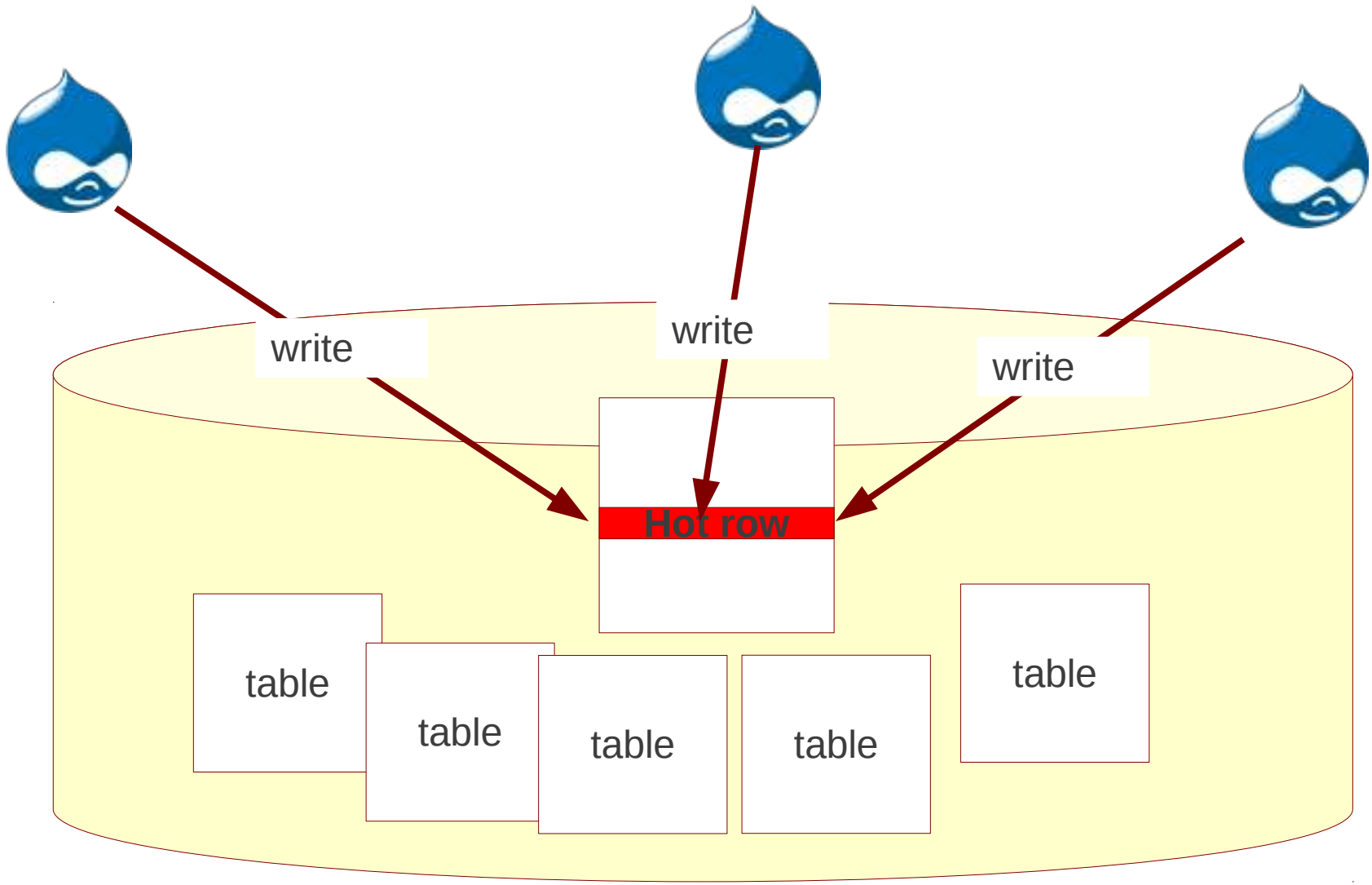
Multi-Master Conflicts

- Galera uses optimistic concurrency control
- If two transactions modify same row on different nodes at the same time, one of the transactions must abort
  - → Victim transaction will get deadlock error
- Application should retry deadlocked transactions, however not all applications have retrying logic inbuilt

Database Hot-Spots

- Some rows where many transactions want to write to simultaneously
- Patterns like queue or ID allocation can be hot-spots

Hot-Spots



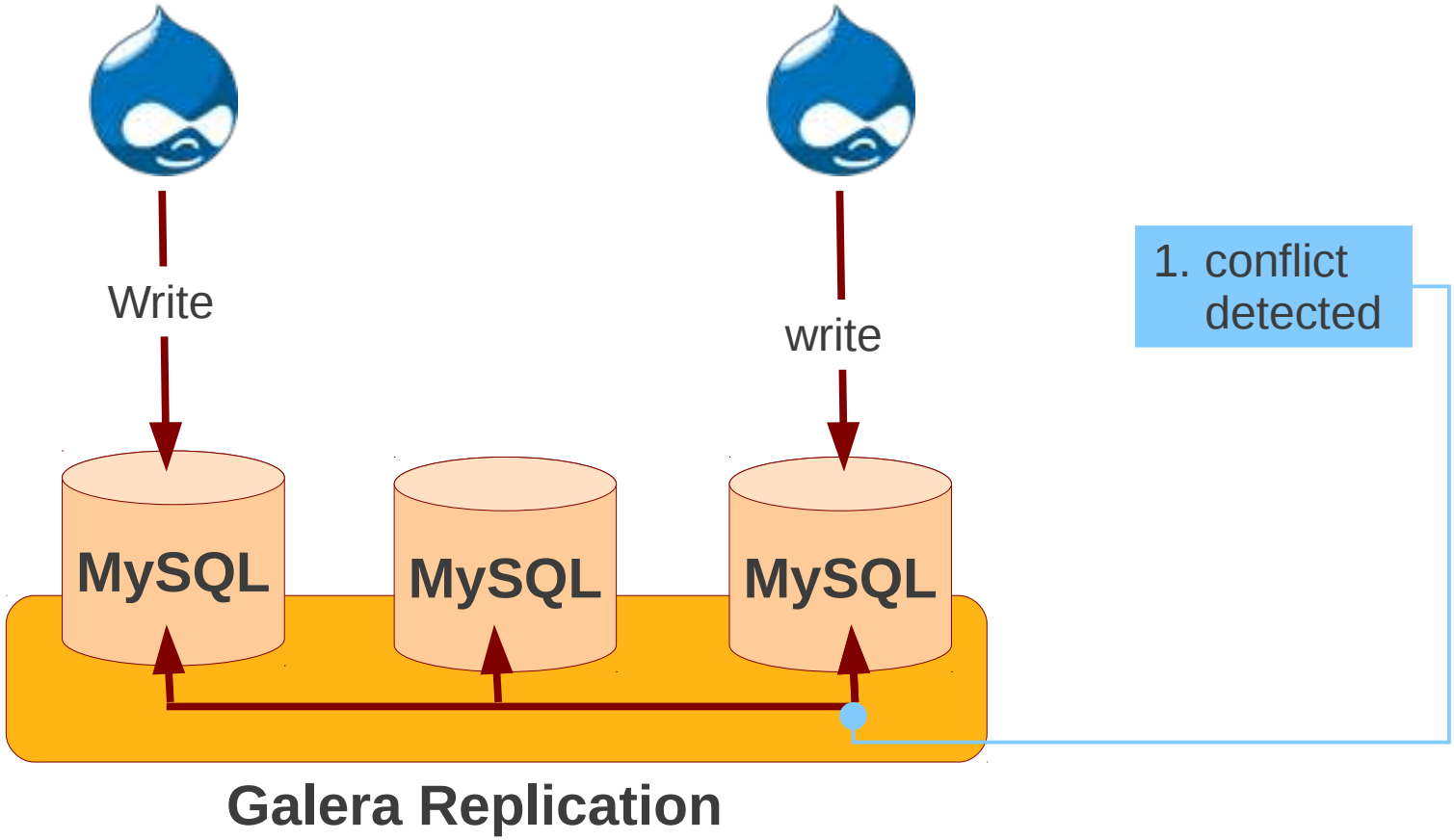
Diagnosing Multi-Master Conflicts

- By wsrep_debug configuration, all conflicts (...and plenty of other information) will be logged
- Next release (5.5.28-23.7) will add new variable: wsrep_log_conflicts which will cause each cluster conflict to be logged in mysql error log
- Monitor conflicts with:
  - wsrep_local_bf_aborts
  - wsrep_local_cert_failures

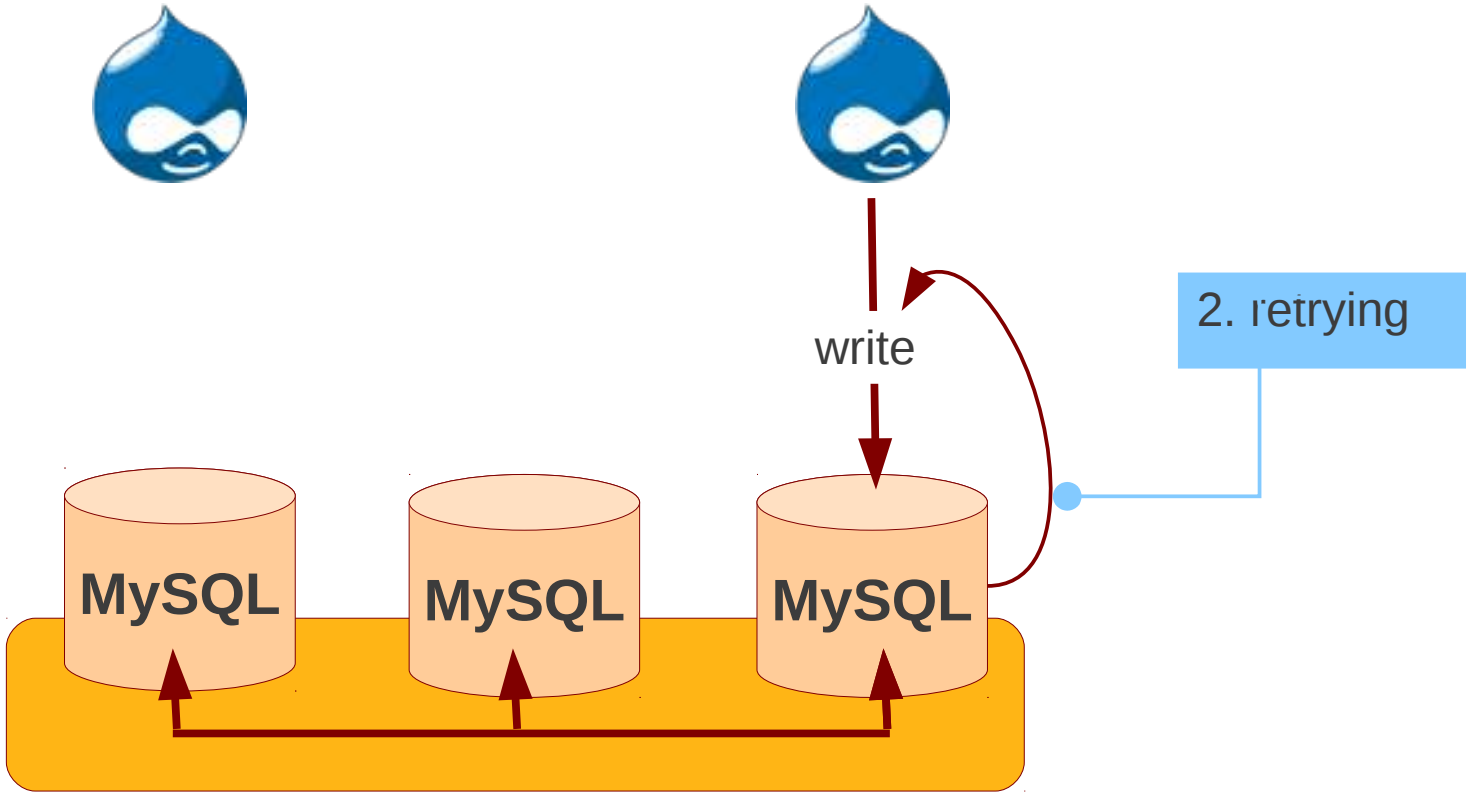
wsrep_retry_autocommit

- Galera can retry autocommit transaction on behalf of the client application, inside of the MySQL server
- MySQL will not return deadlock error, but will silently retry the transaction
- wsrep_retry_autocommit=n will retry the transaction n times before giving up and returning deadlock error
- Retrying applies only to autocommit transactions, as retrying is not safe for multi-statement transactions

Retry Autocommit



Retry Autocommit



Multi-Master Conflicts

1) Analyze the hot-spot
2) Check if application logic can be changed to catch deadlock exception and apply retrying logic in application
3) Try if wsrep_retry_autocommit configuration helps
4) Limit the number of master nodes or change completely to master-slave model

   if you can filter out the access to the hot-spot table, it is enough to treat writes only to hot-spot table as master-slave

State Transfers

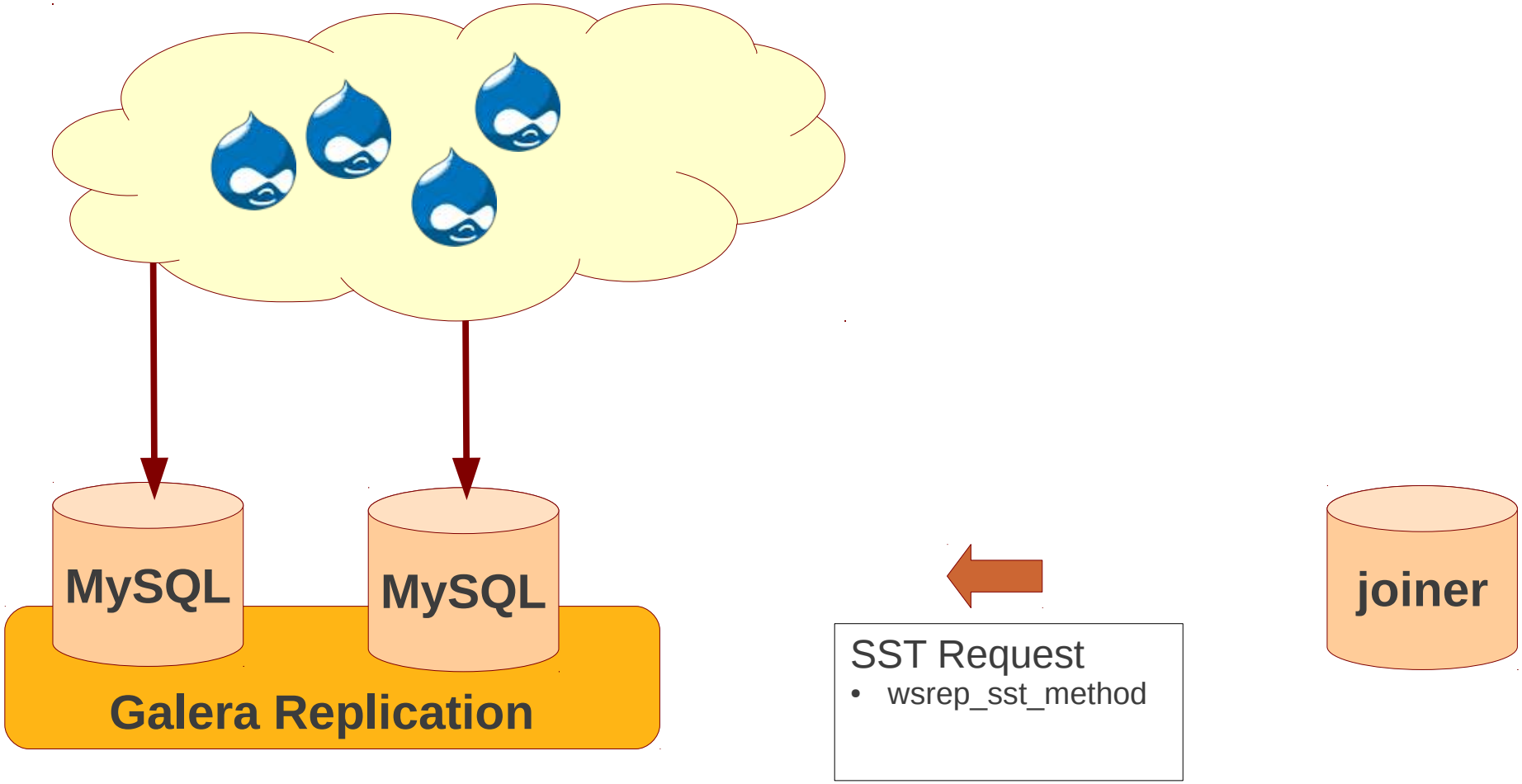
State Transfer

- Joining node needs to get the current database state
- Two choices:
  - IST: incremental state transfer
  - SST: full state transfer
- If joining node had some previous state and gcache spans to that, then IST can be used

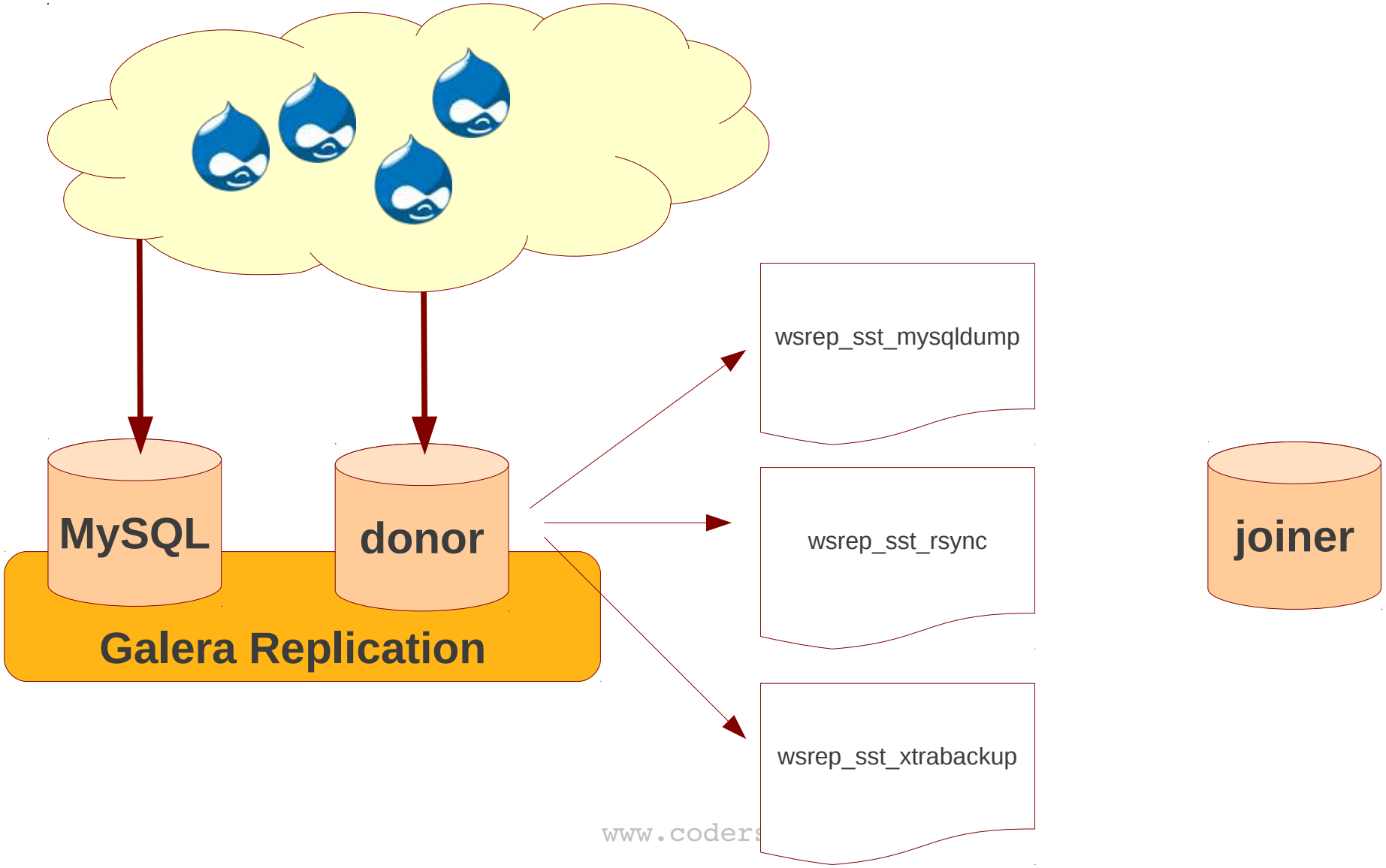
State Snapshot Transfer

- To send full database state
- wsrep_sst_method to choose the method:
  - mysqldump
  - rsync
  - xtrabackup

SST Request


MySQL
MySQL
Galera Replication
SST Request
• wsrep_sst_method
joiner

SST Method



SST API

- SST is open API for shell scripts
- Anyone can write custom SST
- SST API can be used e.g. for:
  - Backups
  - Filtering out part of database

wsrep_sst_mysqldump

- Logical backup
- Slowest method
- Configure authentication
  - wsrep_sst_auth=”root:rootpass”
  - Super privilege needed
- Make sure SST user in donor node can take mysqldump from donor and load it over the network to joiner node
  - You can try this manually beforehand

wsrep_sst_rsync

- Physical backup
- Fast method
- Can only be used when node is starting
  - Rsyncing data directory under running InnoDB is not possible

wsrep_sst_xtrabackup

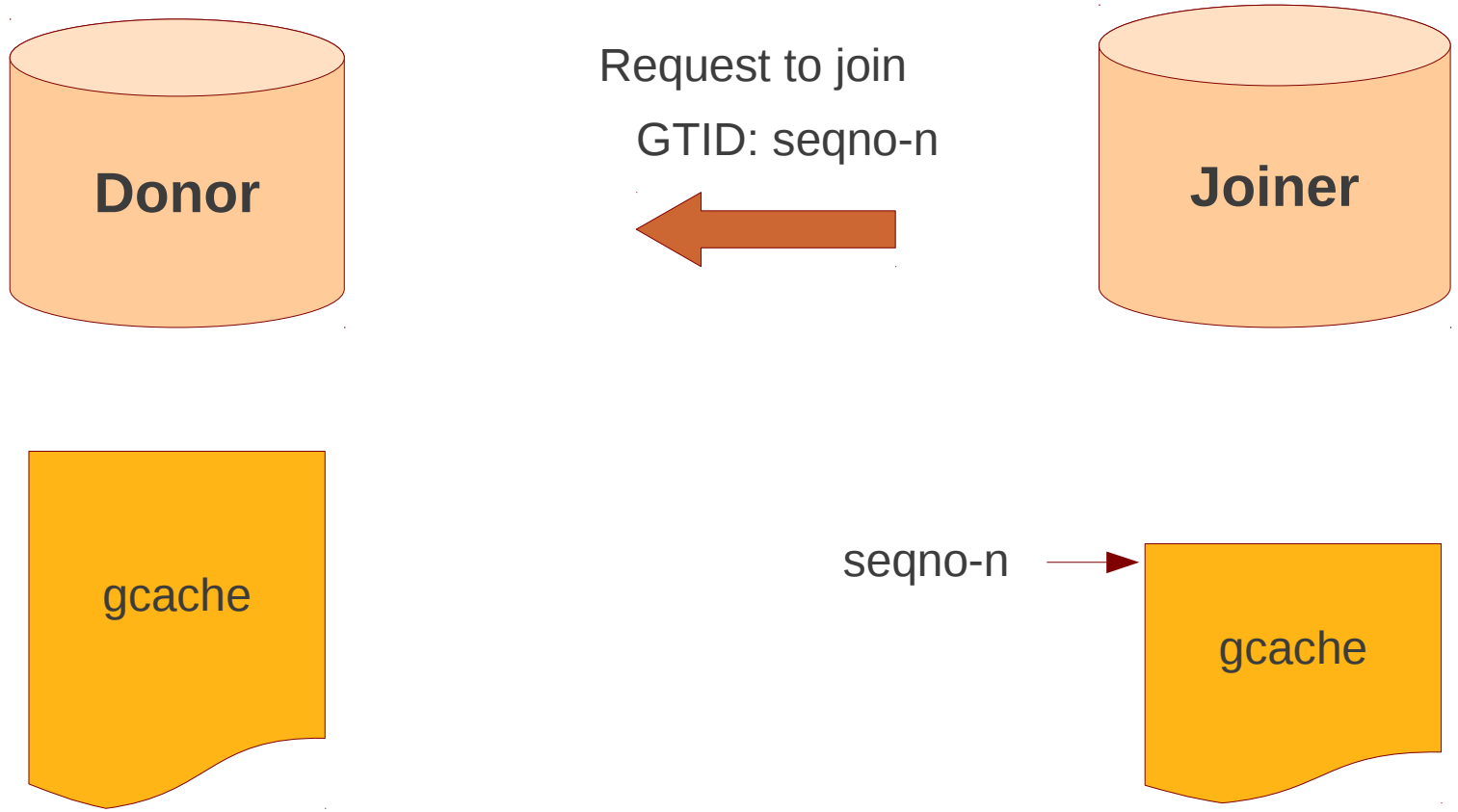
- Contributed by Percona
- Probably the fastest method
- Uses xtrabackup
- Least blocking on Donor side (short readlock is still used when backup starts)

SST Donor

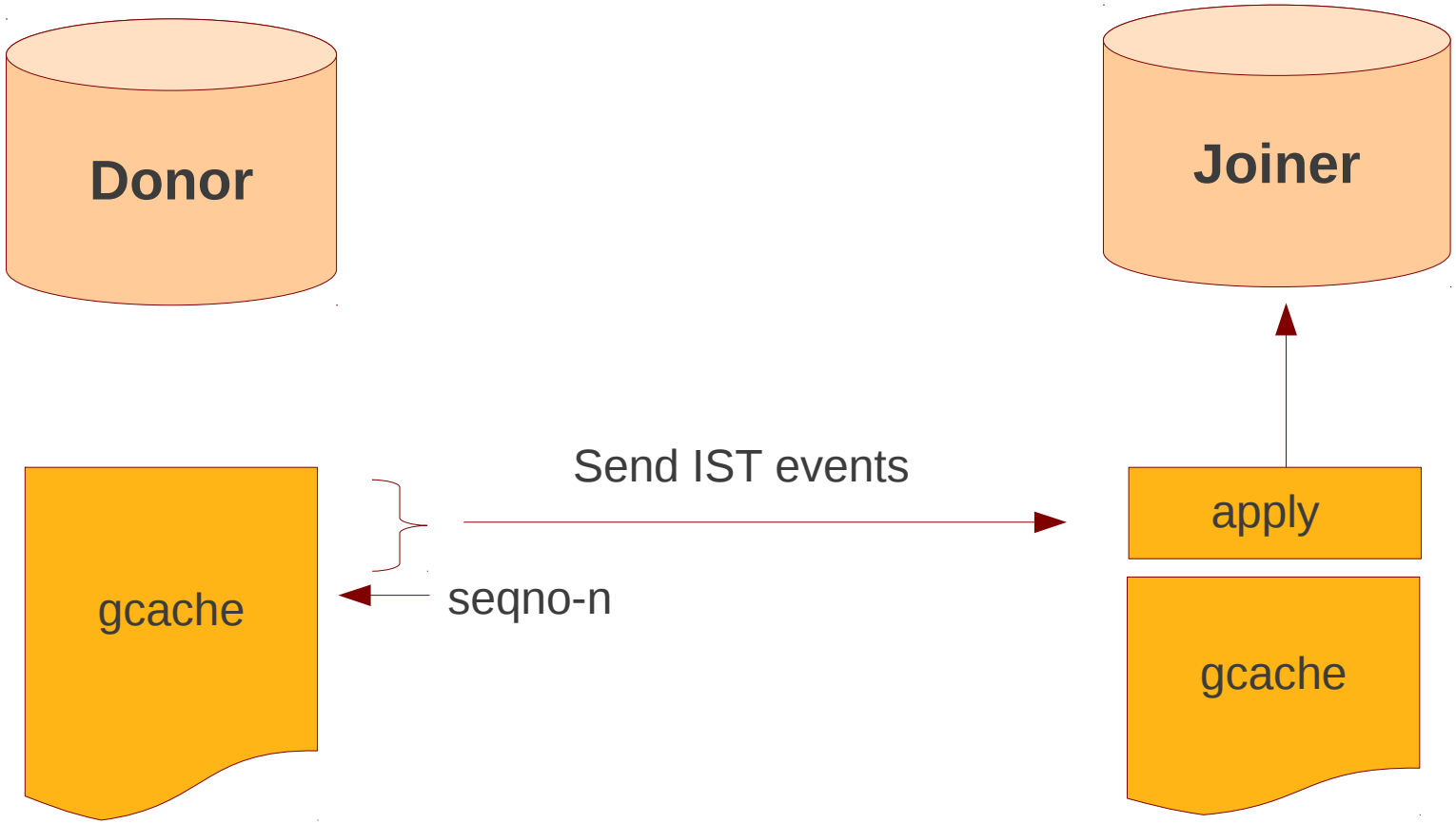
- All SST methods cause some disturbance for donor node
- By default donor accepts client connections, although committing will be prohibited for a while
- If wsrep_sst_donor_rejects_queries is set, donor gives unknown command error to clients

➔ Best practice is to dedicate a reference node for donor and backup activities

Incremental State Transfer



Incremental State Transfer



Incremental State Transfer

- Very effective
- gcache.size parameter defines how big cache will be maintained
- Gcache is mmap, available disk space is upper limit for size allocation

Incremental State Transfer

- Use database size and write rate to optimize gcache:
  - gcache < database
  - Write rate tells how long tail will be stored in cache

Incremental State Transfer

- You can think that IST Is
  - A short asynchronous replication session
  - If communication is bad quality, node can drop and join back fast with IST

Backups Backups Backups

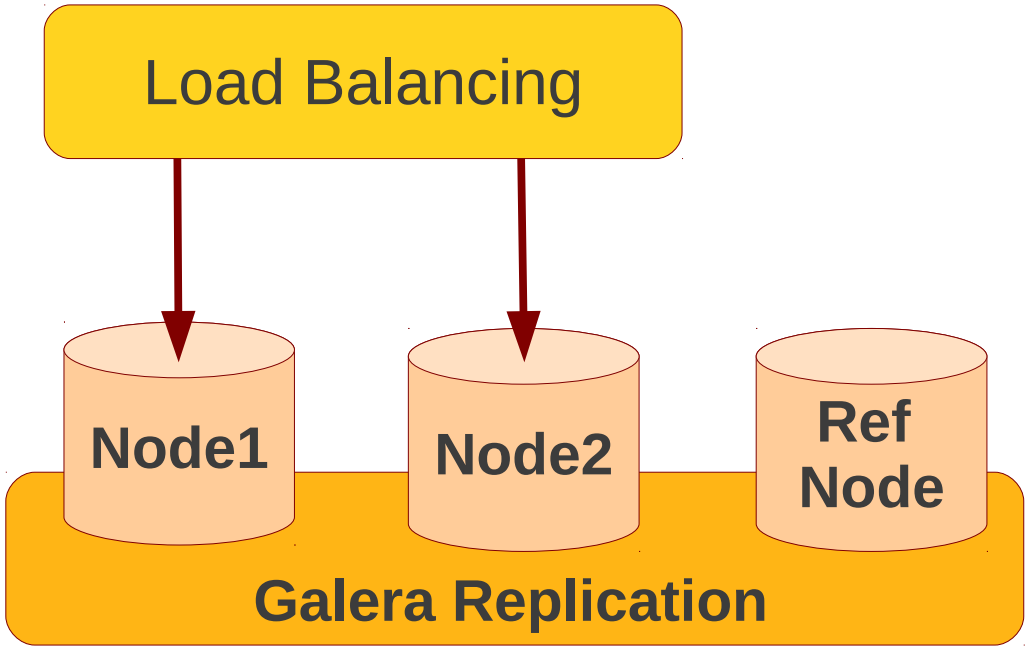
codership

Backups

- All Galera nodes are constantly up to date
- Best practices:
  - Dedicate a reference node for backups
  - Assign global trx ID with the backup
- Possible methods:

1. Disconnecting a node for backup
2. Using SST script interface
3. xtrabackup

Using Reference Node for Backups

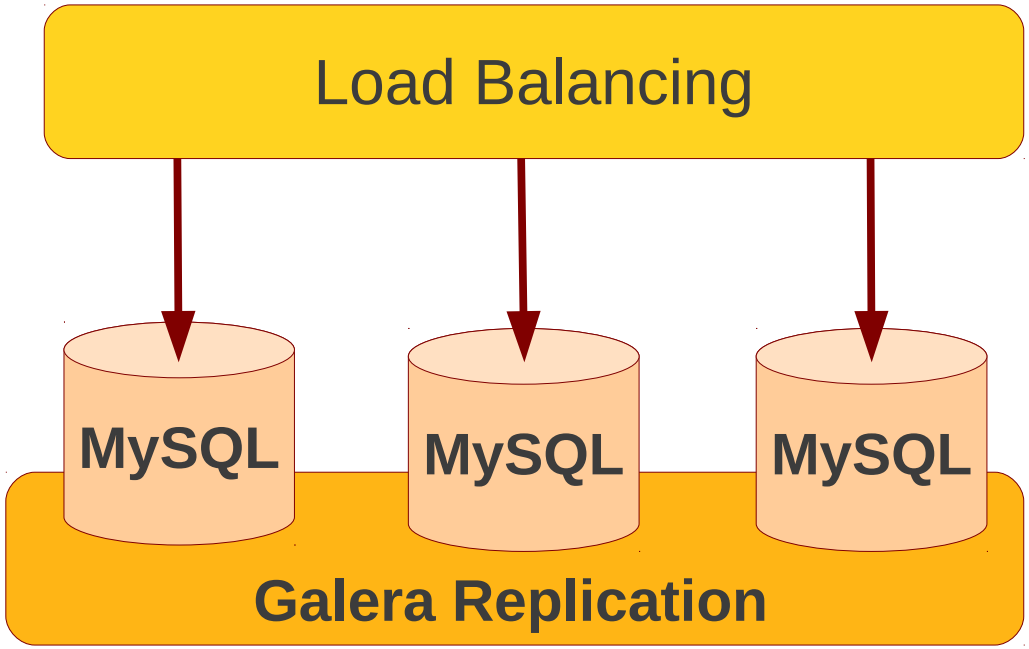


Reference node is a member in the cluster, but does not serve client connections

Backups with global Trx ID

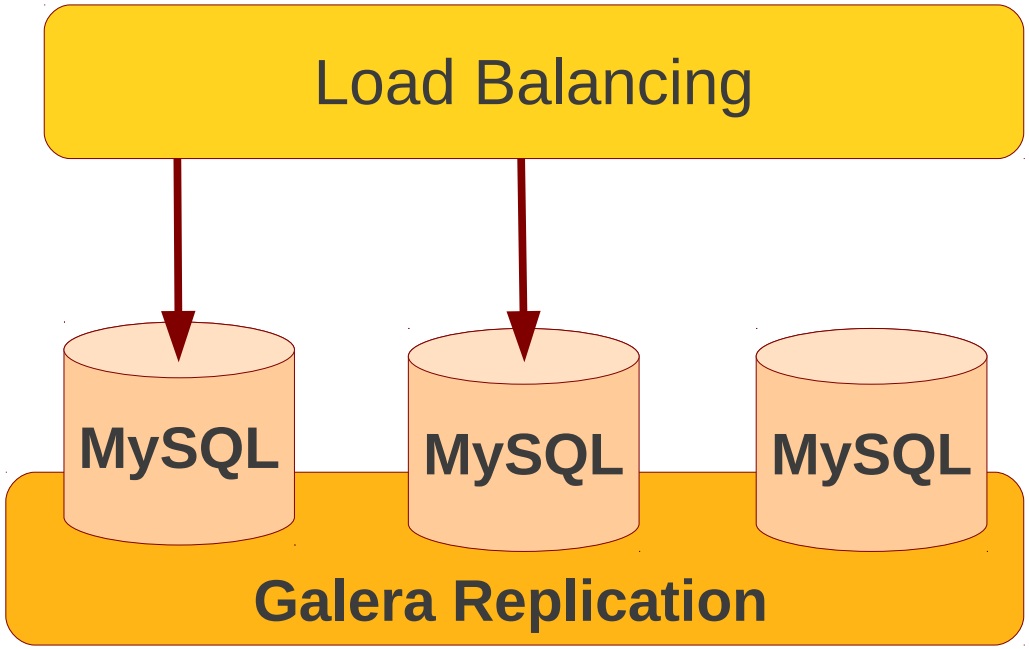
- Global transaction ID (GTID) marks a position in the cluster transaction stream
- Backup with known GTID make it possible to utilize IST when joining new nodes, eg, when:
  - Recovering the node
  - Provisioning new nodes

Backup by Disconnecting a Node



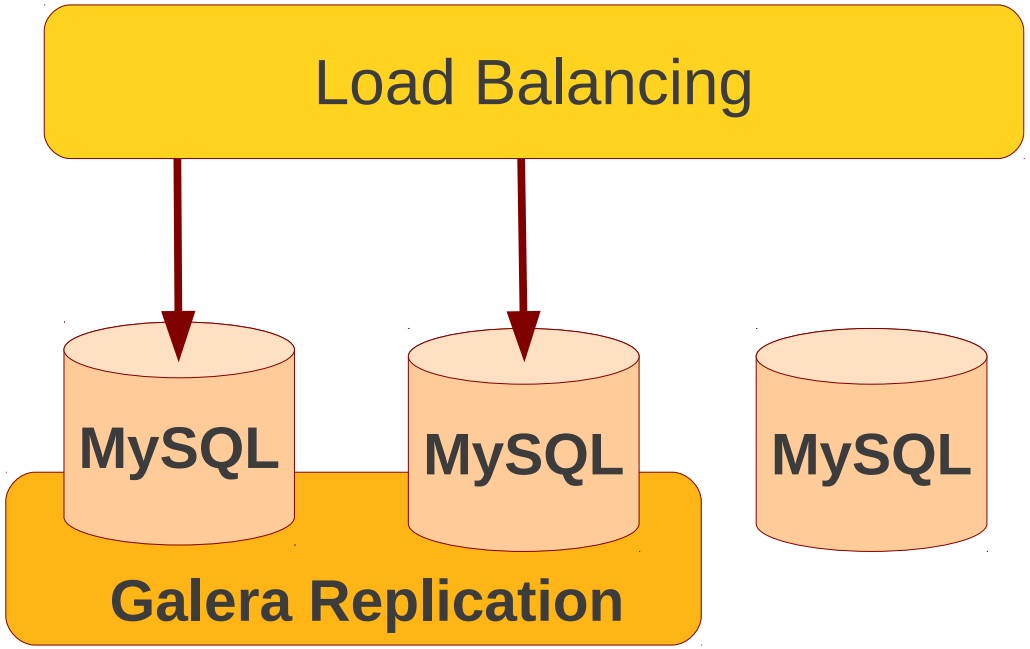
Isolate the backup node

Backup by Disconnecting a Node



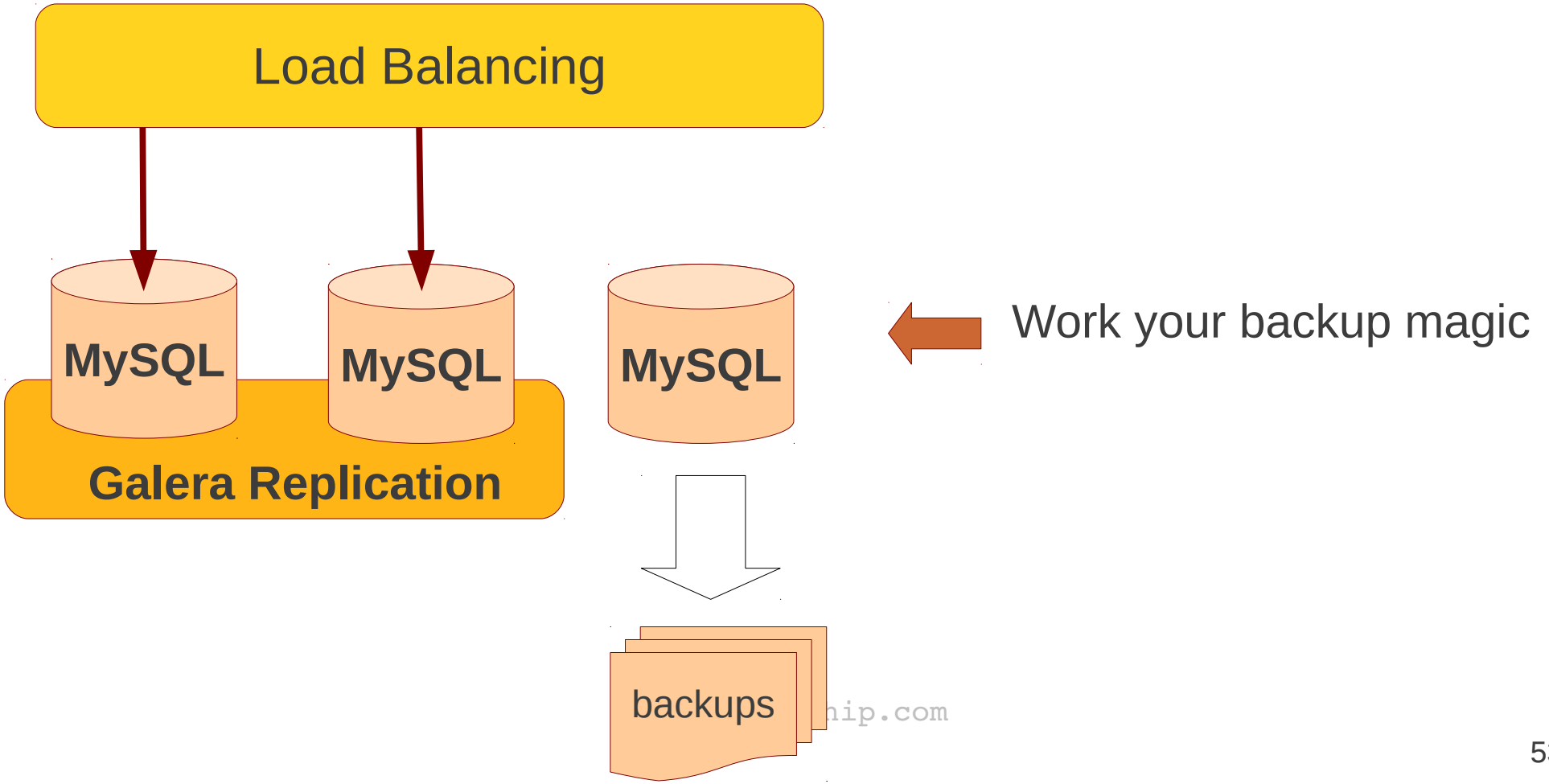
Disconnect from group
e.g. clear wsrep_provider

Backup by Disconnecting a Node

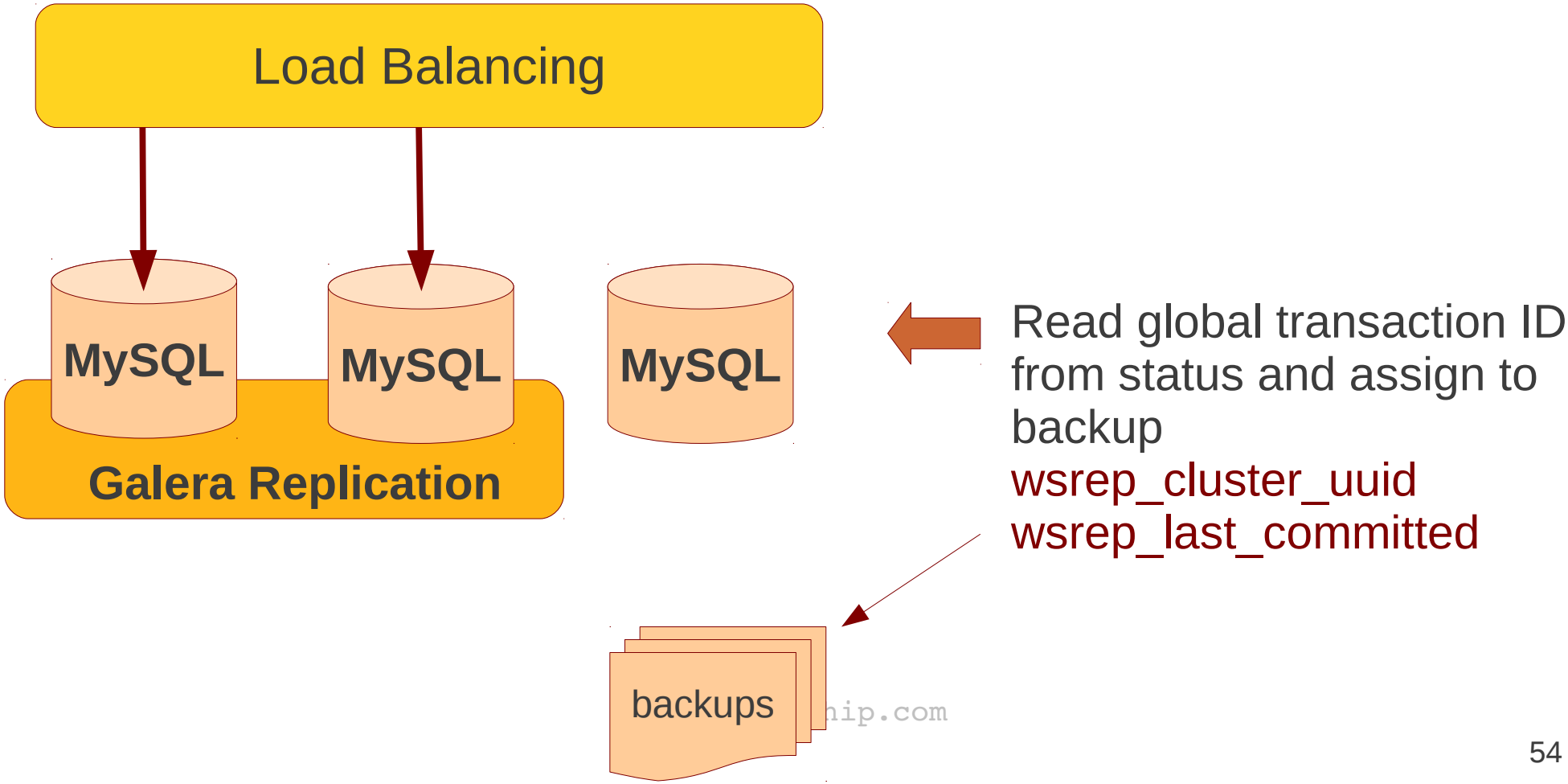


Disconnect from group
e.g. clear wsrep_provider

Backup by Disconnecting a Node



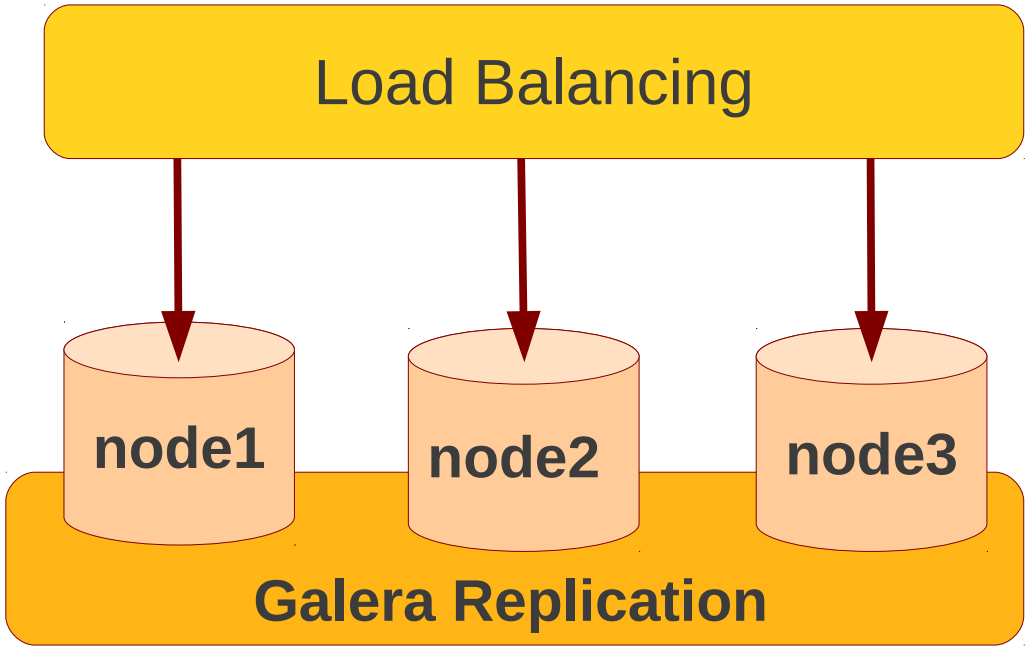
Backup by Disconnecting a Node



Backup by SST

- Donor mode provides isolated processing environment
- A special SST script can be written just to prepare backup in donor node: wsrep_sst_backup
- Garbd can be used to trigger donor node to run the wsrep_sst_backup

Backup by SST API



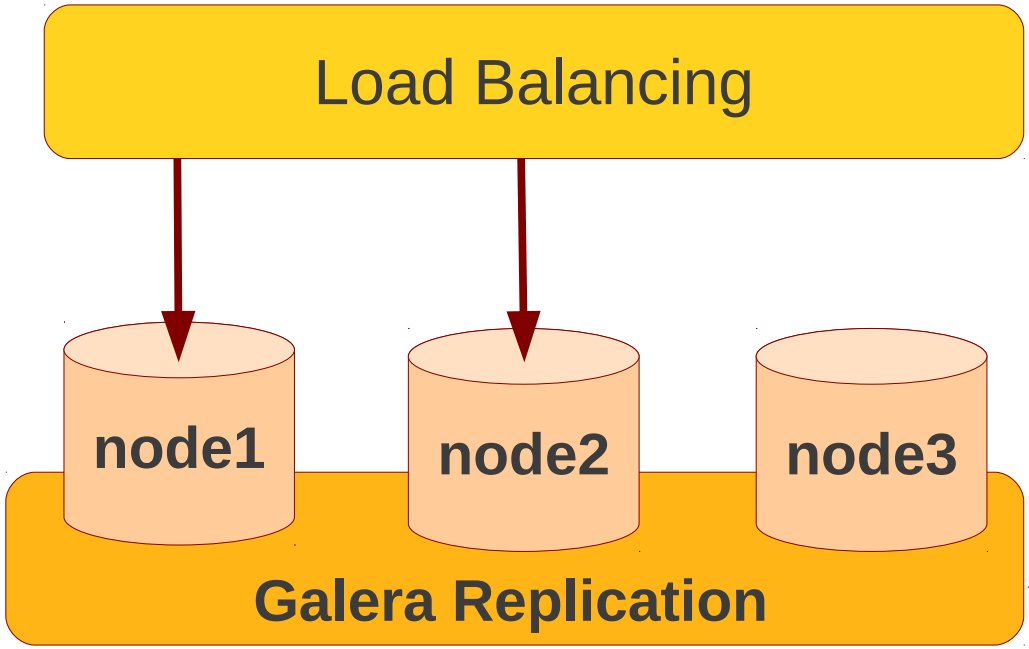
Launch garbd

SST request

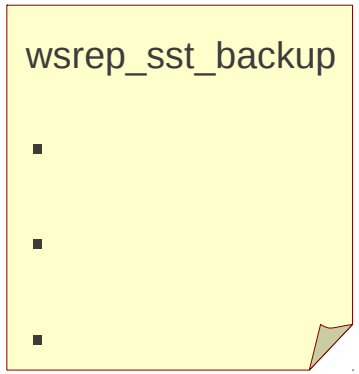
Garbd

wsrep_sst_donor=node3
wsrep_sst_method=backup

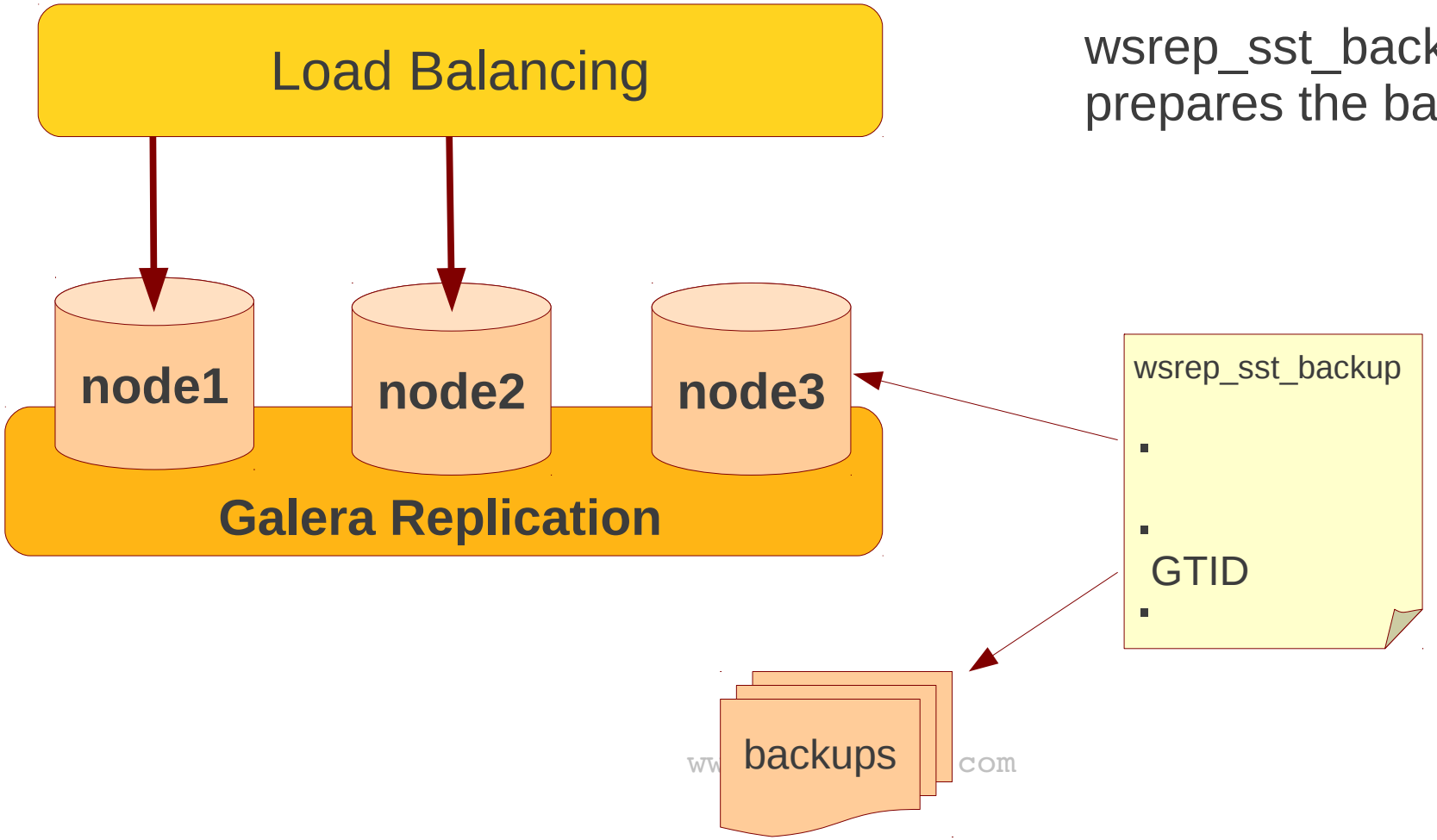
Backup by SST API



Donor launches
wsrep_sst_backup

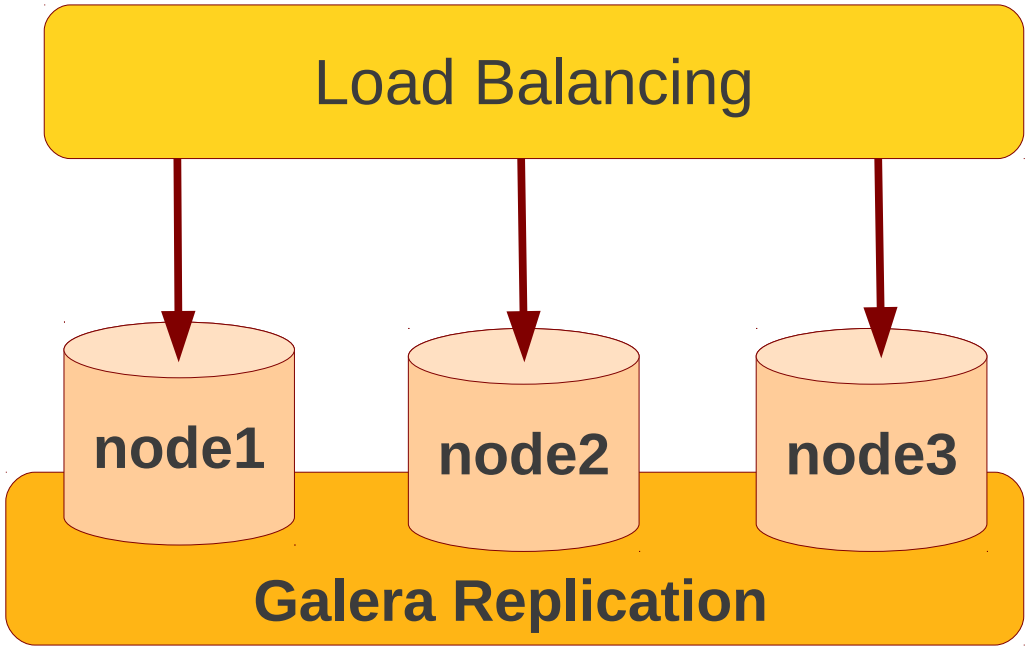


Backup by SST API



wsrep_sst_backup
prepares the backup

Backup by SST API



Backup node returns to cluster

Backup by xtrabackup

- Xtrabackup is hot backup method and can be used anytime
- Simple, efficient
- Use –galera-info option to get global transaction ID logged into separate galera info file

Schema Upgrades

Schema Upgrades

- DDL is non-transactional, and therefore bad
- Galera has two methods for DDL
  - TOI, Total Order Isolation
  - RSU, Rolling Schema Upgrade
- Use wsrep_osu_method to choose either option

Total Order Isolation

- DDL is replicated up-front
  - Each node will get the DDL statement and must process the DDL at same slot in transaction stream
- Galera will isolate the affected table/database for the duration of DDL processing

Rolling Schema Upgrade

- DDL is not replicated
- Galera will take the node out of replication for the duration of DDL processing
- When DDL is done with, node will catch up with missed transactions (like IST)
- DBA should roll RSU operation over all nodes
- Requires backwards compatible schema changes

wsrep_on=OFF

- wsrep_on is a session variable telling if this session will be replicated or not
- I tried to hide this information to the best I can, but somebody has leaked this out
- And so, yes, it is possible to run “poor man's RSU” with wsrep_on set to OFF
- such session may be aborted by replication
- Use only, if you are really sure that:
  - planned SQL is not conflicting
  - SQL will not generate inconsistency

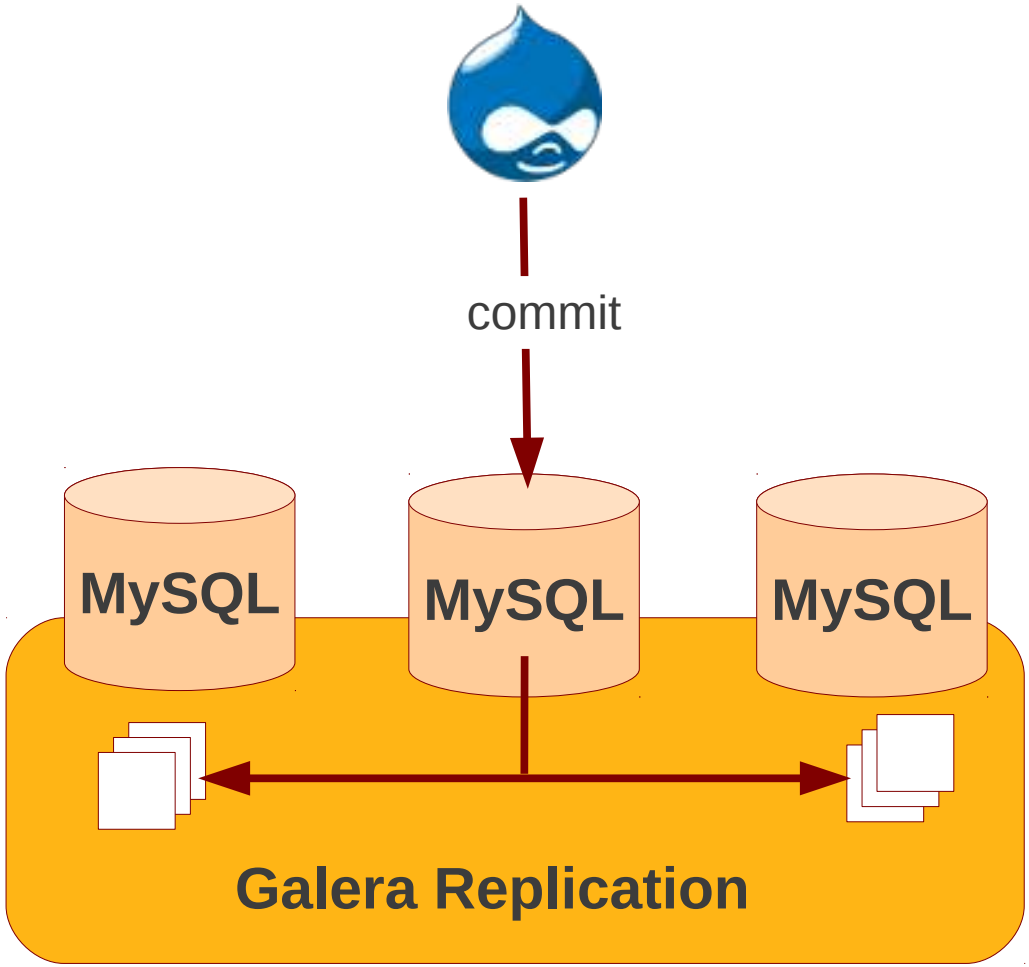
Schema Upgrades

- Best practices:
  - Plan your upgrades
    - Try to be backwards compatible
  - Rehearse your upgrades
    - Find out DDL execution time
  - Go for RSU if possible

Consistent Reads

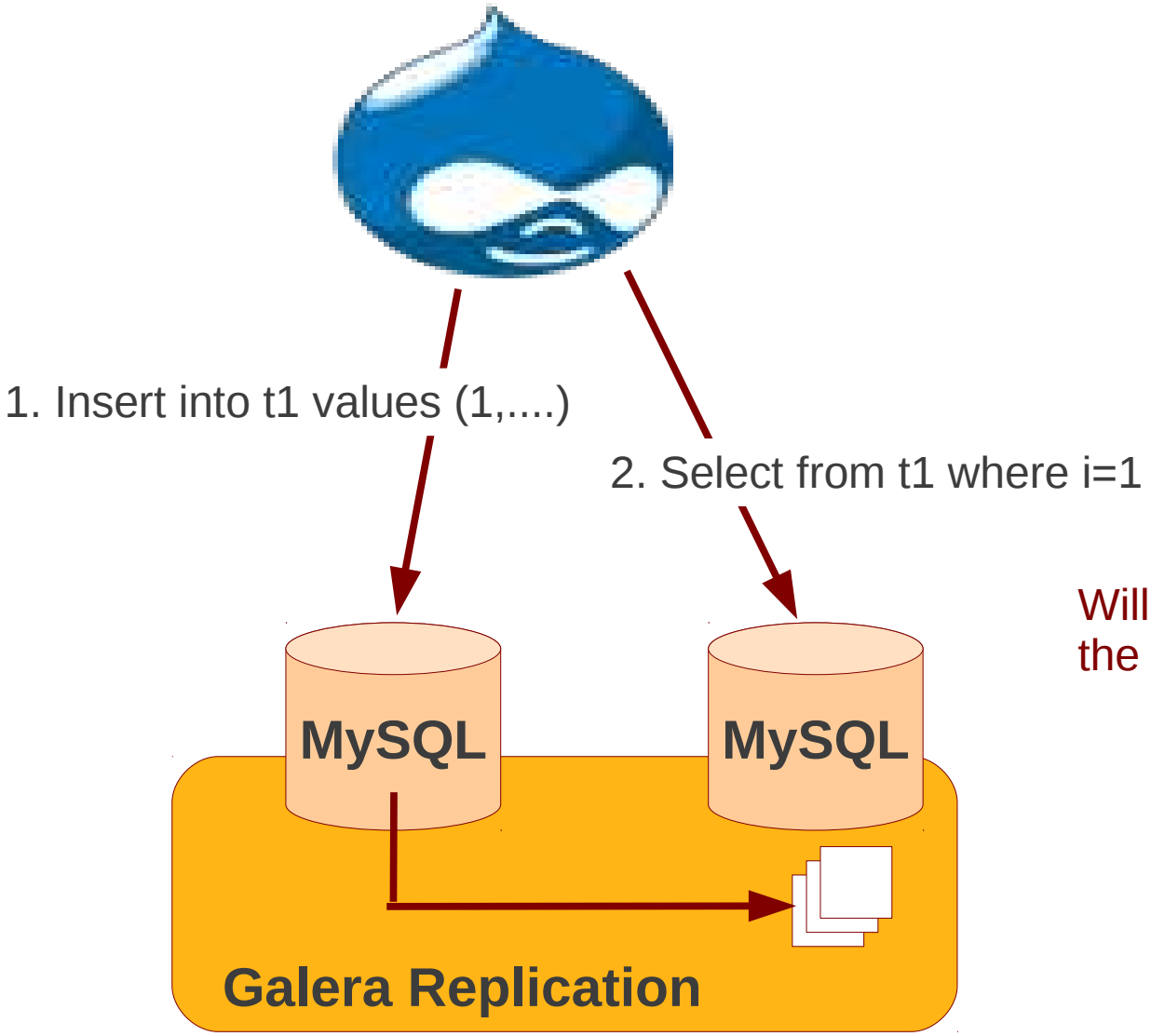
Consistent reads

Replication is virtually synchronous...



Transaction is replicated to whole cluster

Consistent reads



Will the select see the inserted row?

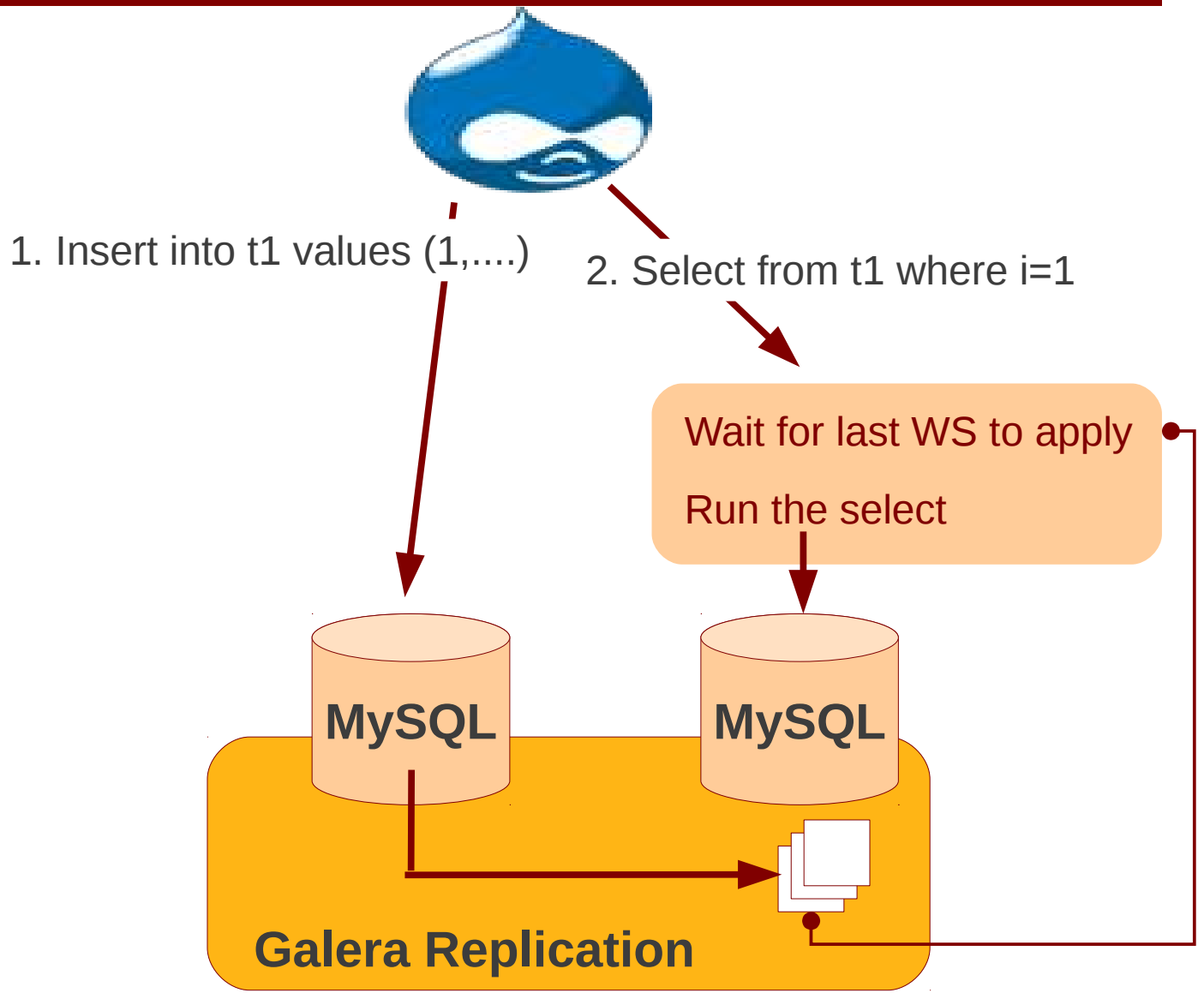
Consistent Reads

- Aka read causality
- There is causal dependency between operations on two database connections
  - Application is expecting to see the values of earlier write

Consistent Reads

- Use: wsrep_causal_reads=ON
  - ➔ Every read (select, show) will wait until slave queue has been fully applied
- There is timeout for max causal read wait:
  - replicator.causal_read_keepalive

Consistent reads



Other Tidbits...

Parallel Applying

- Aka parallel replication
- “true parallel applying”
  - Every application will benefit of it
  - Works not on database, not on table, but on **row level**
- wsrep_slave_threads=n
- How many slaves makes sense:
  - Monitor wsrep_cert_deps_distance
  - Max 2 * cores

MyISAM Replication

- On experimental level
  - MyISAM is phasing out not much demand to complete
- Replicates SQL up-front, like TOI
- Should be used in master-slave model
- No checks for non-deterministic SQL
  - Insert into t (r, time) values (rand(), now());

SSL / TLS

- Replication over SSL is supported
- No authentication (yet), only encryption
- Whole cluster must use SSL

SSL or VPN

- Bundling several nodes through VPN tunnel may cause a vulnerability
- When VPN gateway breaks, a big part of cluster will be blacked out
- Best practice is to go for SSL if VPN does not have alternative routes

UDP Multicast

- Configure with gmcast.mcast_addr
- Full cluster must be configured for multicast or tcp sockets
- Multicast is good for scalability
- Best practice is to go for multicast if planning for large clusters

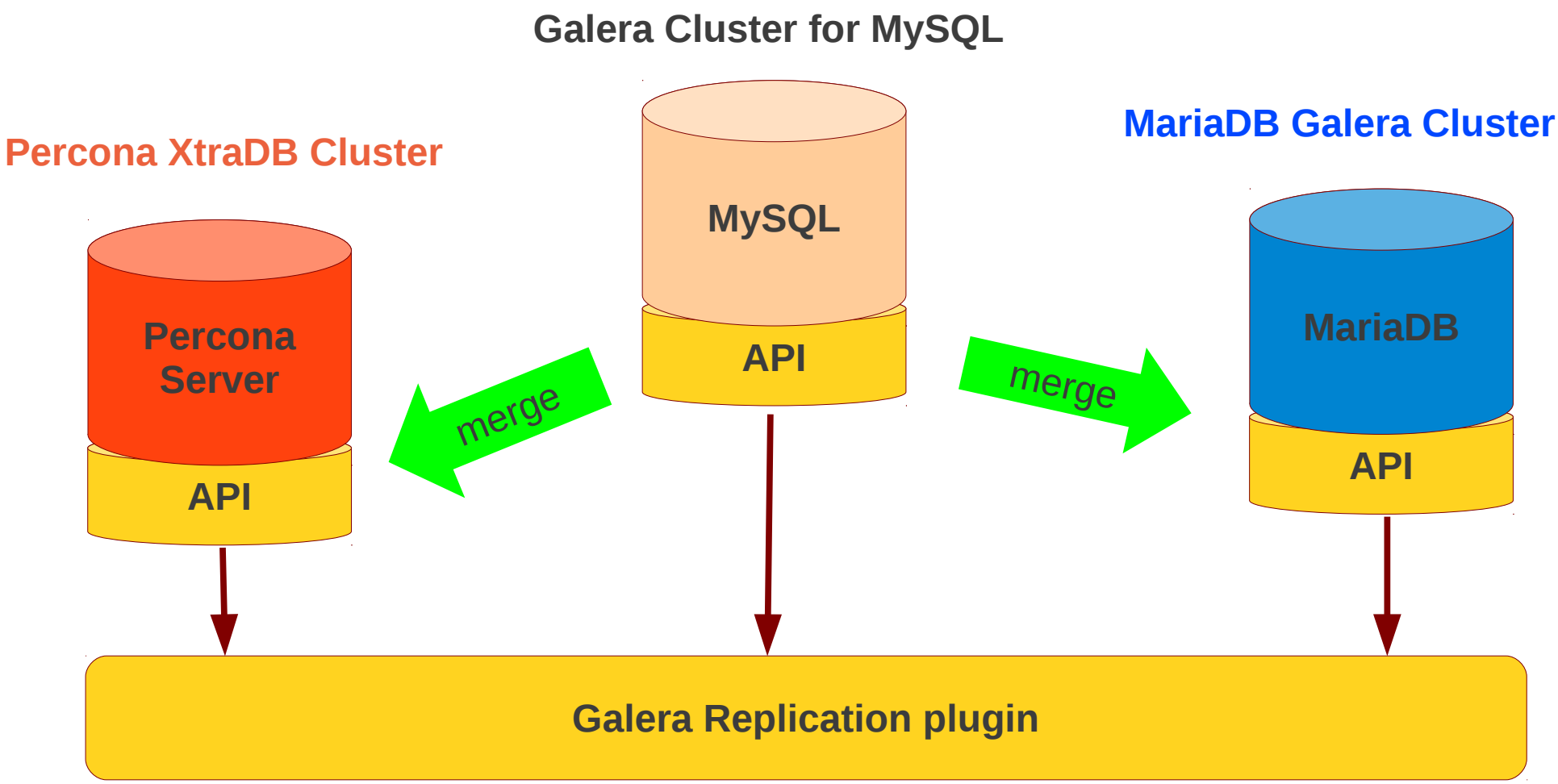
Galera Project

codership

Galera Project

- Galera Cluster for MySQL
  - 5 years development
  - based on MySQL server community edition
  - Fully open source
  - Active community
- ~3 releases per year
  - Latest release 2.2
  - Major release 3.0 in the works
- Galera Replication also used in:
  - Percona XtraDB Cluster
  - MariaDB Galera Cluster

Galera Project



Codership

- Consulting and support services for Galera
- Technology and support partners:
  - Percona
  - SkySQL
  - Monty Program
  - SeveralNines
  - FromDual
  - Capside

Questions?

Thank you for listening!
Happy Clustering :-)