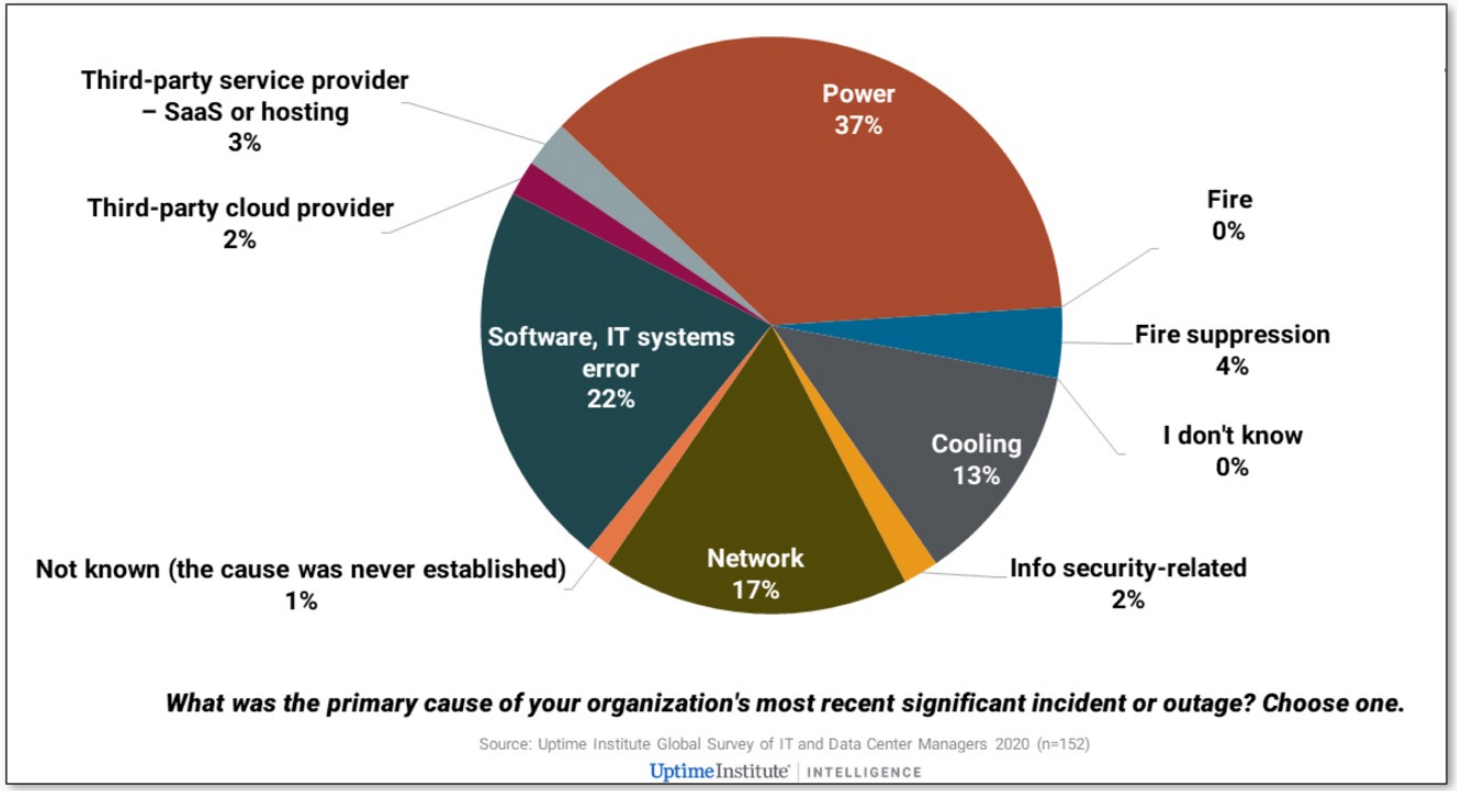
ORACLE DevLive | Level Up | MySQL Summit

MySQL Database High Availability & Disaster Recovery Solutions

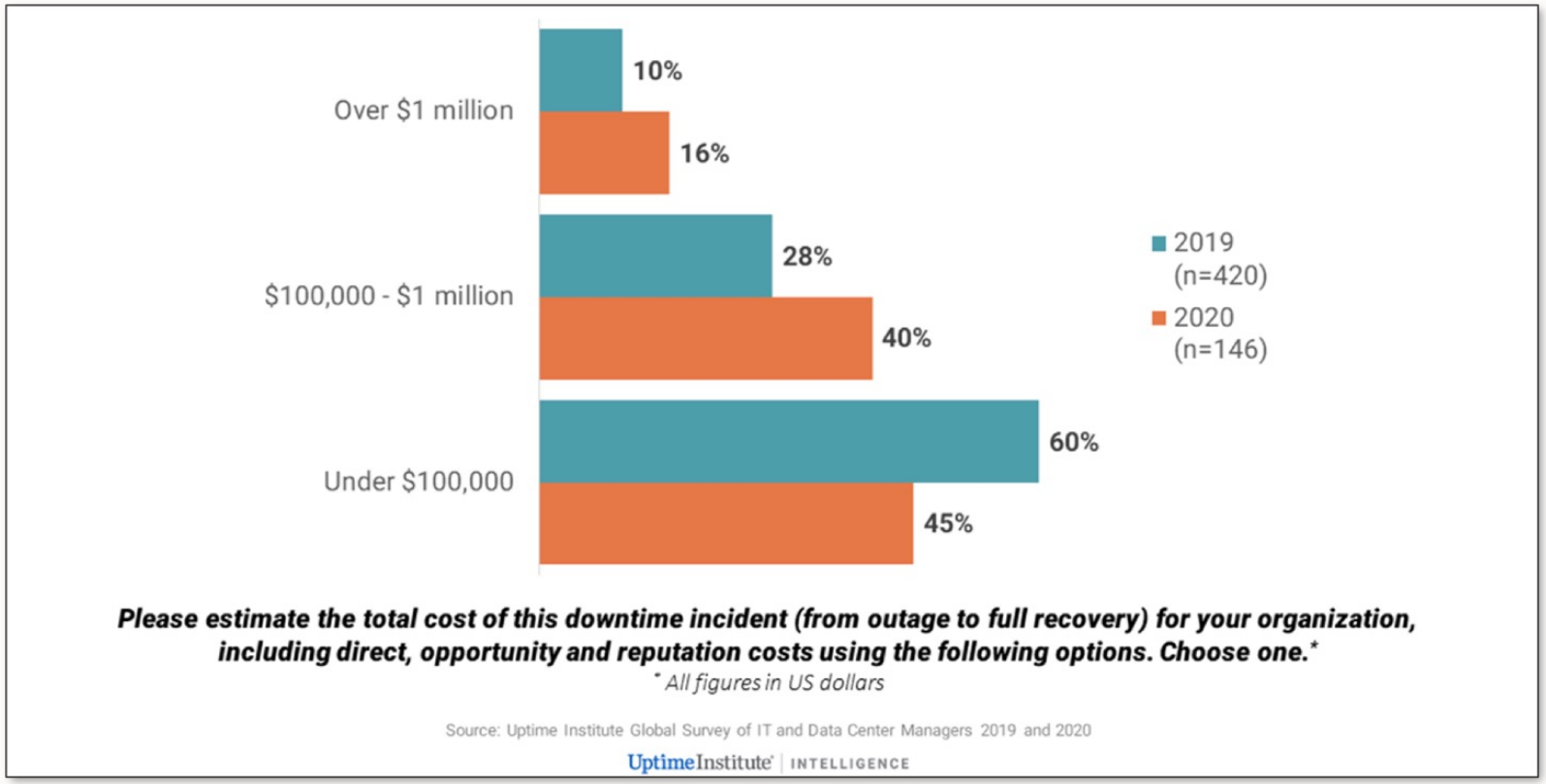
Kenny Gryp
MySQL Product Management Director
March 23, 2023

IT Disasters & Outages: Primary Causes



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

IT Disasters & Outages: Costs are Rising



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

IT Disasters and Outages: Examples

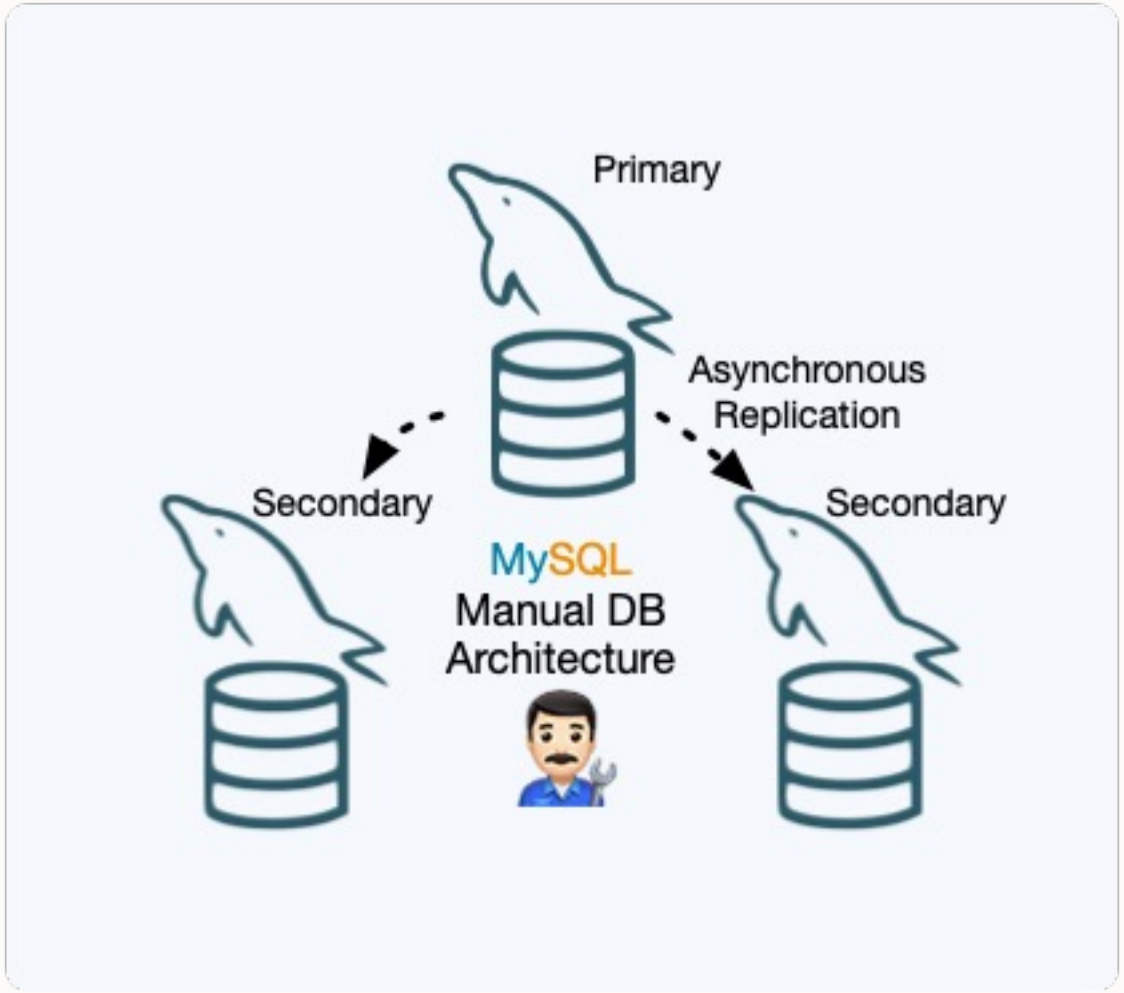
5-Hour computer outage cost $150 million. The airline eventually cancelled about 1,000 flights on the day of the outage and grounded an additional 1,000 flights over the following days.

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

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

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

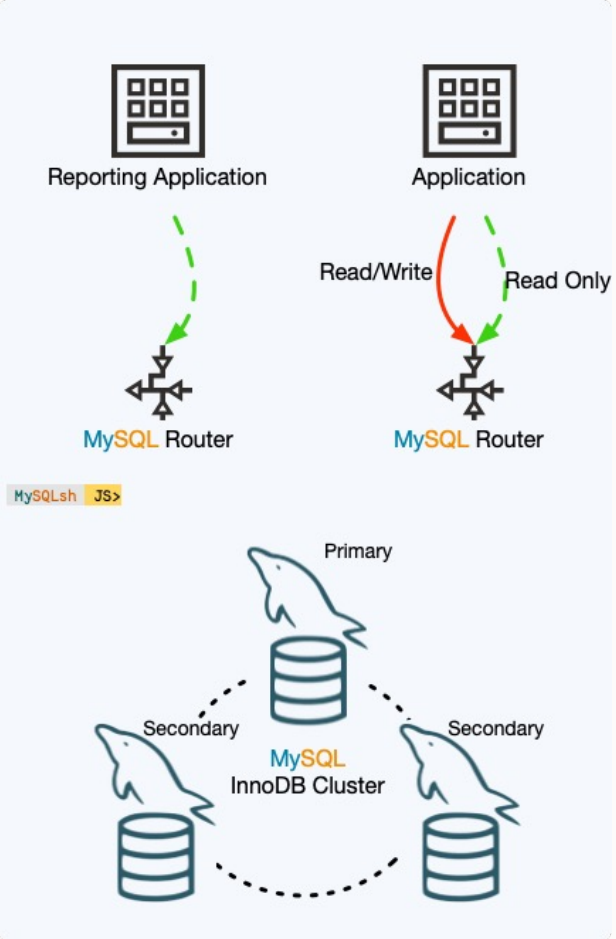
Past - Manual



- 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

MySQL InnoDB Cluster

Present – Solutions!

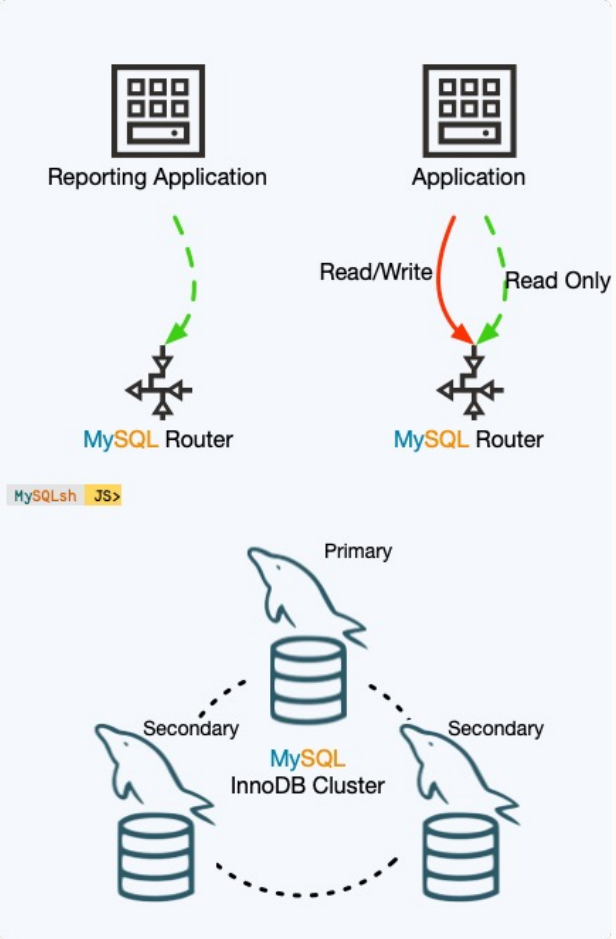


RPO = 0
RTO = seconds (automatic failover)

2016 MySQL InnoDB Cluster

- MySQL Group Replication:
  - Automatic membership changes,
  - Network partition handling,
  - Automatic failover
  - Consistency, no split brain…
- MySQL Shell: a powerful interface that helps in automating and integrating all components
- InnoDB `CLONE` to automatically provision members, fully integrated in InnoDB
- MySQL Router
- MySQL Server

Present – Solutions!

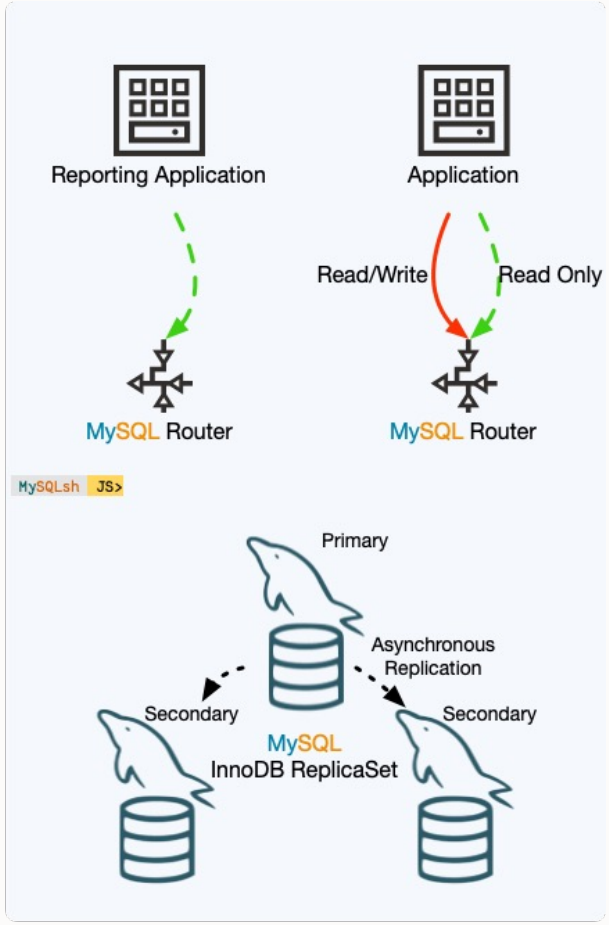


RPO = 0
RTO = seconds (automatic failover)

2016 MySQL InnoDB Cluster

- Replicated Database State Machine
  - Total Order - Writes
  - XCOM - Paxos implementation
- Configurable Consistency Guarantees
  - Eventual consistency
  - 8.0+: per session & global read/write consistency
- Using MySQL replication framework by design: binary logs, relay logs, GTIDs: Global Transaction IDs
- Generally Available since MySQL 5.7
- Supported on all platforms

Present – Solutions!



RPO != 0
RTO = minutes (manual failover)

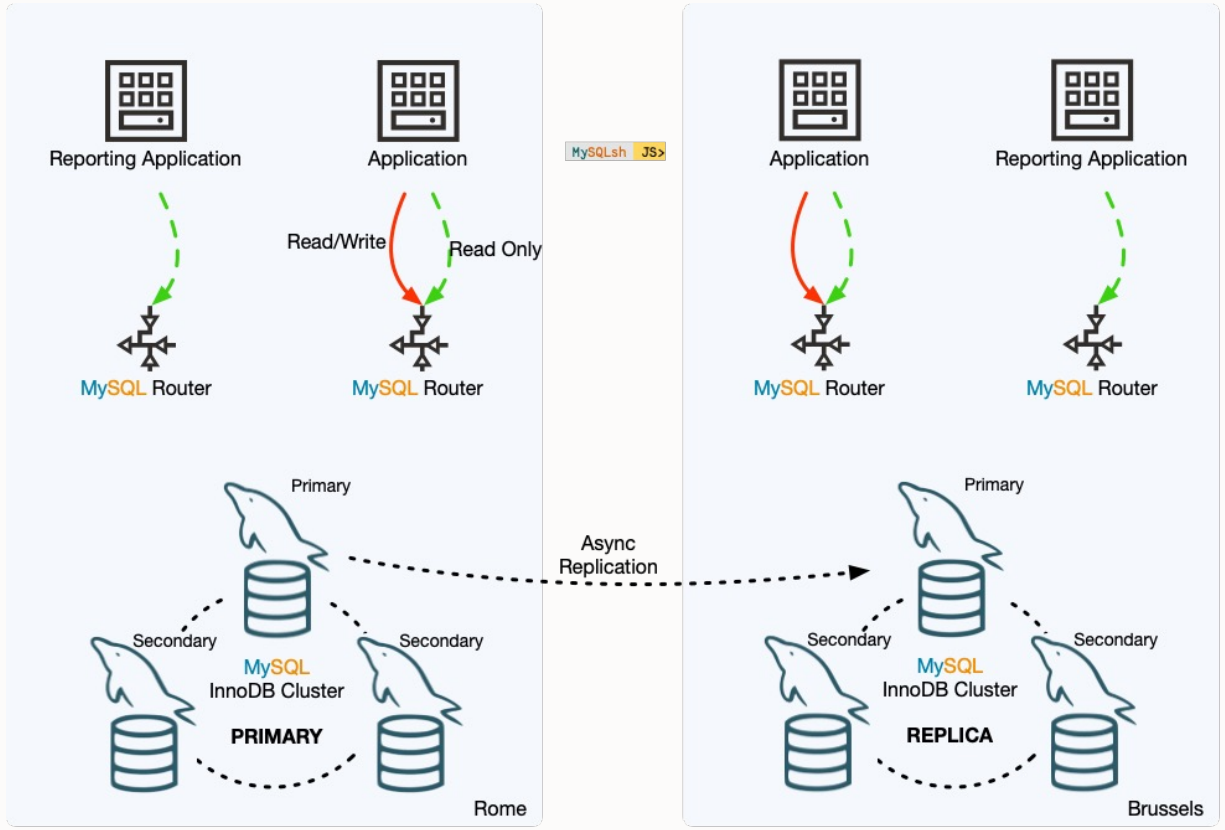
2020 MySQL InnoDB Replicaset

- 'classic', 'asynchronous' Replication based Solution, fully integrated
- MySQL Shell
- MySQL Router
- MySQL Server

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)

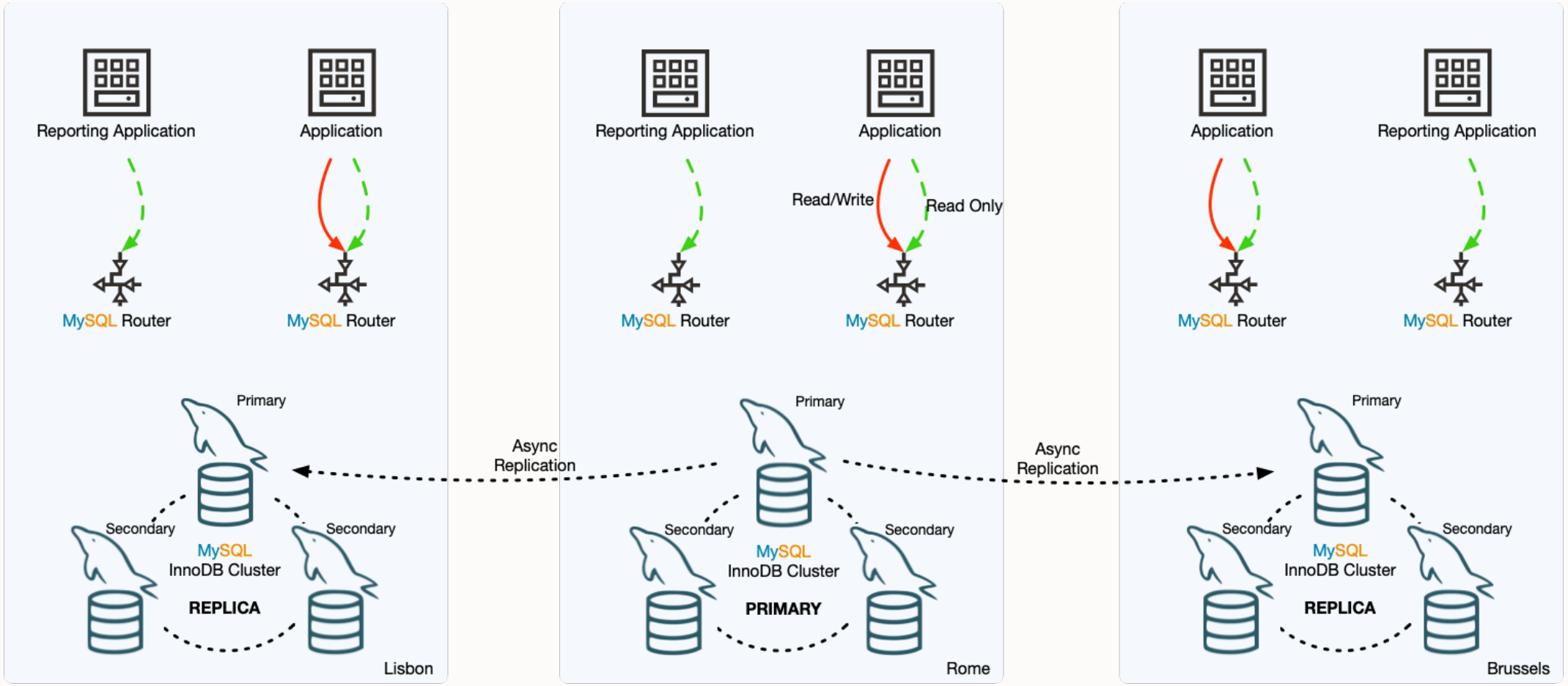
- RPO !=0
- RTO = mintues or more (manual failover)
- No write performance impact

Features

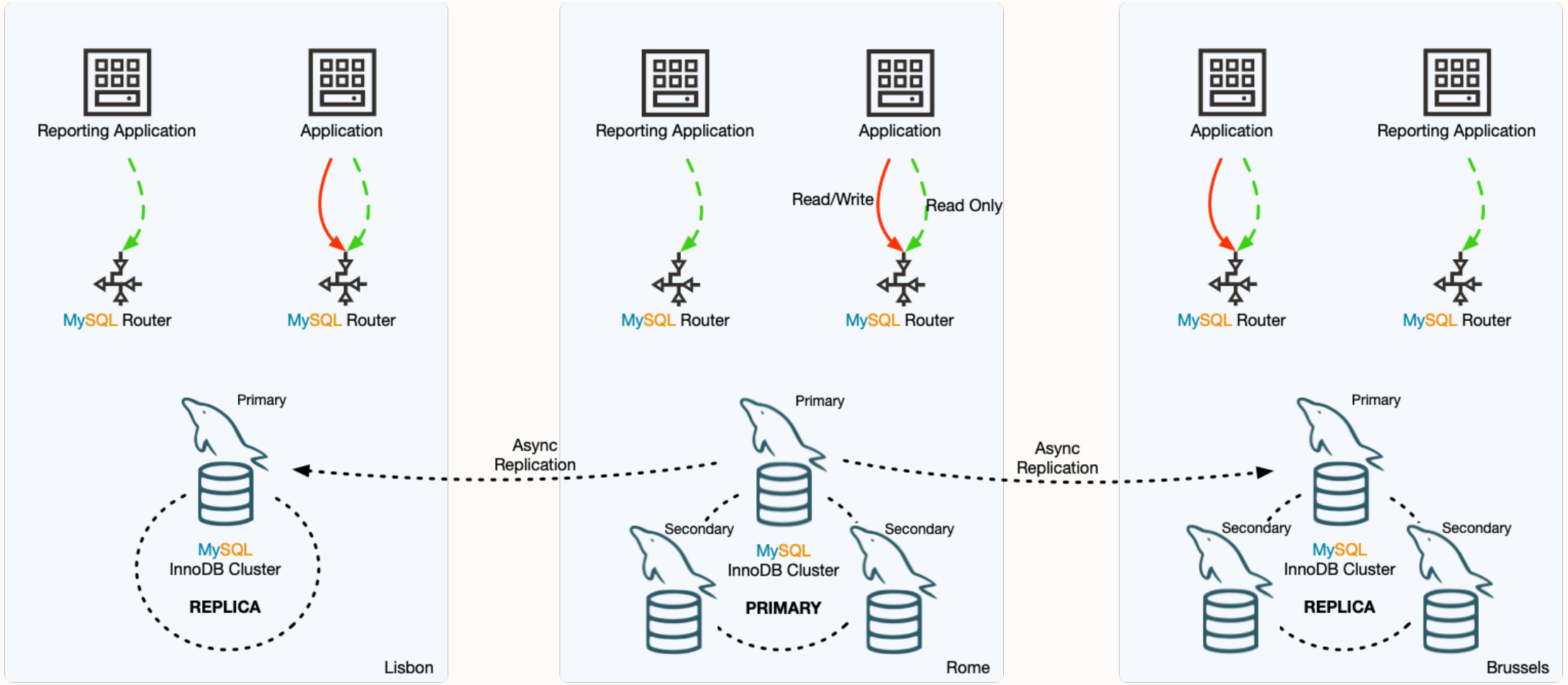
- Easy to use
- Familiar interface and usability

  `mysqlsh, CLONE,…`
- Add/remove nodes/clusters online
- Router integration, no need to reconfigure application if the topology changes

MySQL InnoDB ClusterSet – 3 Data Centers



MySQL InnoDB ClusterSet – Not every Cluster has to be 3 nodes



Create MySQL InnoDB Cluster

Start with setting up a regular MySQL InnoDB Cluster:

```
mysqlsh> \c root@localhost:3331
mysqlsh> \sql create schema sbtest;
mysqlsh> bru = dba.createCluster("BRU")

mysqlsh> bru.addInstance('localhost:3332')
mysqlsh> bru.addInstance('localhost:3333')
mysqlsh> bru.status()
```


Create ClusterSet

```
mysqlsh> clusterset = bru.createClusterSet('clusterset')

A new ClusterSet will be created based on the Cluster 'BRU'.
- Validating Cluster 'BRU' for ClusterSet compliance.
- Creating InnoDB ClusterSet 'clusterset' on 'BRU'...
- Updating metadata...

ClusterSet successfully created. Use ClusterSet.createReplicaCluster() to add
Replica Clusters to it.
<ClusterSet:cluster>
```


Check ClusterSet Status

```
mysqlsh> clusterset.status()

{
    "clusters": {
        "BRU": {
            "clusterRole": "PRIMARY",
            "globalStatus": "OK",
            "primary": "127.0.0.1:3331"
} },
    "domainName": "clusterset",
    "globalPrimaryInstance": "127.0.0.1:3331",
    "primaryCluster": "BRU",
    "status": "HEALTHY",
    "statusText": "All Clusters available."
}
```


Add Replica Cluster

- Supports incremental recovery (binlog) & full recovery (**CLONE**)

```
mysqlsh> lis = clusterset.createReplicaCluster('localhost:4441', 'LIS')
mysqlsh> lis.addInstance('localhost:4442')
mysqlsh> lis.addInstance('localhost:4443')
```


Check ClusterSet Status

```
mysqlsh> clusterset.status()
mysqlsh> bru.status()
mysqlsh> lis.status()
```

```
{
    "clusters": {
        "BRU": {
            "clusterRole": "PRIMARY",
            "globalStatus": "OK",
            "primary": "127.0.0.1:3331"
}, "LIS": {
            "clusterRole": "REPLICA",
            "clusterSetReplicationStatus": "OK",
            "globalStatus": "OK"
} },
    "domainName": "clusterset",
    "globalPrimaryInstance": "127.0.0.1:3331",
    "primaryCluster": "BRU",
    "status": "HEALTHY",
    "statusText": "All Clusters available."
```

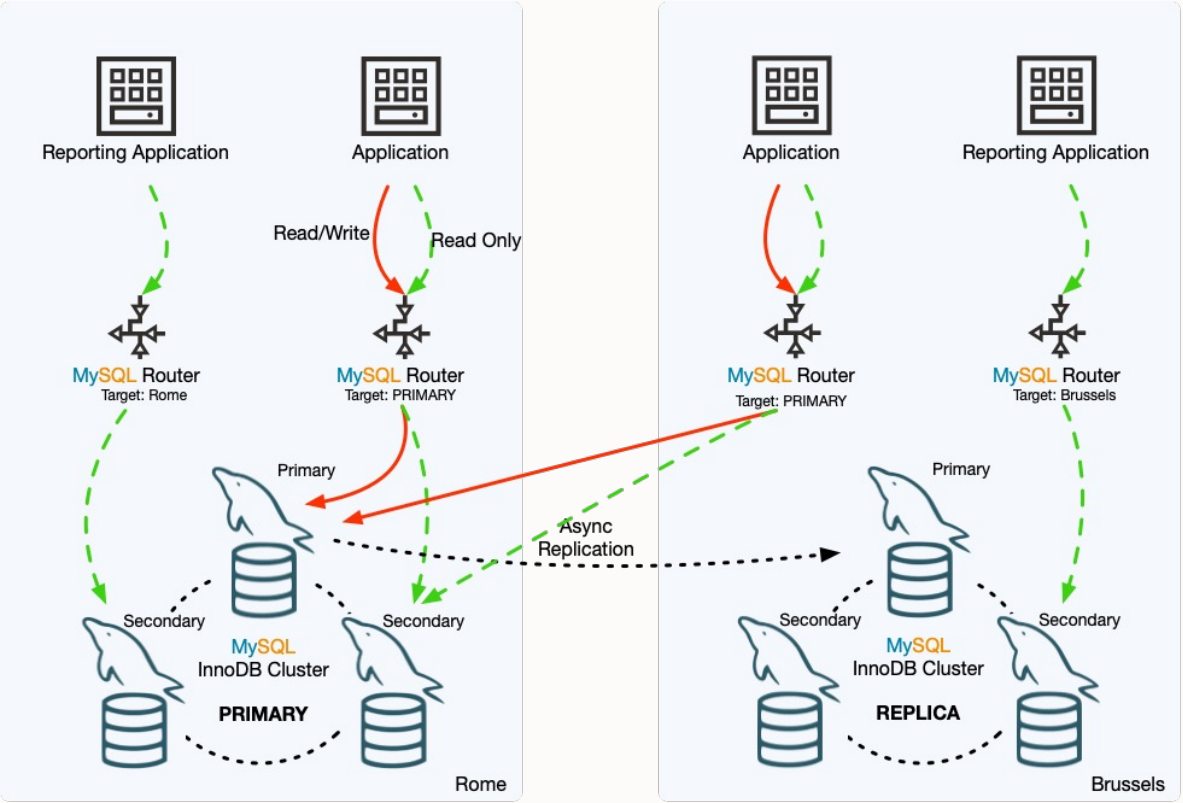
Or, to get everything in one command:

```
mysqlsh> clusterset.status({extended:1})
```


Change PRIMARY member in PRIMARY cluster

```
mysqlsh> bru.setPrimaryInstance('localhost:3332')

Setting instance 'localhost:3332' as the primary instance of cluster 'BRU'...
Instance '127.0.0.1:3331' was switched from PRIMARY to SECONDARY.
Instance '127.0.0.1:3332' was switched from SECONDARY to PRIMARY.
Instance '127.0.0.1:3333' remains SECONDARY.

WARNING: The cluster internal session is not the primary member anymore. For
cluster management operations
please obtain a fresh cluster handle using dba.getCluster().

The instance 'localhost:3332' was successfully elected as primary.
```

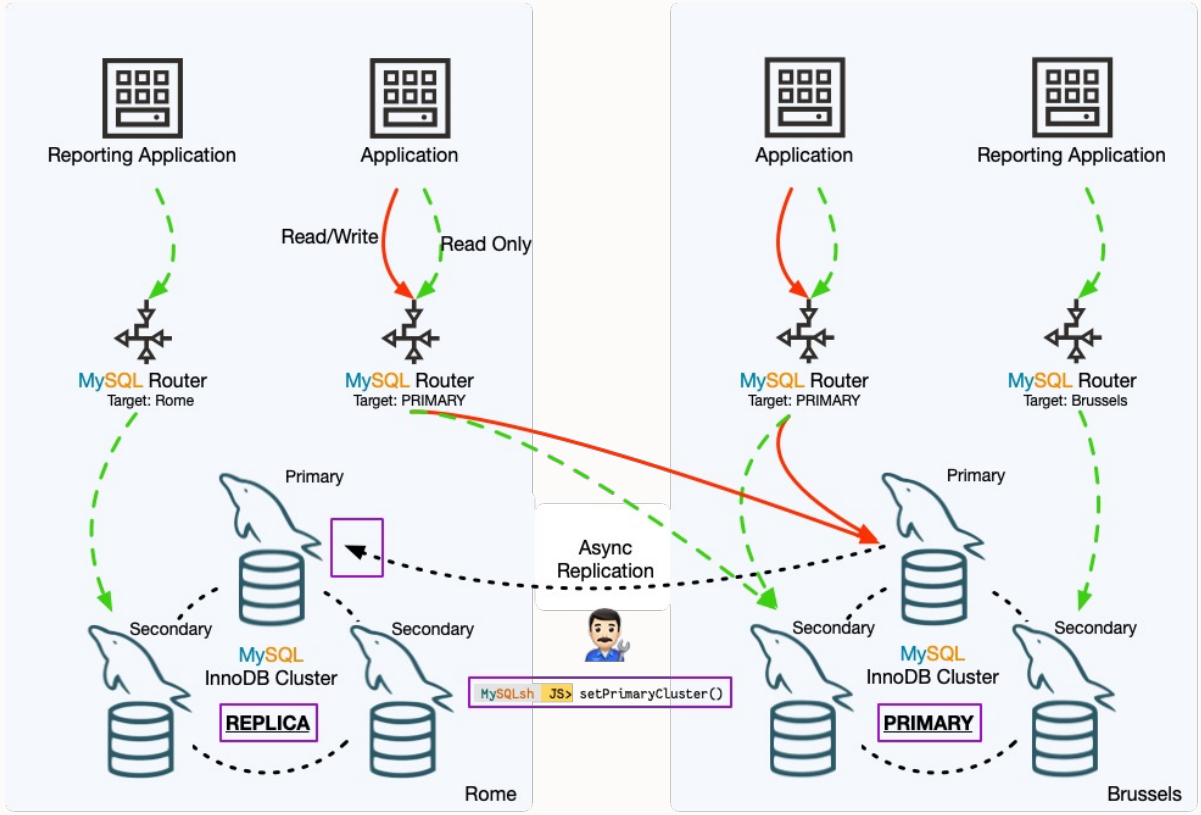

Changing Primary - Change Primary Cluster on Healthy System



Switchover

- one command that does it all: `setPrimaryCluster()`
- Asynchronous replication channels between clusters are automatically reconfigured
- Consistency guaranteed
- All routers will immediately redirect if needed (depending on target mode)

Changing Primary - Change Primary Cluster on Healthy System



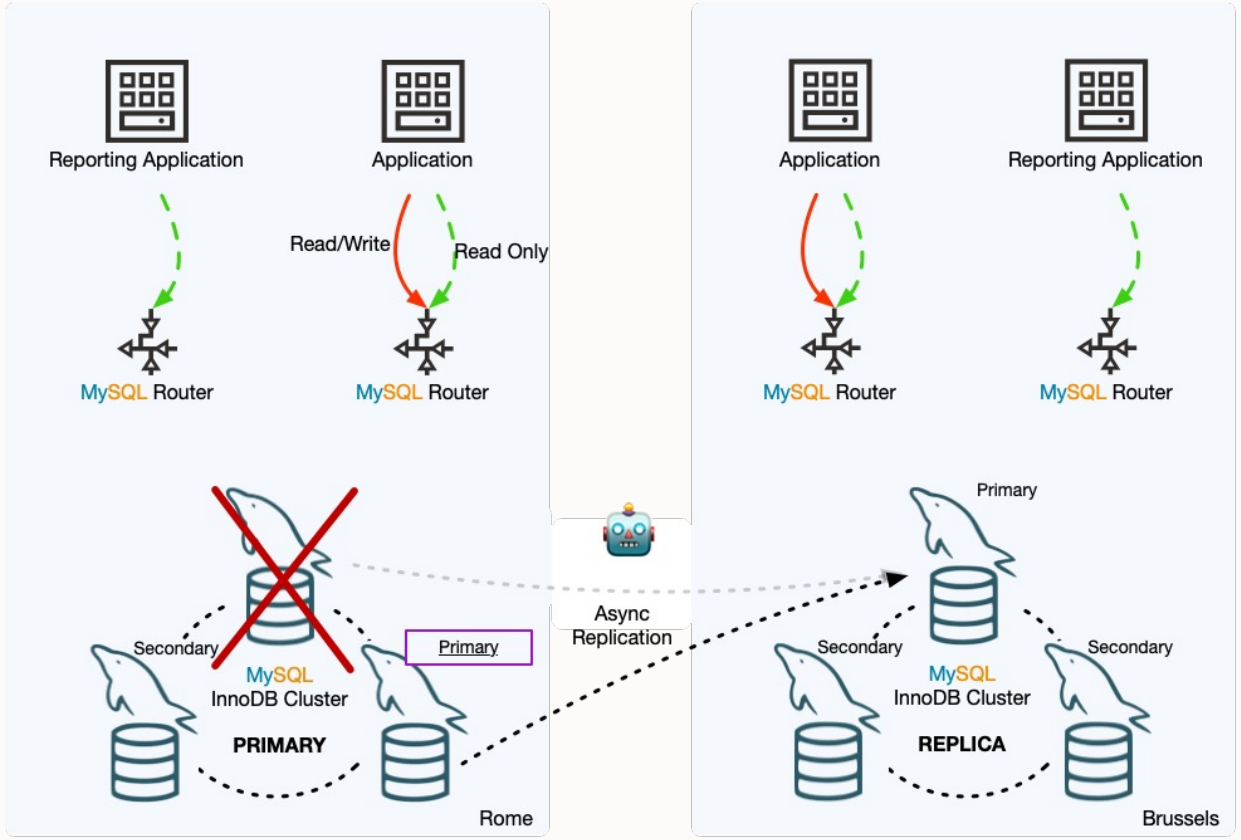
Switchover

- one command that does it all: `setPrimaryCluster()`
- Asynchronous replication channels between clusters are automatically reconfigured
- Consistency guaranteed
- All routers will immediately redirect if needed (depending on target mode)

Switchover - Changing `PRIMARY` Cluster - `setPrimaryCluster()`

```
mysqlsh> clusterset.setPrimaryCluster('LIS')

Switching the primary cluster of the clusterset to 'LIS'
- Verifying clusterset status
-- Checking cluster BRU -  Cluster 'BRU' is available
-- Checking cluster ROM -  Cluster 'ROM' is available
-- Checking cluster LIS -  Cluster 'LIS' is available

- Refreshing replication account of demoted cluster
- Synchronizing transaction backlog at 127.0.0.1:4442

- Updating metadata

- Updating topology
-- Changing replication source of 127.0.0.1:3331 to 127.0.0.1:4442
-- Changing replication source of 127.0.0.1:3333 to 127.0.0.1:4442
-- Changing replication source of 127.0.0.1:3332 to 127.0.0.1:4442
- Acquiring locks in replicaset instances
-- Pre-synchronizing SECONDARIES
-- Acquiring global lock at PRIMARY & SECONDARIES

- Synchronizing remaining transactions at promoted primary

- Updating replica clusters
-- Changing replication source of 127.0.0.1:5552 to 127.0.0.1:4442
-- Changing replication source of 127.0.0.1:5553 to 127.0.0.1:4442
-- Changing replication source of 127.0.0.1:5551 to 127.0.0.1:4442
Cluster 'LIS' was promoted to PRIMARY of the clusterset. The PRIMARY instance is '127.0.0.1:4442'
```


PRIMARY Cluster PRIMARY member Crash/Partition - Automatic

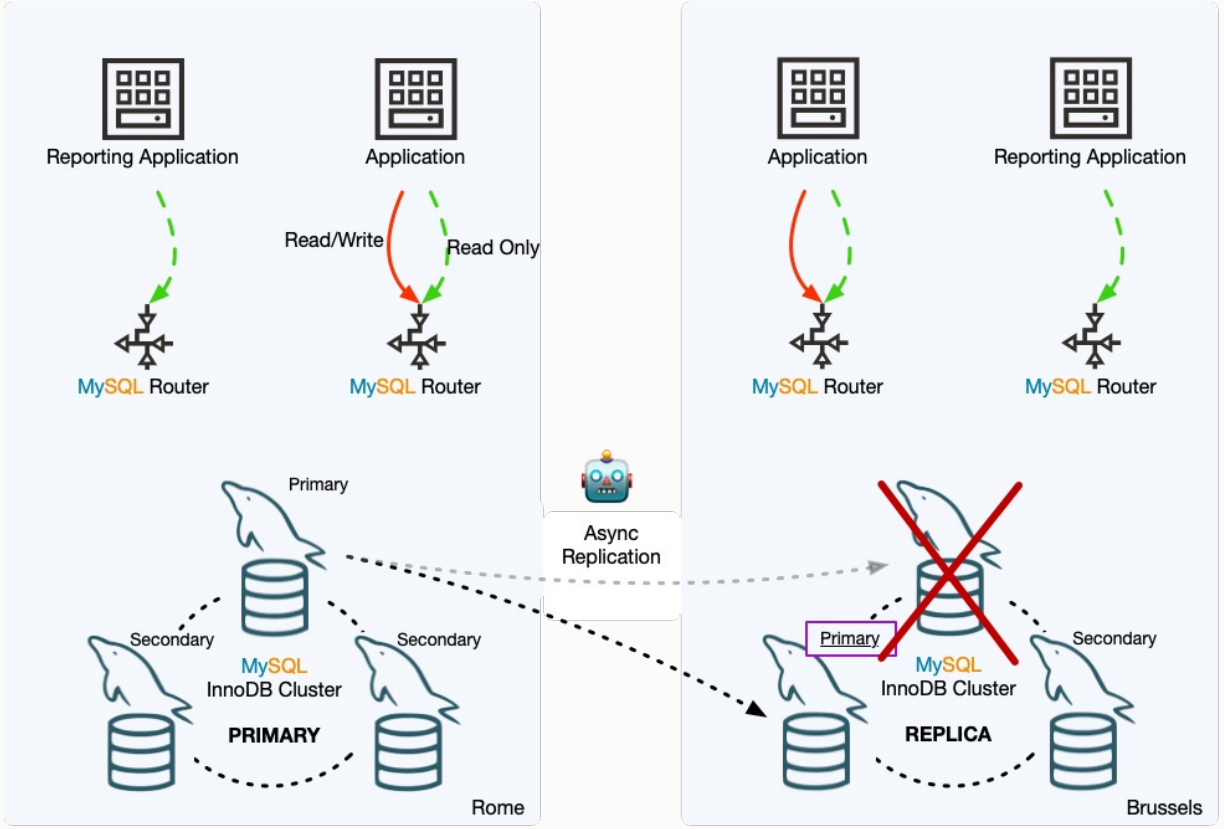


- When there is newly elected PRIMARY member in a cluster
- Works on failures in PRIMARY and REPLICA clusters

Automatic Handling of InnoDB Cluster state changes

- Asynchronous replication is automatically reconfigured after primary change

`REPLICA` Cluster `PRIMARY` member Crash/Partition - Automatic

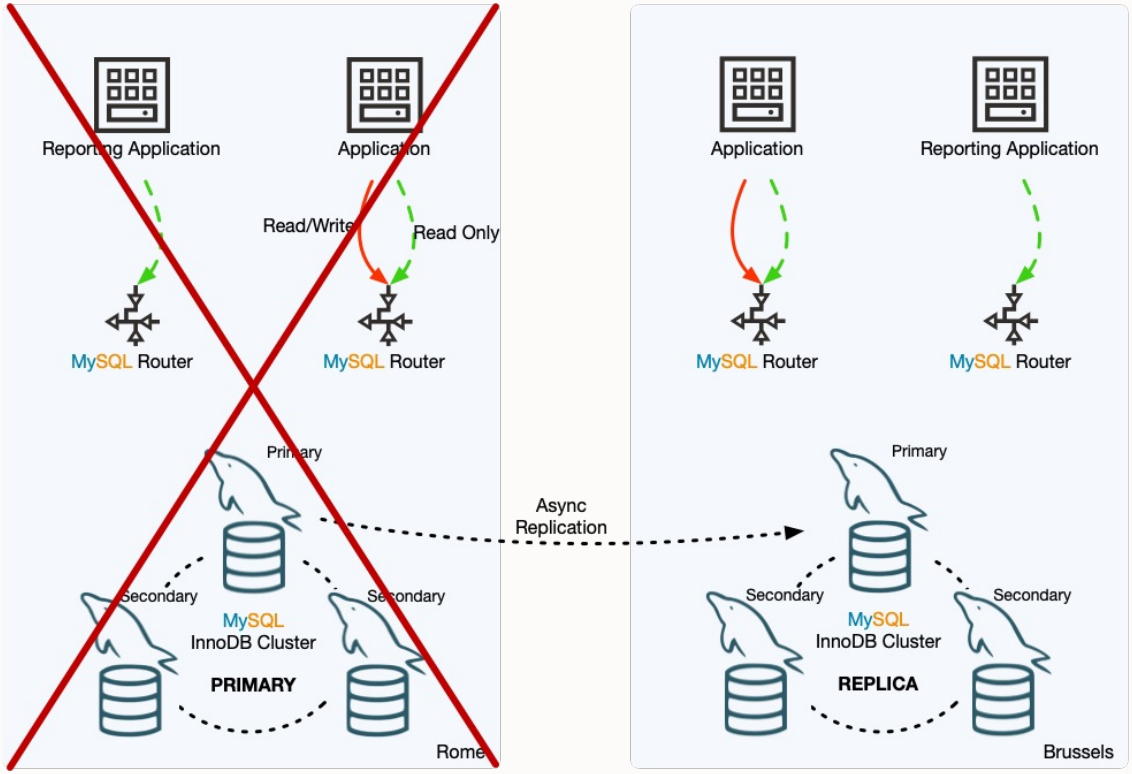


- When there is newly elected `PRIMARY` member in a cluster
- Works on failures in `PRIMARY` and `REPLICA` clusters

Automatic Handling of InnoDB Cluster state changes

- Asynchronous replication is automatically reconfigured after primary change

Datacenter Crash/Partition



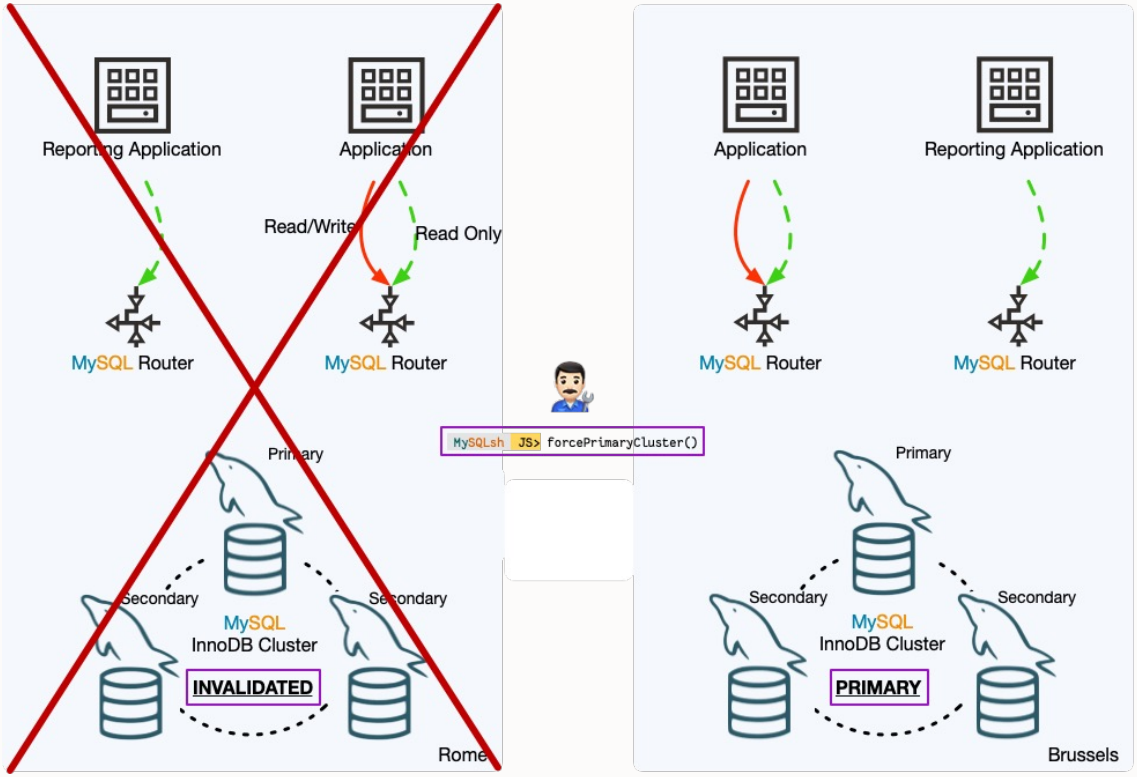
Failover to another Cluster

- one command to invalidate the PRIMARY cluster and promote a new PRIMARY cluster: **`forcePrimaryCluster()`**
- other REPLICA clusters replication will be recon

Split Brain Warning

- local Routers that cannot connect to other clusters will not learn about new topology
- if datacenter is network partitioned, it will continue to operate as PRIMARY

Datacenter Crash/Partition - `forcePrimaryCluster()`



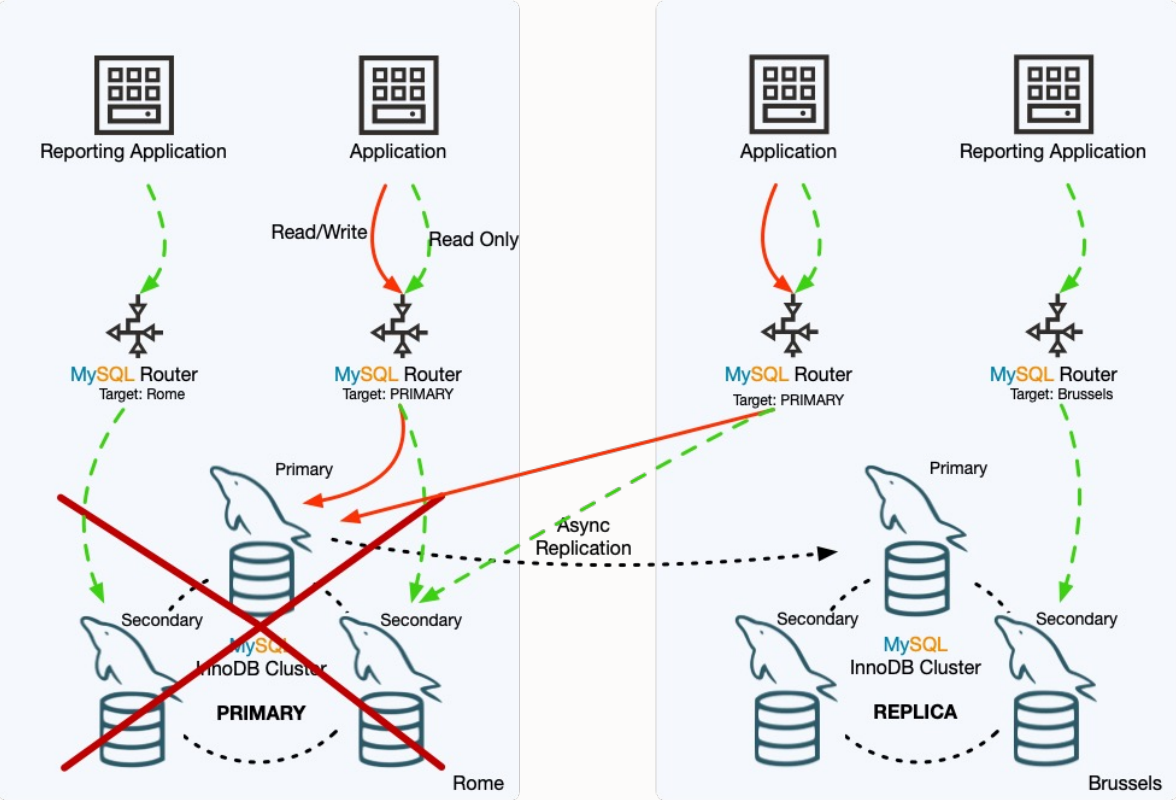
Failover to another Cluster

- one command to invalidate the PRIMARY cluster and promote a new PRIMARY cluster: **`forcePrimaryCluster()`**
- other REPLICA clusters replication will be recon

Split Brain Warning

- local Routers that cannot connect to other clusters will not learn about new topology
- if datacenter is network partitioned, it will continue to operate as PRIMARY

Group Replication Crash/Partition



Router Integration

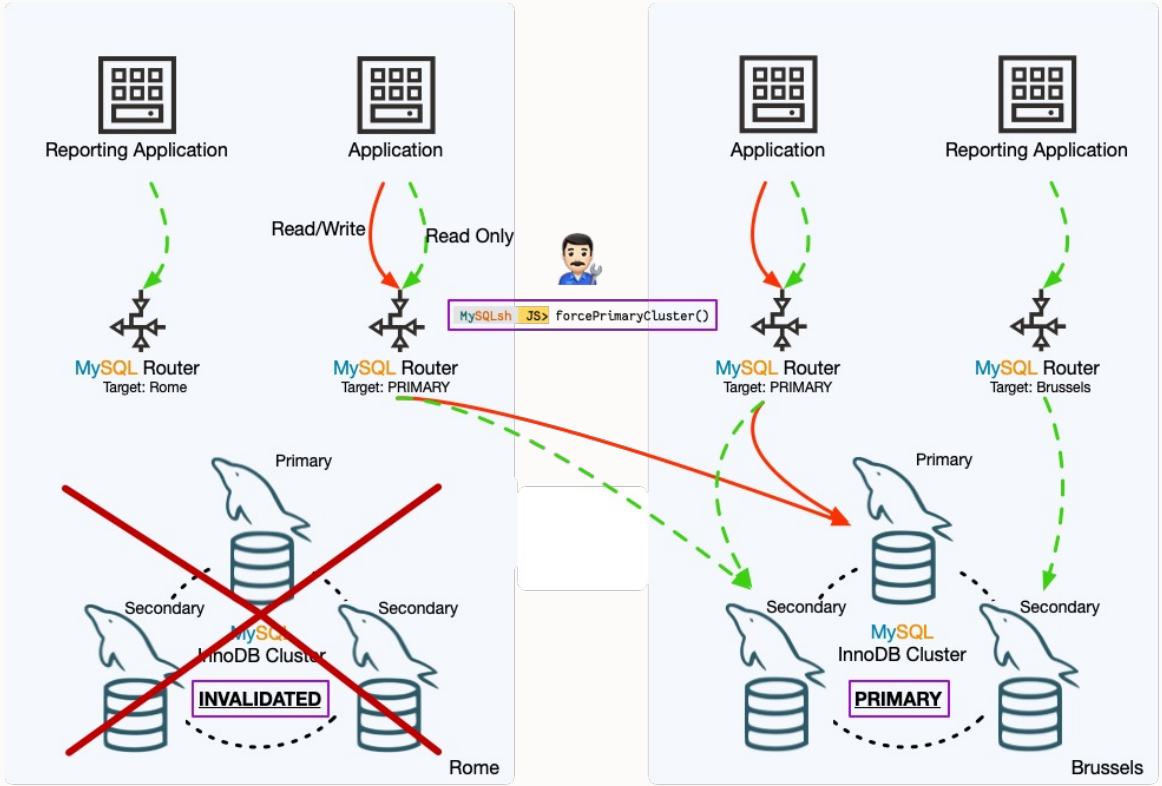
When GR is offline:

- network partition
- no quorum
- full cluster lost (e.g. power outage)

Failover to another Cluster

- one command to invalidate the PRIMARY cluster and promote a new PRIMARY cluster: **`forcePrimaryCluster()`**
- Router instances will follow PRIMARY (depending on target mode)

Group Replication Crash/Partition - `forcePrimaryCluster()` & Router



Router Integration

When GR is offline:

- network partition
- no quorum
- full cluster lost (e.g. power outage)

Failover to another Cluster

- one command to invalidate the PRIMARY cluster and promote a new PRIMARY cluster: **`forcePrimaryCluster()`**
- Router instances will follow PRIMARY (depending on target mode)

Failover to another Cluster

```
mysqlsh> \c root@localhost:3331
mysqlsh> clusterset=dba.getClusterSet()
mysqlsh> clusterset.forcePrimaryCluster('BRU')

Failing-over primary cluster of the clusterset to 'BRU'
- Verifying primary cluster status
None of the instances of the PRIMARY cluster 'LIS' could be reached.
- Verifying clusterset status
-- Checking cluster BRU
  Cluster 'BRU' is available
-- Checking cluster ROM
  Cluster 'ROM' is available
-- Checking whether target cluster has the most recent GTID set
- Promoting cluster 'BRU'
- Updating metadata

-- Changing replication source of 127.0.0.1:5552 to 127.0.0.1:3331
-- Changing replication source of 127.0.0.1:5553 to 127.0.0.1:3331
-- Changing replication source of 127.0.0.1:5551 to 127.0.0.1:3331
PRIMARY cluster failed-over to 'BRU'. The PRIMARY instance is '127.0.0.1:3331'
Former PRIMARY cluster was INVALIDATED, transactions that were not yet replicated may be lost.
```


Business Requirements

Different needs, different solutions

Business Requirements

Concepts – RTO & RPO

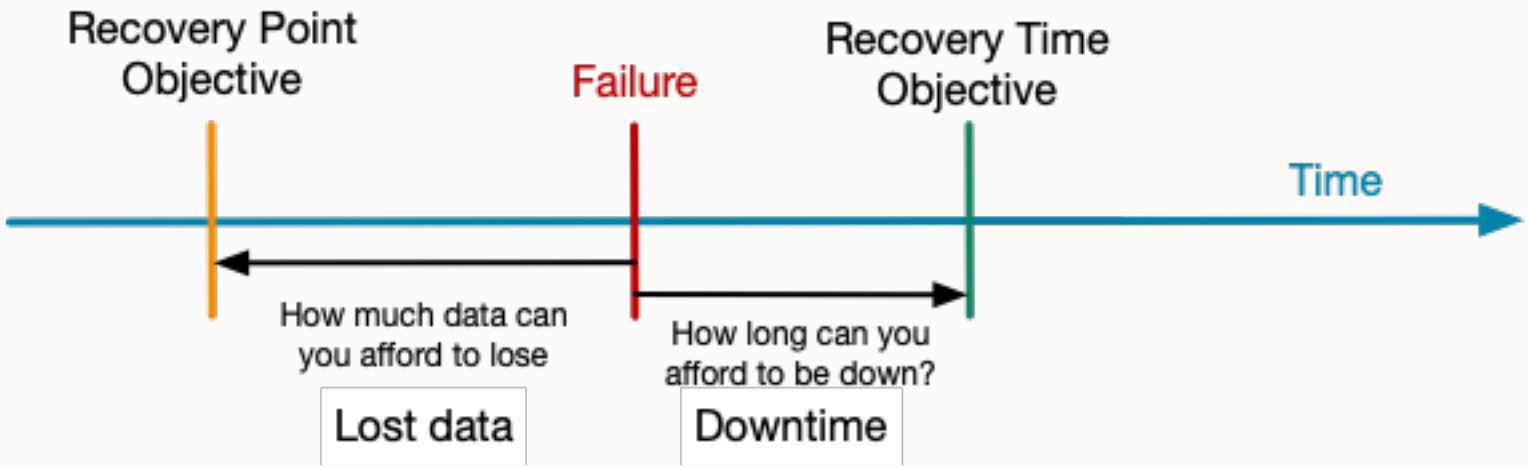
- RTO: Recovery Time Objective
  - How long does it take to recover from a single failure
- RPO: Recovery Point Objective
  - How much data can be lost when a failure occurs

Types of Failure

High Availability: Single Server Failure, Network Partition

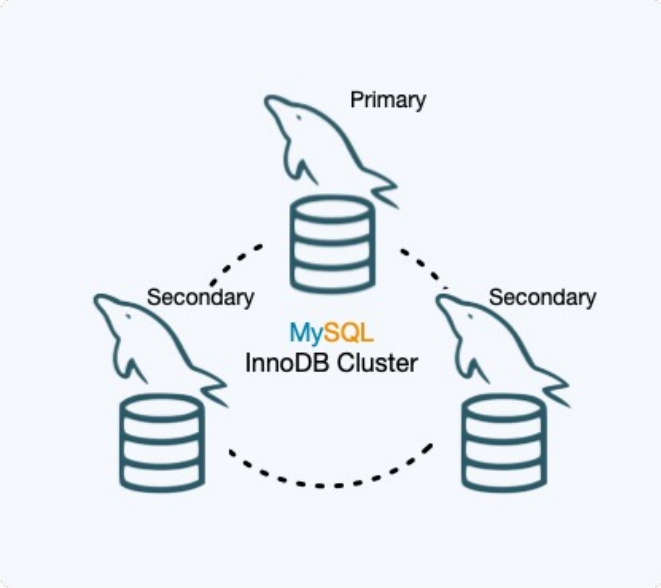
Disaster Recovery: Full Region/Network Failure

Human Error: Little Bobby Tables



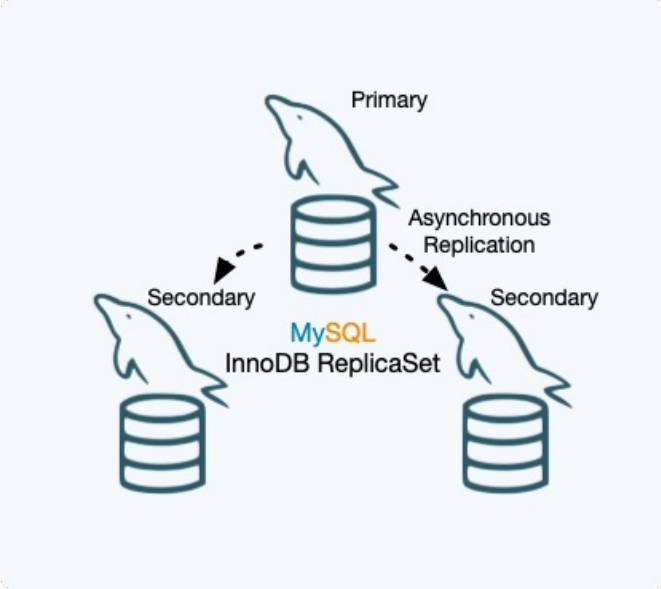
High Availability – Single Region

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

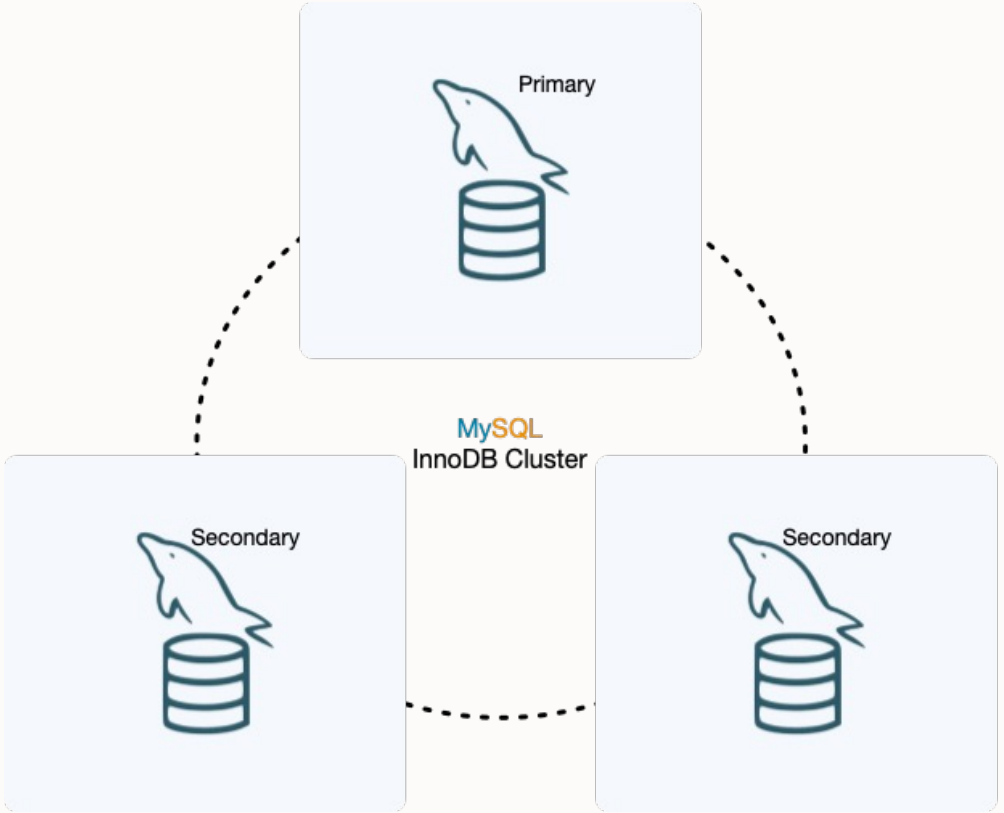
- RPO = 0
- RTO = Seconds

👍 Multi-Region Multi-Primary

👎 3 DC

👎 Requires very stable WAN

👎 Write performance affected by latency between DCs



Disaster Recovery - Multi Region

MySQL InnoDB Cluster ClusterSet

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

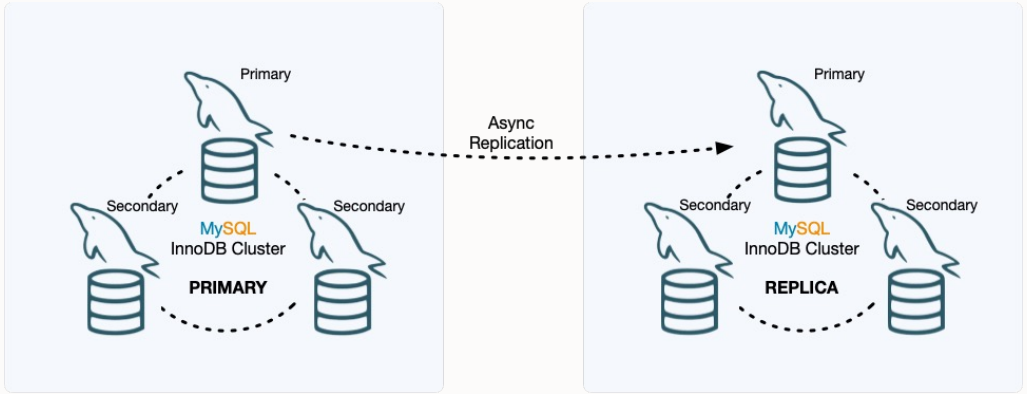
👍 RPO = 0 & RTO = seconds within Region (HA)

👎 Write performance (no sync to other region required)

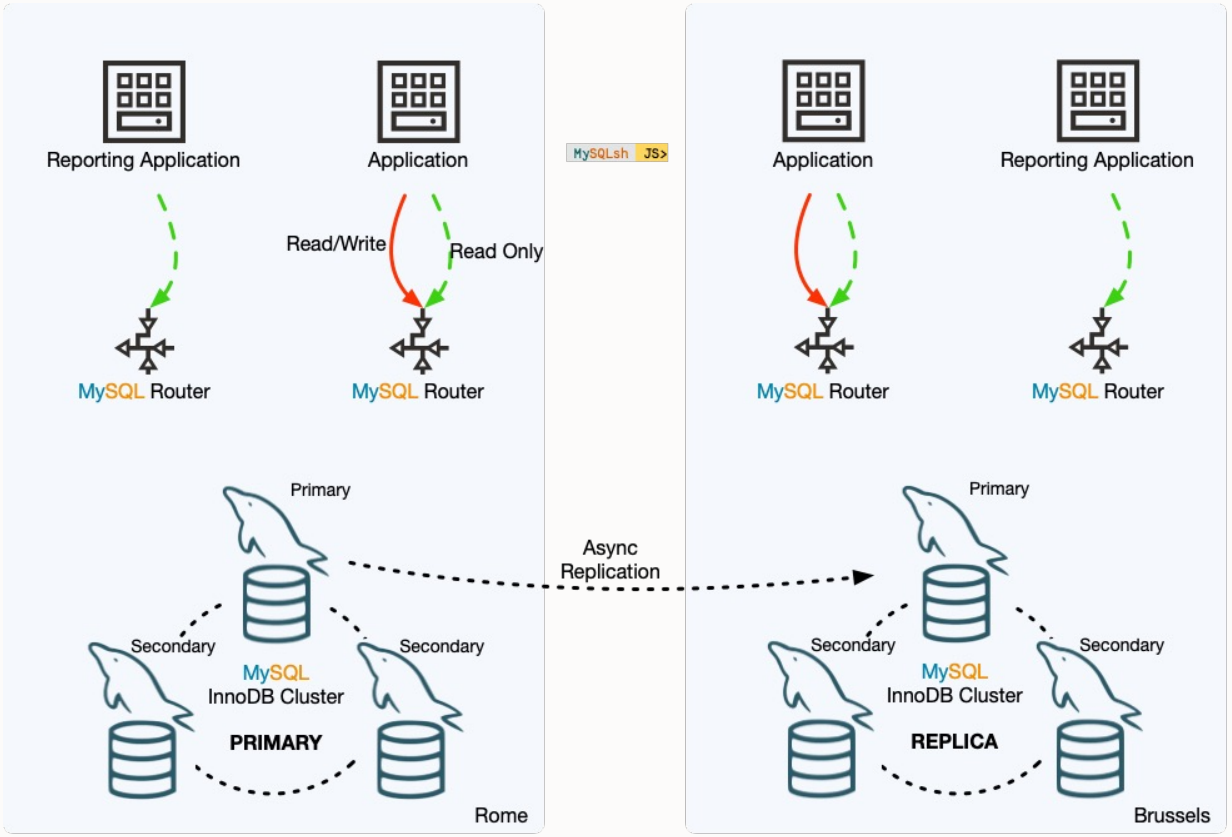
👎 Higher RTO: Manual failover

👎 RPO != 0 when region fails

MySQL InnoDB ClusterSet



MySQL InnoDB ClusterSet



ORACLE DevLive | Level Up | MySQL Summit

Please rate this session.

Session ID – MS05