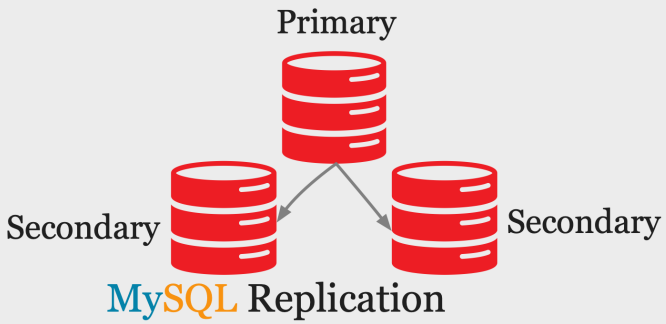

ORACLE
MySQL
MySQL
Database Architectures
#MySQL8isGreat
Kenny Gryp
MySQL Product Manager

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.

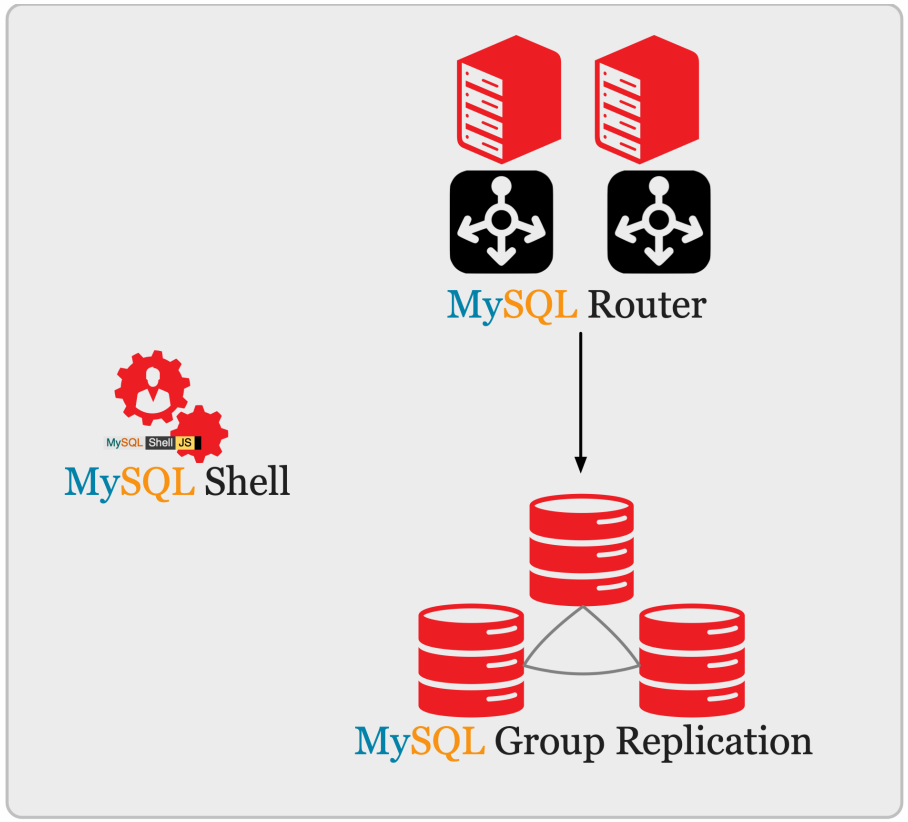
Past, Present & Future

'Past' - Manual



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

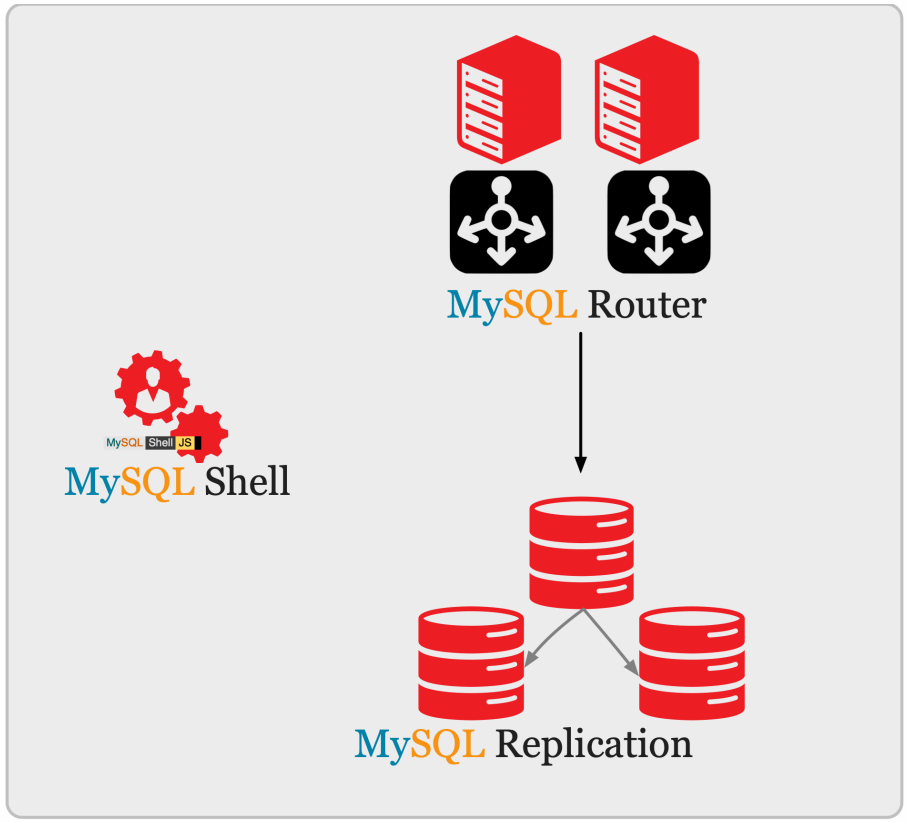
Present - Solutions!



2016 - MySQL InnoDB Cluster

- Group Replication: Automatic membership changes, network partition handling, consistency…
- Shell to provide a powerful interface that helps in automating an integrating all components
- InnoDB CLONE to automatically provision members, fully integrated in InnoDB

Present - Solutions!



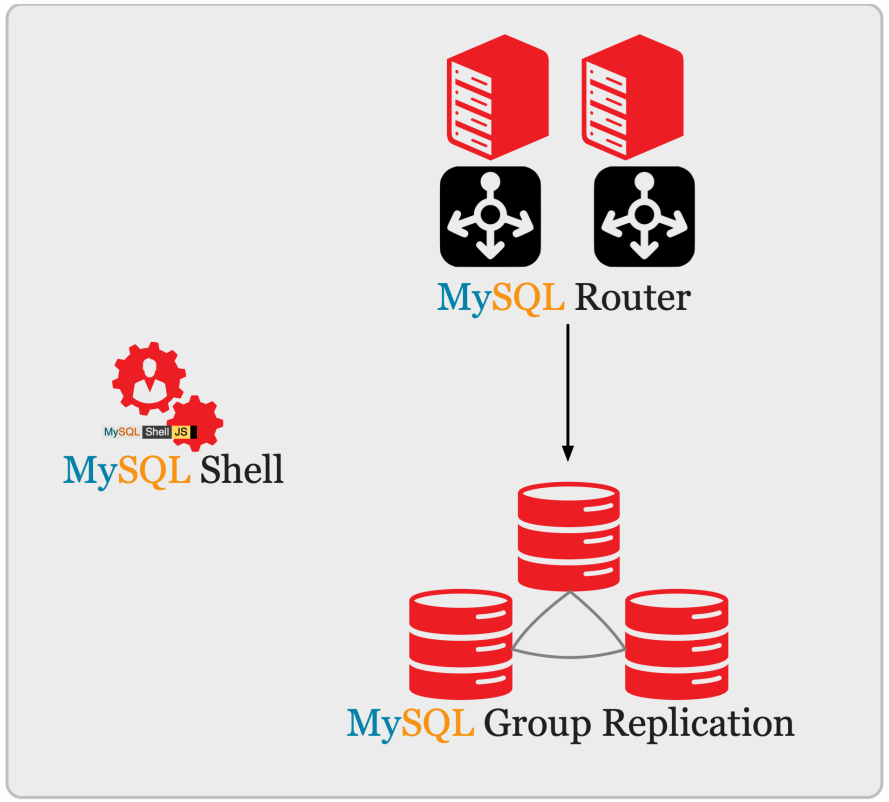
2020 - MySQL InnoDB Replicaset

- 'classic', 'asynchronous' Replication based Solution, fully integrated

MySQL InnoDB Cluster

MySQL InnoDB Cluster

"A single product — MySQL — with **high availability** and **scaling features** baked in;
providing an **integrated end-to-end solution that is easy to use.**"



Components:

- MySQL Server
- MySQL Group Replication
- MySQL Shell
- MySQL Router

MySQL InnoDB Cluster - Goals

One Product: MySQL

- All components developed together
- Integration of all components
- Full stack testing

MySQL InnoDB Cluster - Goals

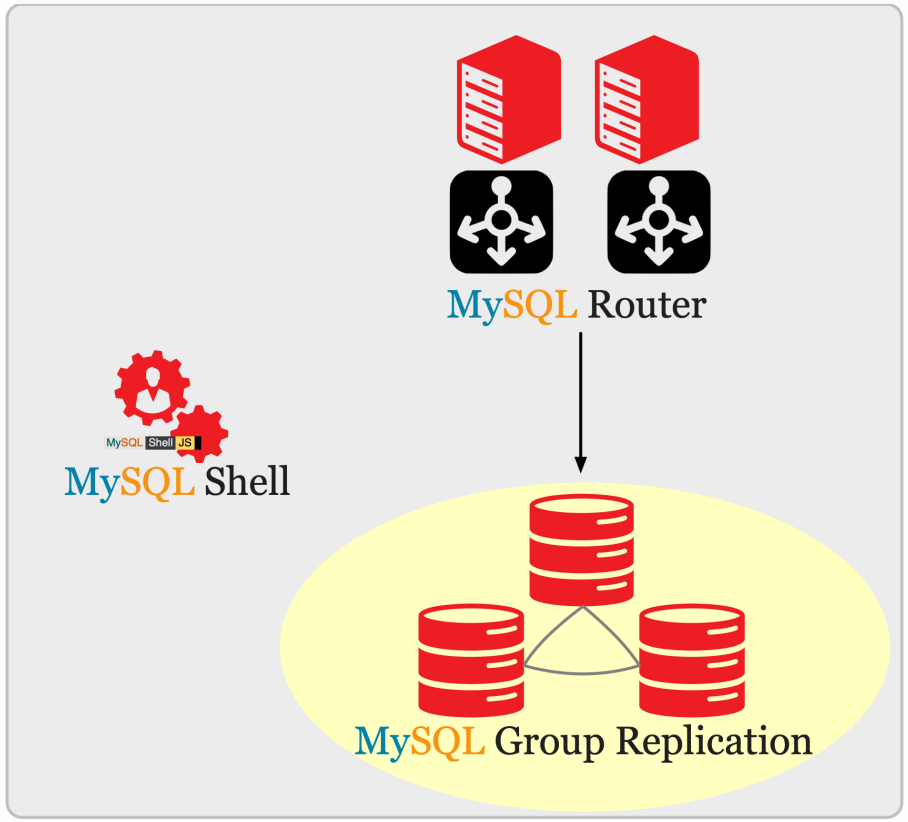
One Product: MySQL

- All components developed together
- Integration of all components
- Full stack testing

Easy to Use

- One client: MySQL Shell
- Integrated orchestration
- Homogenous servers

MySQL Group Replication



High Available Distributed MySQL DB

- Fault tolerance
- Automatic failover
- Active/Active update anywhere (limits apply)
- Automatic membership management
  - Adding/removing members
  - Network partitions, failures
- Conflict detection and resolution
- Prevents data loss

MySQL Group Replication

- Implementation of 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: linux, windows, solaris, macosx, freebsd

MySQL Group Replication - Use Cases

Consistency: No Data Loss (RPO=0)

- in event of failure of (primary) member
- Split brain prevention (Quorum)

MySQL Group Replication - Use Cases

Consistency: No Data Loss (RPO=0)

- in event of failure of (primary) member
- Split brain prevention (Quorum)

Highly Available: Automatic Failover

- Primary members are automatically elected
- Automatic Network Partition handling

MySQL Group Replication - Use Cases

Consistency: No Data Loss (RPO=0)

- in event of failure of (primary) member
- Split brain prevention (Quorum)

Highly Available: Automatic Failover

- Primary members are automatically elected
- Automatic Network Partition handling

Read Scaleout

- Add/Remove members as needed
- Replication Lag handling with Flow Control
- Configurable Consistency Levels
  - Eventual
  - Full Consistency -- no stale reads

MySQL Group Replication - Use Cases

Consistency: No Data Loss (RPO=0)

- in event of failure of (primary) member
- Split brain prevention (Quorum)

Highly Available: Automatic Failover

- Primary members are automatically elected
- Automatic Network Partition handling

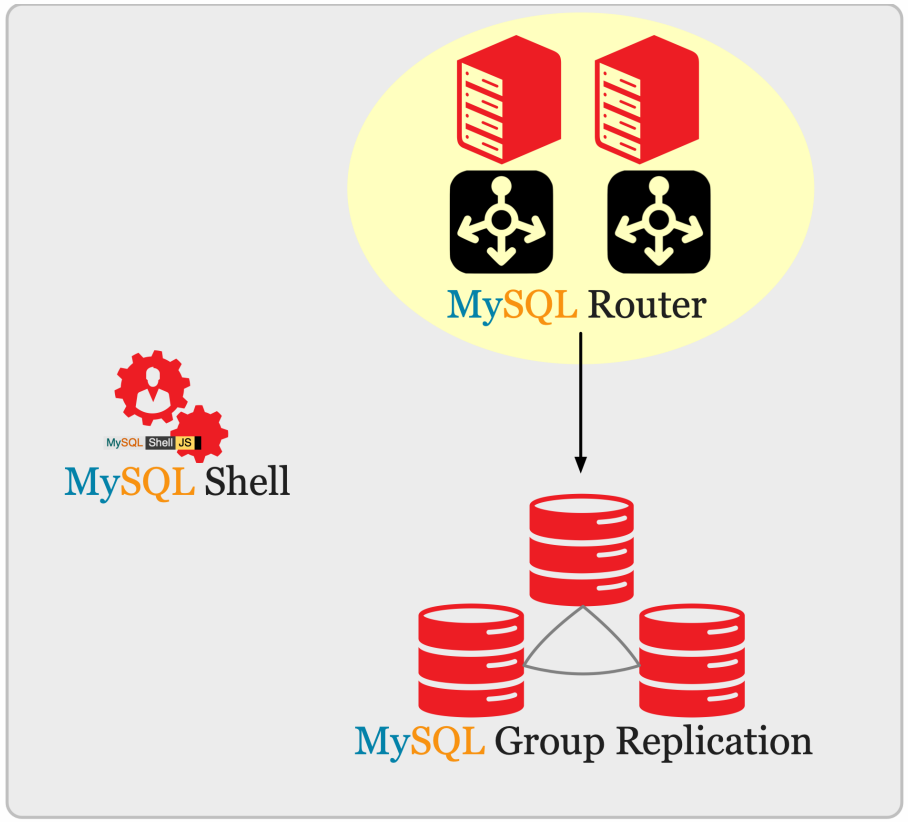
Read Scaleout

- Add/Remove members as needed
- Replication Lag handling with Flow Control
- Configurable Consistency Levels
  - Eventual
  - Full Consistency -- no stale reads

Active/Active environments

- Write to many members at the same time
  - ordered writes within the group (XCOM)
  - guaranteed consistency
- Good write performance
  - due to Optimistic Locking (workload dependent)

MySQL Router

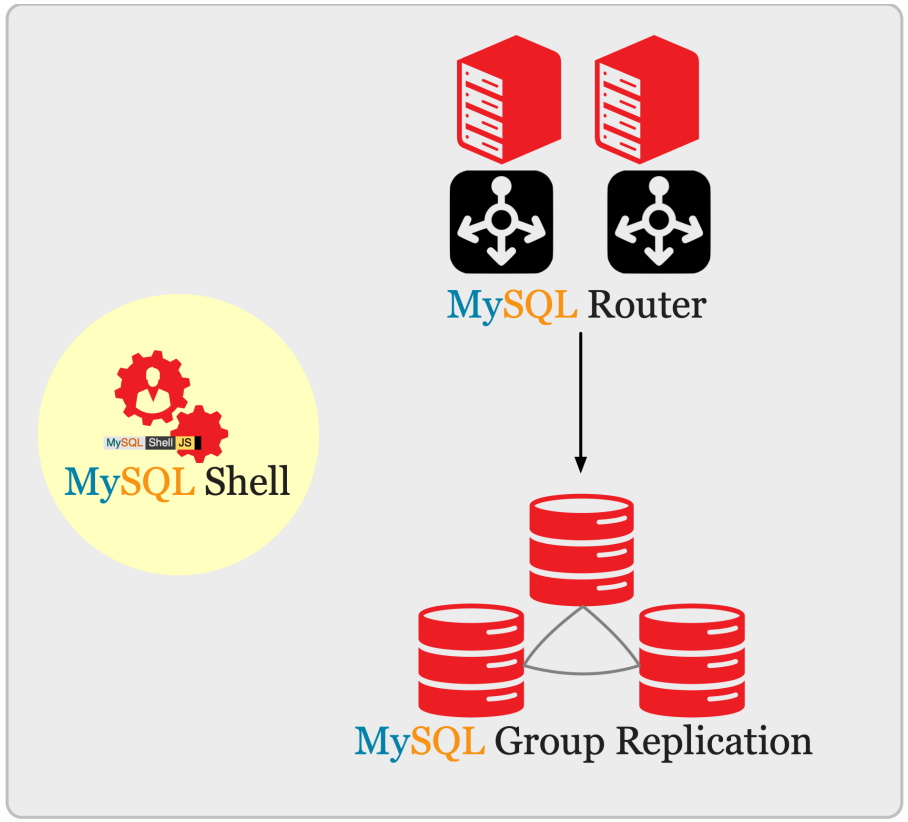


Transparent Access to Database Arch.

"provide transparent routing between your application and back-end MySQL Servers"

- Transparent client connection routing
  - Load balancing
  - Application connection failover
  - Little to no configuration needed
- Stateless design offers easy HA client routing
  - Router as part of the application stack
- Integration into InnoDB Cluster & InnoDB ReplicaSet
- 2 TCP Ports: **PRIMARY** and **NON-PRIMARY** traffic

MySQL Shell



Database Administration Interface

"MySQL Shell provides the developer and DBA with a single intuitive, flexible, and powerful interface for all MySQL related tasks!"

- Multi-Language: JavaScript, Python, and SQL
- Naturally scriptable
- Supports Document and Relational models
- Exposes full Development and Admin API
- Classic MySQL protocol and X protocol

MySQL Shell - Easy to Use

MySQL InnoDB Cluster

```
mysql-js> \c admin@mysql1
mysql-js> cluster = dba.createCluster('cluster')
```


MySQL Shell - Easy to Use

MySQL InnoDB Cluster

```
mysql-js> \c admin@mysql1
mysql-js> cluster = dba.createCluster('cluster')
```

Configure server to add later:

```
mysql-js> dba.configureInstance('admin@mysql2')
```

MySQL Shell - Easy to Use

MySQL InnoDB Cluster

```
mysql-js> \c admin@mysql1
mysql-js> cluster = dba.createCluster('cluster')
```

Configure server to add later:

```
mysql-js> dba.configureInstance('admin@mysql2')
```

Add server to the Cluster:

```
mysql-js> cluster.addInstance('admin@mysql2')
```


MySQL Shell - Easy to Use

MySQL InnoDB Cluster

```
mysql-js> \c admin@mysql1
mysql-js> cluster = dba.createCluster('cluster')
```

Configure server to add later:

```
mysql-js> dba.configureInstance('admin@mysql2')
```

Add server to the Cluster:

```
mysql-js> cluster.addInstance('admin@mysql2')
```

Bootstrap MySQL Router

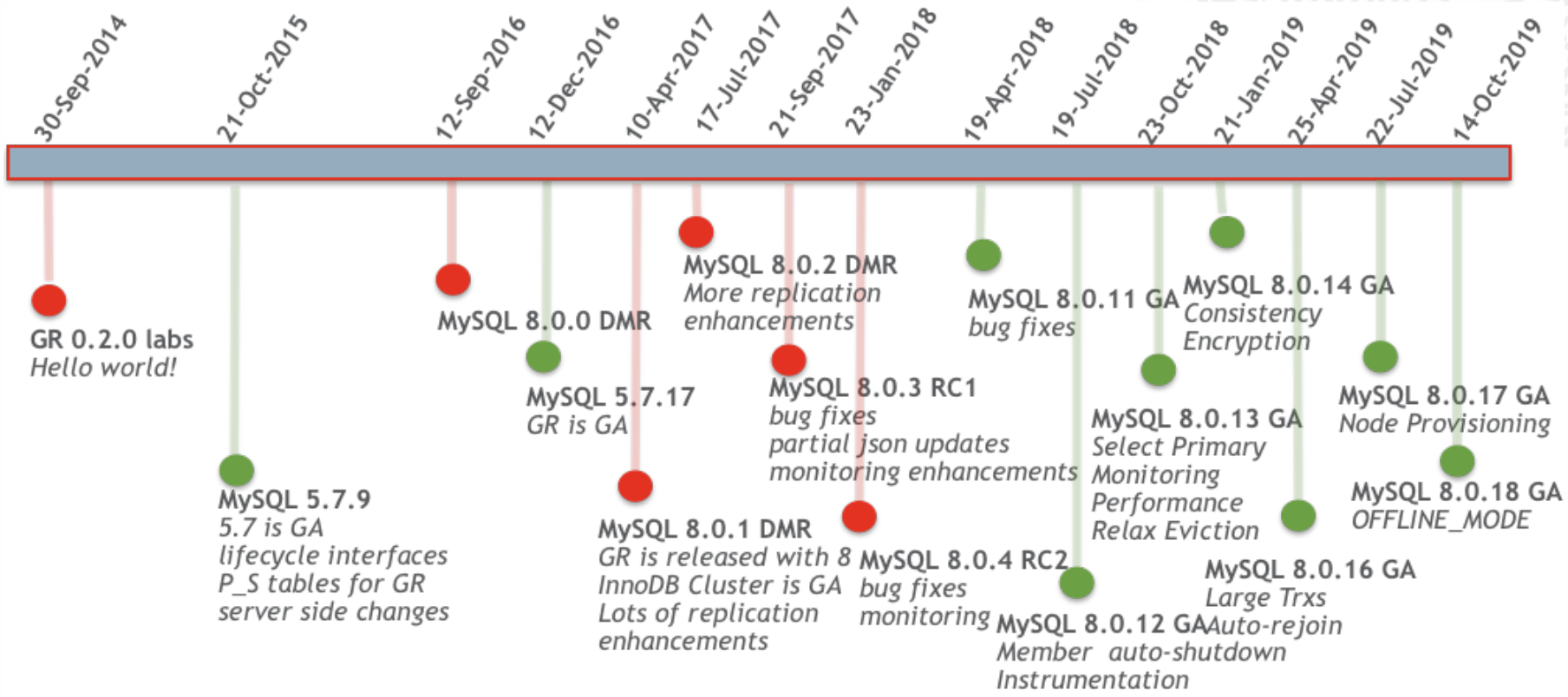
```
$ sudo mysqlrouter --user=mysqlrouter --bootstrap
$ sudo systemctl start mysqlrouter
```

MySQL Shell - Easy to Use

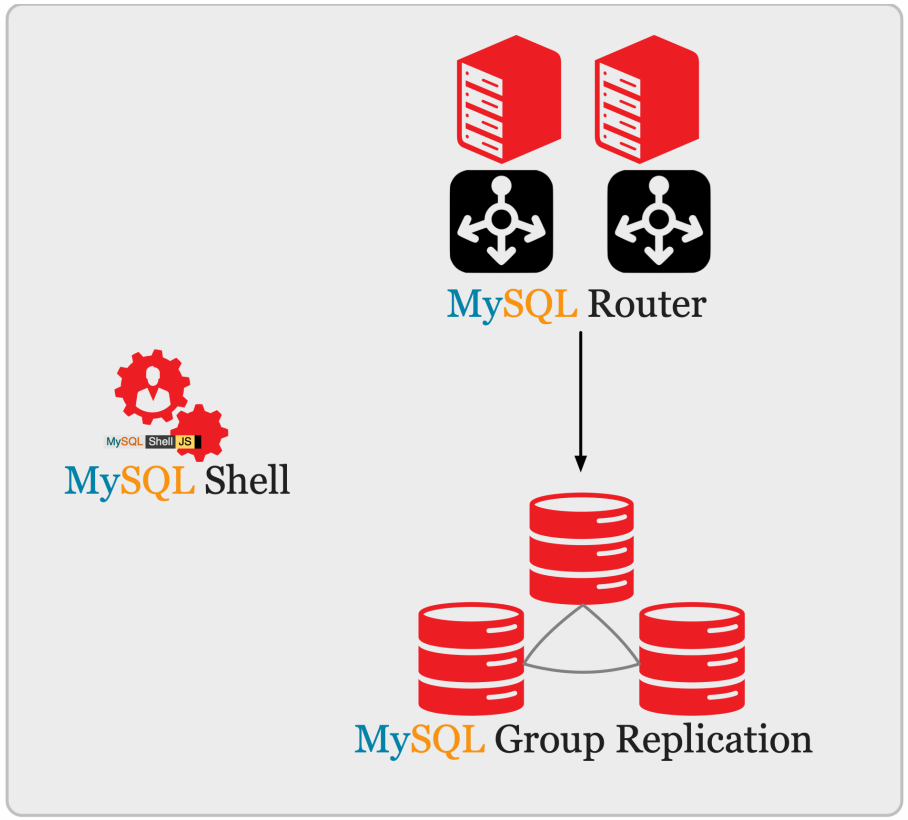
Check the Cluster status:

```
mysql-js> cluster.status()
{
    "clusterName": "cluster",
    "defaultReplicaSet": {
        "name": "default",
        "primary": "mysql1:3306",
        "ssl": "REQUIRED",
        "status": "OK",
        "statusText": "Cluster is ONLINE and can
                       tolerate up to ONE failure.",
        "topology": {
            "mysql1:3306": {
                "address": "mysql1:3306",
                "mode": "R/W",
                "readReplicas": {},
                "role": "HA",
                "status": "ONLINE"
            },
            "mysql2:3306": {
                "address": "mysql2:3306",
                "mode": "R/O",
                "readReplicas": {},
                "role": "HA",
                "status": "ONLINE"
            },
            "mysql3:3306": {
                "address": "mysql3:3306",
                "mode": "R/O",
                "readReplicas": {},
                "role": "HA",
                "status": "ONLINE"
            }
        }
}
```

MySQL InnoDB Cluster - Evolving



MySQL InnoDB Cluster - Adoption

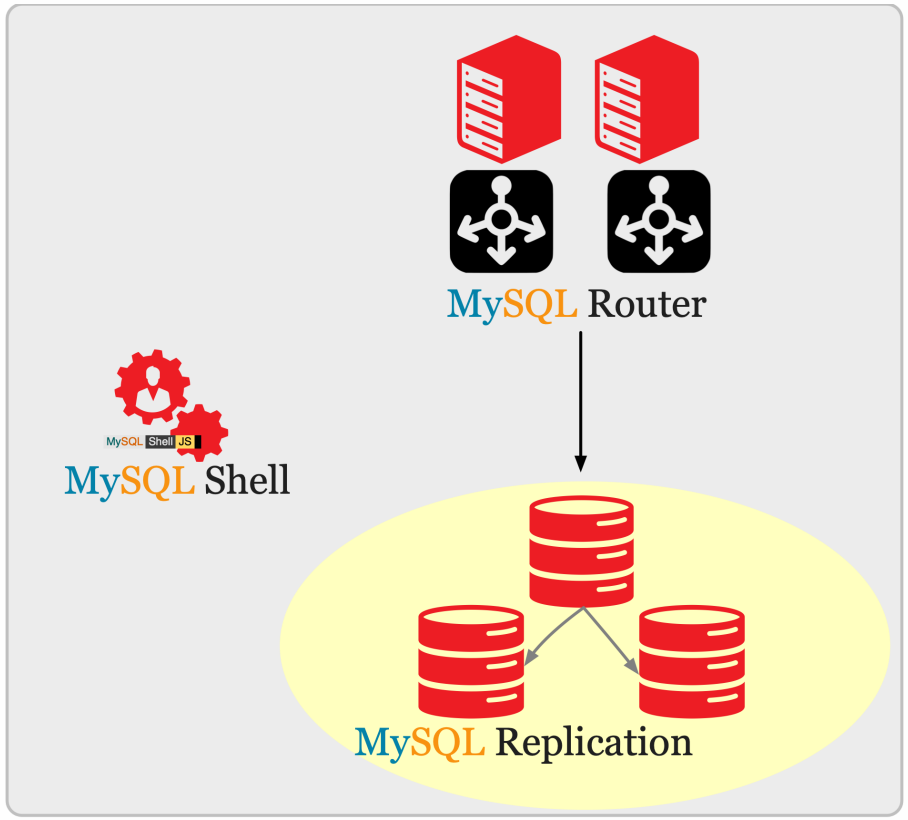


Many of our customers have adopted InnoDB Cluster!

- Fully integrated MySQL Router
  - Automatic Routing
- Ease of use with MySQL Shell
  - Configuring, Adding, Removing members
- Group Replication Architecture
  - Providing Consistency
  - Automatic Failover
  - Network Partition Handling
  - No data loss in case of failure
  - Automatic Member Provisioning (CLONE)

MySQL InnoDB ReplicaSet

MySQL InnoDB ReplicaSet



- Fully integrated MySQL Router
  - Automatic Routing
- Ease of use with MySQL Shell
  - Configuring, Adding, Removing members
  - Automatic Member Provisioning (CLONE)
- Replication Architecture:
  - (manual) Switchover & Failover
  - (asynchronous) Read Scaleout
  - 'Simple' Replication architecture:
    - no network/hardware requirements
      - Provides Availability on PRIMARY during issues with members or network

MySQL InnoDB ReplicaSet - Features

Past

- Restore a backup to provision a member

MySQL InnoDB ReplicaSet

- Automatically provisioning new members: InnoDB CLONE

MySQL InnoDB ReplicaSet - Features

Past

- Restore a backup to provision a member

- Configure Replication Users
- Configure Replication

MySQL InnoDB ReplicaSet

- Automatically provisioning new members:
  InnoDB CLONE

- MySQL Shell Automatically configures users & replication

MySQL InnoDB ReplicaSet - Features

Past

- Restore a backup to provision a member

- Configure Replication Users
- Configure Replication

MySQL InnoDB ReplicaSet

- Automatically provisioning new members: InnoDB CLONE

- MySQL Shell Automatically configures users & replication

- Manually configuring, adding removing servers in Application, MySQL Router (or other proxy)

- Integrated MySQL Router load balancing
  - Only need to bootstrap Router
  - Router is stateless, adapts to topology changes

MySQL InnoDB ReplicaSet - Features

Past

- Restore a backup to provision a member

- Configure Replication Users
- Configure Replication

- Manually or relying on external tools to make topology changes

  - Easy to use manual switchover/failover

MySQL InnoDB ReplicaSet

- Automatically provisioning new members: InnoDB CLONE

- MySQL Shell Automatically configures users & replication

- Manually configuring, adding removing servers in Application, MySQL Router (or other proxy)

- Integrated MySQL Router load balancing
  - Only need to bootstrap Router
  - Router is stateless, adapts to topology changes

MySQL InnoDB ReplicaSet - Features

Past

MySQL InnoDB ReplicaSet

MySQL InnoDB ReplicaSet - Features

Past

- Use additional monitoring tool log in on all machines to check topology status

MySQL InnoDB ReplicaSet

- See status of the topology through MySQL Shell status()

MySQL InnoDB ReplicaSet - Features

Past

- Use additional monitoring tool log in on all machines to check topology status
- complexity: user is responsible for the full configuration of every component and it's settings

MySQL InnoDB ReplicaSet

- See status of the topology through MySQL Shell `status()`
- Shell configures Server, Router, Replication in a standardized best practice setup, prevents mistakes

MySQL InnoDB ReplicaSet - Features

Past

- Use additional monitoring tool log in on all machines to check topology status
- complexity: user is responsible for the full configuration of every component and it's settings
- every setup is a customized setup

MySQL InnoDB ReplicaSet

- See status of the topology through MySQL Shell `status()`
- Shell configures Server, Router, Replication in a standardized best practice setup, prevents mistakes
- Standard Solution -- Supported & QA'ed by Oracle

MySQL InnoDB ReplicaSet - Features

Past

- Use additional monitoring tool log in on all machines to check topology status
- complexity: user is responsible for the full configuration of every component and it's settings
- every setup is a customized setup
- A lot of manual steps and additional software required, always customized and often overengineered by MySQL DBA's

MySQL InnoDB ReplicaSet

- See status of the topology through MySQL Shell `status()`
- Shell configures Server, Router, Replication in a standardized best practice setup, prevents mistakes
- Standard Solution -- Supported & QA'ed by Oracle
- Easy to use, even for MySQL beginner

MySQL InnoDB ReplicaSet - Requirements & Limitations

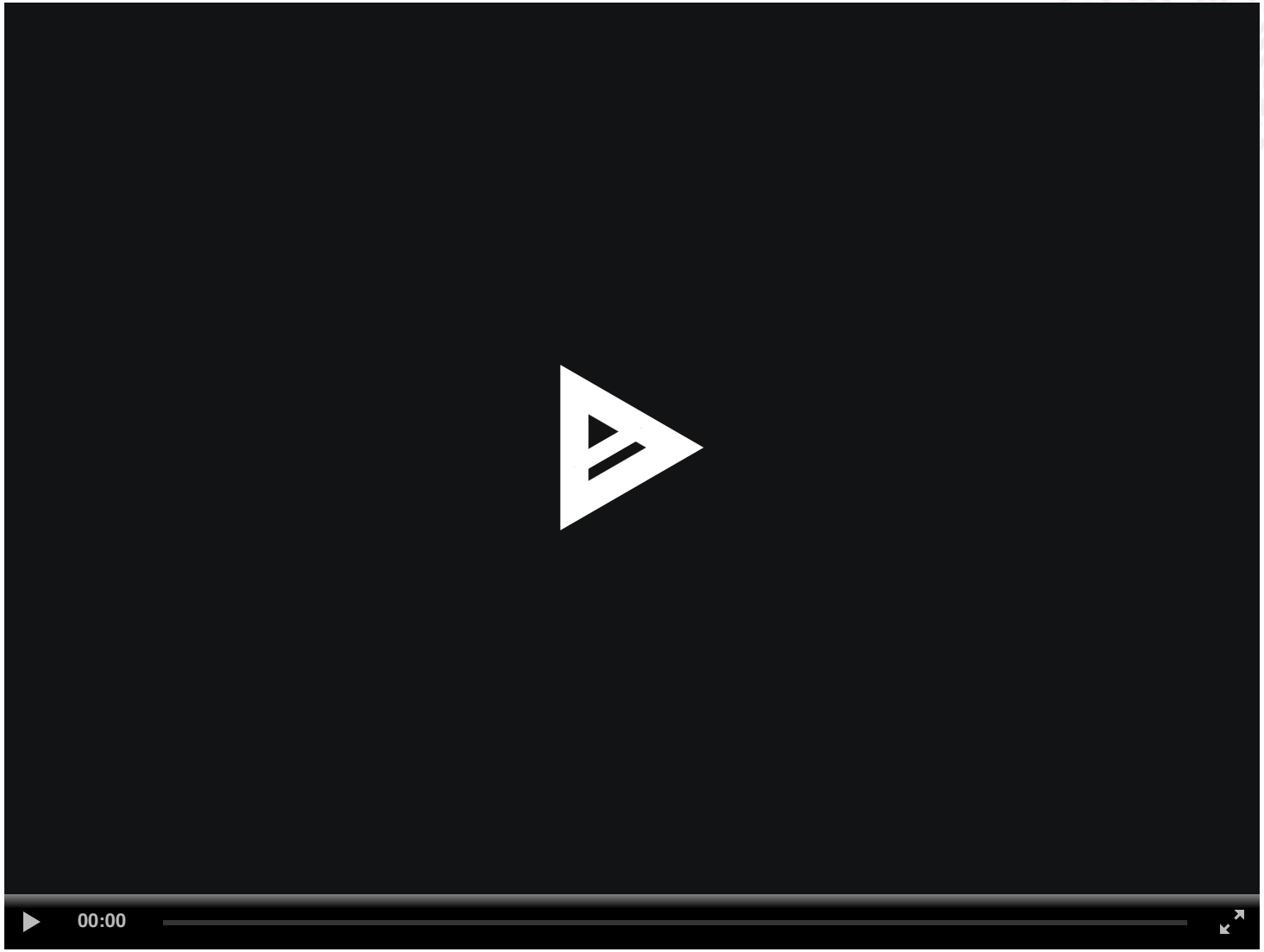
Requirements:

- MySQL 8 (SET PERSIST!)
- GTID

Limitations:

- Manual failover
- No multi-primary as such topology cannot guarantee data consistency
- All secondary members replicate from primary

MySQL InnoDB ReplicaSet - Demo



MySQL Shell - Easy to Use

MySQL Shell - Easy to Use

MySQL InnoDB Cluster

```
mysql-js> \c admin@mysql1
mysql-js> cluster = dba.createCluster('cluster')
```

MySQL InnoDB ReplicaSet

```
mysql-js> \c admin@mysql1
mysql-js> rs = dba.createReplicaSet('replicaset')
```


MySQL Shell - Easy to Use

MySQL InnoDB Cluster

```
mysql-js> \c admin@mysql1
mysql-js> cluster = dba.createCluster('cluster')
```

MySQL InnoDB ReplicaSet

```
mysql-js> \c admin@mysql1
mysql-js> rs = dba.createReplicaSet('replicaset')
```

Configure server to add later

```
mysql-js> dba.configureInstance('admin@mysql2')
```

```
mysql-js> dba.configureReplicaSetInstance('admin@mysql2')
```


MySQL Shell - Easy to Use

MySQL InnoDB Cluster

```
mysql-js> \c admin@mysql1
mysql-js> cluster = dba.createCluster('cluster')
```

Configure server to add later

```
mysql-js> dba.configureInstance('admin@mysql2')
```

Add server to the Cluster

```
mysql-js> cluster.addInstance('admin@mysql2')
```

MySQL InnoDB ReplicaSet

```
mysql-js> \c admin@mysql1
mysql-js> rs = dba.createReplicaSet('replicaset')
```

```
mysql-js> dba.configureReplicaSetInstance('admin@mysql2')
```

```
mysql-js> rs.addInstance('admin@mysql2')
```


MySQL Shell - Easy to Use

MySQL InnoDB Cluster

```
mysql-js> \c admin@mysql1
mysql-js> cluster = dba.createCluster('cluster')
```

Configure server to add later

```
mysql-js> dba.configureInstance('admin@mysql2')
```

Add server to the Cluster

```
mysql-js> cluster.addInstance('admin@mysql2')
```

Bootstrap MySQL Router

```
$ sudo mysqlrouter --user=mysqlrouter --bootstrap
$ sudo systemctl start mysqlrouter
```

MySQL InnoDB ReplicaSet

```
mysql-js> \c admin@mysql1
mysql-js> rs = dba.createReplicaSet('replicaset')
```

```
mysql-js> dba.configureReplicaSetInstance('admin@mysql2')
```

```
mysql-js> rs.addInstance('admin@mysql2')
```

```
$ sudo mysqlrouter --user=mysqlrouter --bootstrap
$ sudo systemctl start mysqlrouter
```

Demo: https://mysqlserverteam.com/introducing-mysql-innodb-replicaset

What does your business need?

Business Requirements

Recovery Time Objective (RTO)

How fast should the service recover from a failure

Recovery Point Objective (RPO)

How much data loss can the service lose from a failure

Business Requirements

Recovery Time Objective (RTO)

How fast should the service recover from a failure

Recovery Point Objective (RPO)

How much data loss can the service lose from a failure

Types of Failures

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

How Much

- None
- few seconds
- minutes
- hours
- day
- ...

Which Solution fits my business?

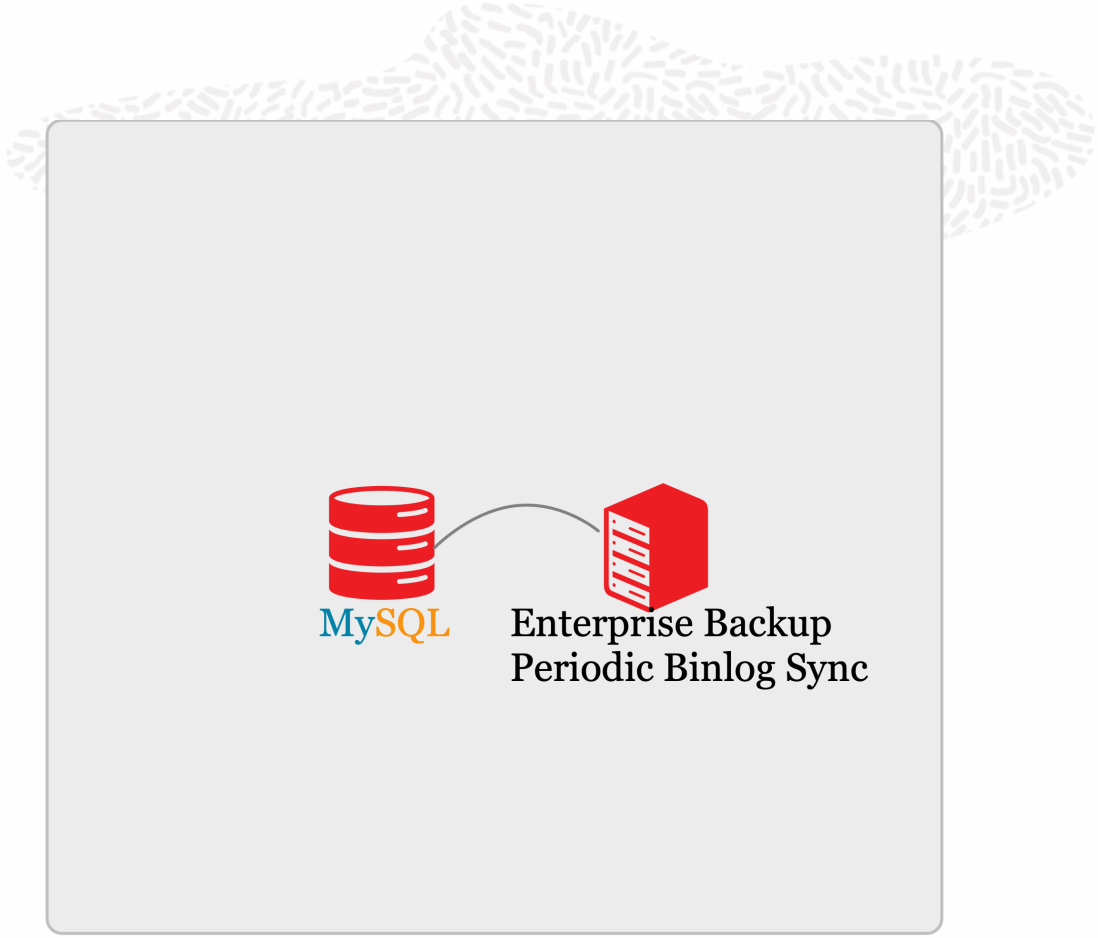
Single Region

Single Region

- RTO = hours
- RPO = minutes

Single MySQL Server

- with backups
- syncing binary logs (change stream) to other host

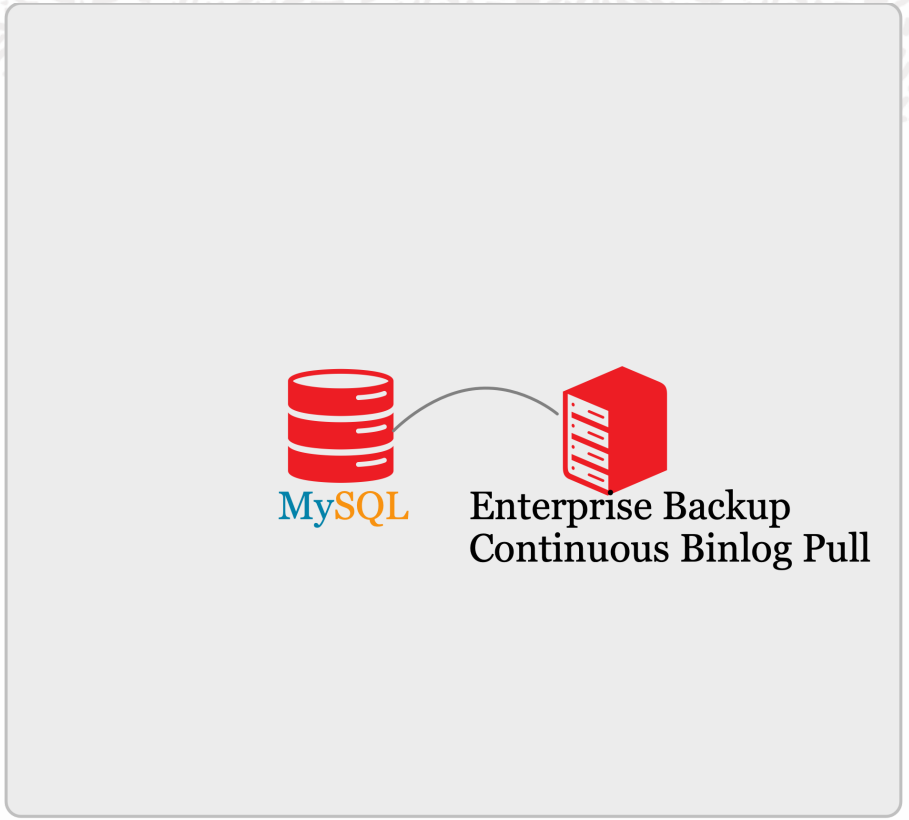


Single Region

- RTO = hours
- RPO = less than a second

Single MySQL Server

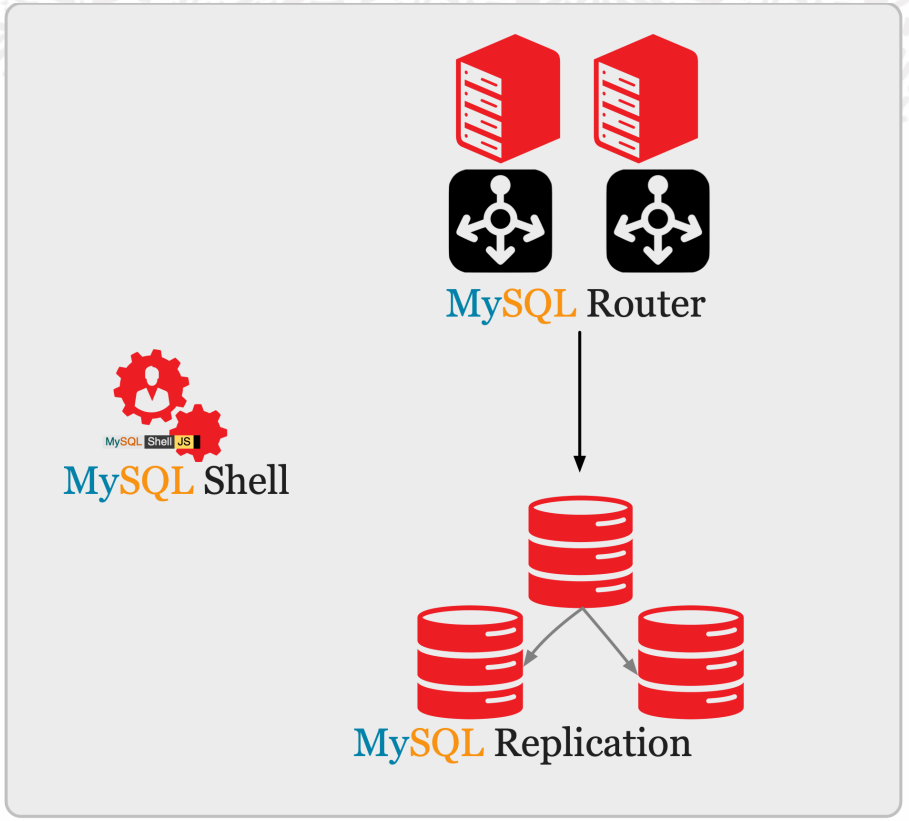
- with frequent backups
- remote binary log pulling (point in time recovery)



Single Region

- RTO = minutes
- RPO = less than a second

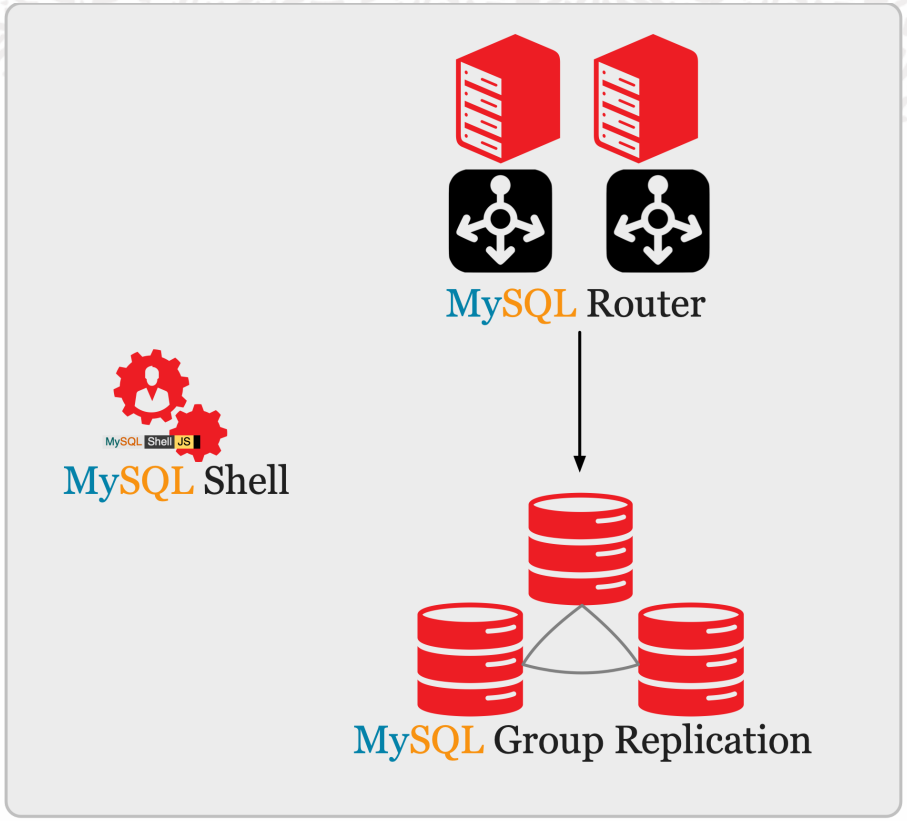
MySQL InnoDB ReplicaSet



Single Region

- RTO = seconds
- RPO = 0

MySQL InnoDB Cluster



Which Solution fits my business?

Multi Region

Multi Region

Region Failure:

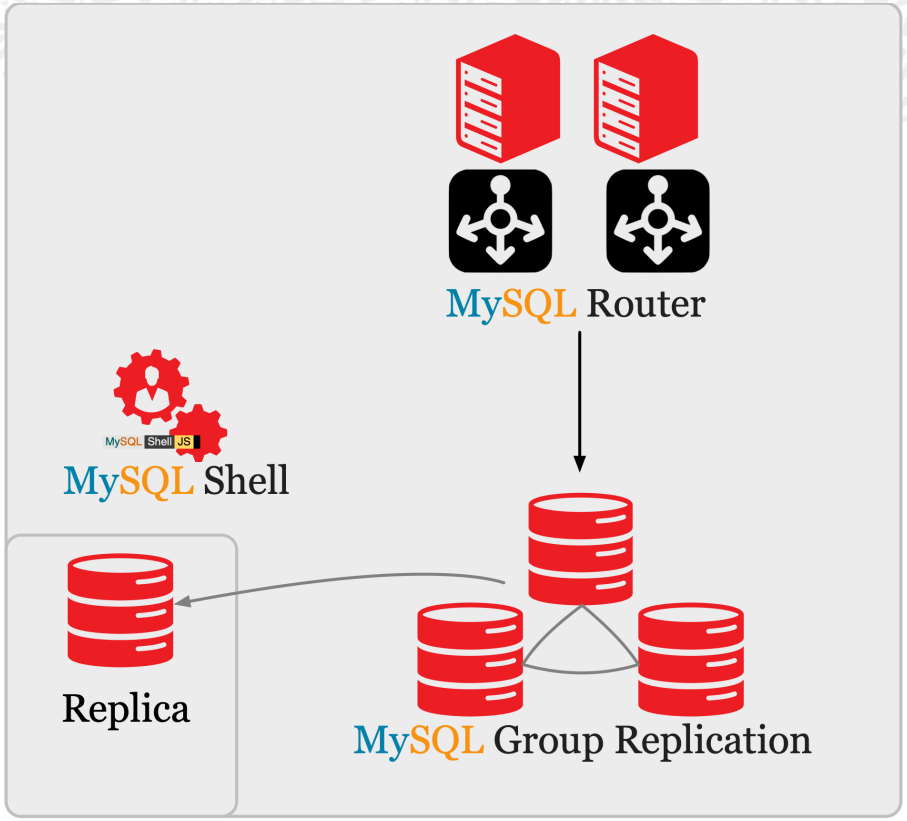
- RTO = minutes,
- RPO = seconds

MySQL InnoDB Cluster:

- with asynchronous Replica

Note:

- no write throughput impact on write transactions



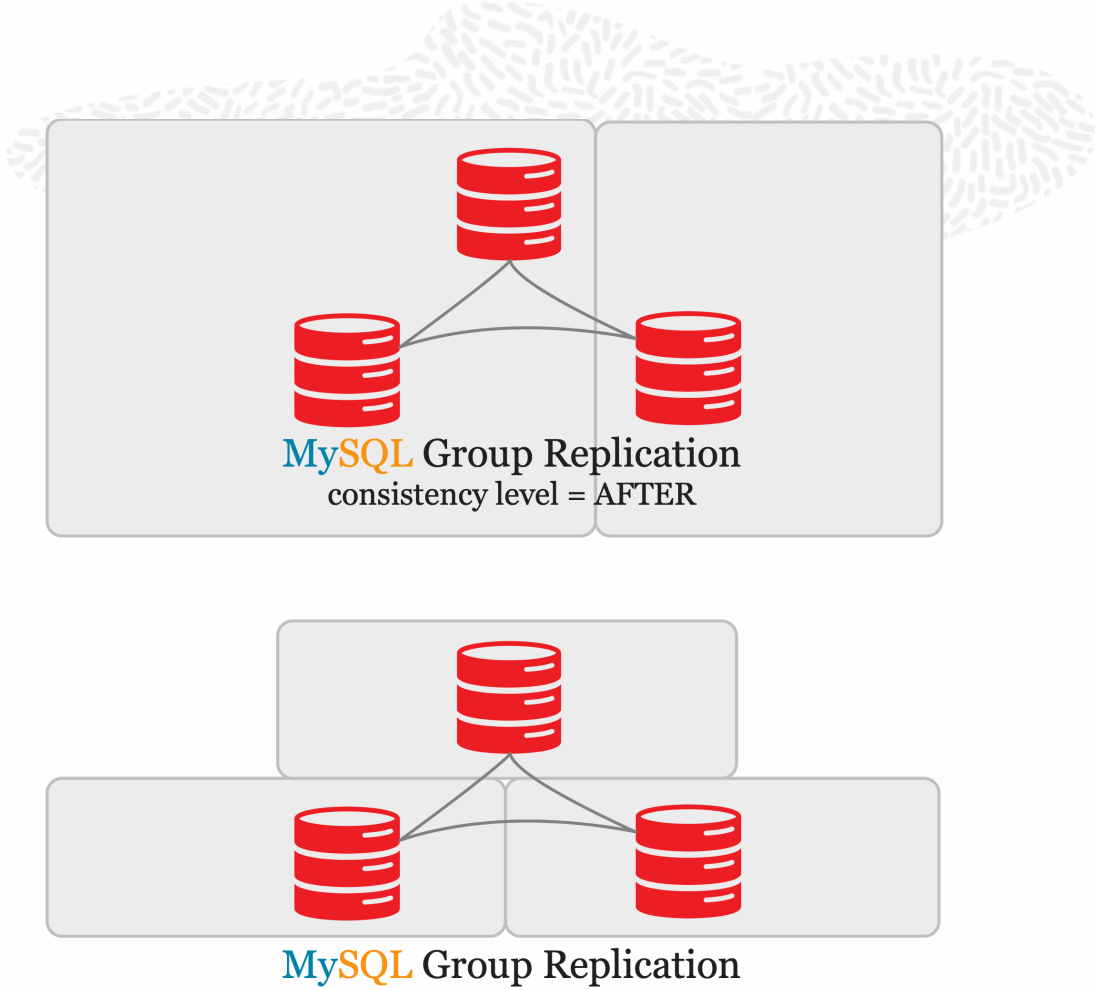
Multi Region

Region Failure

- RTO = seconds and/or
- Region Failure RPO = 0

MySQL InnoDB Cluster deployed accross multiple regions

- consistency level AFTER with 2 regions
  OR 3 regions with each region having 1-3 members.
- write throughput impact on write transactions to guarantee transactions are synced

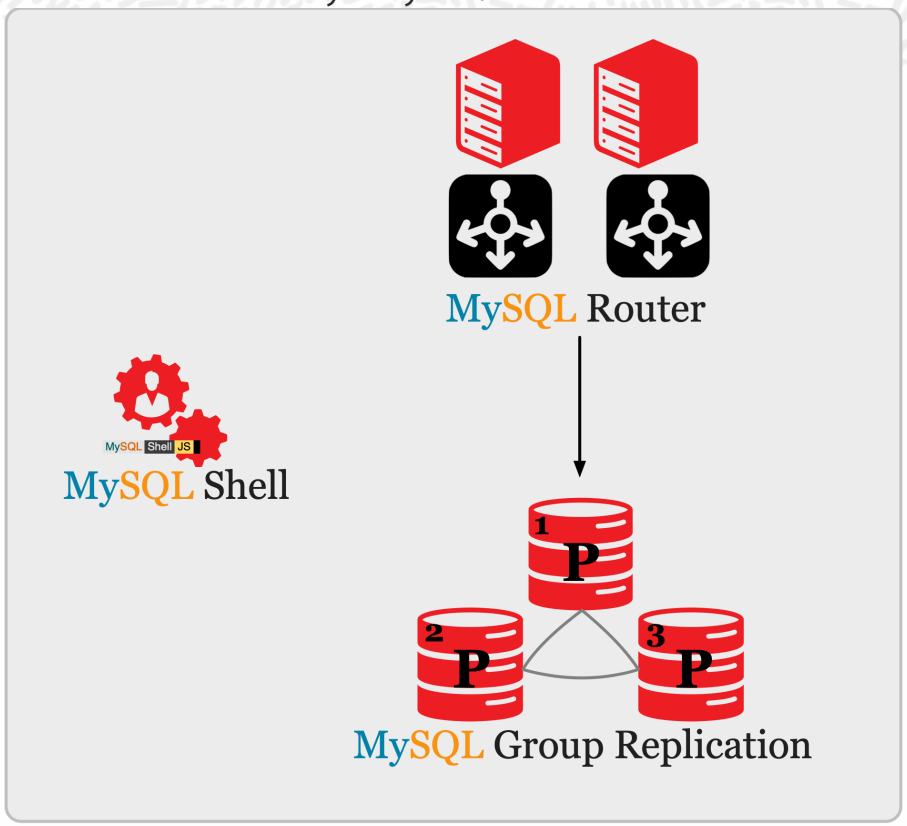


Which Solution fits my business?

MySQL InnoDB Cluster Other Requirements

MySQL InnoDB Cluster Other Requirements

- Fully Consistent Reads
- Multi Primary Single Region
- Multi Primary Multi Region



MySQL Database Architectures

Summary

MySQL Database Architectures

Single Region

Requirement	Solution
RTO = hours, RPO = minutes	MySQL Server w. Backups & Binary Log Sync
RTO = hours, RPO = less than a second	MySQL Server w. Backups & Binary Log Stream
RTO = minutes, RPO = less than a second	MySQL InnoDB ReplicaSet
RTO = seconds, RPO = 0	MySQL InnoDB Cluster

Multi Region

Requirement	Solution
RTO = minutes, RPO = seconds	MySQL InnoDB Cluster w. asynchronous replica
RTO = seconds and/or RPO = 0	Multi Region MySQL InnoDB Cluster w: - 2 region: consistency level AFTER - 3 region deployment

MySQL Database Architectures

Other Requirements

Requirement	Solution
Fully Consistent Reads	MySQL InnoDB Cluster w. custom consistency levels
Multi Primary Single Region	MySQL InnoDB Cluster
Multi Primary Multi Region	Multi Region MySQL InnoDB Cluster

Q&A

ORACLE

Up Next

Panel: SQL or NoSQL? Schema or Schemaless?

Morgan Tocker, Markus Winand,
Bill Karwin, Moderator: Frédéric Descamps

MySQL Performance Tuning

Jesper Wisborg Krogh

Please visit the website for event agenda:
http://developer.oracle.com/devlivemysql

Click the join button on the agenda to join either of the upcoming sessions.

#OracleDevLive

ORACLE

Up Next

Panel: SQL or NoSQL? Schema or Schemaless?

Morgan Tocker, Markus Winand,
Bill Karwin, Moderator: Frédéric Descamps

MySQL Shell: The Best MySQL DBA Tool

Miguel Araújo

Please visit the website for event agenda:
http://developer.oracle.com/devlivemysql

Click the join button on the agenda to join either of the upcoming sessions.

#OracleDevLive