Disaster Recovery Solutions
MySQL InnoDB ClusterSet

Kenny Gryp
MySQL Product Manager

Miguel Araújo
MySQL Software Engineer
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.
IT Disasters & Outages: Primary Causes

On-site power failure is the biggest cause of significant outages
Over half who had experienced an outage costing more than $100,000.
IT Disasters and Outages: Examples

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.

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

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

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, Present & Future
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!

2016 - **MySQL InnoDB Cluster**

- **MySQL Group Replication**: Automatic membership changes, network partition handling, consistency...
- **MySQL Shell**: Provides a powerful interface that helps in automating and integrating all components
- **InnoDB CLONE**: Automatically provisions members, fully integrated in InnoDB
- **MySQL Router**
- **MySQL Server**

RPO = 0
RTO = seconds (automatic failover)
Present - Solutions!

2020 - **MySQL InnoDB Replicaset**
- 'classic', 'asynchronous' Replication based Solution, fully integrated
- **MySQL Shell**
- **MySQL Router**
- **MySQL Server**

RPO != 0
RTO = minutes (manual failover)
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/clusters online
- Router integration, no need to reconfigure application if the topology changes
MySQL InnoDB ClusterSet - 3 Datacenters
MySQL InnoDB ClusterSet - Not every Cluster has to be 3 nodes

Each replica is a MySQL InnoDB Cluster that can have 1-9 members.
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 Demo
ClusterSet Demo

Environment, 3 regions, 3 mysql databases each, listening on different ports:

- ROM:
  - rome:3331
  - rome:3332
  - rome:3333

- BRU:
  - brussels:4441
  - brussels:4442
  - brussels:4443

- LIS:
  - lisbon:5551
  - lisbon:5552
  - lisbon:5553

Commands used in demo available on https://github.com/miguelaraujo/ClusterSet-Demo
Demo

Initial Setup

- Create MySQL InnoDB Cluster
- Create ClusterSet with 3 clusters
- ClusterSet Status
- Router Bootstrap

Change PRIMARYs

- Change PRIMARY member in PRIMARY cluster
- Change PRIMARY member in REPLICA cluster
- Change PRIMARY Cluster - setPrimaryCluster()
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 - `setPrimaryCluster()`

- 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)
Demo

Initial Setup

- Create MySQL InnoDB Cluster
- Create ClusterSet with 3 clusters
- ClusterSet Status
- Router Bootstrap

Change PRIMARYs

- Change PRIMARY member in PRIMARY cluster
- Change PRIMARY member in REPLICA cluster
- Change PRIMARY Cluster - setPrimaryCluster()
Configure your application to connect to a local MySQL Router to connect to the ClusterSet.
Router Integration

Router Target Modes:

- follow the PRIMARY cluster
  - Writes & Reads go to the PRIMARY Cluster
- connect to the configured target cluster
  - When target cluster is not PRIMARY:
    - only read traffic is open
    - writes will be denied
  - when target cluster is PRIMARY
    - write port opens

Features:

- Configurable per Router instance
- Configuration can be changed ONLINE in mysqlsh
- Deploy 2 types of routers:
  - target PRIMARY to send writes to PRIMARY
  - define target cluster to keep read traffic local
- INVALIDATED clusters can still be used for read traffic (configurable)
Demo

Initial Setup

- Create MySQL InnoDB Cluster
- Create ClusterSet with 3 clusters
- ClusterSet Status
- Router Bootstrap

Change PRIMARYs

- Change PRIMARY member in PRIMARY cluster
- Change PRIMARY member in REPLICA cluster
- Change PRIMARY Cluster - setPrimaryCluster()

Router

- Changing Router Configuration Options
- Router Status with Cluster changes
- Router Logs

Failure Scenarios

- Automatic Handling of PRIMARY member in PRIMARY cluster
- Automatic Handling of PRIMARY member in REPLICA cluster
- Disaster - PRIMARY Cluster Failure - Failover
- Bring back INVALIDATED Cluster
- Rejoin INVALIDATED Cluster to ClusterSet
ClusterSet Scenarios
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
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
Failover to another Cluster

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

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
Failover to another Cluster

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

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
Router Integration

- Routers will learn about new topology and redirect traffic
- Routers that come back, will learn about new topology and abandon the old PRIMARY Cluster (e.g. failed DC comes back online)
Datacenter Crash/Partition - Multiple REPLICA clusters Support
Demo

Initial Setup

- Create MySQL InnoDB Cluster
- Create ClusterSet with 3 clusters
- ClusterSet Status
- Router Bootstrap

Change PRIMARYs

- Change PRIMARY member in PRIMARY cluster
- Change PRIMARY member in REPLICA cluster
- Change PRIMARY Cluster - setPrimaryCluster()