Secure Your Data Workshop

MySQL Replication for High Availability

Dale Dasker
Manager MySQL Solution Engineering
dale.dasker@oracle.com
August 29, 2024

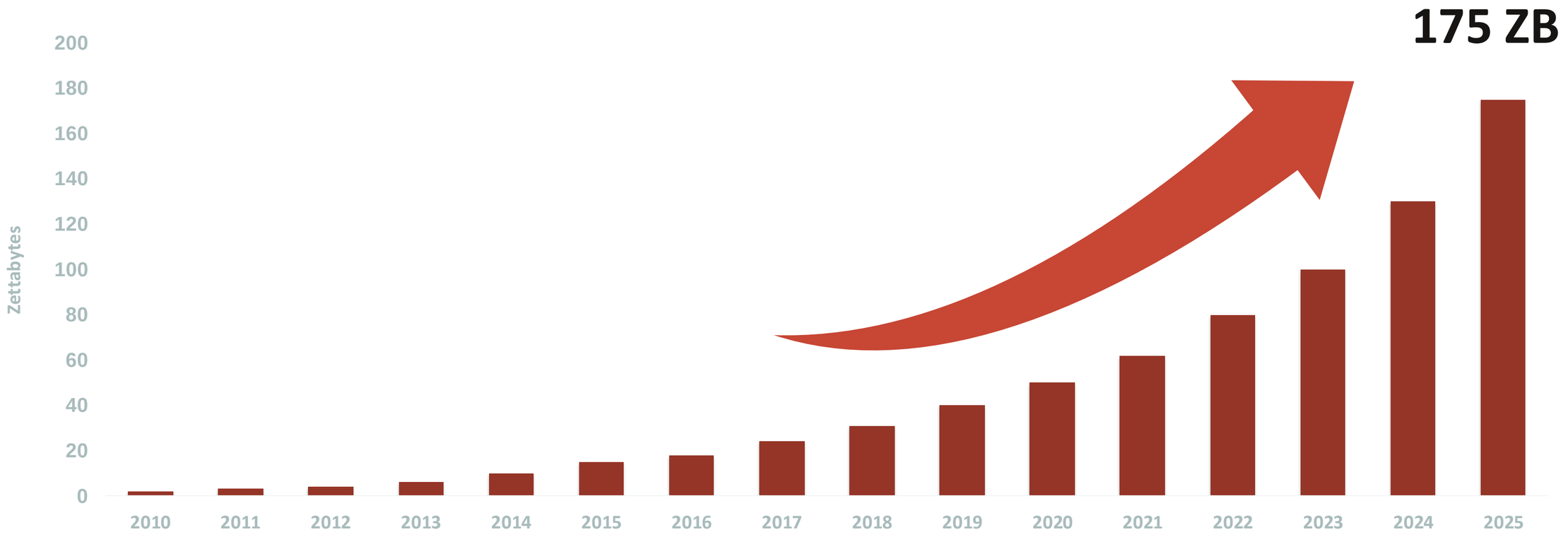
Eric Yanta
Principal MySQL Solution Engineer
eric.yanta@oracle.com

Agenda

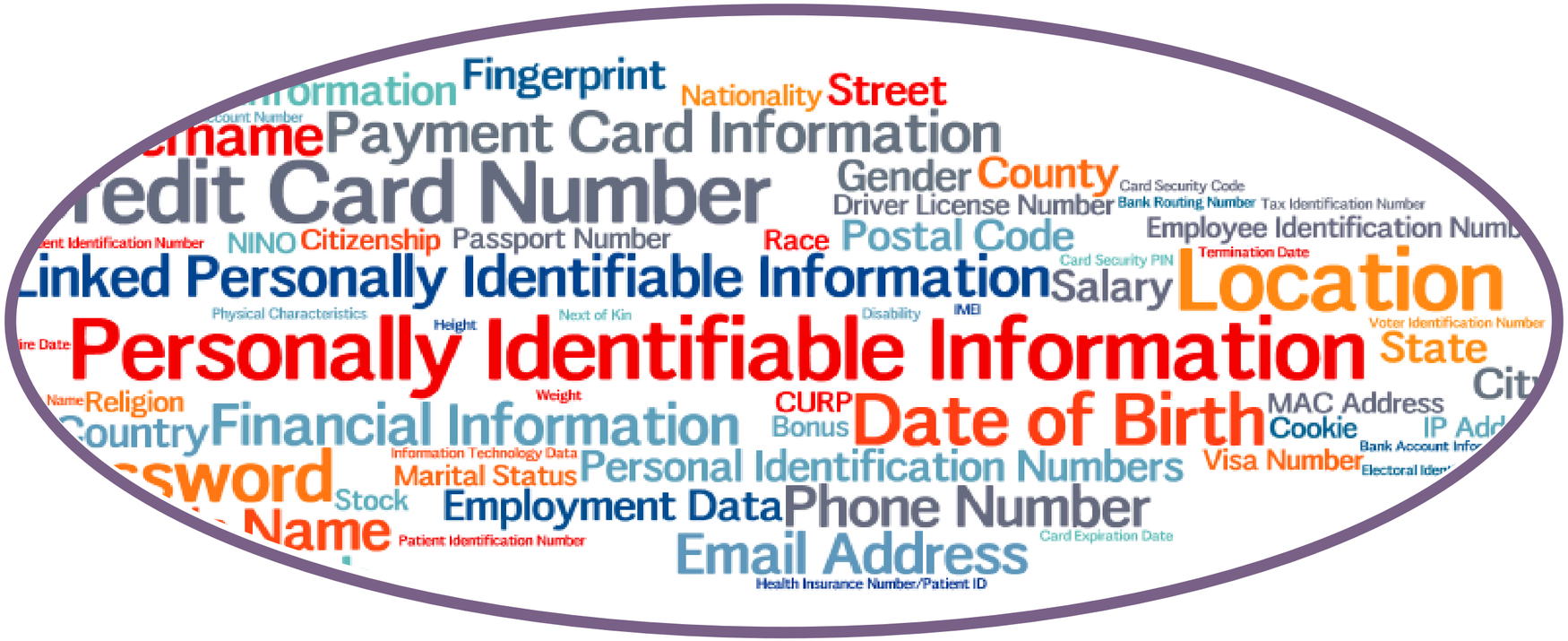
- MySQL Replication Overview
- Workshop Overview
- Setup and Installation of:
  - MySQL ReplicaSets
  - MySQL InnoDB Clusters
  - MySQL InnoDB ClusterSets

Why?

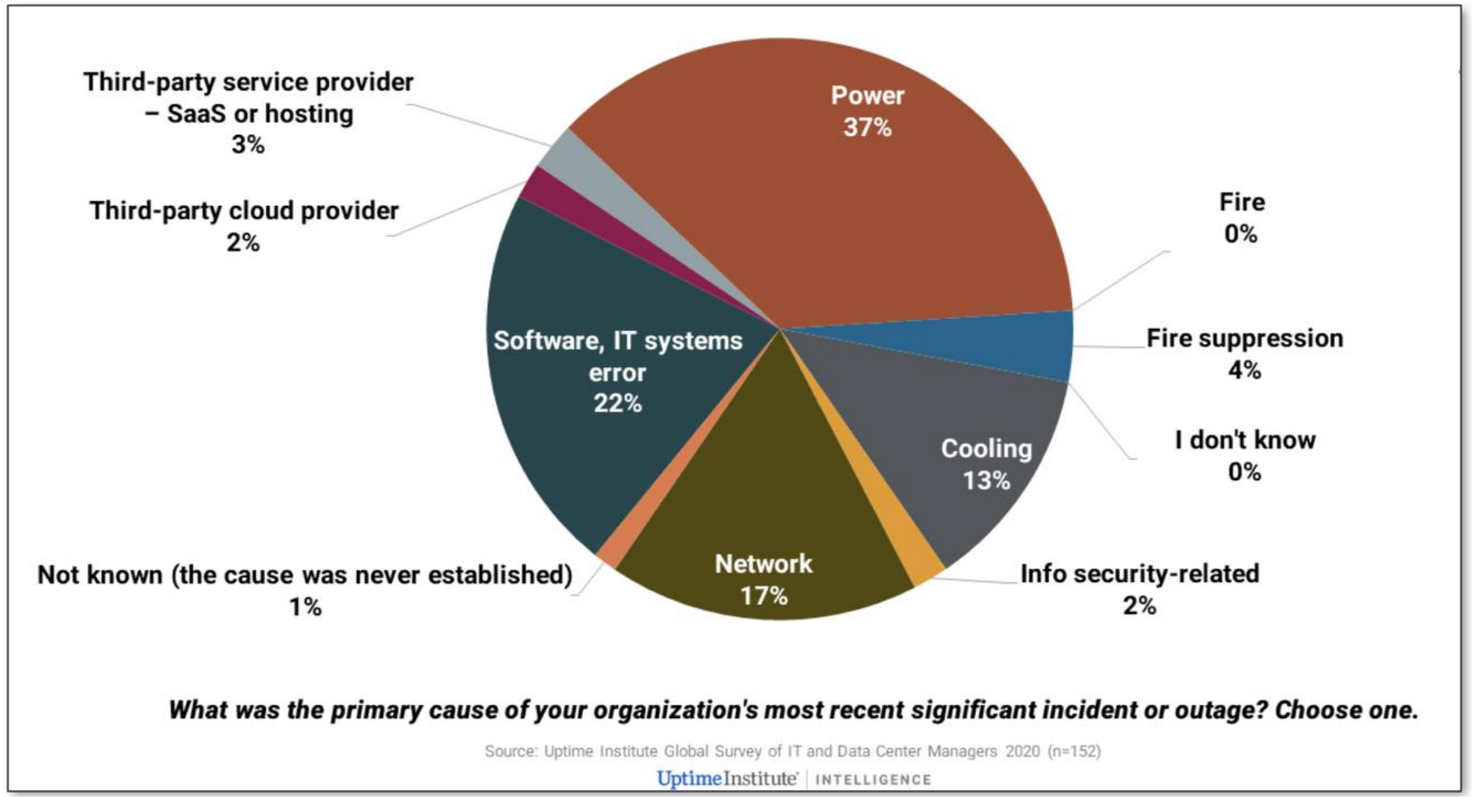
Global Datasphere



Data: Your Most Valuable Asset

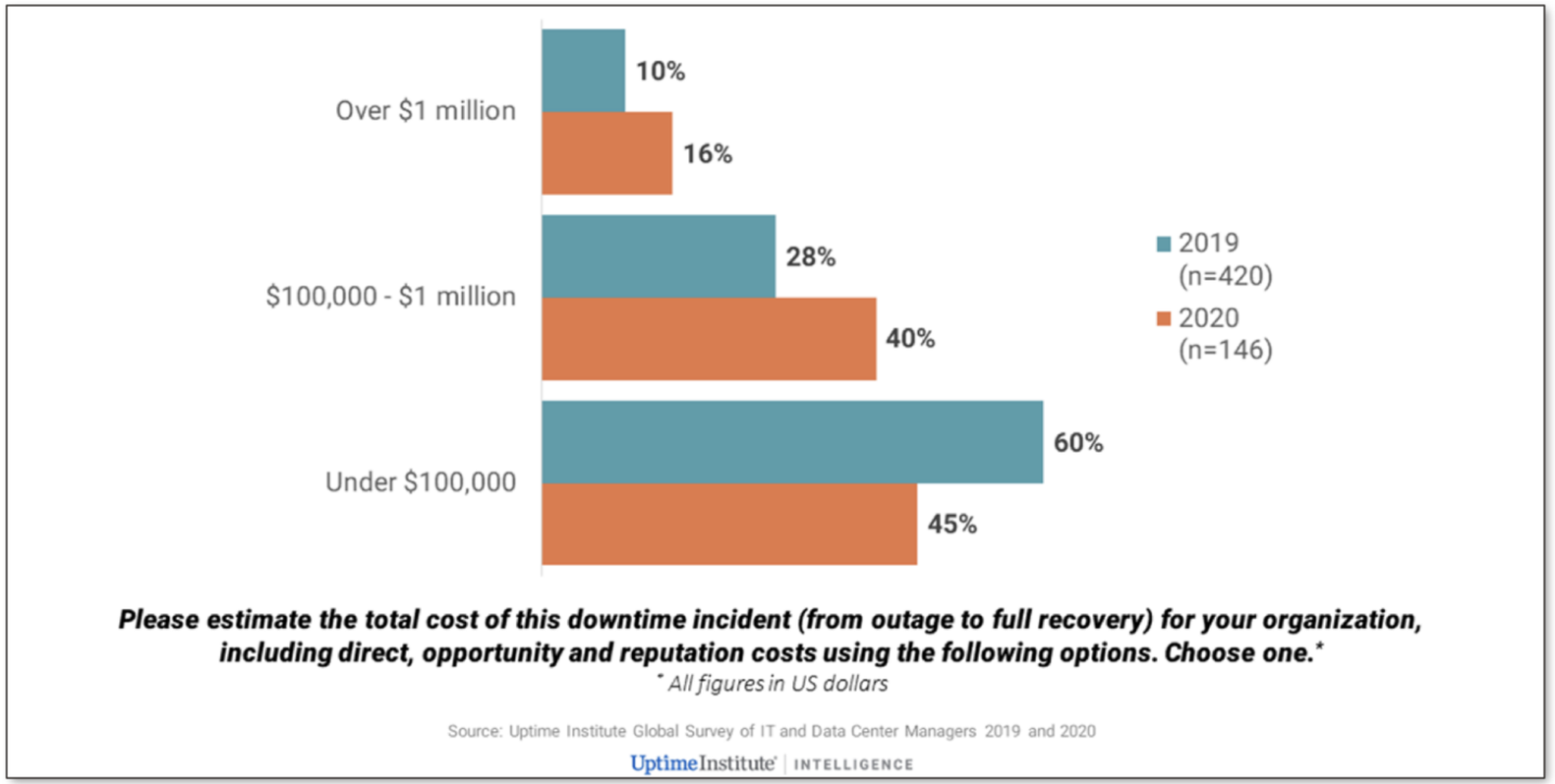


IT Disasters & Outages: Primary Causes



On-site power failure is the biggest cause of significant outages

IT Disasters & Outages: Costs are Rising



Over half who had experienced an outage costing more than $100,000.

IT Disasters and Outages: Examples

DELTA

5-hour computer outage cost us $150 million. The airline eventually canceled about 1,000 flights on the day of the outage and ground an additional 1,000 flights over the following two days.

OVHcloud

Millions of websites offline after fire at French cloud services firm. The fire is expected to cost the company more than €105 million.

BRITISH AIRWAYS

Tens of thousands of passengers were stranded in cities around the world due to cancellation of about 130 flights and the delay of 200.

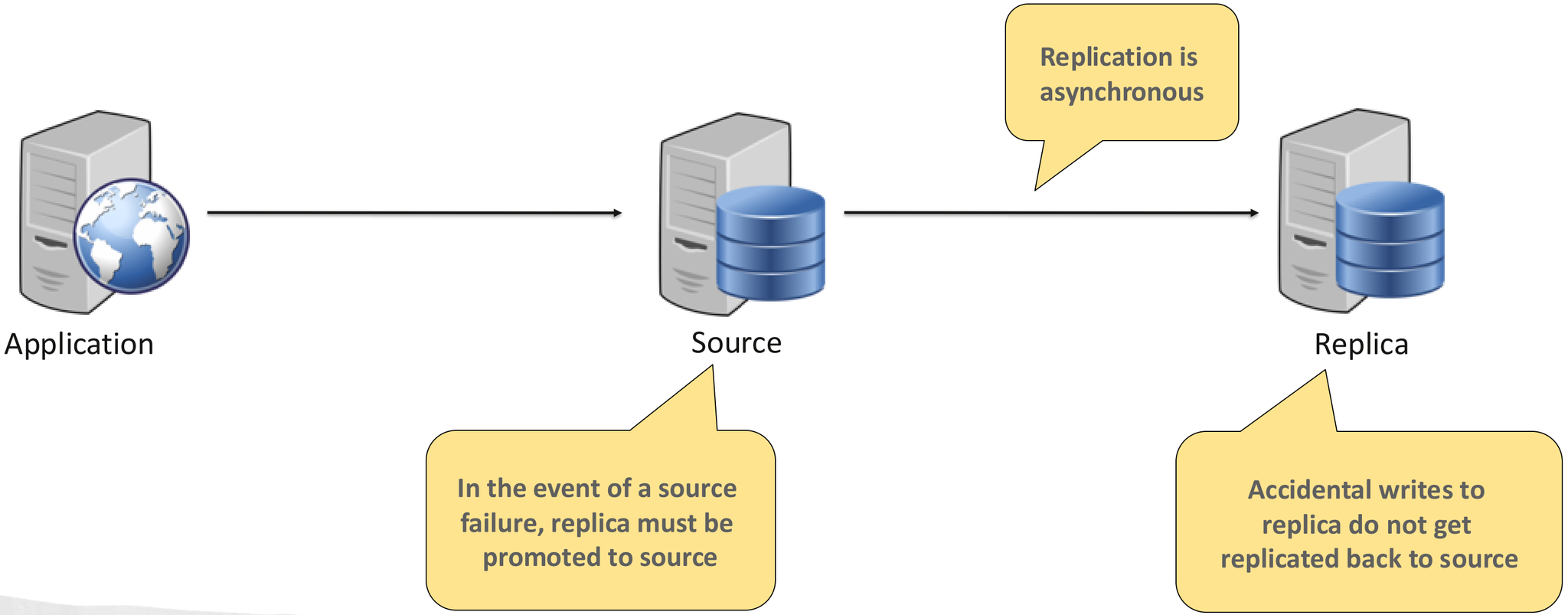
HSBC

Millions of bank customers were unable to access online accounts. The bank took almost 2 days to recover and get back to normal functioning.

MySQL Replication Overview

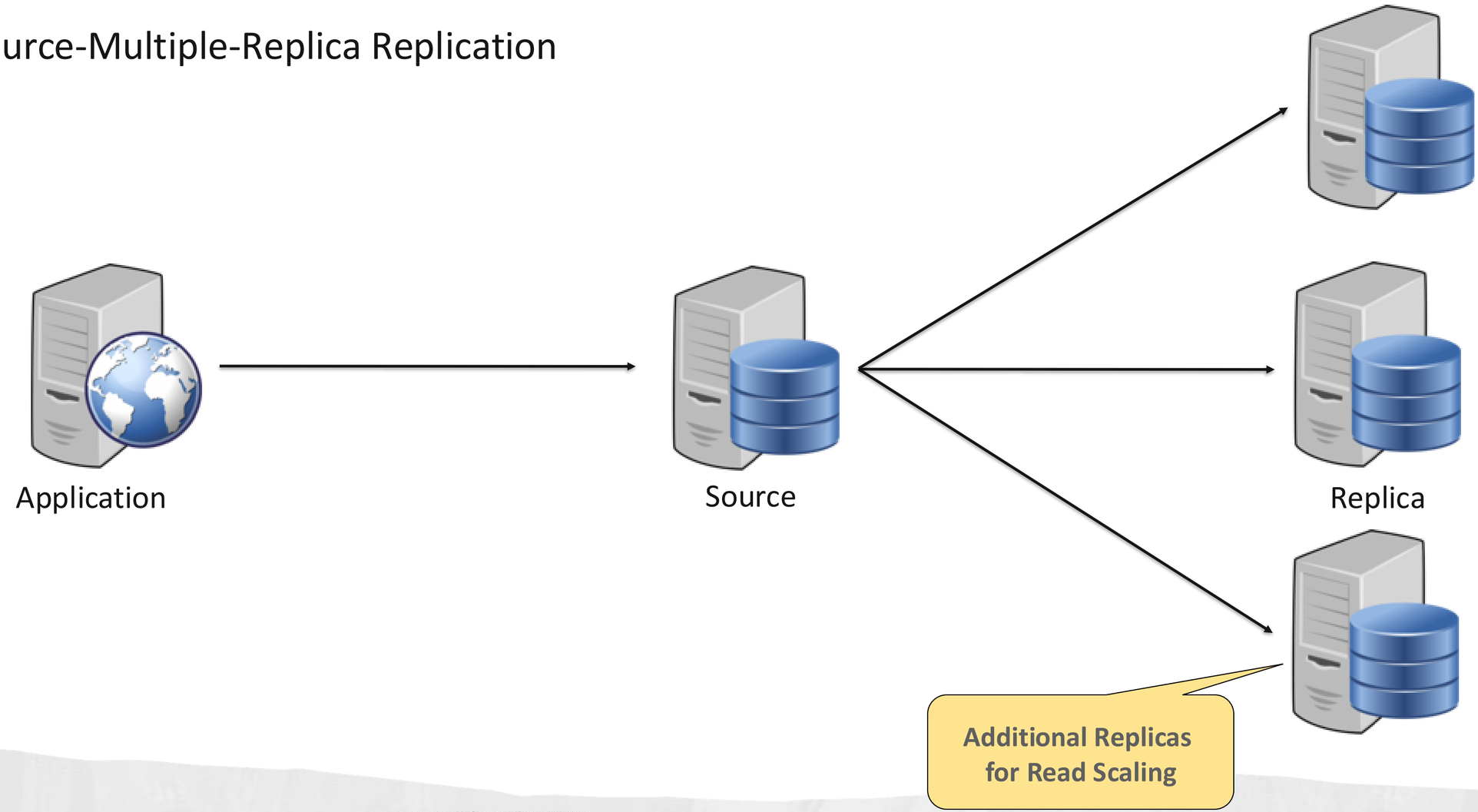
Background: Popular Solutions

Source-Replica Replication



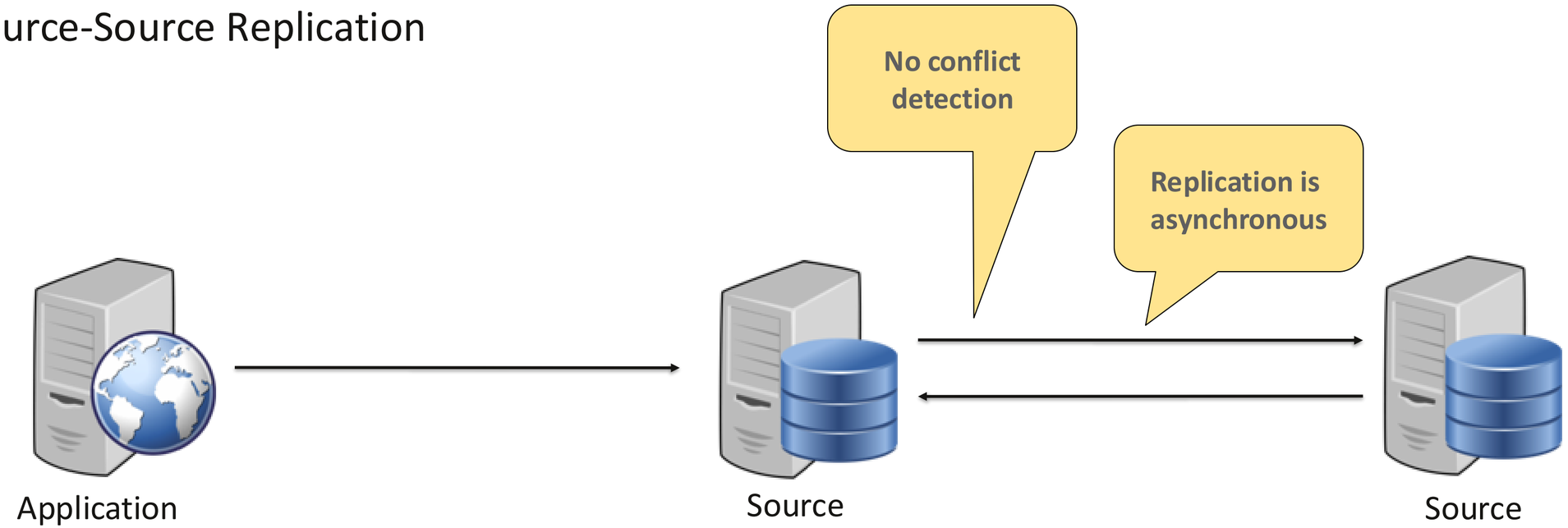
Background: Popular Solutions

Source-Multiple-Replica Replication



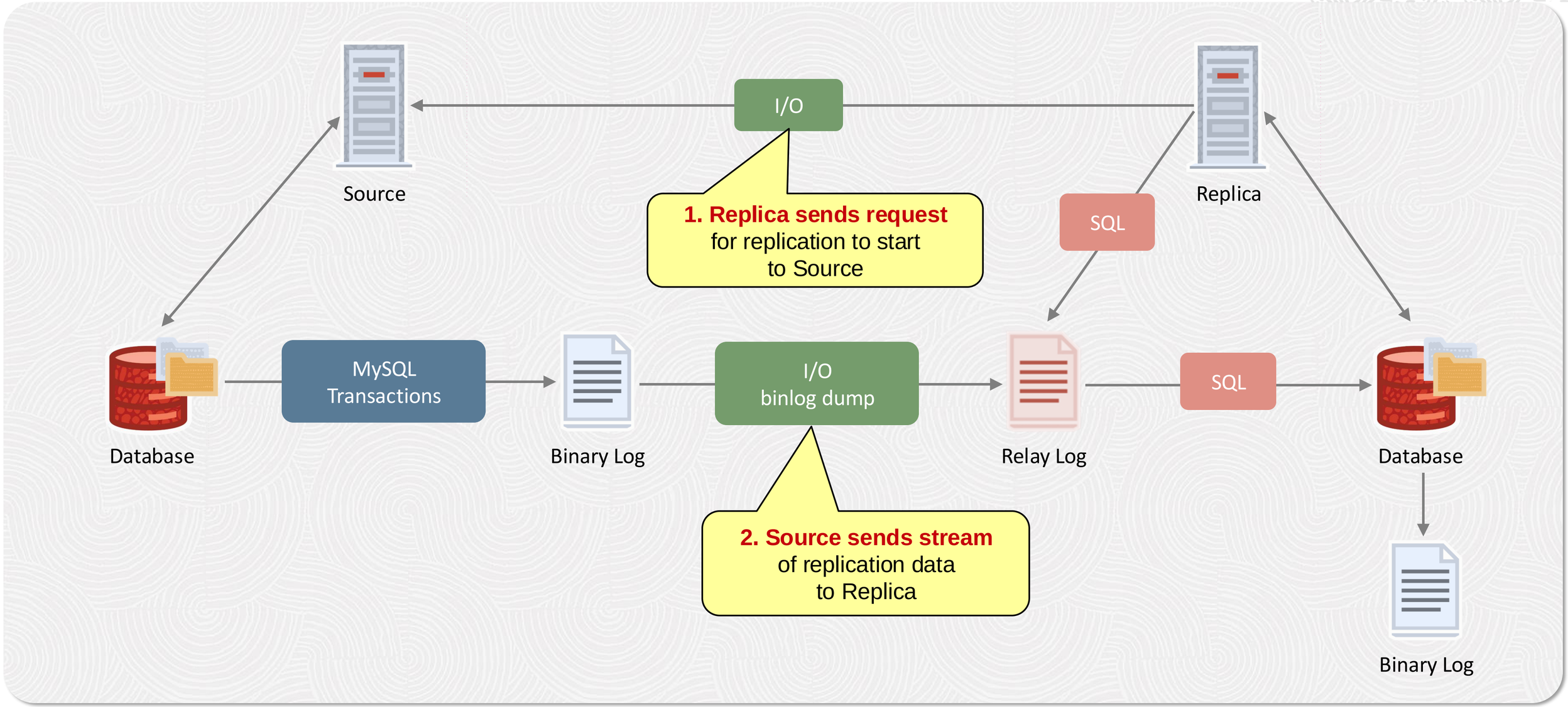
Background: Popular Solutions

Source-Source Replication

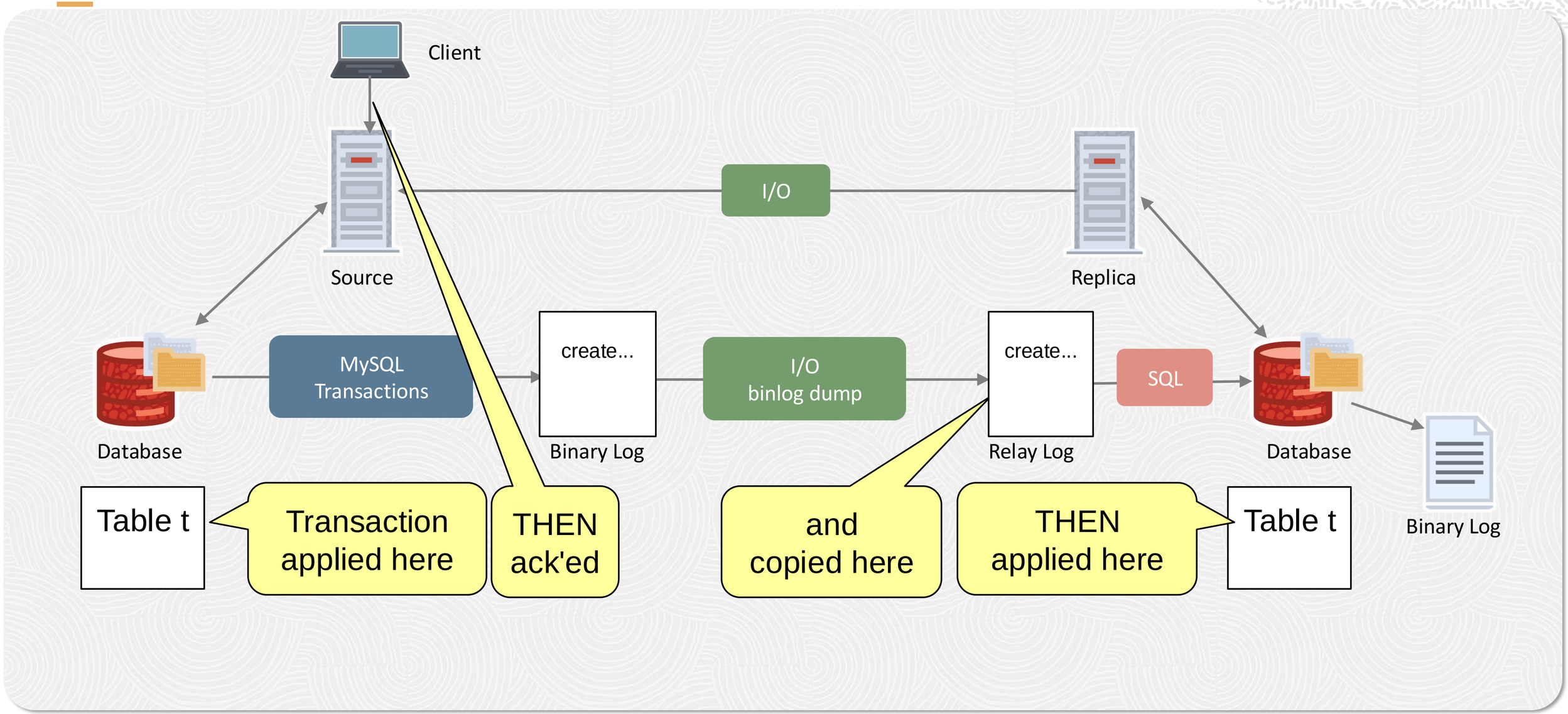


How Does Replication Work?

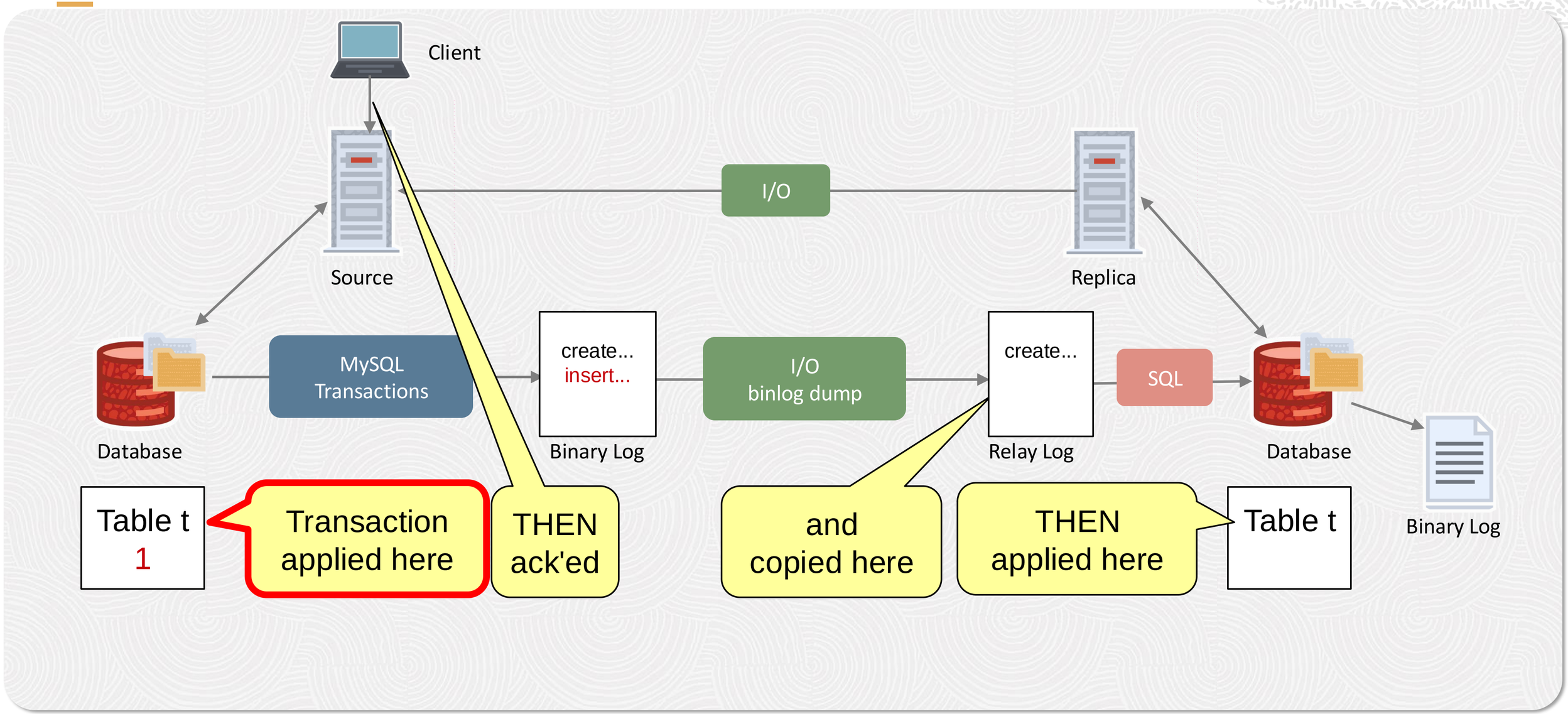
Replica Initiates Replication



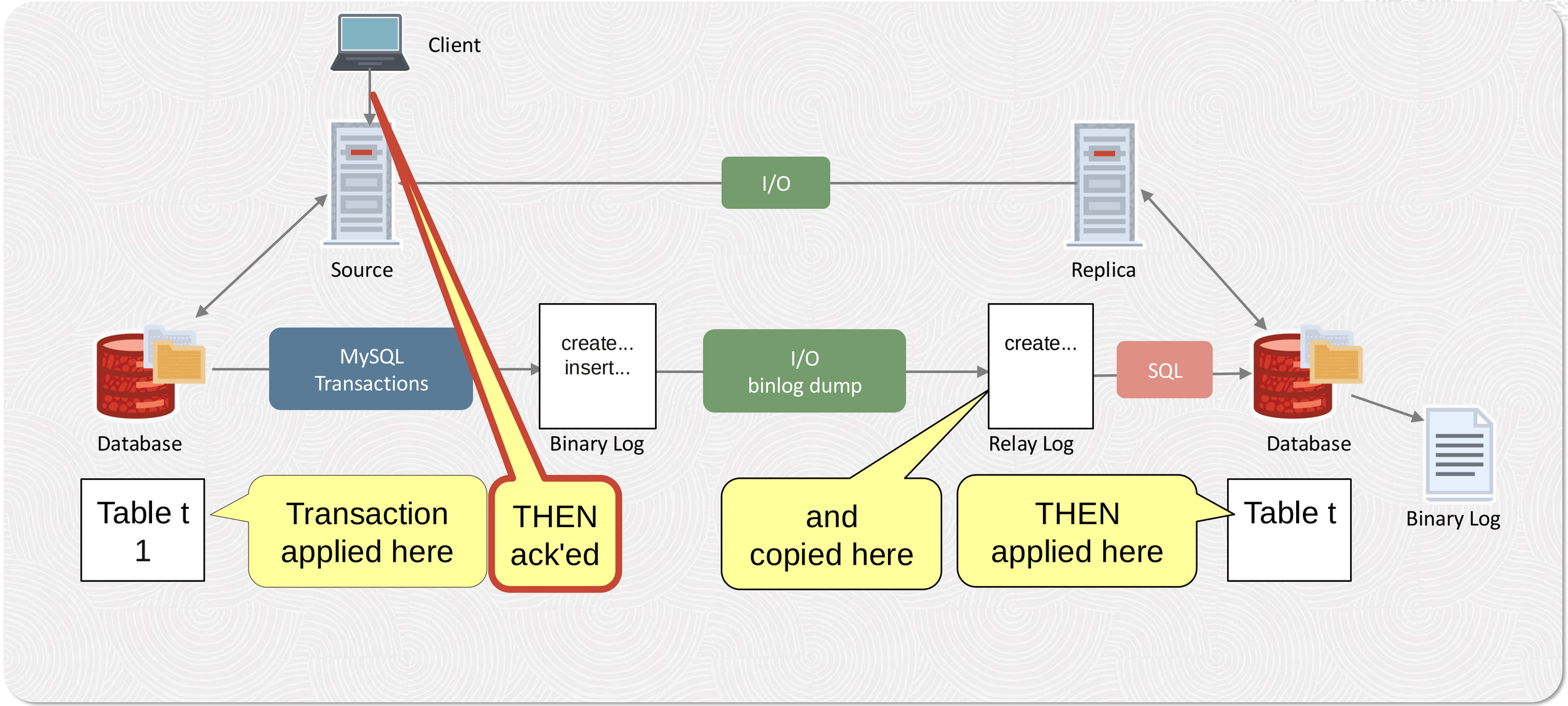
MySQL Asynchronous Replication



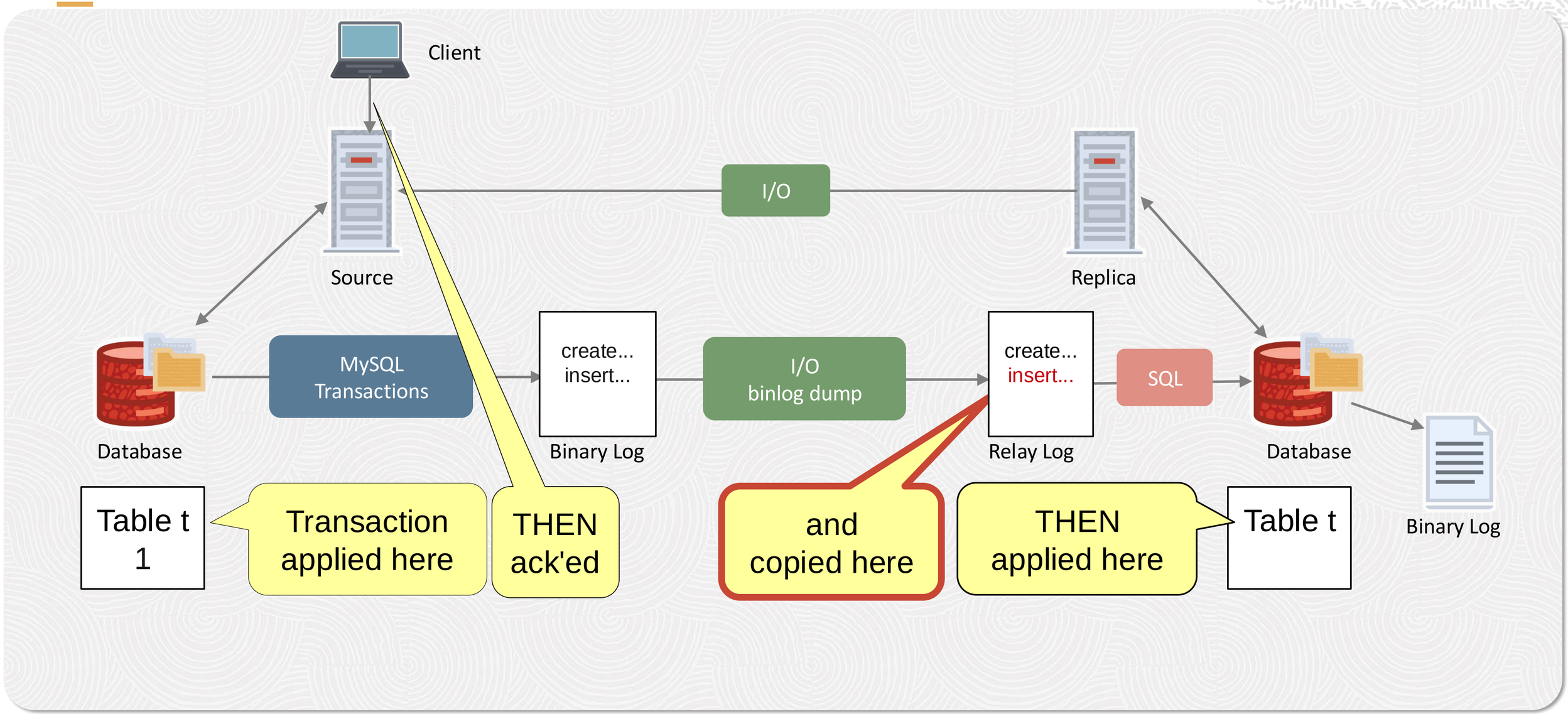
MySQL Asynchronous Replication



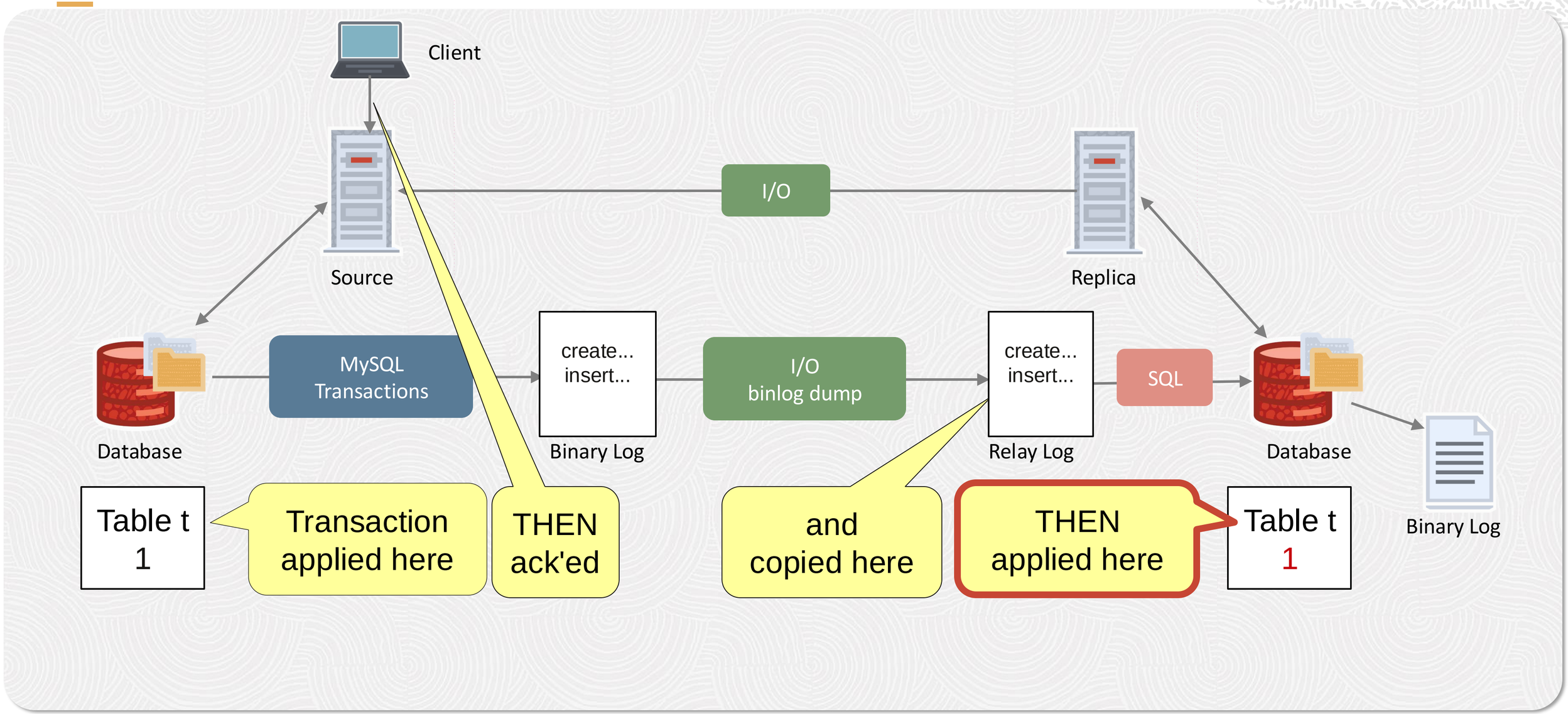
MySQL Asynchronous Replication



MySQL Asynchronous Replication



MySQL Asynchronous Replication



Change Propagation

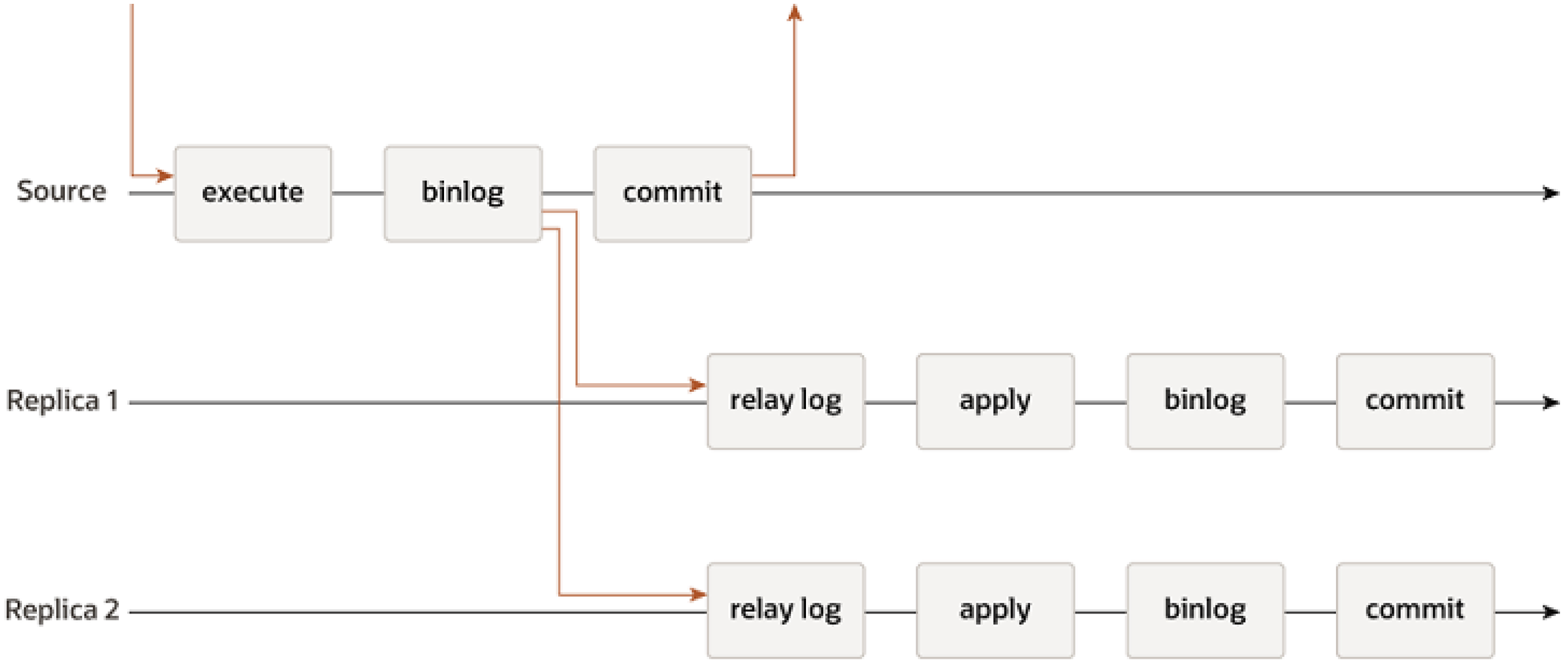
Asynchronous Replication

- Transactions committed immediately
- Events are propagated after the commit operation
- Faster, but vulnerable to lost updates on server crashes and inconsistency
- Native in MySQL Server

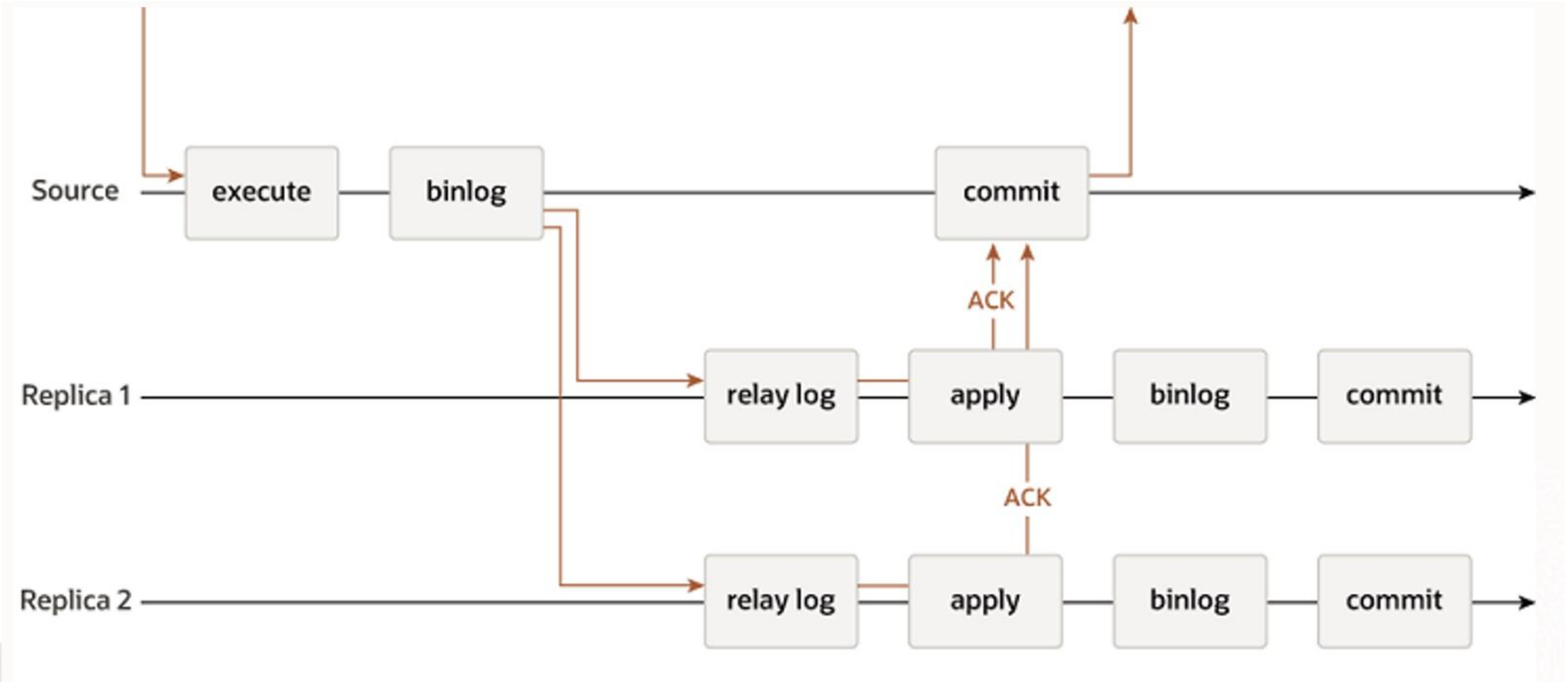
Semi-synchronous Replication

- Similar to asynchronous but commits are not acknowledged until N replicas have received and stored the correspondent events
- Introduced in MySQL 5.5

Replication Technologies: Native Replication



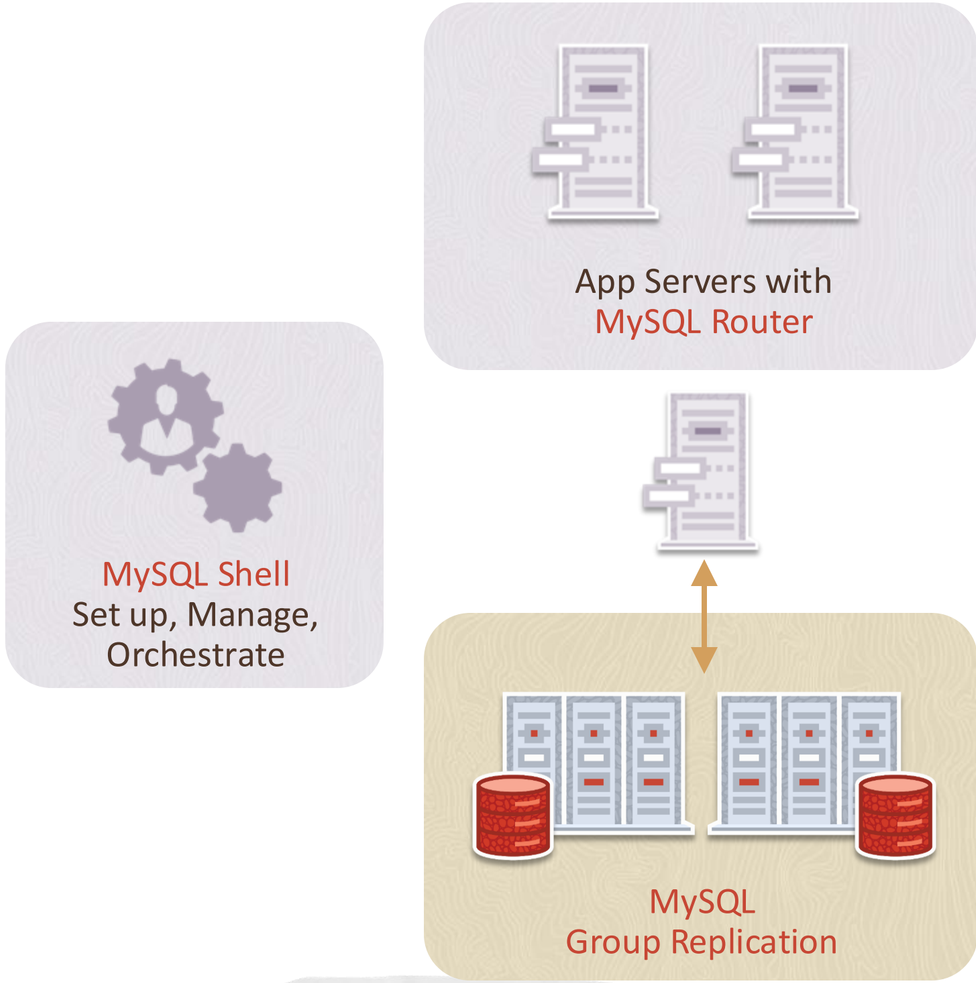
Replication Technologies: Semi-Synchronous Replication



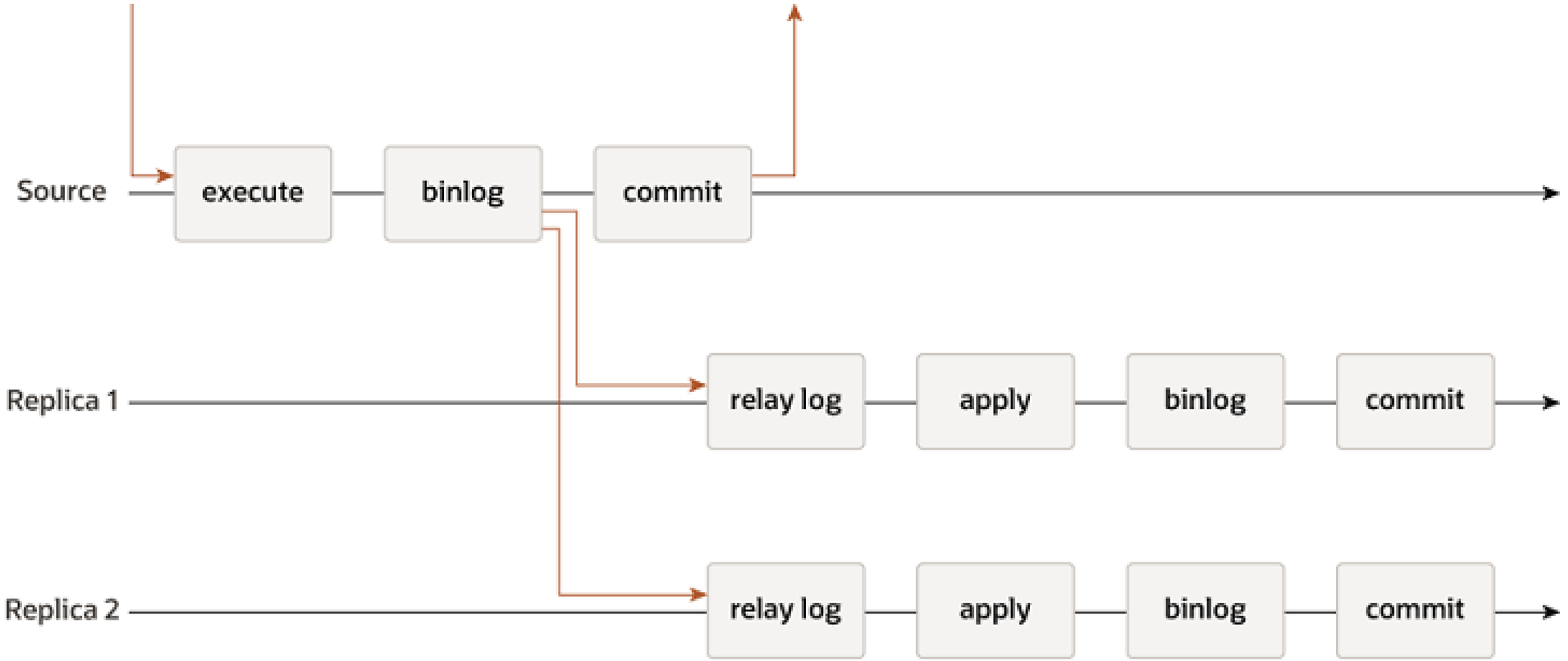
Group Replication

MySQL Group Replication

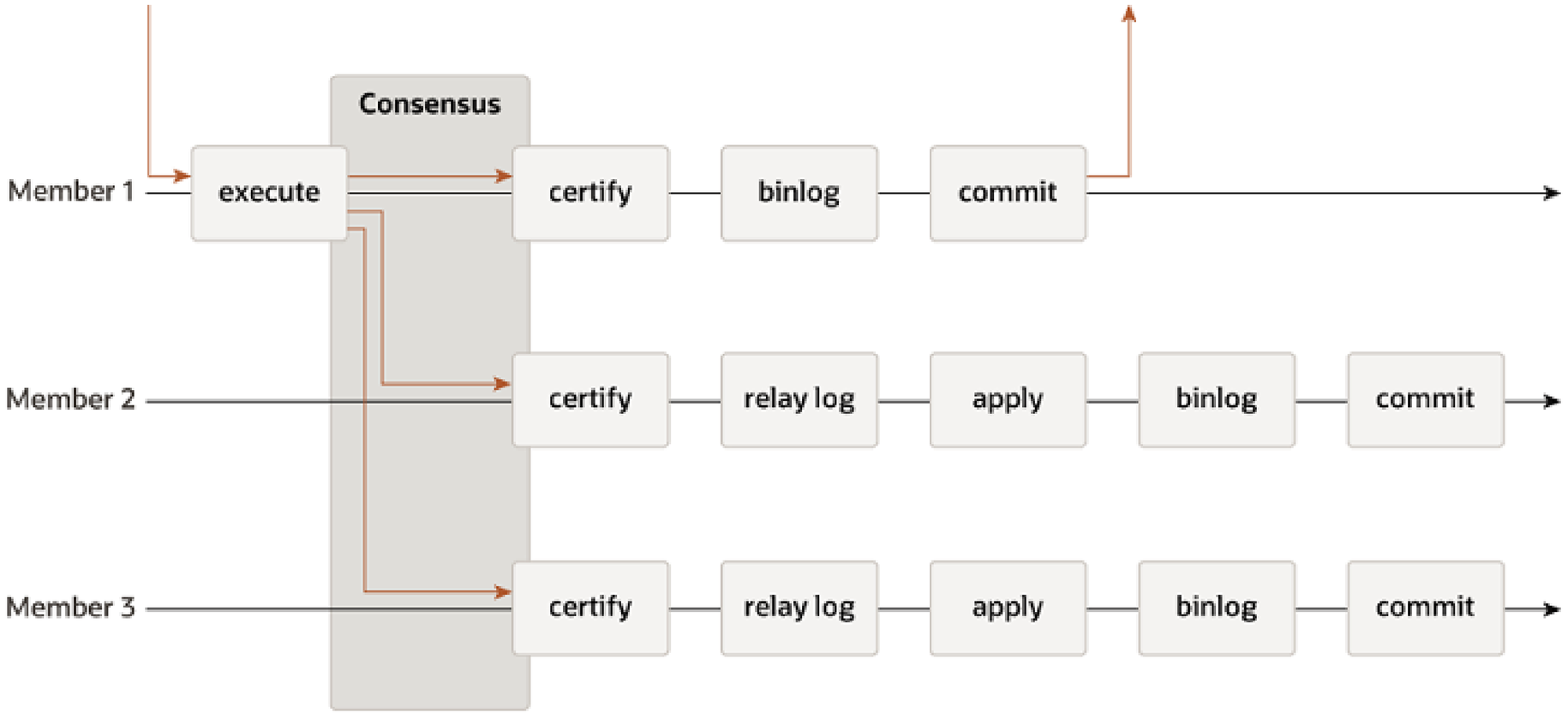
- Group Replication library:
  - Provides *virtually* synchronous replication for MySQL
  - Using MySQL replication framework by design
    - Binary & Relay logs
    - GTIDs: Global Trans+ Transaction IDs Generally
  - Is supported on all **MySQL platforms**
    - Linux, Windows, Solaris, OSX, FreeBSD
  - Automates operations
    - Conflict detection and resolution
    - Failure detection, failover, recovery
    - Group membership management and reconfiguration



Replication Technologies: Native Replication



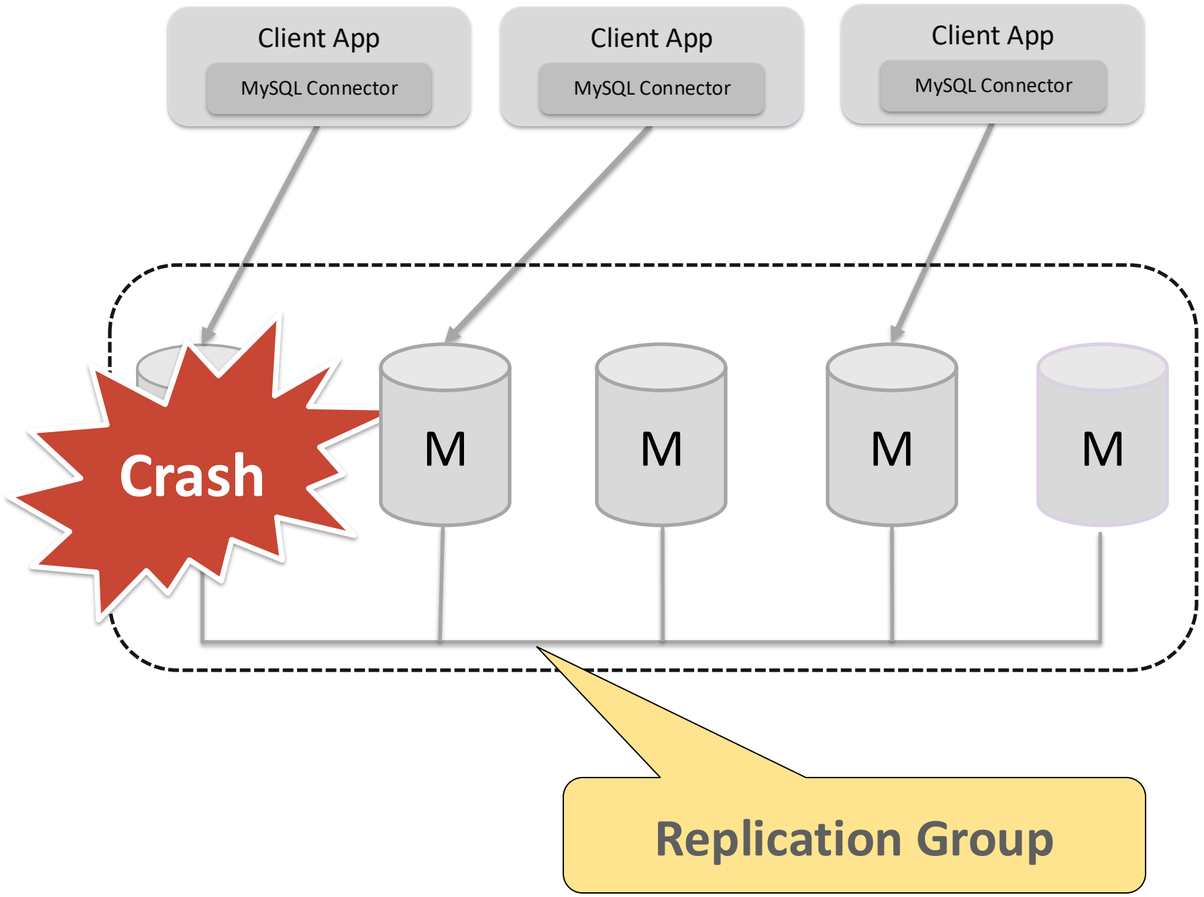
Replication Technologies: Group Replication



Understand Group Replication High Availability

Simpler Failover

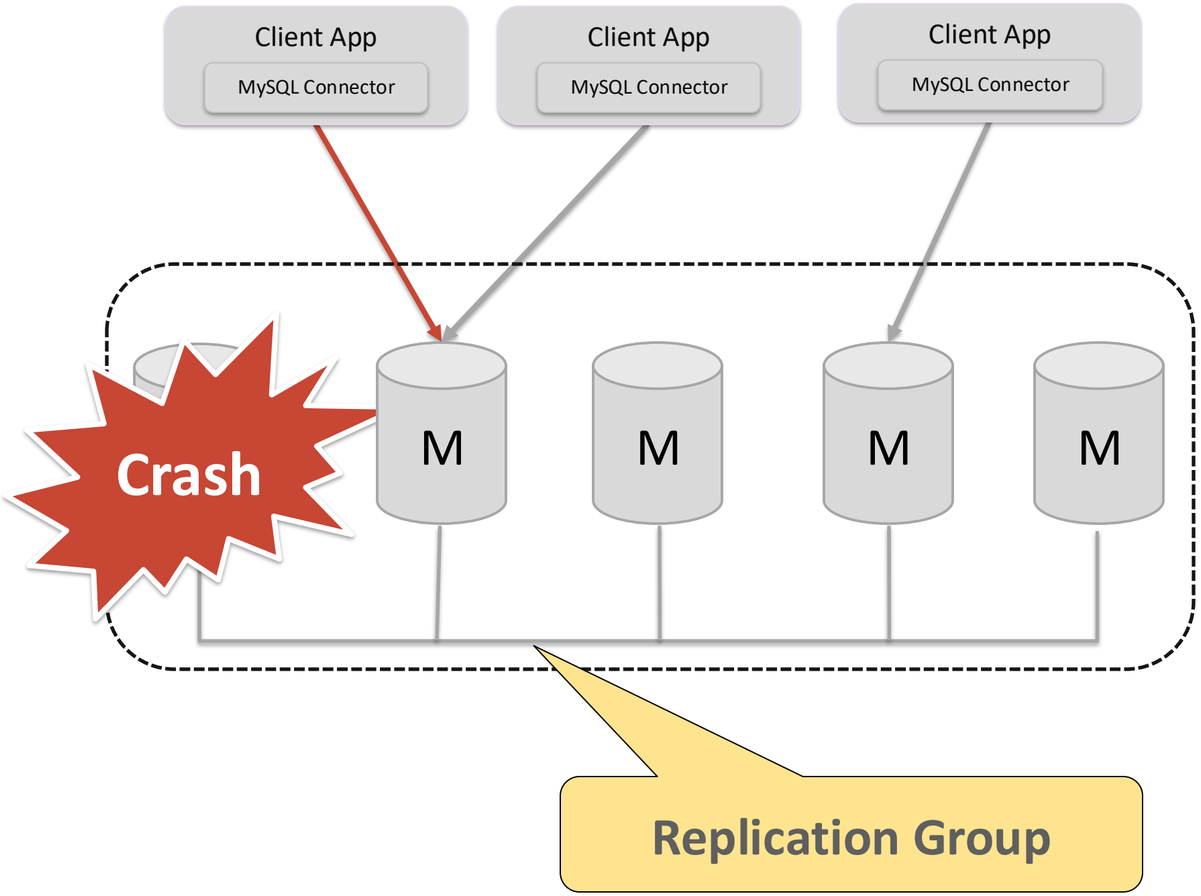
- No need to choose a new primary
- No need to configure the new primary
- No need to switch slaves to new primary



Understand Group Replication High Availability

Simpler Failover

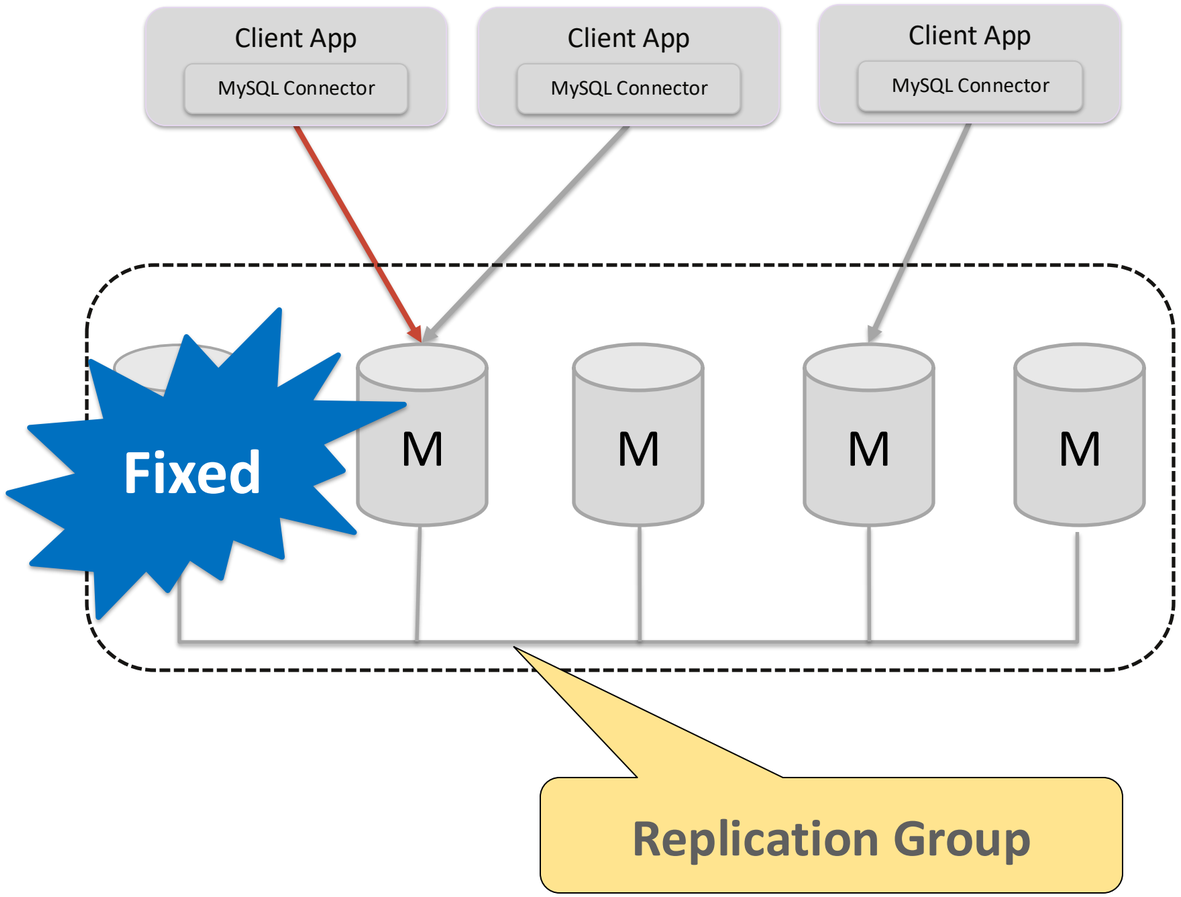
- No need to choose a new primary
- No need to configure the new primary
- No need to switch slaves to new primary
- Only need to switch crashed server’s connections to other members.



Understand Group Replication High Availability

Automatic Recovery

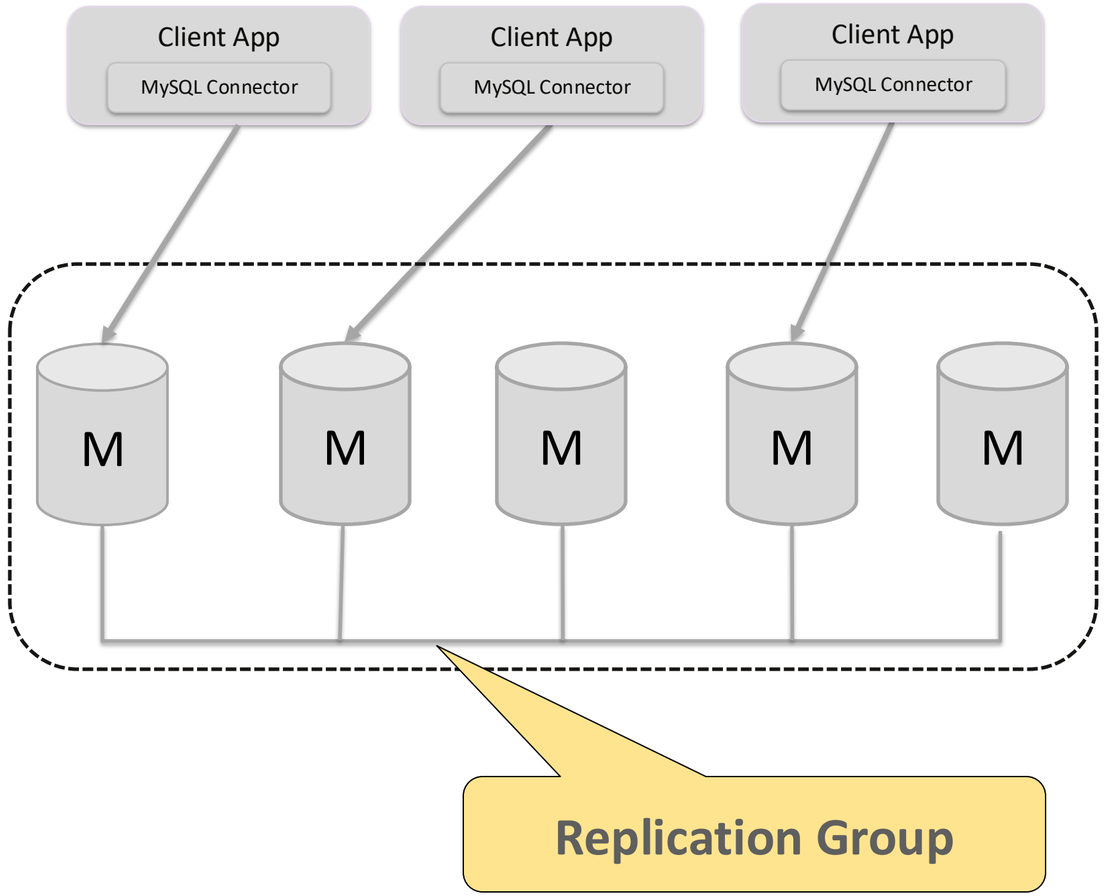
- No need to check and truncate binlog events which are not replicated
- No need to switch to new primary



Understand Group Replication High Availability

Automatic Recovery

- No need to check and truncate binlog events which are not replicated
- No need to switch to new primary
- Just need to rejoin the group
  - START GROUP_REPLICATION

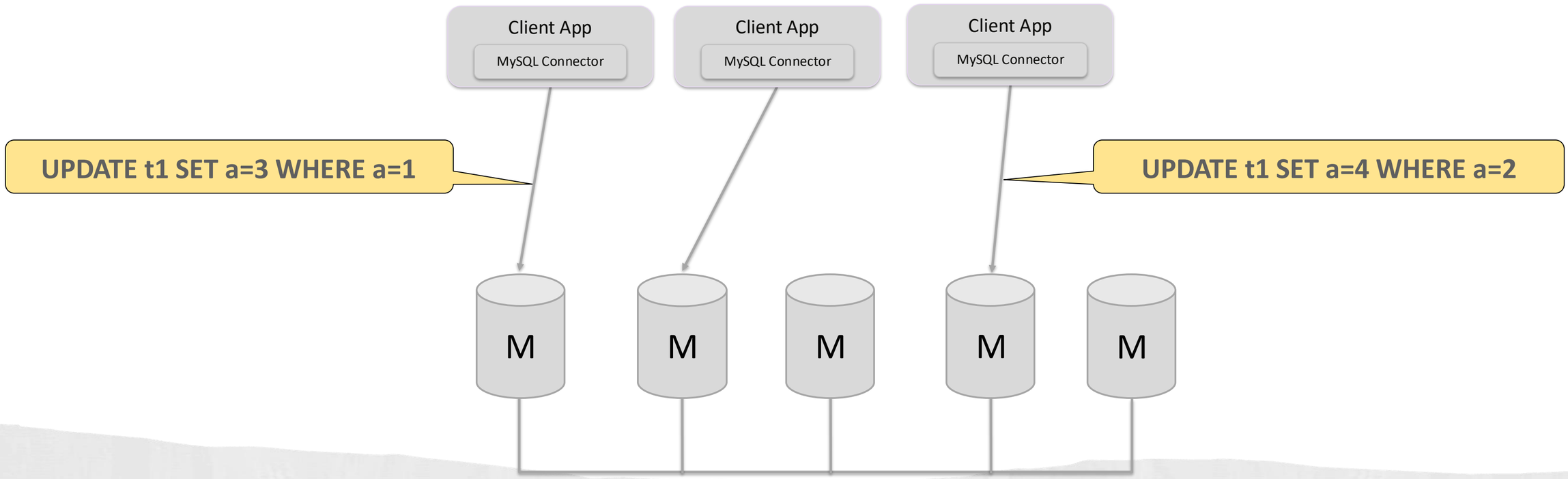


Multi-Master update everywhere!

Any two transactions on different servers can write to the same tuple.

Conflicts will be detected and dealt with.

First committer wins rule.

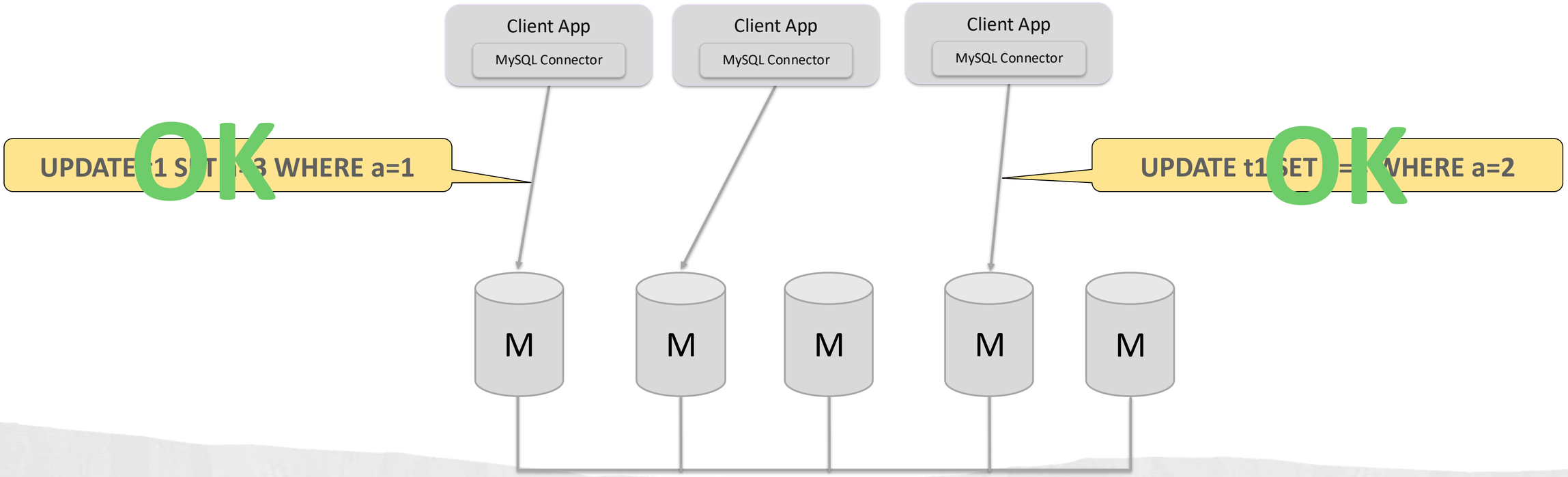


Multi-Master update everywhere!

Any two transactions on different servers can write to the same tuple.

Conflicts will be detected and dealt with.

First committer wins rule.

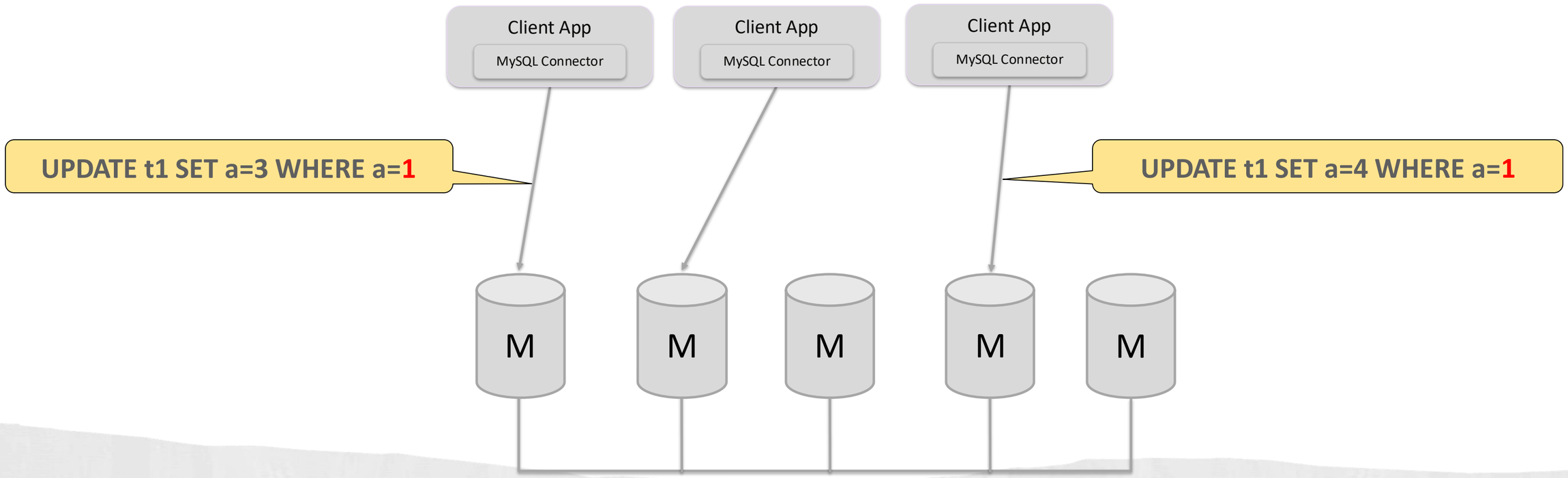


Multi-Master update everywhere!

Any two transactions on different servers can write to the same tuple.

Conflicts will be detected and dealt with.

First committer wins rule.

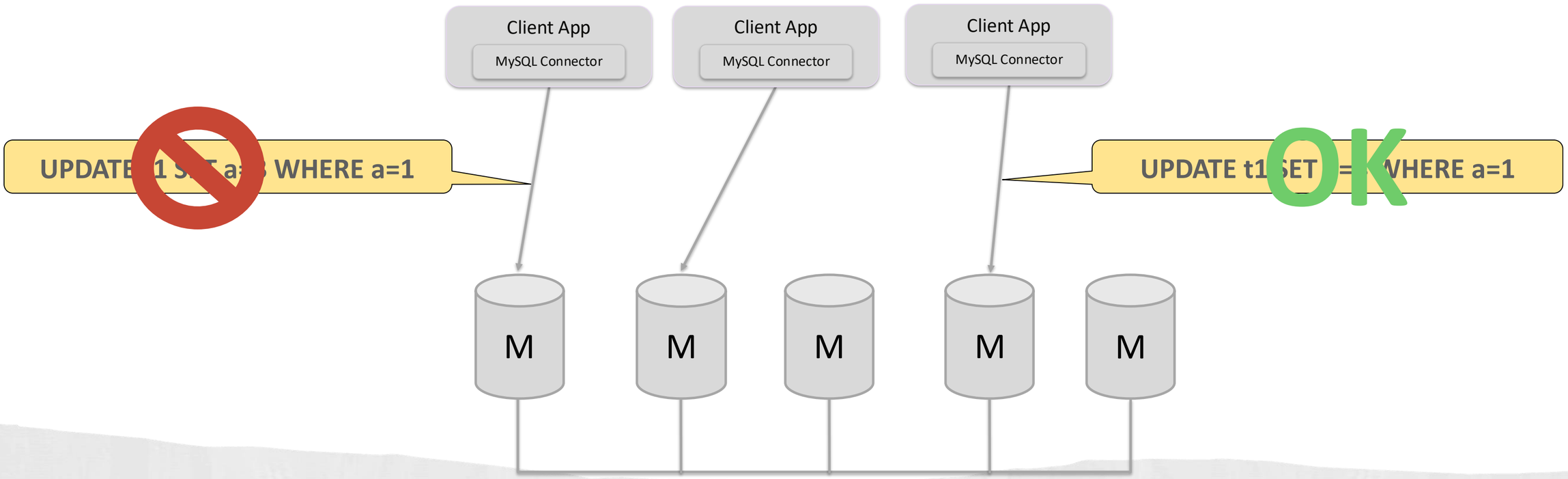


Multi-Master update everywhere!

Any two transactions on different servers can write to the same tuple.

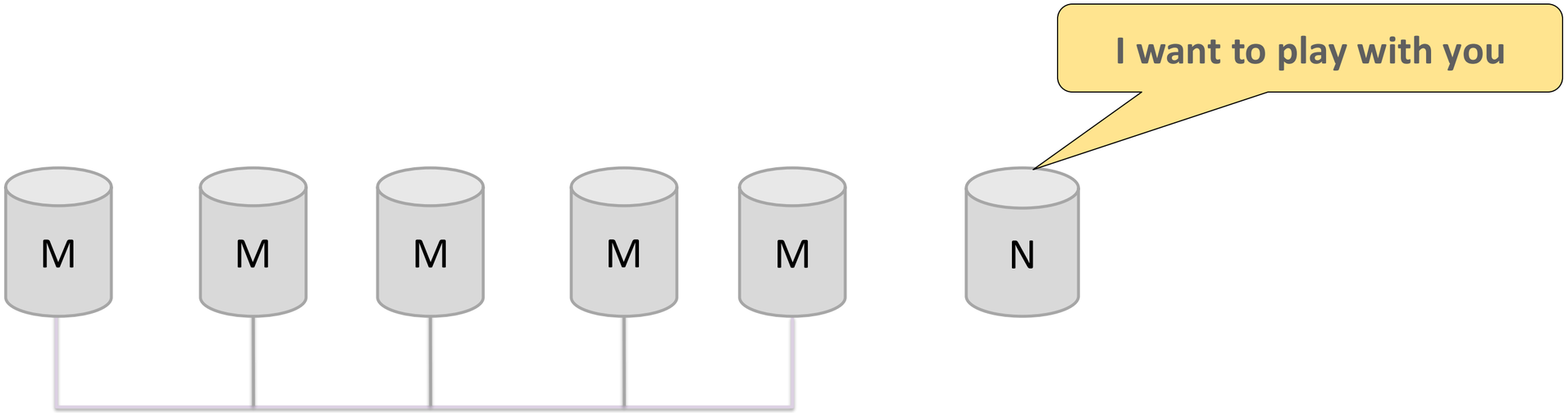
Conflicts will be detected and dealt with.

First committer wins rule.



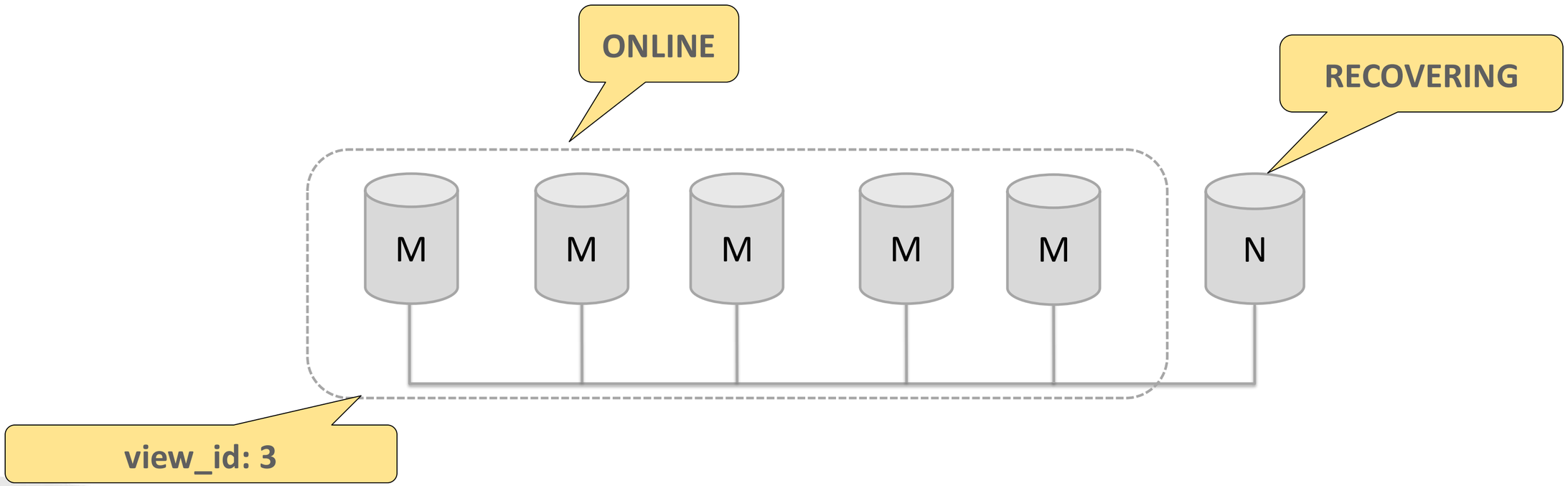
Automatic distributed server recovery!

Server that joins the group will automatically synchronize with the others.



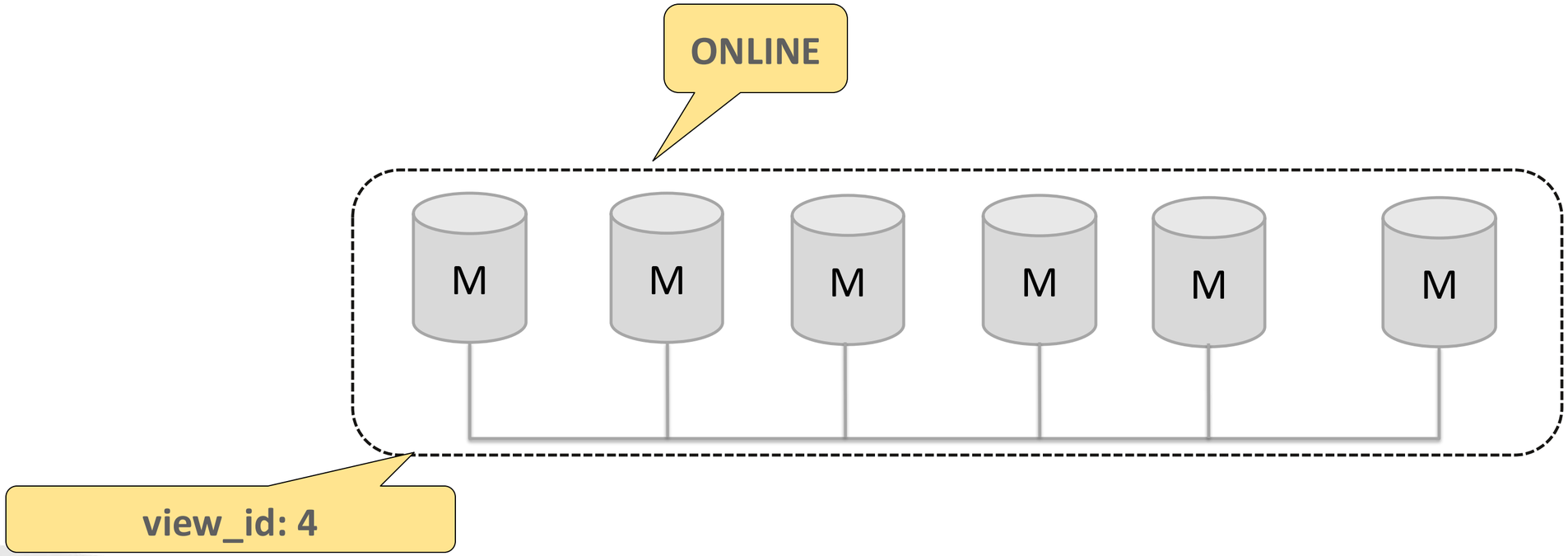
Automatic distributed server recovery!

Server that joins the group will automatically synchronize with the others.



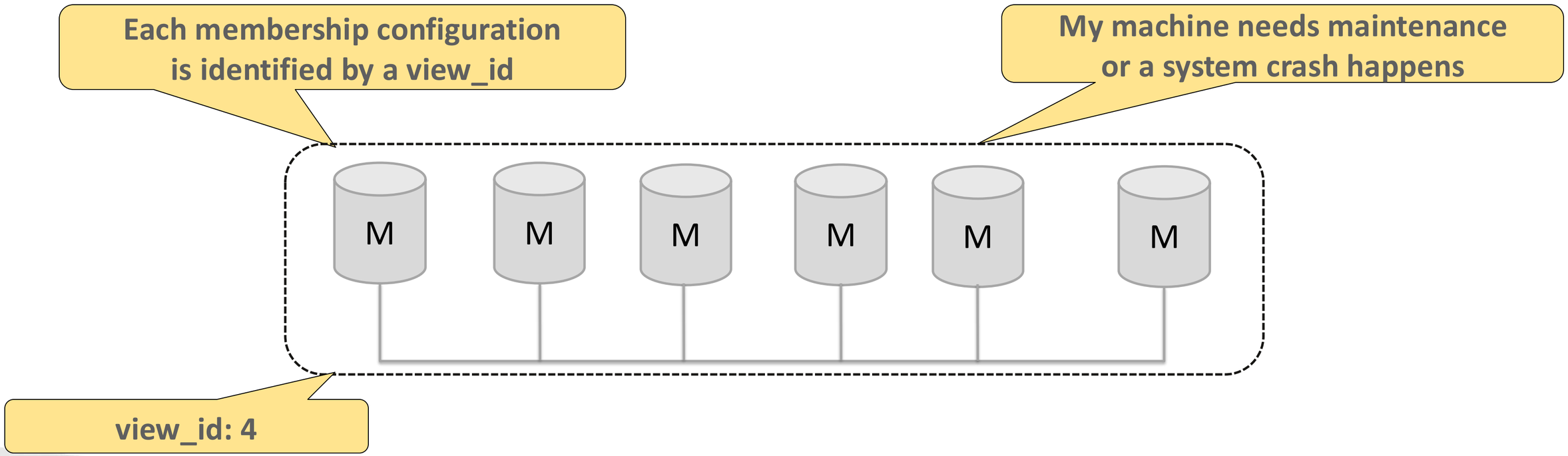
Automatic distributed server recovery!

Server that joins the group will automatically synchronize with the others.



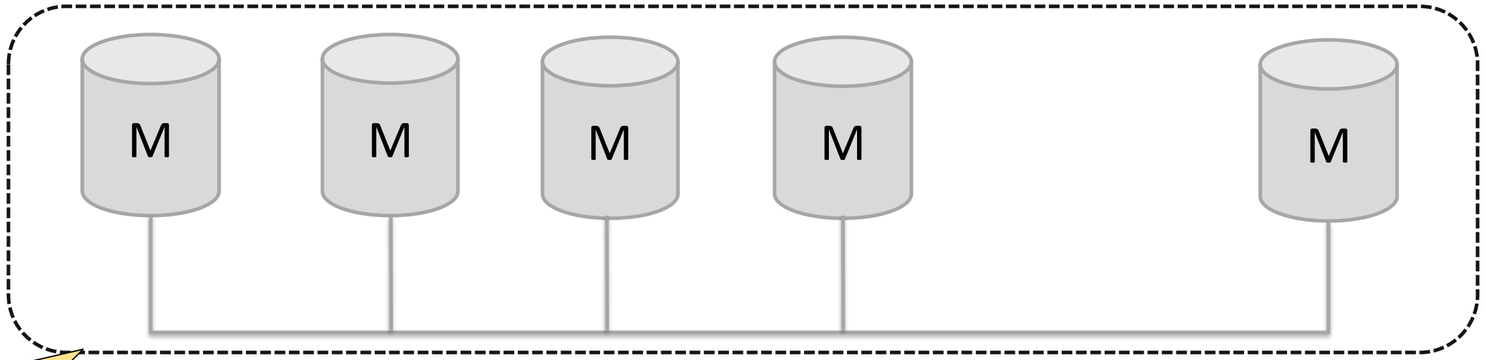
Automatic distributed server recovery!

If a server leaves the group, the others will automatically be informed.



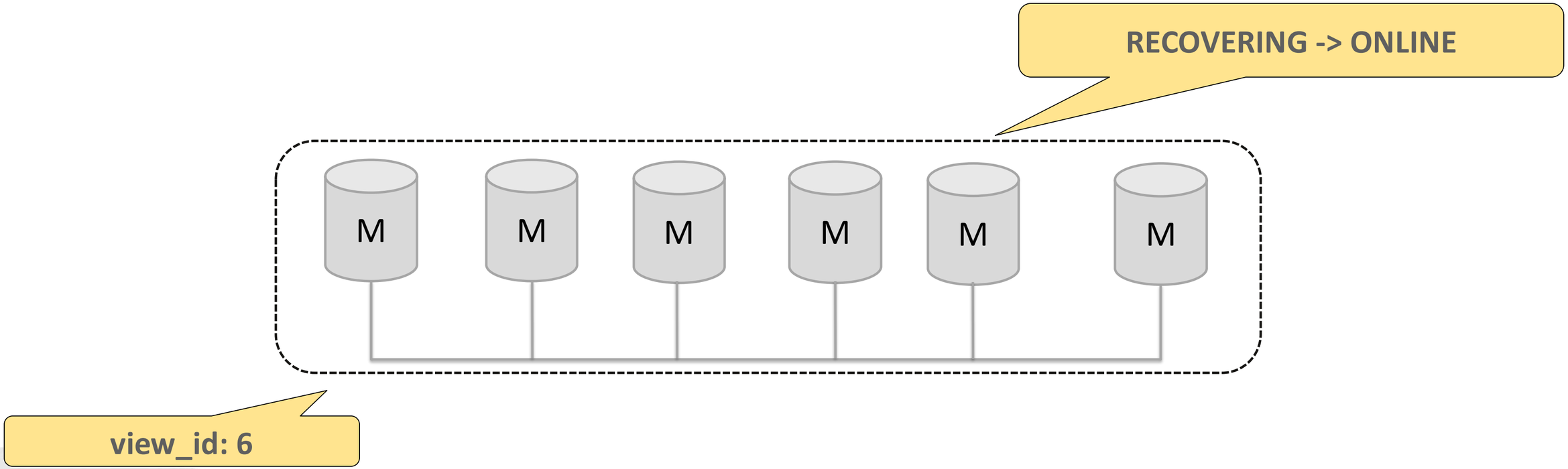
Automatic distributed server recovery!

If a server leaves the group, the others will automatically be informed.



Automatic distributed server recovery!

Server that (re)joins the group will automatically synchronize with the others.



Consistency Levels

Eventual Consistency (default)

- Transaction does not wait at all.
- Executes on the current snapshot of the data on that member.

Before Consistency (Synchronize on Reads)

- Transaction waits for all preceding transactions to complete.
- Executes on the most up to date snapshot of the data in the group.

After Consistency (Synchronize on Writes)

- Transaction waits until all members have executed it.
- Executes on the current snapshot of the data on that member.

Consistency Levels

Before and After (Yes, you can **combine** both)

- Transaction waits for all preceding transactions and for all members to execute it.
- Executes on the most up to date snapshot of the data in the group and updates everywhere before returning to the application.

Before On Primary Fail-over

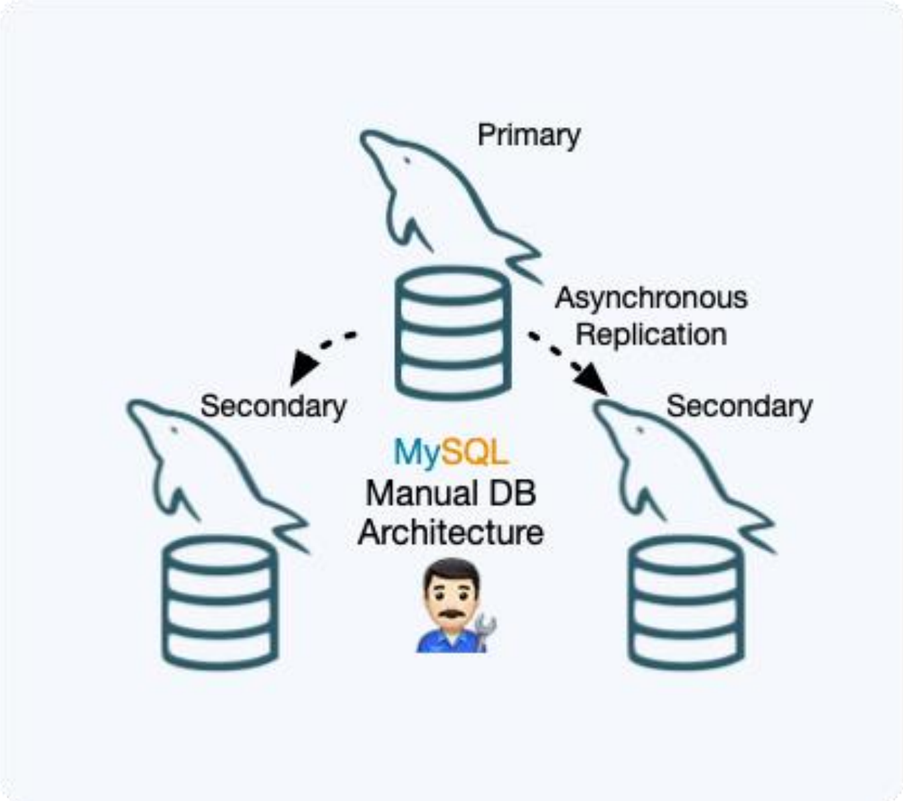
- Transaction waits for all transactions in the new primary’s replication backlog to be executed.
- Executes on the snapshot of the data that the old primary was in when it stepped down (or crashed).

Hardware Configuration

- All servers in a Group should have similar specifications.
- Server resources should be isolated for MySQL use.
- The number of servers in the group must be **3, 5, 7, or 9**.
- Ideally, storage hardware should be SSDs.
- Use redundant hardware where possible:
  - Storage
  - Power supplies
  - Network cards

Past & Present

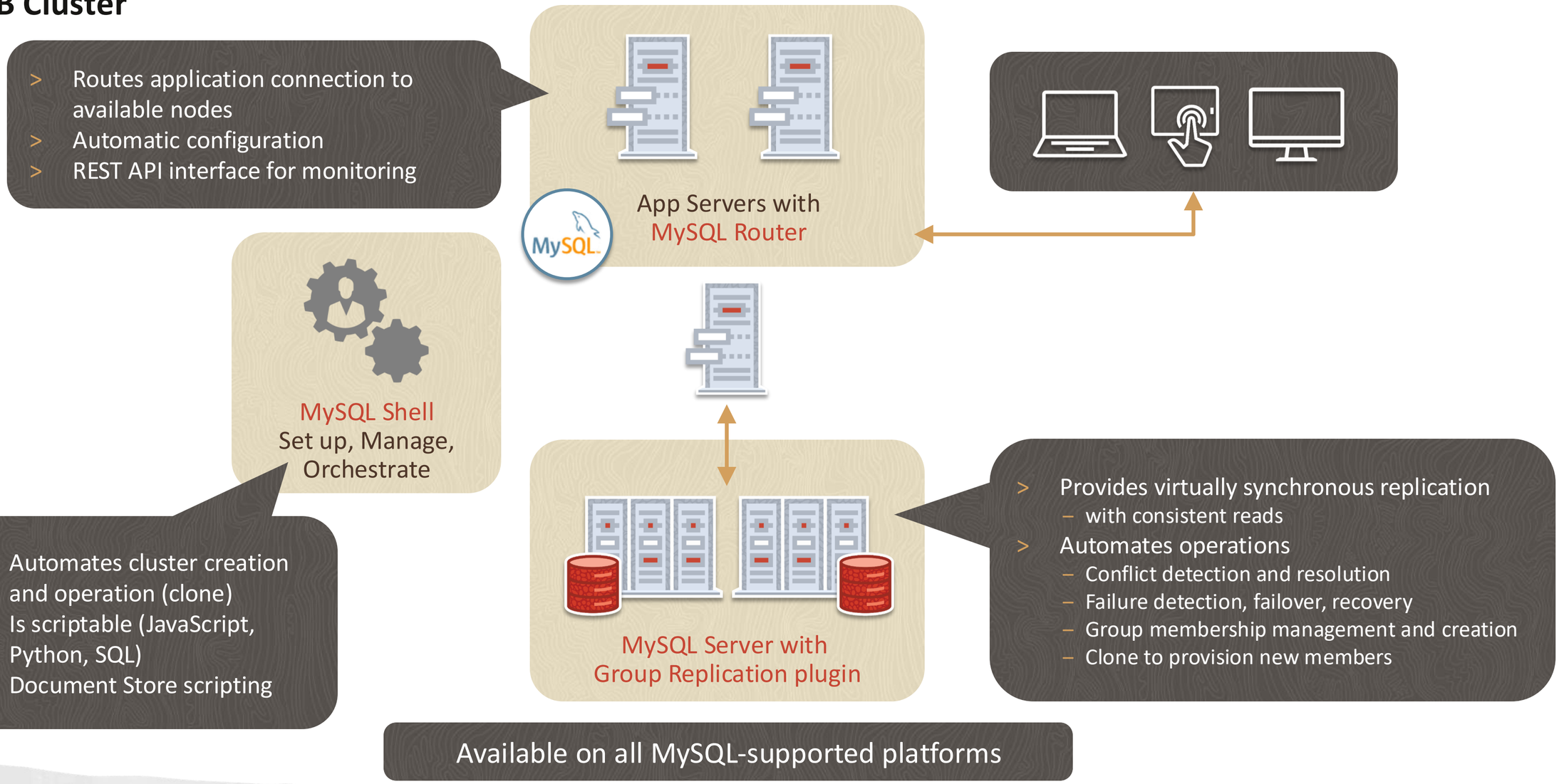
'Past' - Manual



- Setting up Replication topology was usually done manually, taking many steps
  - including user management, restoring backups, configuring replication…
- MySQL only offered the technical pieces, leaving it up to the user to setup an (always customized) architecture
- Even required other software … bringing lot's of work for DBA's and experts, who spent their time automating and integrating their customized architecture

Present – Solutions!

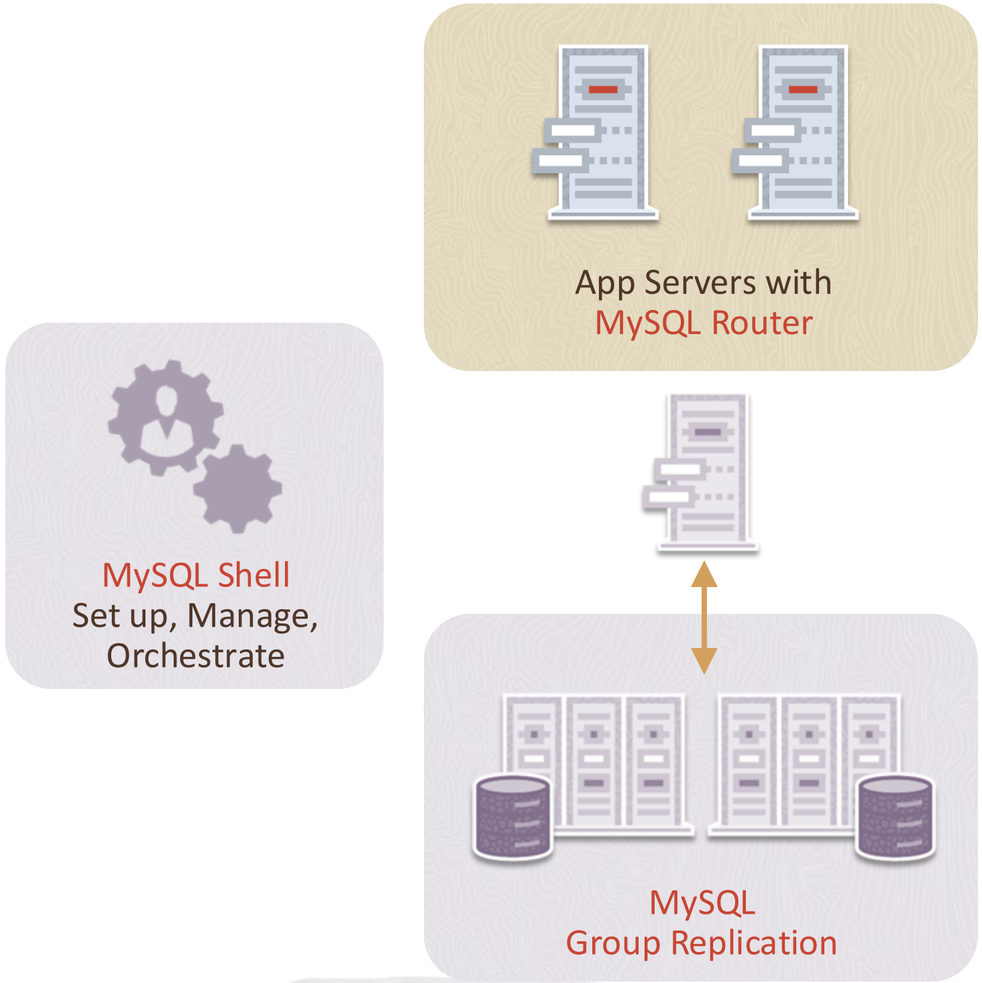
InnoDB Cluster



MySQL Router

Transparent access to HA databases for MySQL Applications

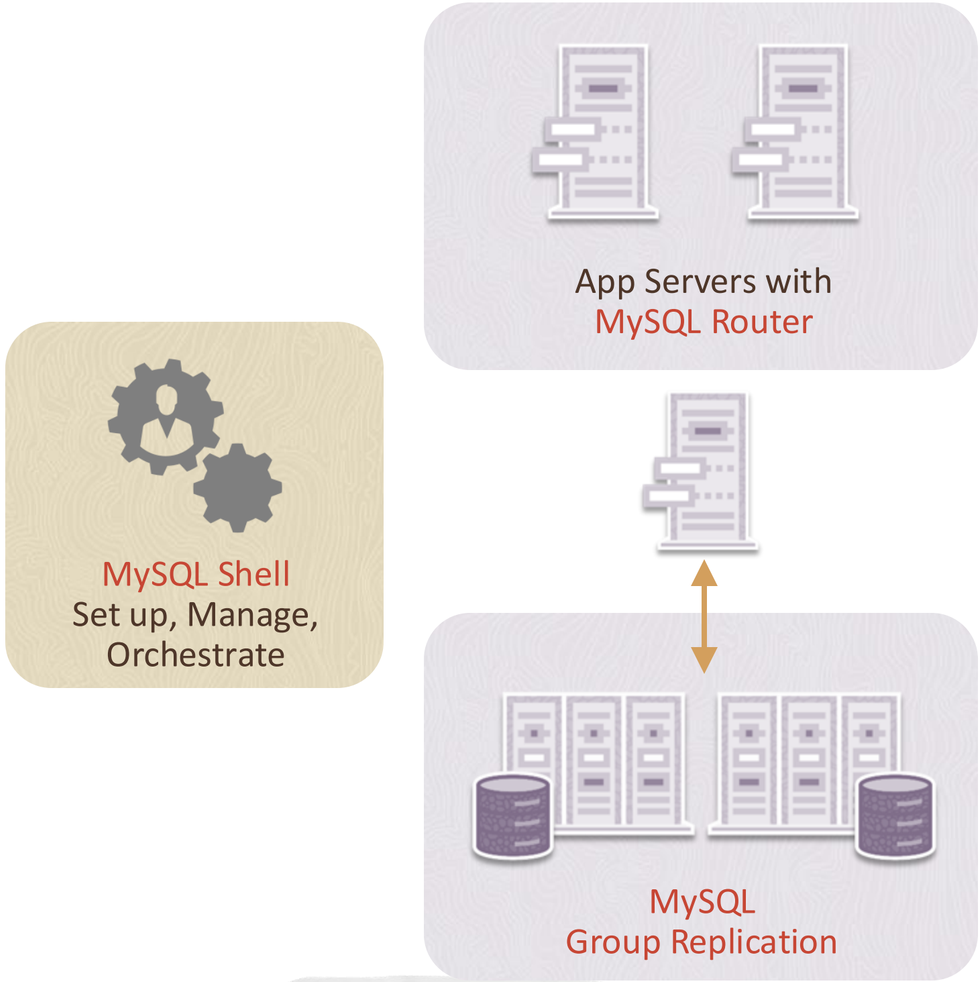
- It offers transparent client connection routing.
  - Load balancing
  - Application connection failover
- Stateless design offers easy HA client routing.
  - A local router becomes part of the application stack (installed in every application server).
  - It can be installed on dedicated servers.
    - But HA has to be provided by third-party SW.



MySQL Shell

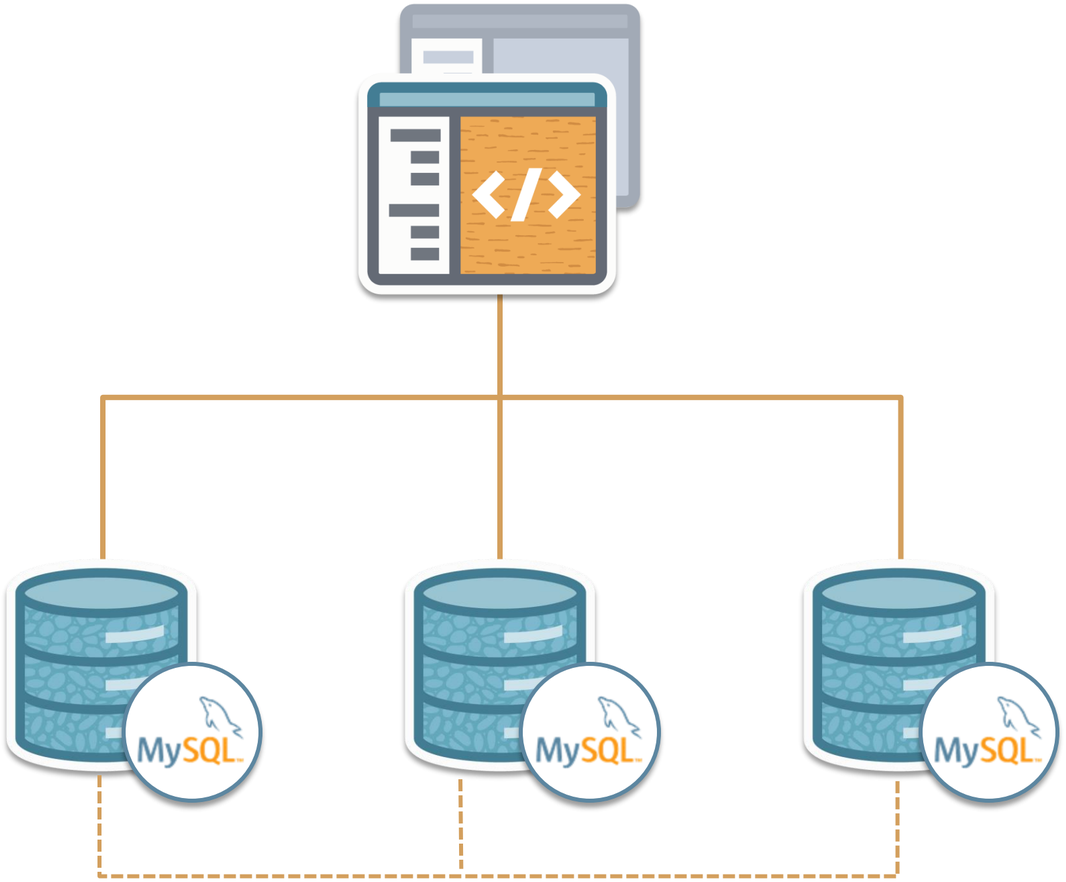
A single unified client for all administrative and operations tasks

- Multi-language: JavaScript, Python, and SQL
  - Naturally scriptable
- Supports both Document and Relational models
- Exposes full Development and Admin API
- Performs DBA operations
- Creates and manages MySQL InnoDB clusters, ReplicaSet and ClusterSet



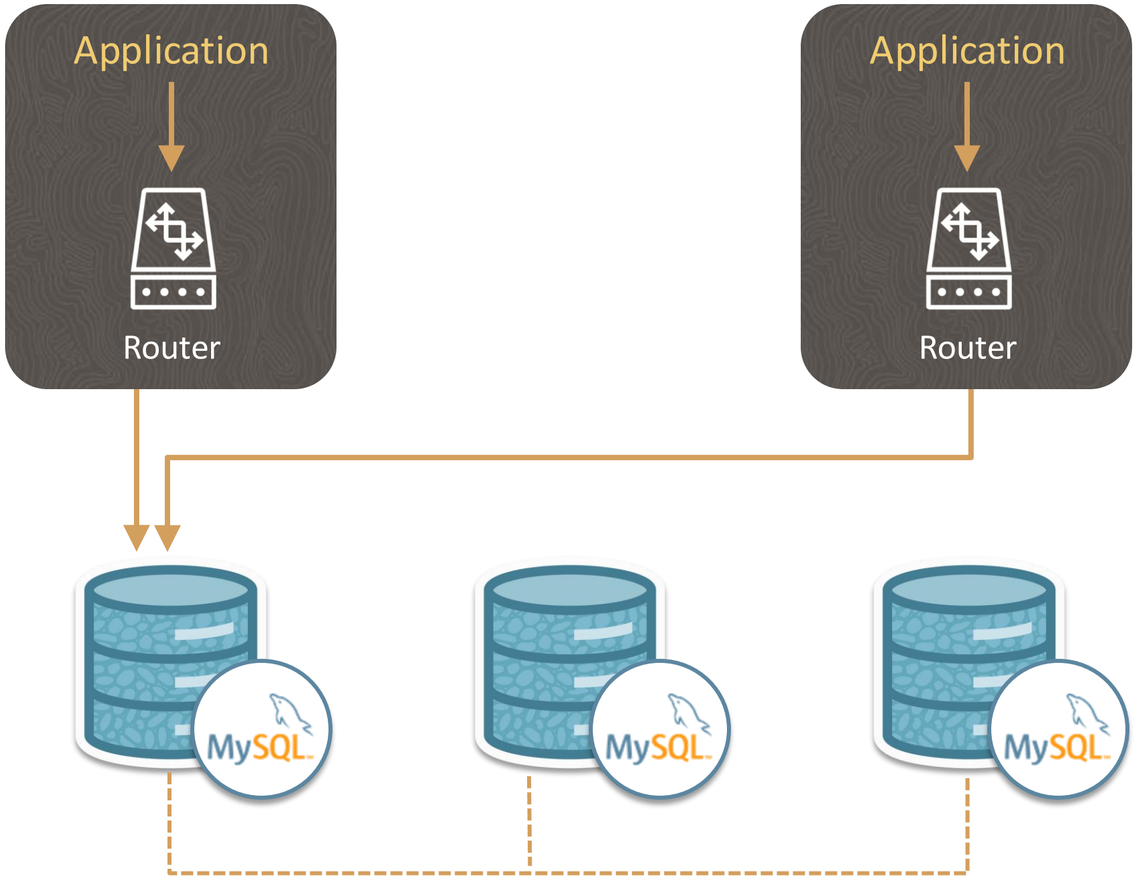
InnoDB Cluster: Setup

- Management of the cluster is easy with MySQL Shell: Creation, status check, configuration changes, and cloning.
- A cluster is set up with a remote connection to three instances.



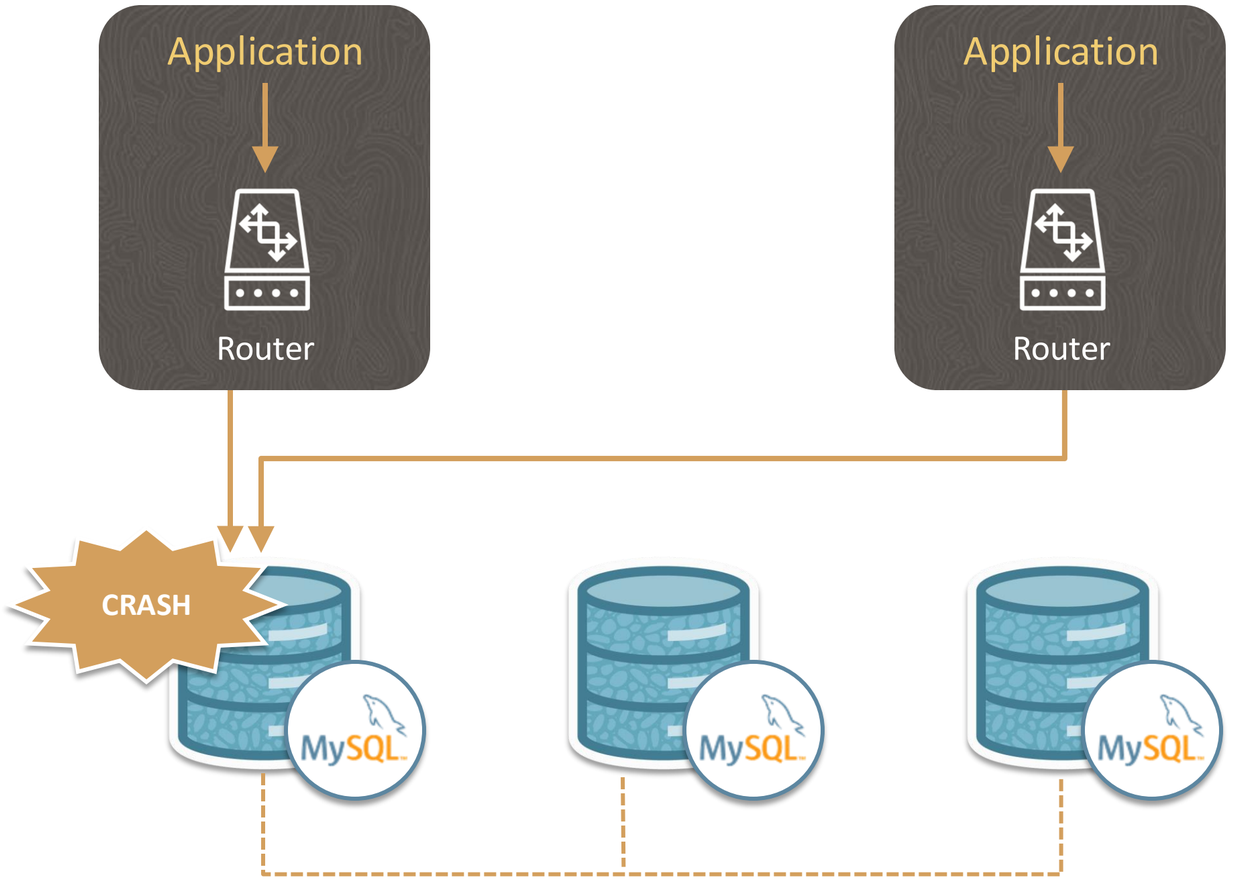
InnoDB Cluster: Application Access

- Start Router with the “bootstrap” option that automatically creates the Router configuration.



InnoDB Cluster: Application Access

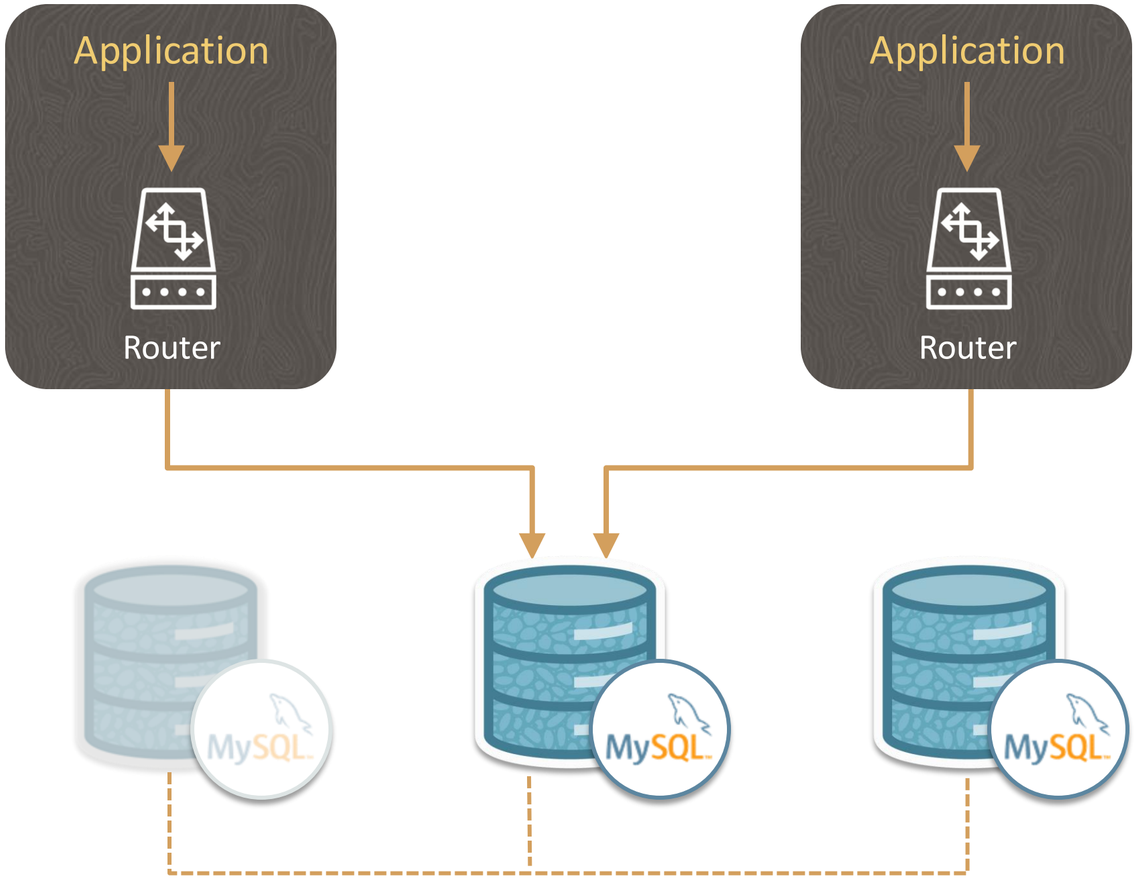
- The application accesses the database through MySQL Router.
- If the primary server fails, MySQL Router performs a *failover*.



https://dev.mysql.com/doc/refman/8.0/en/group-replication-failure-detection.html

InnoDB Cluster: Application Access after Failover

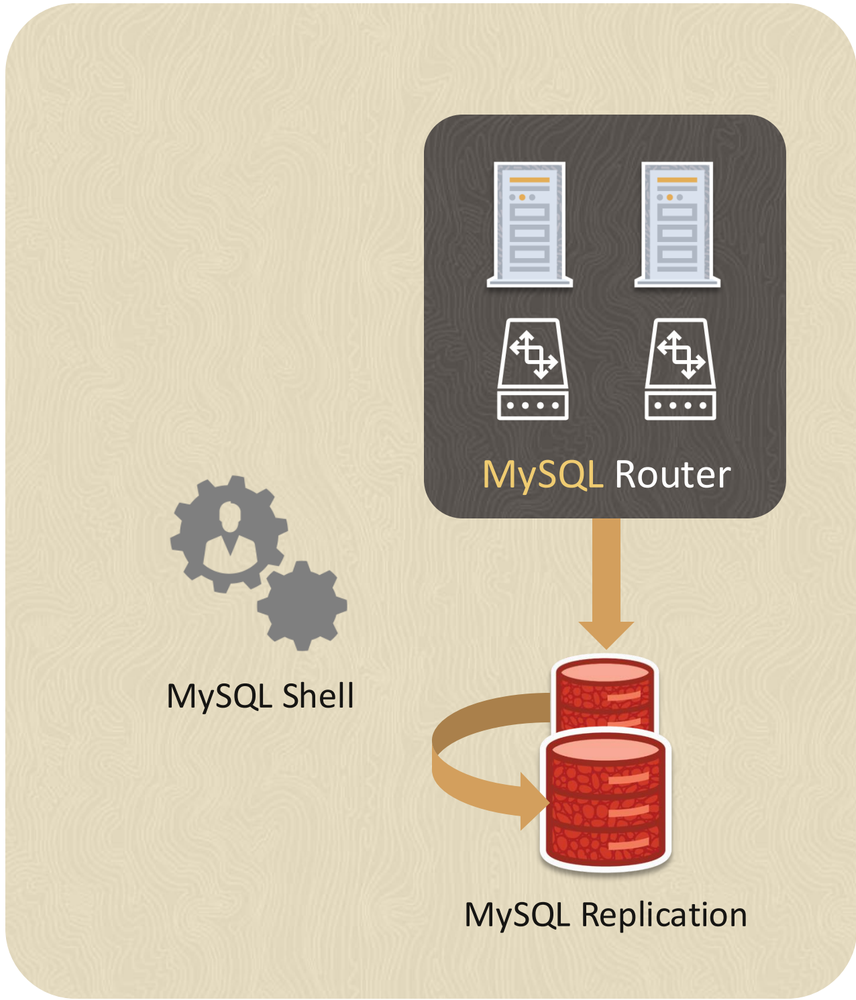
- After failover, MySQL Router redirects application queries to an available node.
- No application configuration changes are required; the failover happens automatically.



Present – Solutions!

MySQL Replicaset

- 'Classic', 'asynchronous' Replication-based solution, fully integrated
- Integrated MySQL Router:
  - Automatic routing
  - (asynchronous) Read scaleout
- Ease of use with MySQL Shell:
  - Configuring, adding, removing members
  - Automatic Member Provisioning (CLONE)
- Switchover and failover (manual)
- Introduced 2020

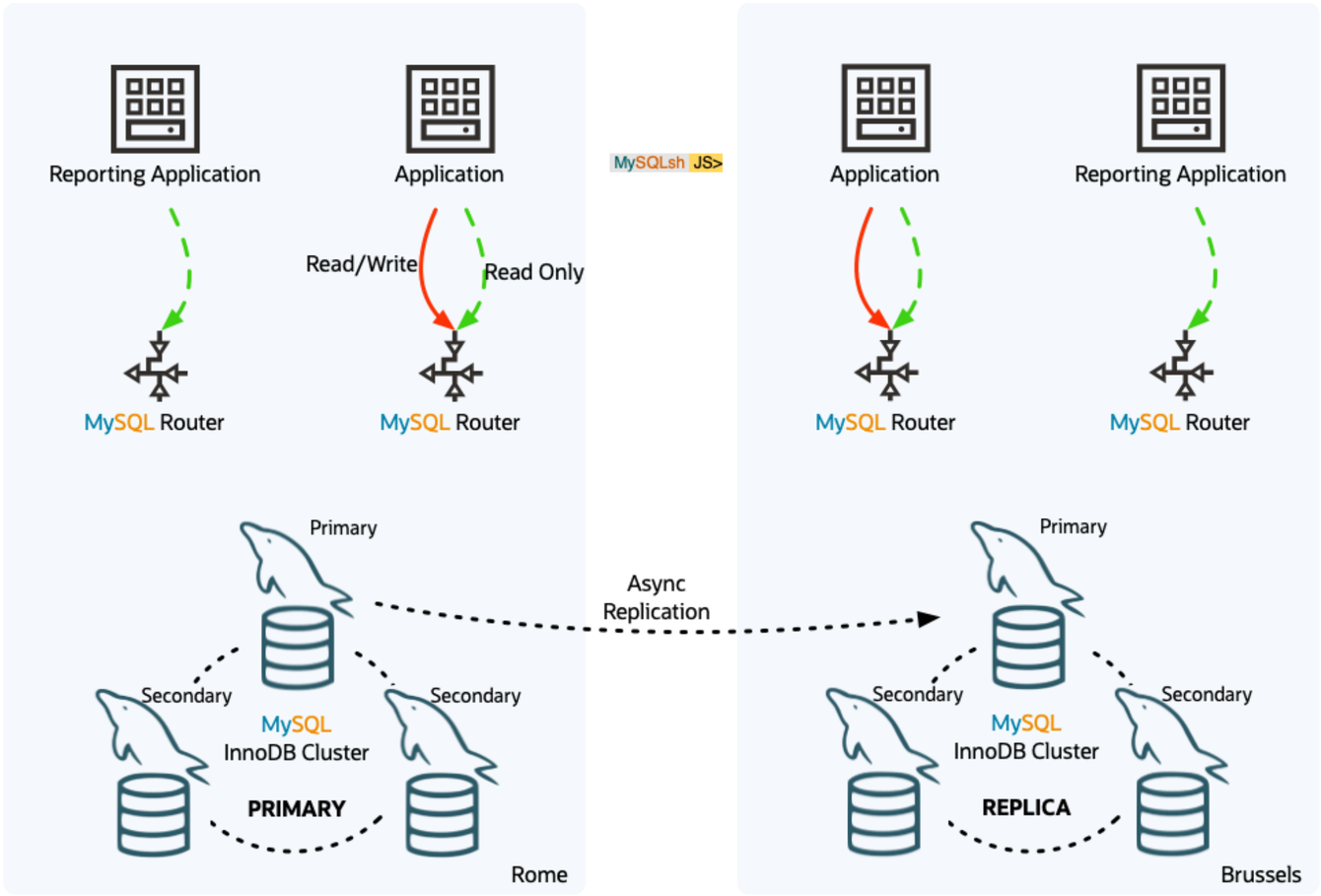


https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-innodb-replicaset.html

MySQL InnoDB ClusterSet

MySQL InnoDB ClusterSet

One or more REPLICA MySQL InnoDB Clusters attached to a PRIMARY MySQL InnoDB Cluster



High Availability (Failure Within a Region)

- RPO = 0
- RTO = seconds (automatic failover)

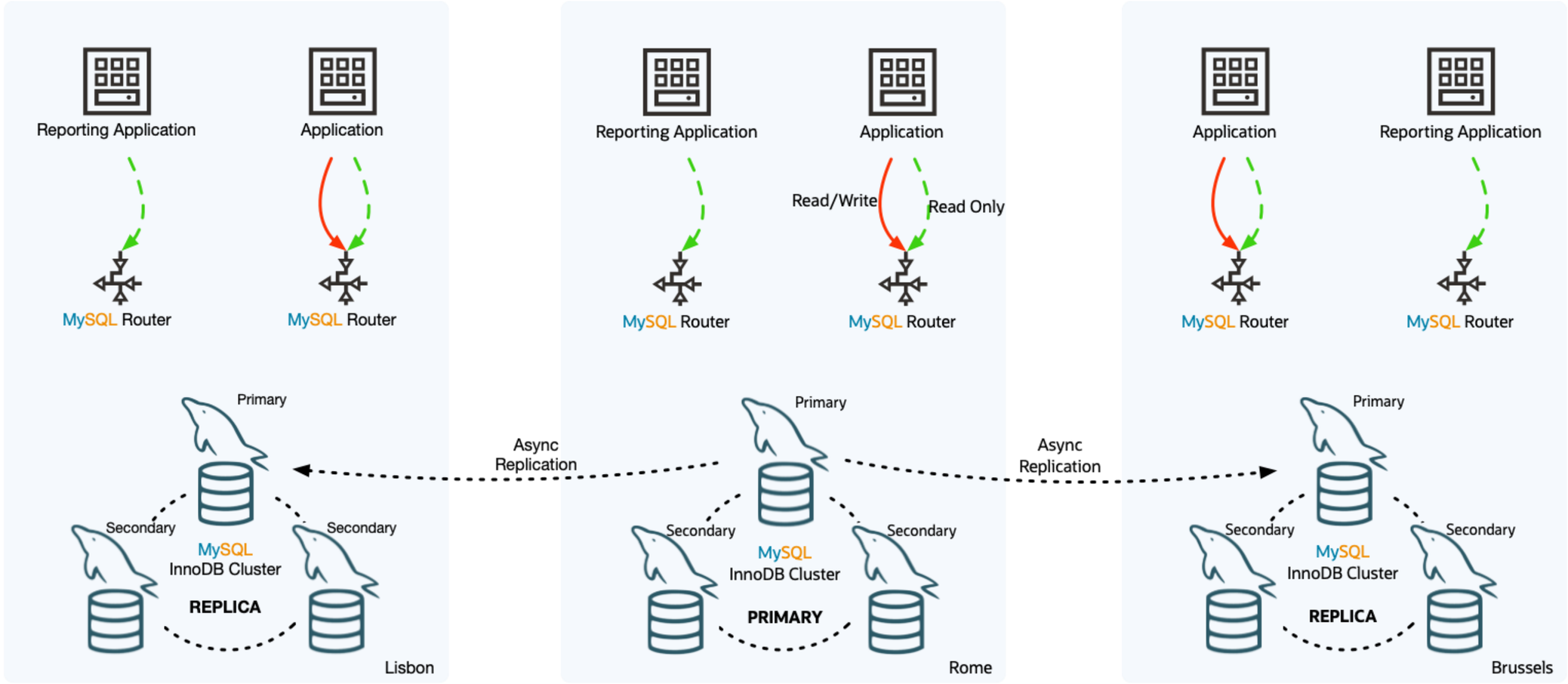
Disaster Recovery (Region Failure)

- RPO != 0
- RTO = minutes or more (manual failover)
- No write performance impact

Features

- Easy to use
- Familiar interface and usability
  `mysqlsh`, `CLONE`, ...
- Add/remove nodes/dusters online
- Router integration, no need to reconigure application if the topology changes

MySQL InnoDB ClusterSet - 3 Datacenters



Business Requirements

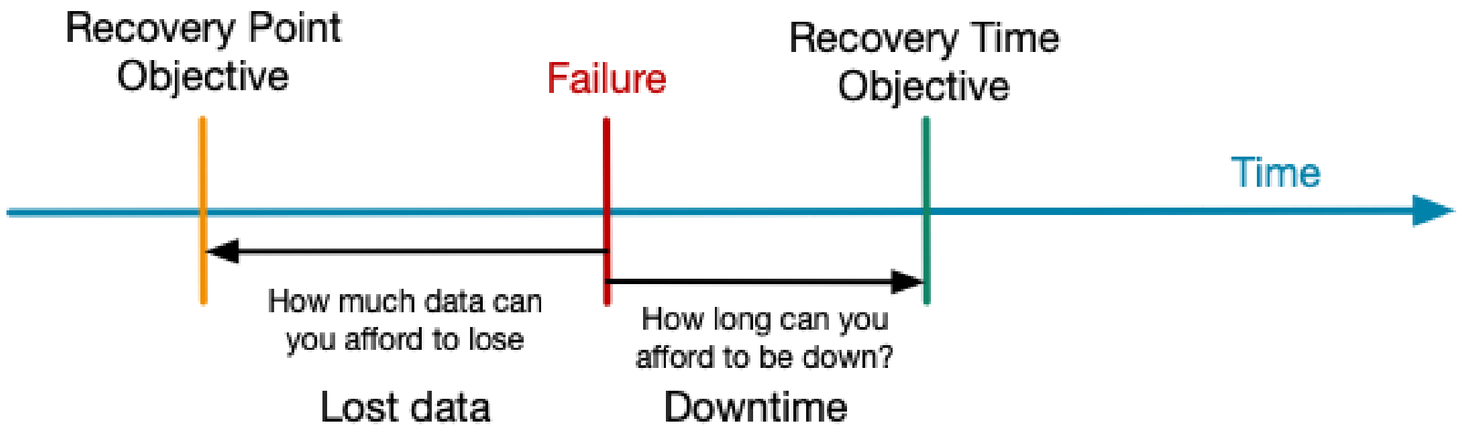
Business Requirements

Concepts - RTO & RPO

- RTO: Recovery Time Objective
  - How long does it take to recover from a single failure
- RPO: Recovery Point Objective
  - How much data can be lost when a failure occurs

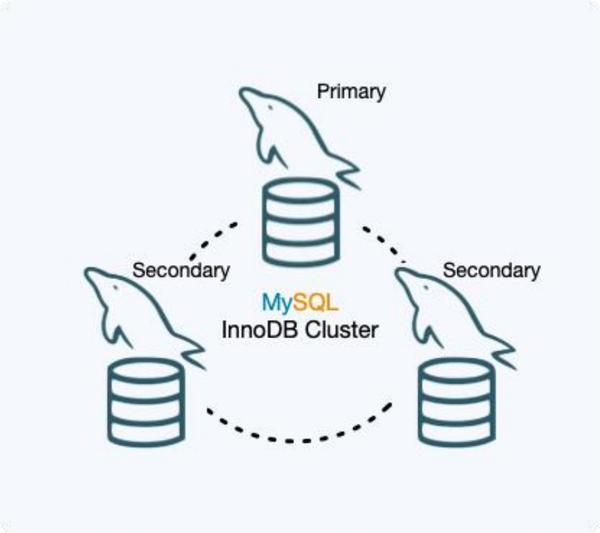
Types of Failure:

High Availability: Single Server Failure, Network Partition
Disaster Recovery: Full Region/Network Failure
Human Error: Little Bobby Tables



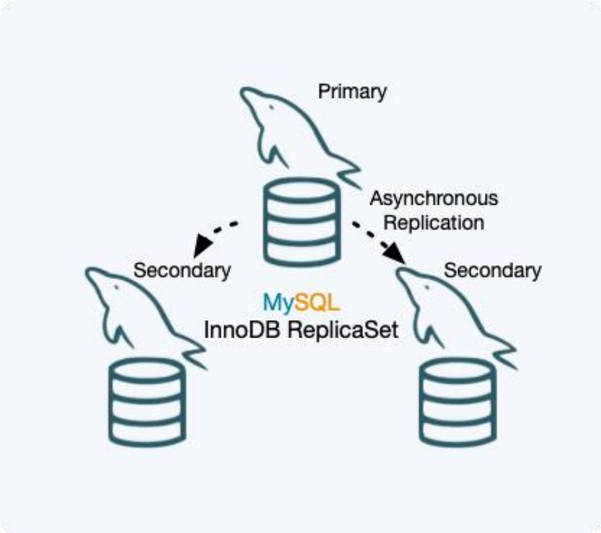
High Availability - Single Region

MySQL InnoDB Cluster



- RPO = 0
- RTO = Seconds

MySQL InnoDB ReplicaSet



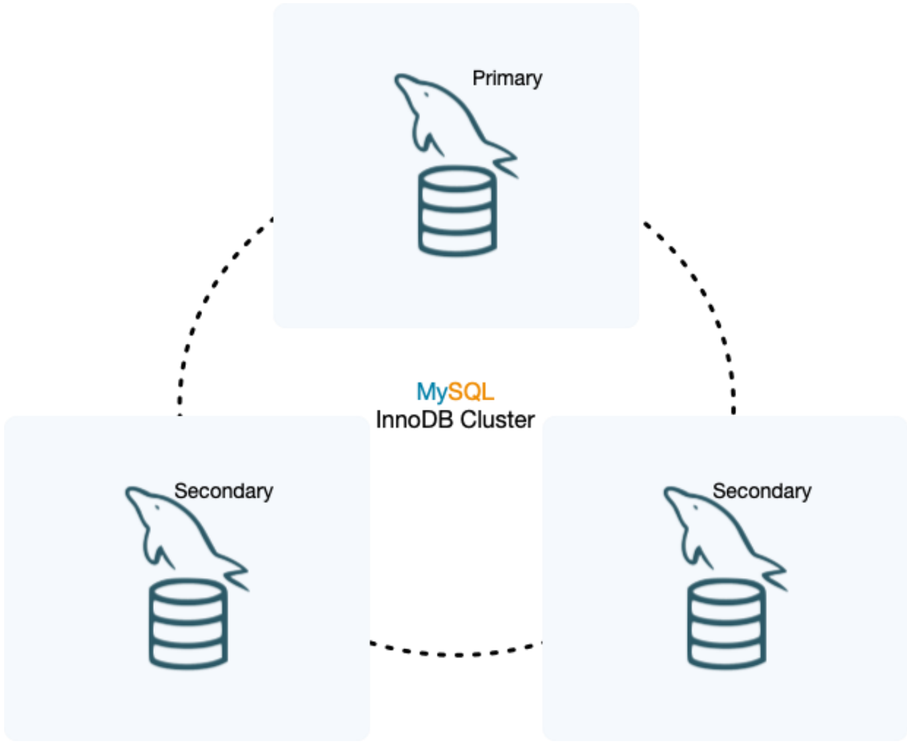
- RPO != 0
- RTO = Minutes+ (manual failover)
- Best write performance
- Manual failover

Disaster Recovery - Multi Region

MySQL InnoDB Cluster

- RPO = 0
- RTO = Seconds

- Multi-Region Multi-Primary
- 3 DC
- Requires very stable WAN
- Write performance affected by latency between dc's



Disaster Recovery - Multi Region

MySQL InnoDB ClusterSet

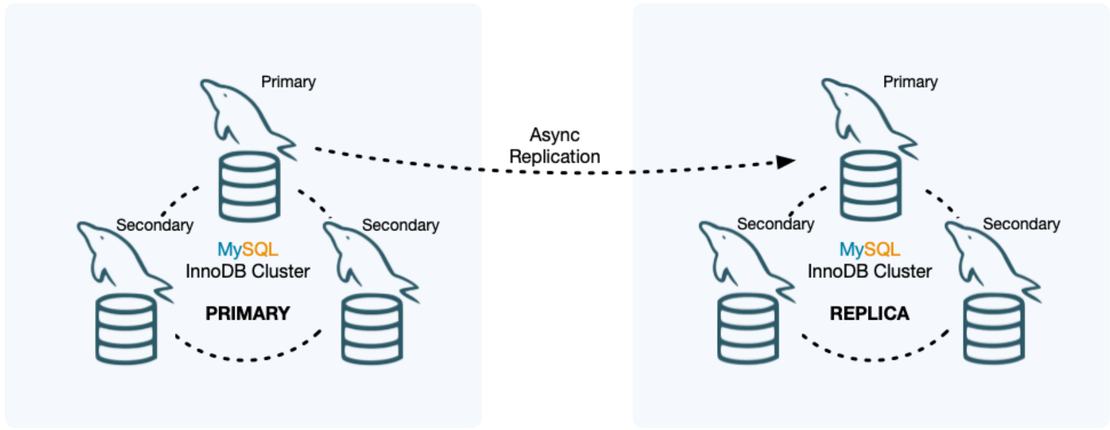
- RPO != 0
- RTO = Minutes+ (manual failover)

- RPO = 0 & RTO = seconds within Region (HA)
- Write performance (no sync to other region required)

Higher RTO: Manual failover

RPO != 0 when region fails

MySQL InnoDB ClusterSet



Workshop Overview

Workshop Overview

- Goals
  1. Create a OCI Compute server for hosting MySQL Enterprise Edition
  2. Install MySQL Enterprise Edition, MySQL Shell and World Database
     1. https://edelivery.oracle.com/
     2. https://dev.mysql.com/doc/world-setup/en/
     3. https://dev.mysql.com/doc/mysql-shell/9.0/en/Overview & Setup
  4. InnoDB ReplicaSets
  5. InnoDB Cluster
  6. InnoDB ClusterSets.

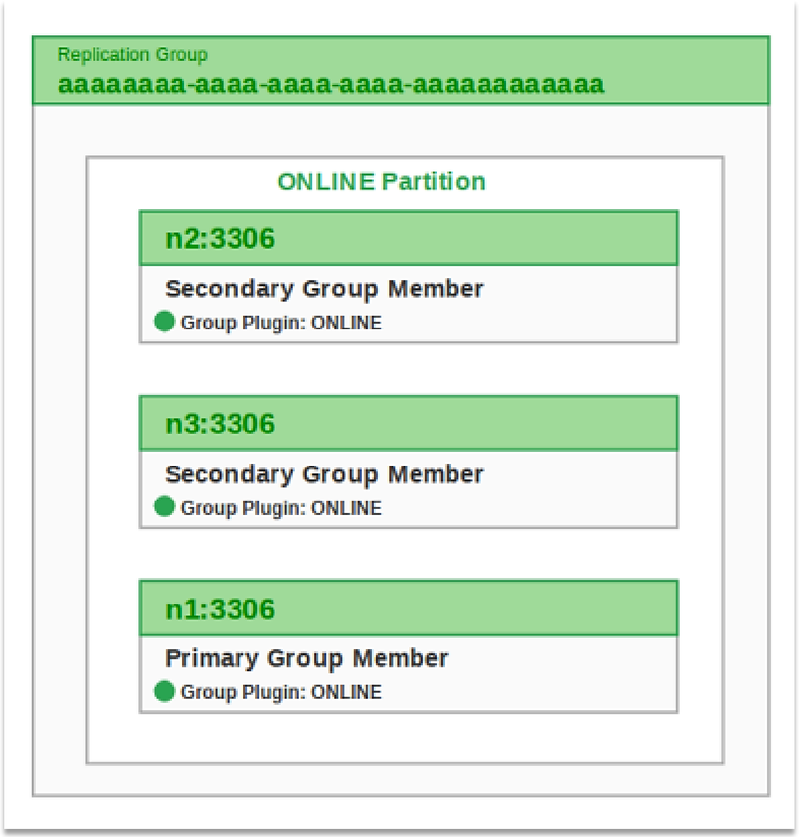
- What this Workshop is not:
  - In-depth tutorial on Oracle Cloud Infrastructure
  - MySQL Training Class
- Lab:
  - https://bit.ly/HA_Workshop

Extra Material

Monitoring

Monitoring Group Replication

- Performance Schema Tables
  - `performance_schema.replication_group_member_stats`
  - `performance_schema.replication_group_members`
  - `performance_schema.replication_connection_status`
  - `performance_schema.replication_applier_status`
- Replication Channels Created:
  - group_replication_recovery - changes to replication
  - group_replication_applier - applies incoming transactions



Group Replication Server States

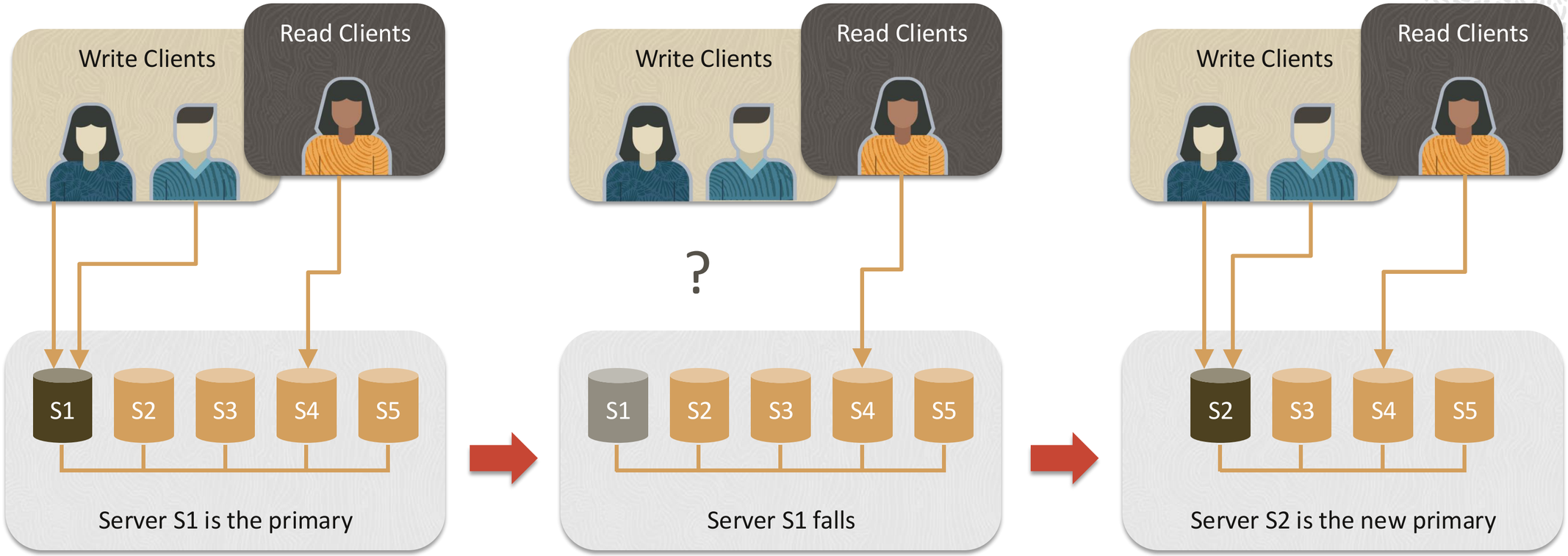
STATUS	DESCRIPTION	GROUP SYNCHRONIZED
ONLINE	The server is an active member of a group and in a fully functioning state.	Y
RECOVERING	The server has joined a group and is in the process of becoming an active member. Distributed recovery is currently taking place using a remote cloning operation or binary log.	N
OFFLINE	The Group Replication plugin is loaded, but the member does not belong to any group.	N
ERROR	The member is in an error state and is not functioning correctly as a group member.	N
UNREACHABLE	The local failure detector suspects that the member cannot be contacted, because the group's messages are timing out.	N

InnoDB Cluster: Shared Data

- MySQL Shell saves metadata for InnoDB Cluster in the database mysql_innodb_cluster_metadata.
  - Metadata are replicated on every node.
- The users used to manage replication are automatically generated.
  - `mysql_innodb_cluster_…`
  - `mysql_router1_...`
  - Use `<cluster>.resetRecoveryAccountsPassword()` to reset the passwords for the internal recovery accounts created by InnoDB Cluster.
- Administrative user for both cluster and router can be created and managed with MySQL Shell:
  - `dba.configureInstance()`, `<cluster>.setupRouterAccount()`, and `<cluster>.setupAdminAccount()`

Single-primary and Multi-Primary Configurations

Single-Primary Mode



https://dev.mysql.com/doc/refman/8.0/en/group-replication-single-primary-mode.html

Multi-Primary Mode