

ORACLE

MySQL™ Belgian Days 2024

# MySQL Shell/AdminAPI

MySQL Architectures Made Easy For All!

**Miguel Araújo**

Senior Principal Software Engineer

MySQL, Oracle

February 2, 2024

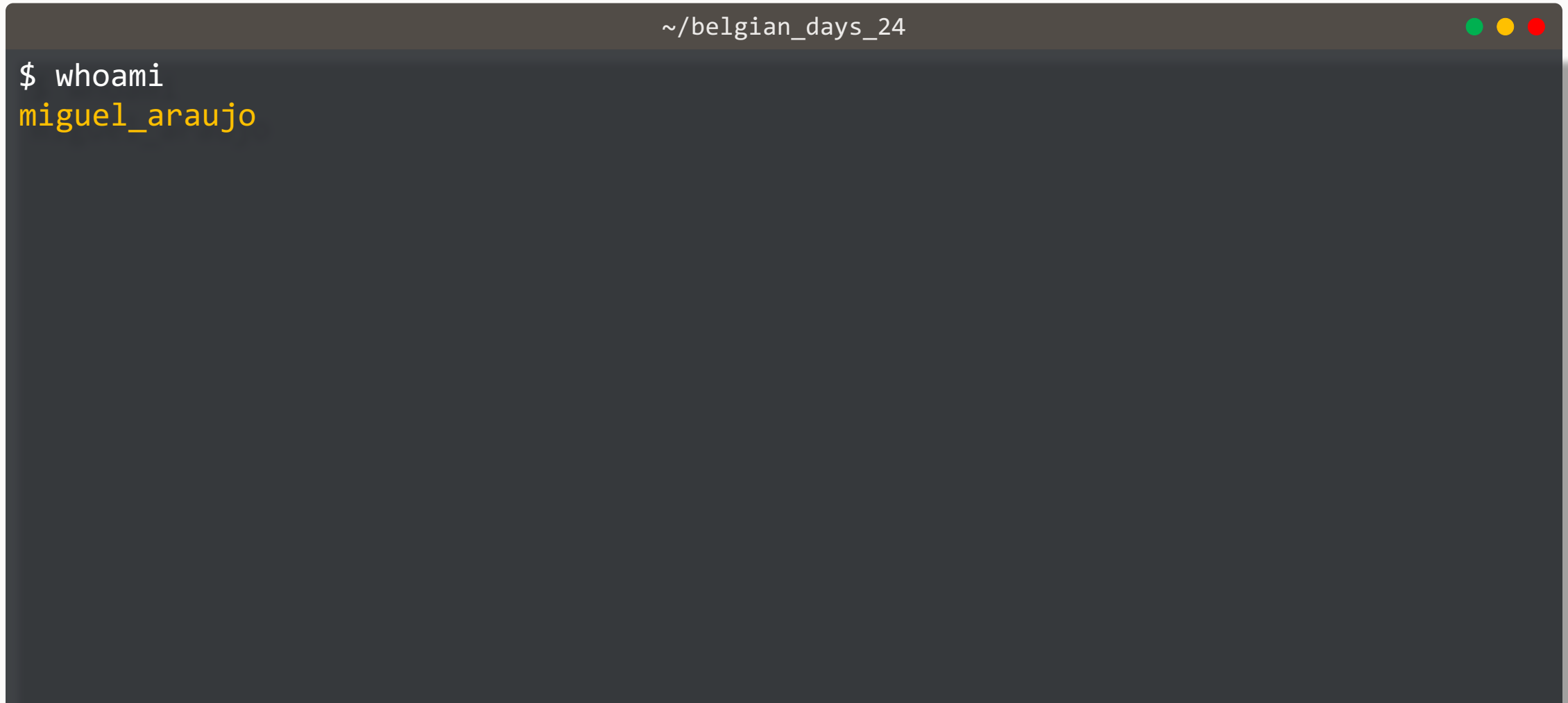


# Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purpose only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied up in making purchasing decisions. The development, release and timing of any features or functionality described for Oracle's product remains at the sole discretion of Oracle.



\$ whoami && history



A terminal window with a dark gray background and a brown title bar. The title bar contains the text '~ /belgian\_days\_24' and three window control buttons (green, yellow, red). The terminal shows the command '\$ whoami' and its output 'miguelaraujo' in yellow text.

```
~/belgian_days_24  
$ whoami  
miguelaraujo
```

\$ whoami && history

~/belgian\_days\_24

\$ whoami

miguelaraujo

\$ history -E

07.11.2011      Joined MySQL



## \$ whoami && history

~/belgian\_days\_24

```
$ whoami
```

```
miguel_araujo
```

```
$ history -E
```

```
07.11.2011    Joined MySQL
```

```
27.05.2014    MySQL Fabric 1st GA
```

## \$ whoami && history

```
~/belgian_days_24
$ whoami
miguel_araujo
$ history -E
07.11.2011      Joined MySQL
27.05.2014      MySQL Fabric 1st GA
30.09.2014      'Hello World' blog post from Luís: 1st preview release of GR
```



## \$ whoami && history

```
~/belgian_days_24
$ whoami
miguel_araujo
$ history -E
07.11.2011    Joined MySQL
27.05.2014    MySQL Fabric 1st GA
30.09.2014    'Hello World' blog post from Luís: 1st preview release of GR
20.09.2015    MySQL Router 1st labs release
```



## \$ whoami && history

```
~/belgian_days_24

$ whoami
miguel_araujo

$ history -E

07.11.2011    Joined MySQL
27.05.2014    MySQL Fabric 1st GA
30.09.2014    'Hello World' blog post from Luís: 1st preview release of GR
20.09.2015    MySQL Router 1st labs release
15.12.2015    MySQL Fabric aimed for perfection, instead of...
```



## \$ whoami && history

```
~/belgian_days_24

$ whoami
miguel_araujo

$ history -E

07.11.2011    Joined MySQL
27.05.2014    MySQL Fabric 1st GA
30.09.2014    'Hello World' blog post from Luís: 1st preview release of GR
20.09.2015    MySQL Router 1st labs release
15.12.2015    MySQL Fabric aimed for perfection, instead of...
11.04.2016    MySQL Shell 1st preview release: 1.0.3m1
```

## \$ whoami && history

```
~/belgian_days_24

$ whoami
miguel_araujo

$ history -E

07.11.2011    Joined MySQL
27.05.2014    MySQL Fabric 1st GA
30.09.2014    'Hello World' blog post from Luís: 1st preview release of GR
20.09.2015    MySQL Router 1st labs release
15.12.2015    MySQL Fabric aimed for perfection, instead of...
11.04.2016    MySQL Shell 1st preview release: 1.0.3m1
06.09.2016    AdminAPI 1st preview release with Shell 1.0.8 + MySQL Router 2.1.0
```



## \$ whoami && history

```
~/belgian_days_24

$ whoami
miguel_araujo

$ history -E

07.11.2011    Joined MySQL
27.05.2014    MySQL Fabric 1st GA
30.09.2014    'Hello World' blog post from Luís: 1st preview release of GR
20.09.2015    MySQL Router 1st labs release
15.12.2015    MySQL Fabric aimed for perfection, instead of...
11.04.2016    MySQL Shell 1st preview release: 1.0.3m1
06.09.2016    AdminAPI 1st preview release with Shell 1.0.8 + MySQL Router 2.1.0
12.04.2017    MySQL InnoDB Cluster 1st GA
```



# Business Requirements



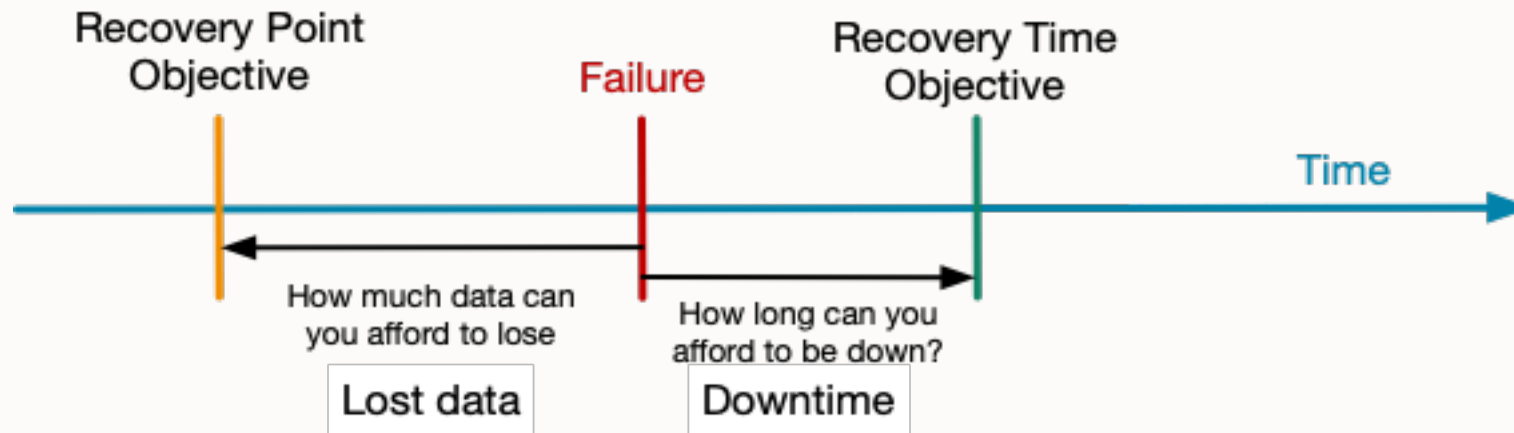
# Business Requirements

## 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

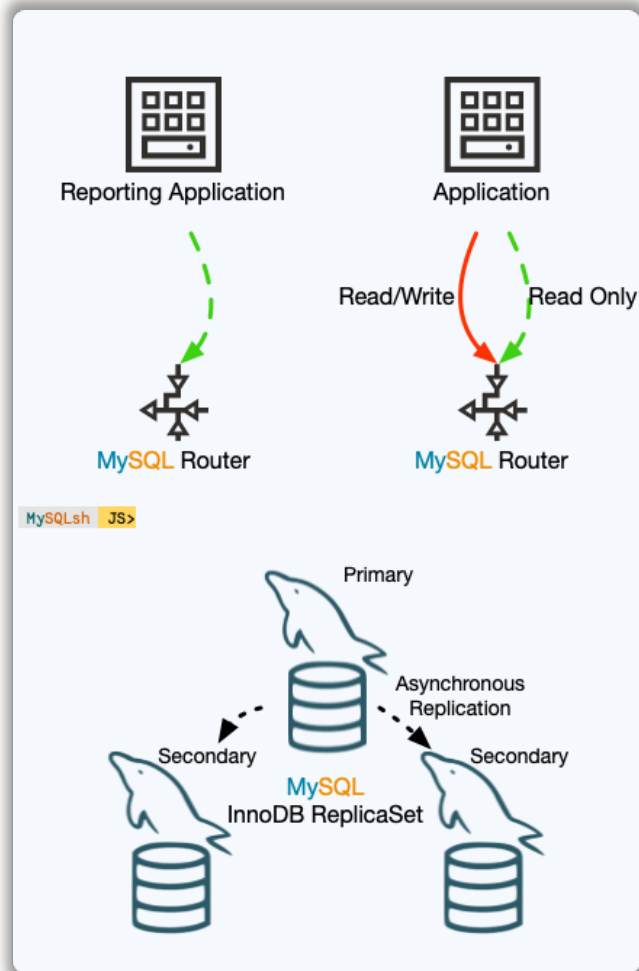
## Types of Failures

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



# MySQL Architectures

# MySQL InnoDB ReplicaSet



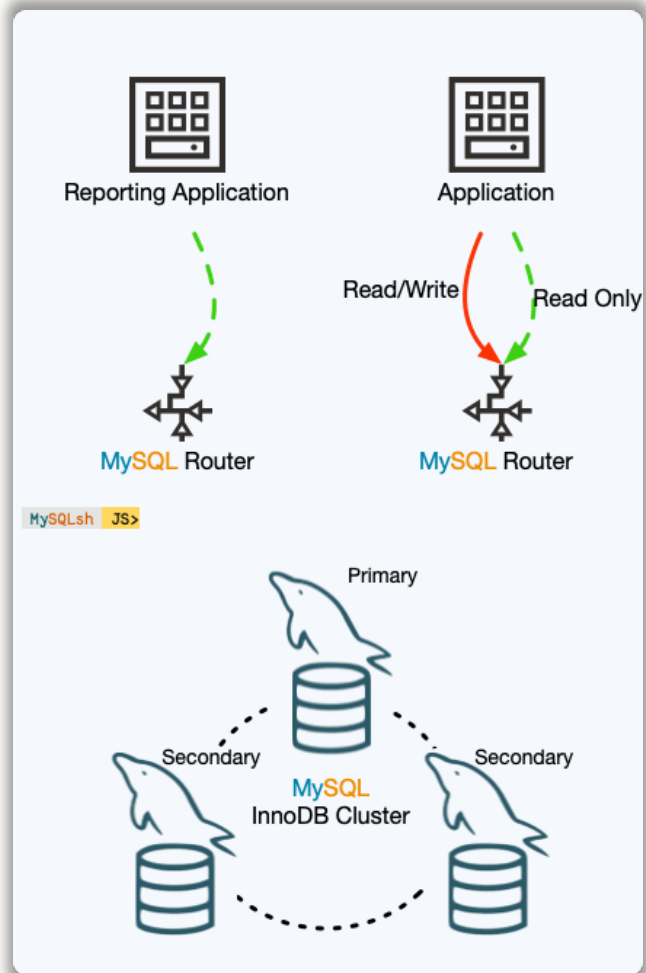
## 'classic', 'asynchronous' Replication based Solution

- Manual failover & switchover
- Asynchronous reads
- Good write performance

RPO != 0

RT0 = minutes or more (manual failover)

# MySQL InnoDB Cluster



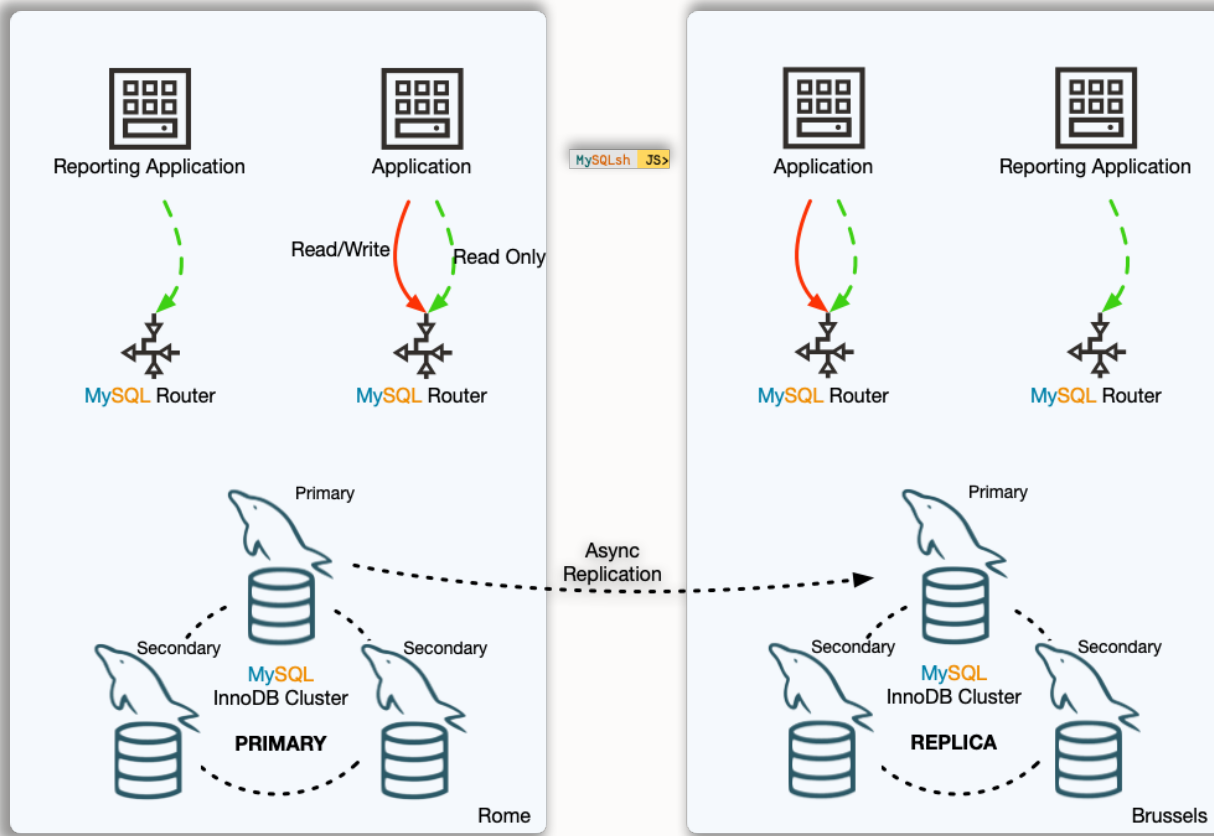
## High Availability solution based on Group Replication

- Automatic failover / Fault Tolerance
- Automatic membership changes
- Network partition handling
- Consistency

RPO = 0

RT0 = seconds (automatic failover)

# MySQL InnoDB ClusterSet

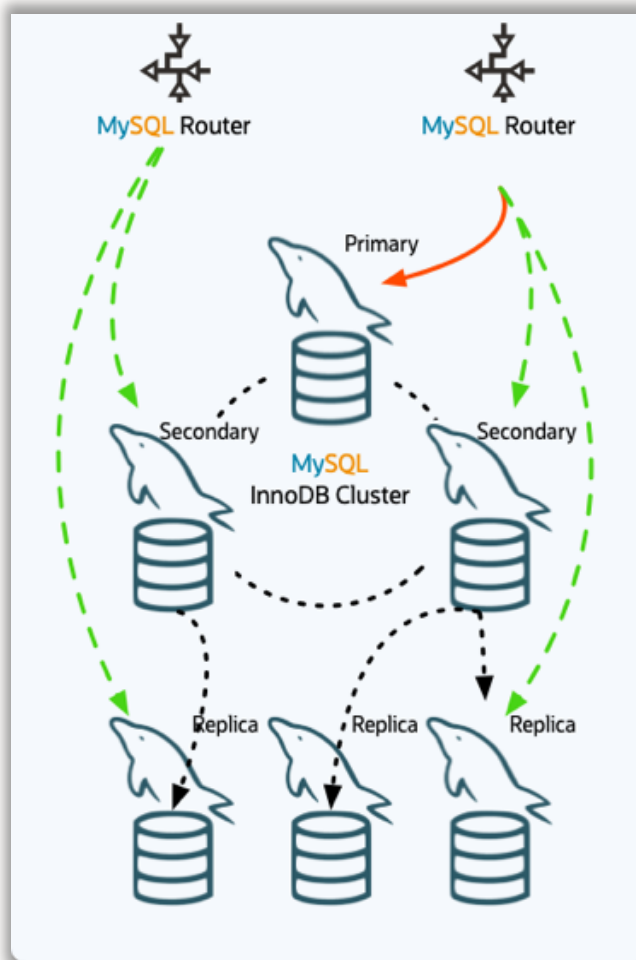


## Disaster Tolerance Solution for InnoDB Clusters deployments in alternate locations

- 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

# MySQL InnoDB Cluster Read Replicas

New in 8.1.0 !



## Read Scale-out

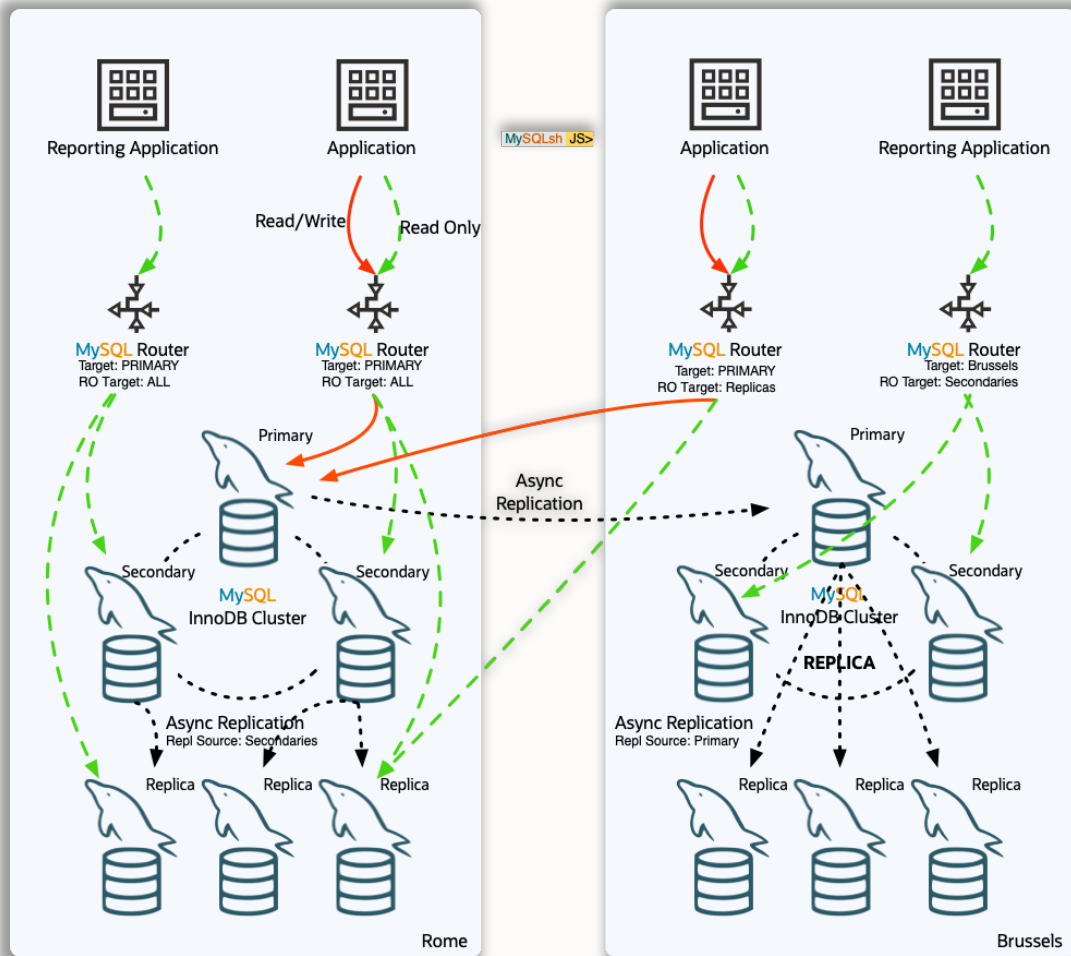
- Add any amount of async read replicas to a Cluster
- Replicate/Failover from
  - PRIMARY
  - SECONDARIES
  - LIST of candidates

## Fully supported on

- InnoDB Cluster
- InnoDB ClusterSet

# MySQL InnoDB ClusterSet with Read Replicas

New in 8.1.0 !



## Flexible

- Add/Remove Read Replicas online
- Configure Router behavior dynamically
- Choose where to route traffic

## Failover

- Automatic connection failover
- List of potential sources automatic or manually populated



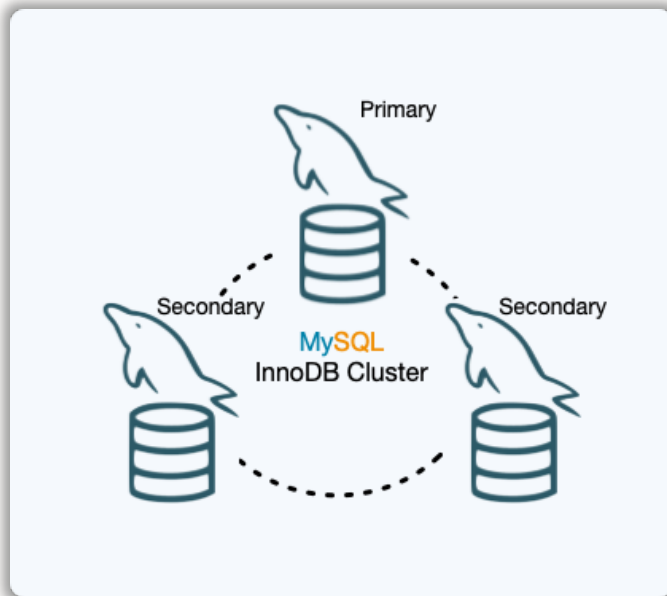
# What architecture fits my requirements?



# High Availability

## Single Region

### MySQL InnoDB Cluster

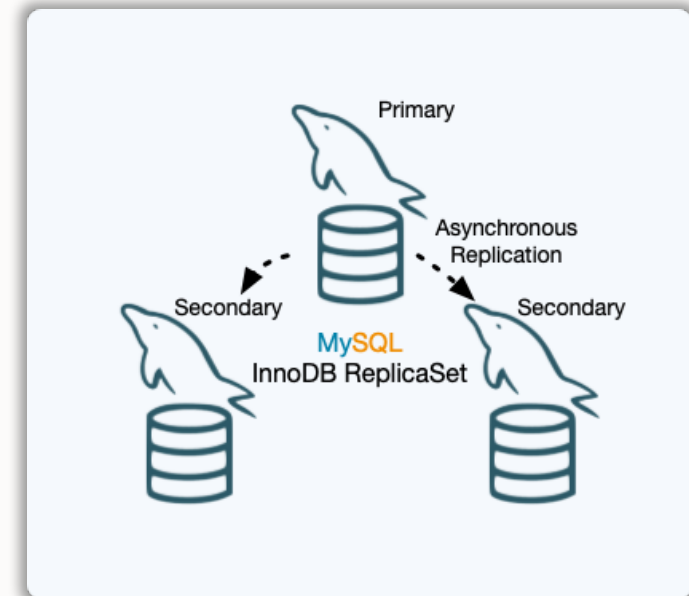


- RPO = 0
- RTO = Seconds



Automatic failover

### MySQL InnoDB ReplicaSet



- RPO != 0
- RTO = Minutes or more (Manual failover)



Best write performance



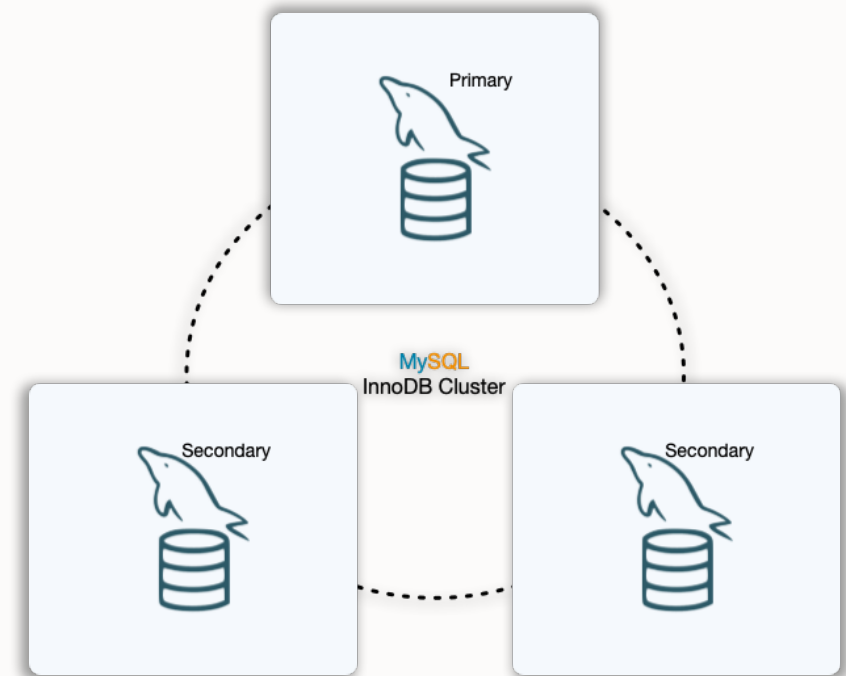
Manual Failover

# High Availability

## Multi Region

**MySQL InnoDB Cluster:** *Deployed over multiple regions*

- Multi-Region Multi-Primary
- 3 DC
- Requires very stable WAN
- Write performance affected by latency between DCs
  - RPO = 0
  - RTO = Seconds

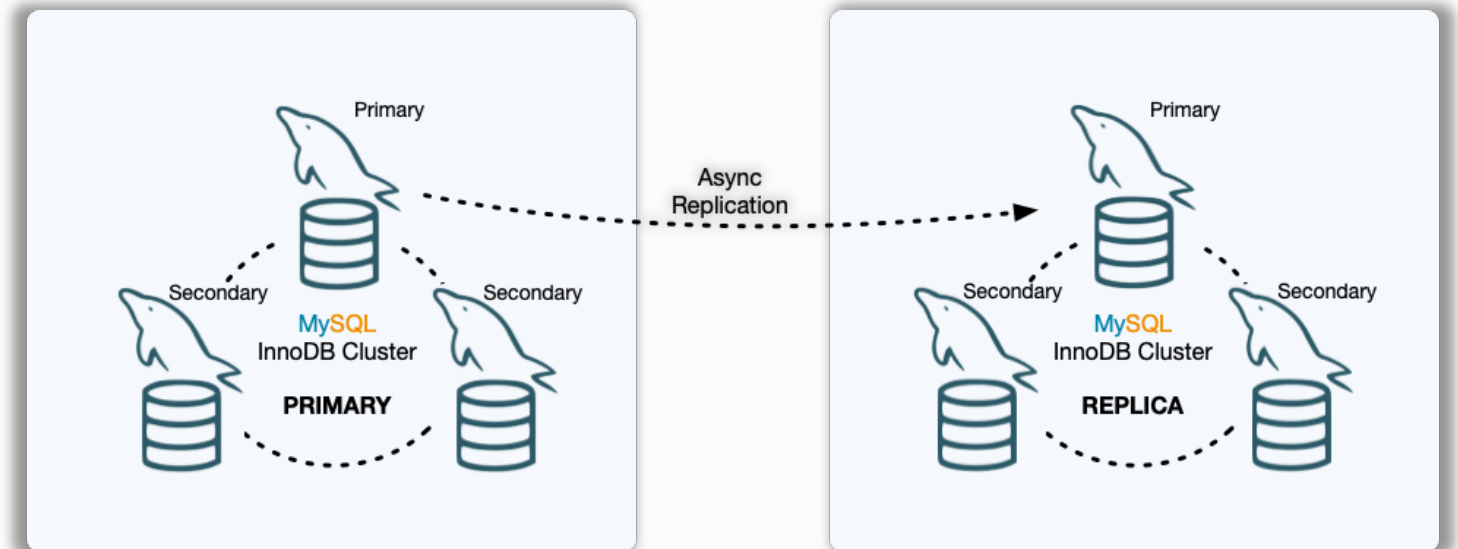


# Disaster Recovery

## Multi Region

### MySQL InnoDB ClusterSet

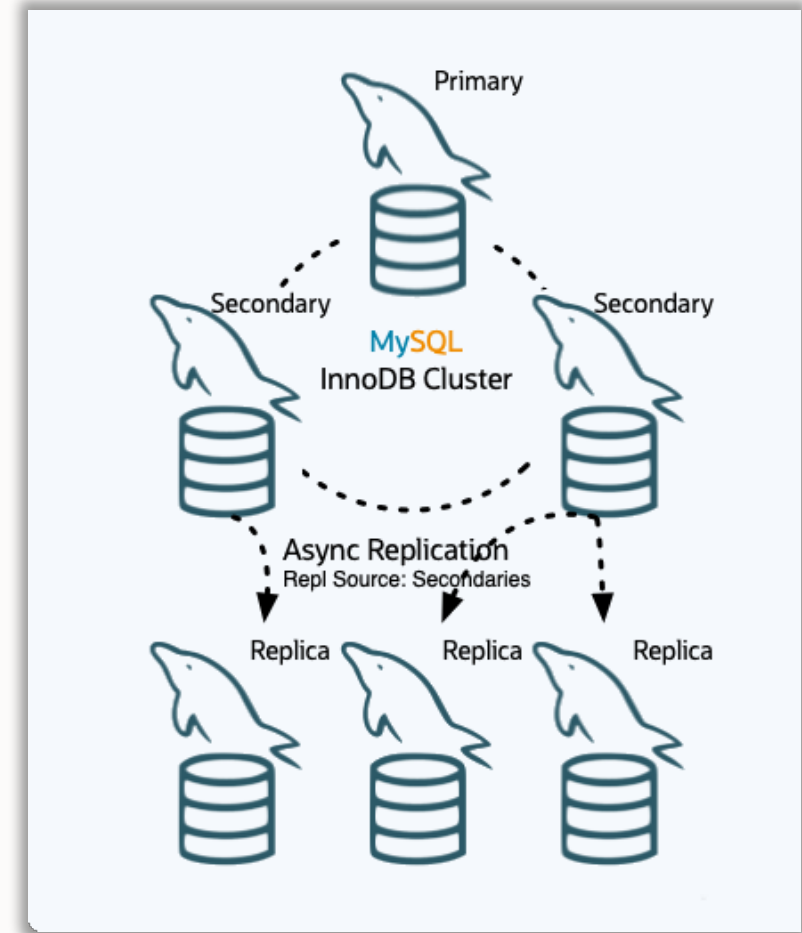
- RPO = 0 & RTO = seconds within Region (HA)
- Write performance (no sync to other region required)
- Higher RTO: Manual failover
- RPO != 0 when region fails
  - RPO != 0
  - RTO = Minutes or more (Manual Failover)



# Read Scale-Out

## MySQL InnoDB Cluster Read Replicas

- Read Intensive Workloads
- Offload Primary or Secondaries
- Dedicated instances for other purposes
- Additional redundancy for the dataset



# Complex?

Absolutely...

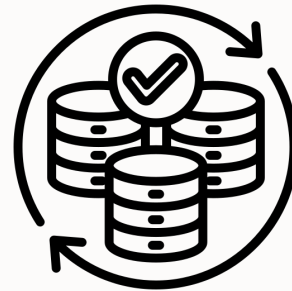
# User Requirements

## 1 Easy to deploy



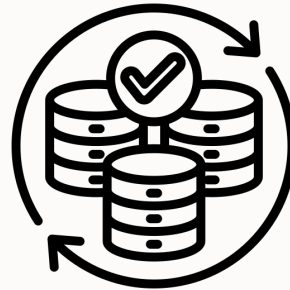
# User Requirements

- 1 Easy to deploy
- 2 Easy to maintain



# User Requirements

- 1 Easy to deploy
- 2 Easy to maintain
- 3 Easy to monitor



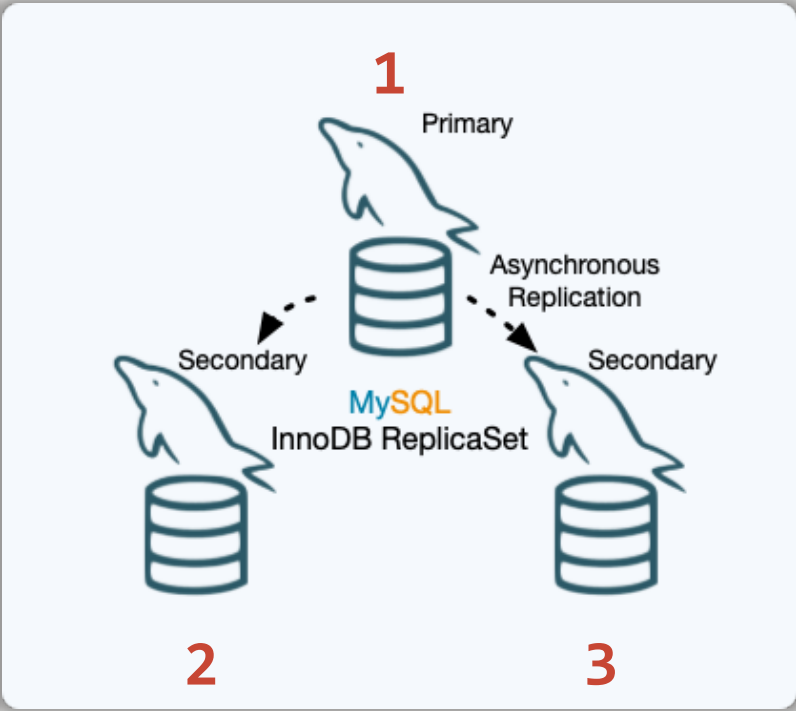


# MySQL Shell AdminAPI

# MySQL Shell AdminAPI

Makes it easy

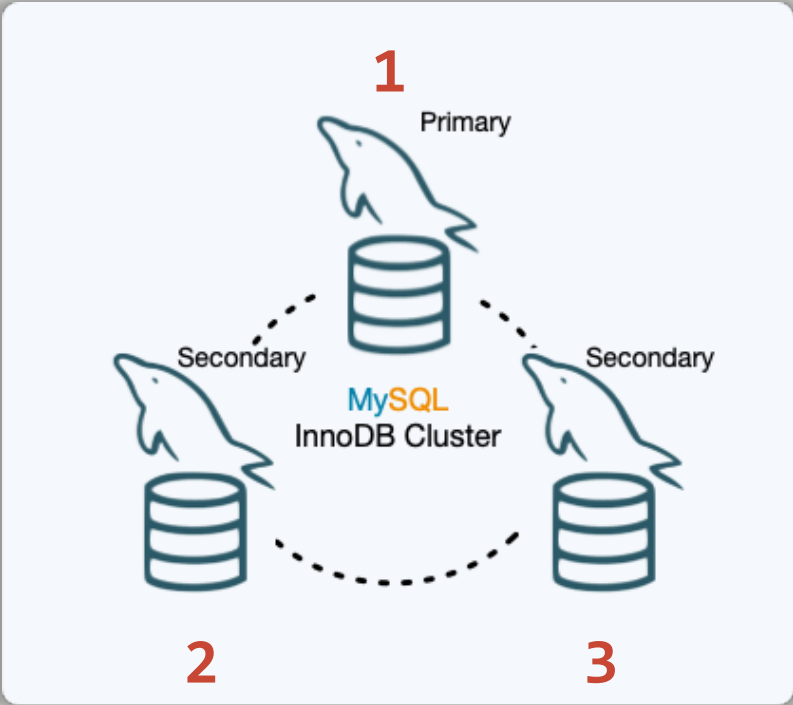
InnoDB ReplicaSet		
Action	Command	# Calls
Configure instances	<code>dba.configureReplicaSetInstance(...)</code>	3
Create Topology	<code>dba.createReplicaSet(...)</code>	1
Setup Admin Account	<code>rs.setupAdminAccount(...)</code>	1
Add instances	<code>rs.addInstance(...)</code>	2
SUM:		7



# MySQL Shell AdminAPI

Makes it easy

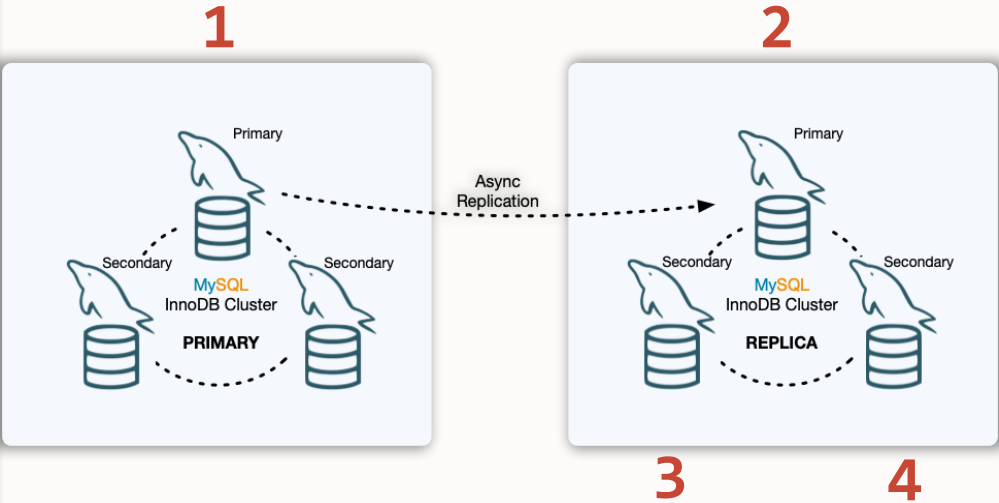
InnoDB Cluster		
Action	Command	# Calls
Configure instances	<code>dba.configureInstance(...)</code>	3
Create Topology	<code>dba.createCluster(...)</code>	1
Setup Admin Account	<code>c.setupAdminAccount(...)</code>	1
Add instances	<code>c.addInstance(...)</code>	2
SUM:		7



# MySQL Shell AdminAPI

Makes it easy

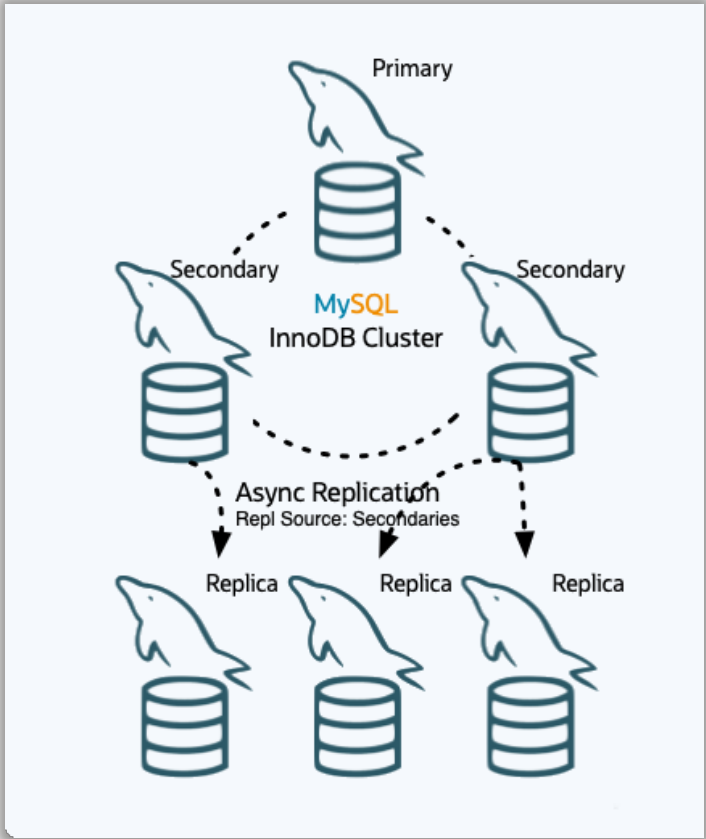
InnoDB ClusterSet		
Action	Command	# Calls
Create Topology	<code>c.createClusterSet(...)</code>	1
Create Replica Cluster	<code>cs.createReplicaCluster(...)</code>	2
Add instances to Replica	<code>rc.addInstance(...)</code>	3
SUM:		6



# MySQL Shell AdminAPI

Makes it easy

InnoDB Cluster Read Replicas		
Action	Command	# Calls
Add Read Replicas	<code>c.addReplicaInstance(...)</code>	3
Configure Router behavior	<code>c.setRoutingOption(...)</code>	1
SUM:		4



1 2 3



# Features

- Sandbox management
- Configuration checker & applier
- Account management
- MySQL Architectures management
- Integrated provisioning
- Configuration management
- MySQL Router management
- Follows best practices



# MySQL Shell AdminAPI

It's not just a bunch of scripts...

~/ngshell/modules/adminapi				
-----				
Language	files	blank	comment	code
-----				
C++	144	13213	12466	63402
C/C++ Header	175	4670	9796	14447
SQL	7	393	1298	1447
-----				
SUM:	326	18276	23560	79296
-----				

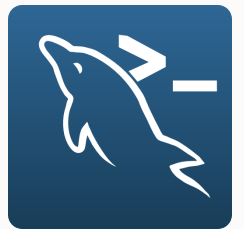


# Latest Additions



# Security

- **MySQL Communication Stack used by default 8.0.30**
- **Full TLS/SSL Support 8.0.33**
  - Encrypt Group Replication and Asynchronous replication channels
  - Certificate-based authentication for intra-node communication
  - Certificate-based authentication for Admin and Router accounts



# Security

```
MySQL localhost:5001 ssl JS > primary = dba.createCluster("primary", { memberSslMode: "VERIFY_IDENTITY", memberAuthType: "CERT_SUBJECT_PASSWORD", certIssuer: "/CN=MyName", certSubject: "/CN=sl-hostname" });  
A new InnoDB Cluster will be created on instance 'localhost:5001'.
```

Validating instance configuration at localhost:5001 ...

**NOTE:** Instance detected as a sandbox.

Please note that sandbox instances are only suitable for deploying test clusters for use within the same host.

This instance reports its own address as 'localhost:5001'.

Instance configuration is suitable.

**NOTE:** Group Replication will communicate with other members using 'localhost:5001'. Use the localAddress option to override.

\* Checking connectivity and SSL configuration ...

Creating InnoDB Cluster 'primary' on 'localhost:5001' ...

Adding Seed Instance ...

Cluster successfully created. Use Cluster.addInstance() to add MySQL instances.

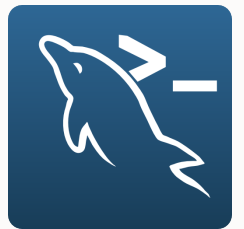
At least 3 instances are needed for the cluster to be able to withstand up to one server failure.

<Cluster:primary>

```
MySQL localhost:5001 ssl JS > |
```

# Concurrency Control & Atomicity

- **Locking mechanism** 8.0.33
  - Prevent conflicting operations to run concurrently resulting in unexpected outcome
  - Supported on the whole API
- **Operations rollback**
  - Avoid leaving the system / instance in a transient state



# Router management

- **More control over Router configuration**

- stats\_updates\_frequency **8.1.0**
- read\_only\_targets **8.1.0**
  - all
  - read\_replicas
  - secondaries
- unreachable\_quorum\_allowed\_traffic **8.2.0**



# Router management

```
MySQL localhost:3310 JS \? clusterset.setRoutingOption
```

**NAME**

**setRoutingOption** - Changes the value of either a global Routing option or of a single Router instance.

**SYNTAX**

```
<ClusterSet>.setRoutingOption([router], option, value)
```

**WHERE**

router: Identifier of the target router instance (e.g. 192.168.45.70::system).  
option: The Router option to be changed.  
value: The value that the option shall get (or null to unset).

**RETURNS**

Nothing.

**DESCRIPTION**

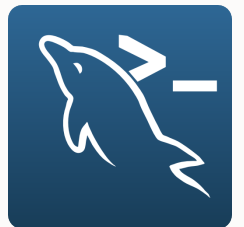
The accepted options are:

- target\_cluster: Target Cluster for Router routing operations. Default value is 'primary'.
- invalidated\_cluster\_policy: Routing policy to be taken when the target cluster is detected as being invalidated. Default value is 'drop\_all'.
- stats\_updates\_frequency: Number of seconds between updates that the Router is to make to its statistics in the InnoDB Cluster metadata.
- use\_replica\_primary\_as\_rw: Enable/Disable the RW Port in Replica Clusters. Disabled by default.
- tags: Associates an arbitrary JSON object with custom key/value pairs with the ClusterSet metadata.
- read\_only\_targets: Routing policy to define Router's usage of Read Replicas. Default is 'append'.



# MySQL Architectures

- **InnoDB Cluster Read Replicas 8.1.0**
- **InnoDB ReplicaSet new commands: 8.3.0**
  - `.rescan()`
  - `.dissolve()`
  - `.describe()`
- **Support fine-grained replication options 8.2.0**
  - `SOURCE_*`, `NETWORK_NAMESPACE`
  - ClusterSet async channel & ReplicaSet



# InnoDB Cluster Read Replicas

```
~/belgian_days_24

// Default, follow primary
mysqlsh-js> cluster.addReplicaInstance("brussels:3006")
(...)

// Change to follow secondaries
mysqlsh-js> cluster.setInstanceOption("brussels:3006", {replicationSources:
"secondary"})
mysqlsh-js> cluster.rejoinInstance("brussels:3006")
(...)

// Configure Router to use only Read Replicas for R/O traffic
mysqlsh-js> cluster.setRoutingOption("read_only_targets", "read_replicas")
(...)
```

# Deprecations **8.2.0**

- **5.7 support EOL'd Oct 2023**
- `dba.configureLocalInstance()`
- `Cluster.checkInstanceState()`
- Command options:
  - `interactive`
  - `password`
  - `clearReadOnly`





# Thank you!

 MySQL™ Belgian Days 2024

Questions?  
Suggestions?  
Requests?



February 1 & 2  
ICAB Incubator  
4 Rue des Pères Blancs 1040 Bruxelles



ORACLE