ORACLE

Observing and Operating Group Replication

A continuation from last year's session! New network metrics and more!

Luís Soares
Senior Software Development Director
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.

Who am I?

Luís Soares
MySQL Replication Team Lead
Oracle

- Born and raised in Portugal
- Sports: Football, Basket, Karate, Running, Biking
- Physics, Astronomy
- Fault-Tolerance, High Availability, Computers
- Read, Travel, Being with People
- Long time MySQLer

Agenda

- Introduction
- Observe
- Automate
- Operate
- MySQL Heatwave Service
- Conclusion

Introduction

MySQL Heatwave Service

Observe, Automate, Operate

MySQL Replication powers key features like, but not limited to:

- High Availability
- Inbound Replication
- Outbound Replication
- Managed Read Replicas
- Point-in-time Recovery

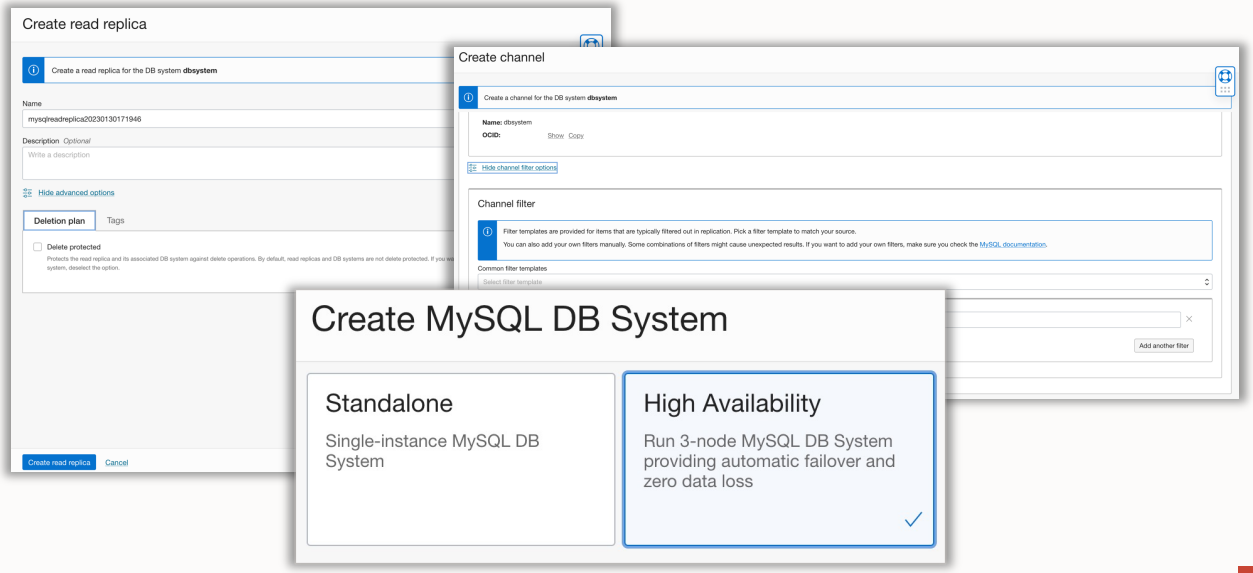
Simple, intuitive, one-click operations:

- Create DB Systems
- Create Read Replica
- Create Inbound Channel

Requires a solid, stable and scale aware framework behind it.

The need to run, monitor and operate:

- At scale
- Exposed to heterogeneous workloads
- Coping with network bursts or packet delays
- Dealing with the “world” splitting
- Through maintenance



MySQL Everywhere

Observe, Automate, Operate

- Not only on the the MySQL Heatwave Service
  - MySQL is deployed everywhere with similar requirements

- The toolset makes it remarkably easier
  - InnoDB Cluster (HA, resilient, fault-tolerant)
  - InnoDB ReplicaSet (Asynchronous)
  - InnoDB ClusterSet (Across clusters, Across regions)

- **There is still, the need to run, monitor and operate:**
  - Possibly at scale
  - Exposed to (potentially bounded but still) heterogeneous workloads
  - Coping with unstable network
  - Dealing with the multiple versions of the world
  - While doing maintenance

MySQL Everywhere

Observe, Automate, Operate

Observe

- Instrument, emit
- Learn, understand, diagnose
- Trends and historical data
- Robots first, then Humans

Automate

- Balance
- Predict
- Self-heal
- Stabilize

Operate

- Plan
- Troubleshoot
- Press buttons, turn nods, flip switches

A continuation from last year…

Yes, this is a continuation from last year's session.

You can find the slides from last year here.

Observe

Network, Paxos and other Protocols Metrics

Servers exchange messages:

- Data (transactions)
- Control Messages (the rest)

XCom (Paxos) for distributed coordination.

- Roles: Leader, Acceptor, Learner
- Terms/Slots/Rounds/Ballots
- Phase 1 (Leader Election)
- Phase 2 (Acceptance)
- Phase 3 (Learn/Dissemination)
- Optimizations:
  - **Multi-Paxos** (accept phase on stable leader)
  - **Mencius** (streams, empty proposals/no-ops)

Proposing

- Empty proposals may exist to make progress
  - All servers propose (multi-leader, default)
    - Nothing to propose → empty proposal
  - One server proposes (single leader)

Agreeing

- **Fast**: accept/learn
  - Stable and reliable network; stable servers; fast servers; non-thrashing hosts
- **Slower, extra phase**: prepare, accept/learn
  - Unstable network; packet loss; jitter; high latency; packet bursts; unstable servers;

Network, Paxos and other Protocols Metrics

There is in an excellent session, delivered at this event too sometime ago, on this topic alone:

"**Group Replication: A Journey to the Group Communication Core**" (slides)

And there is a great blog post about it too:

"**The king is dead, long live the king": Our Paxos-based consensus**"

Network, Paxos and other Protocols Metrics

New MySQL Status Variables

Data Messages

- Sent counter
- Sent bytes
- Sent round-trip time

Control Messages

- Sent counter
- Sent bytes
- Sent round-trip time

Transactions Commited Everywhere

- Garbage collections count
- Garbage collections time

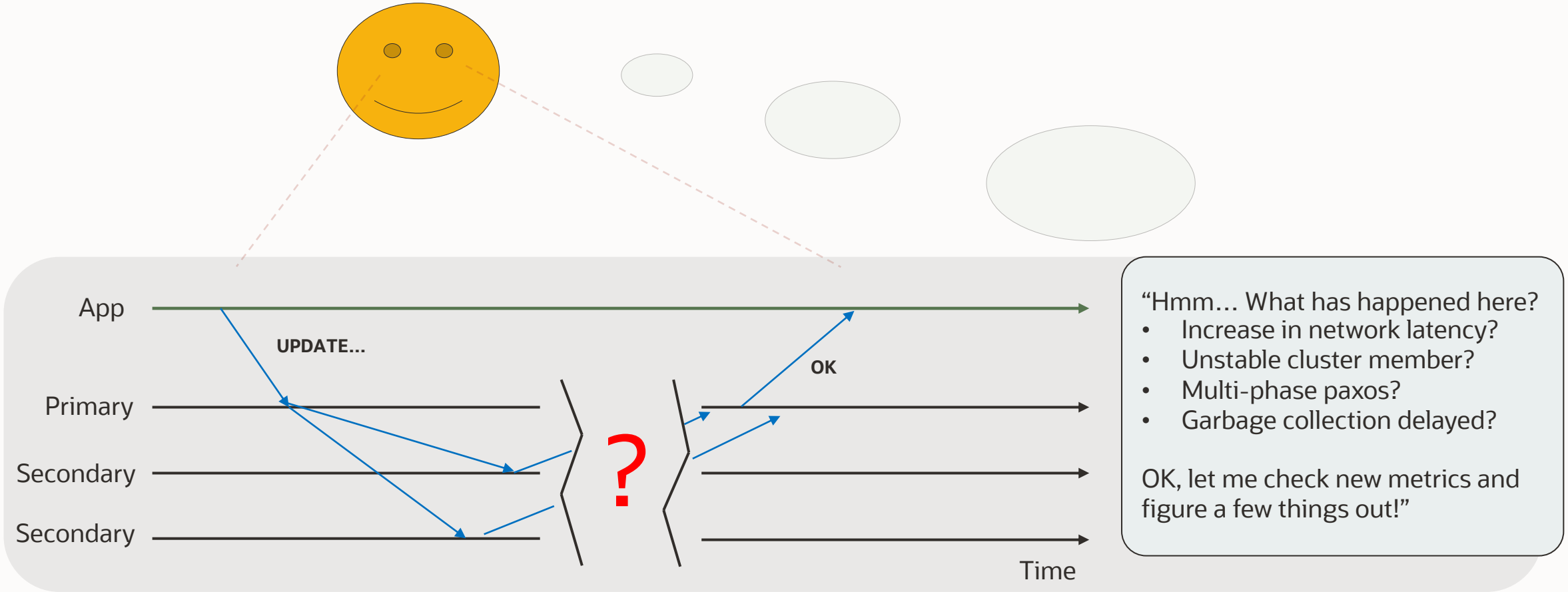
Consensus / Paxos

- Total rounds Time
- Total rounds counter
- Bytes sent
- Bytes received
- 3-phase (slower) rounds count
- Empty proposals (no-ops) rounds count
- Last successful round timestamp

Consistency Protocol Messages

- AFTER sync, wait time and count
- BEFORE sync, wait time and count
- AFTER termination, wait time and count

Network, Paxos and other Protocols Metrics



"Hmm… What has happened here?
- Increase in network latency?
- Unstable cluster member?
- Multi-phase paxos?
- Garbage collection delayed?

OK, let me check new metrics and figure a few things out!"

Detection Metrics

Servers can “disappear” … and maybe even also comeback before a reconfiguration is triggered!

- Suspected as failed, Detected as failed, Indeed failed (and therefore evicted).
- Suspicions can have a cost, since suspected servers may need to propose no-ops.

```
mysql> select MEMBER_ID,MEMBER_HOST,MEMBER_STATE,MEMBER_ROLE from performance_schema.replication_group_members; /* on server 3 */
+--------------------------------------+-------------+--------------+-------------+
| MEMBER_ID                            | MEMBER_HOST | MEMBER_STATE | MEMBER_ROLE |
+--------------------------------------+-------------+--------------+-------------+
| 00…server1                           | 192.168.0.1 | ONLINE       | PRIMARY     |
| 00…server2                           | 192.168.0.2 | UNREACHABLE  | SECONDARY   |
| 00…server3                           | 192.168.0.3 | ONLINE       | SECONDARY   |
+--------------------------------------+-------------+--------------+-------------+
3 rows in set (0.00 sec)
```

Up to MySQL 8.0.X

Detection Metrics

Servers can “disappear” … and maybe even also comeback before a reconfiguration is triggered!

- Suspected as failed, Detected as failed, Indeed failed (and therefore evicted).
- Suspicions can have a cost, since suspected servers may need to propose no-ops.

```
mysql> select MEMBER_ID,MEMBER_HOST,MEMBER_STATE,MEMBER_ROLE from performance_schema.replication_group_members; /* on server 3 */
+--------------------------------------+-------------+--------------+-------------+
| MEMBER_ID                            | MEMBER_HOST | MEMBER_STATE | MEMBER_ROLE |
+--------------------------------------+-------------+--------------+-------------+
| 00…server1                           | 192.168.0.1 | ONLINE       | PRIMARY     |
| 00…server2                           | 192.168.0.2 | UNREACHABLE  | SECONDARY   |
| 00…server3                           | 192.168.0.3 | ONLINE       | SECONDARY   |
+--------------------------------------+-------------+--------------+-------------+
3 rows in set (0.00 sec)
```

Up to MySQL 8.0.X

```
mysql> SELECT *  FROM performance_schema.replication_group_communication_information\G /* on server 3 */
*************************** 1. row ***************************
                    WRITE_CONCURRENCY: 10
                     PROTOCOL_VERSION: 8.0.27
    WRITE_CONSENSUS_LEADERS_PREFERRED: 00…server1,00…server2,00…server3
       WRITE_CONSENSUS_LEADERS_ACTUAL: 00…server1,00…server2,00…server3
WRITE_CONSENSUS_SINGLE_LEADER_CAPABLE: 0
      MEMBER_FAILURE_SUSPICIONS_COUNT: {"00…server1":0,”00…server2":4,”00…server3":0}
1 row in set (0.00 sec).
```

From MySQL 8.1.0 onwards

Detection Metrics

Servers can “disappear” … and maybe even also comeback before a reconfiguration is triggered!

- Suspected as failed, Detected as failed, Indeed failed (and therefore evicted).
- Suspicions can have a cost, since suspected servers may need to propose no-ops.

```
mysql> select MEMBER_ID,MEMBER_HOST,MEMBER_STATE,MEMBER_ROLE from performance_schema.replication_group_members; /* on server 3 */
+--------------------------------------+-------------+--------------+-------------+
| MEMBER_ID                            | MEMBER_HOST | MEMBER_STATE | MEMBER_ROLE |
+--------------------------------------+-------------+--------------+-------------+
| 00…server1                           | 192.168.0.1 | ONLINE       | PRIMARY     |
| 00…server2                           | 192.168.0.2 | UNREACHABLE  | SECONDARY   |
| 00…server3                           | 192.168.0.3 | ONLINE       | SECONDARY   |
+--------------------------------------+-------------+--------------+-------------+
3 rows in set (0.00 sec)
```

```
mysql> SELECT *  FROM performance_schema.replication_group_communication_information\G /* o
*************************** 1. row ***************************
                    WRITE_CONCURRENCY: 10
                     PROTOCOL_VERSION: 8.0.27
    WRITE_CONSENSUS_LEADERS_PREFERRED: 00…server1,00…server2,00…server3
       WRITE_CONSENSUS_LEADERS_ACTUAL: 00…server1,00…server2,00…server3
WRITE_CONSENSUS_SINGLE_LEADER_CAPABLE: 0
      MEMBER_FAILURE_SUSPICIONS_COUNT: {"00…server1":0,”00…server2":4,”00…server3":0}
1 row in set (0.00 sec).
```

“Hmm… What is happening here? Something is not quite right between server3 and server2. Server2 has (transient) troubles speaking to server3!

Wonder if this is causing 3-phase paxos rounds? ”

Detection Metrics

Servers can “disappear” … and maybe even also comeback before a reconfiguration is triggered!

- Suspected as failed, Detected as failed, Indeed failed (and therefore evicted).
- Suspicions can have a cost, since suspected servers may need to propose no-ops.

```
mysql> select * from performance_schema.global_status where variable_name like 'gr_extended%'; /* on server 3 */
+------------------------------+----------------+
| VARIABLE_NAME                | VARIABLE_VALUE |
+------------------------------+----------------+
| Gr_extended_consensus_count | 95             |
+------------------------------+----------------+
1 row in set (0.00 sec)
```

“Indeed, I see the counter growing and therefore seems this server3 is taking over and proposing no-ops on behalf of server2.

I should fix the network connectivity and perhaps consider deploying single leader mode… Could make sense in this scenario to avoid such transient connectivity issues.”

Automate

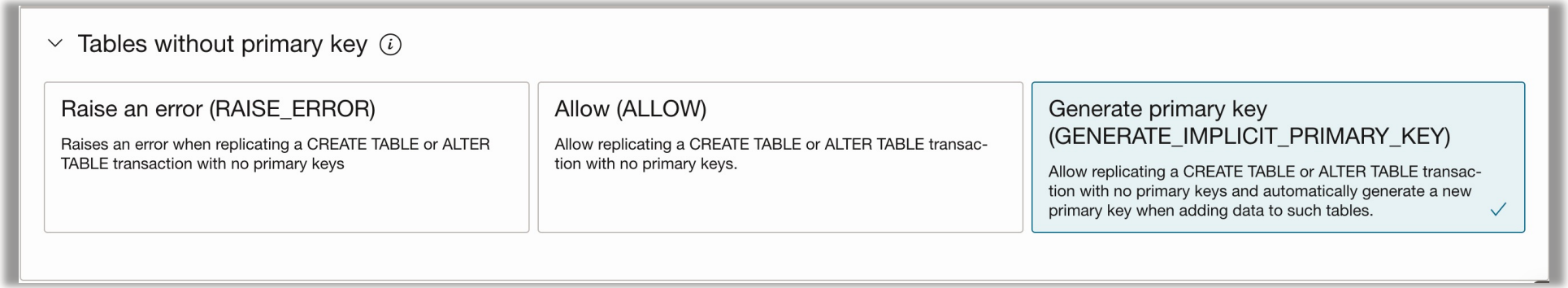
Replication Applier Automatically Adds Primary Keys

Replication Applier Automatically Adds Primary Keys

Replication Applier Automatically Adds Primary Keys

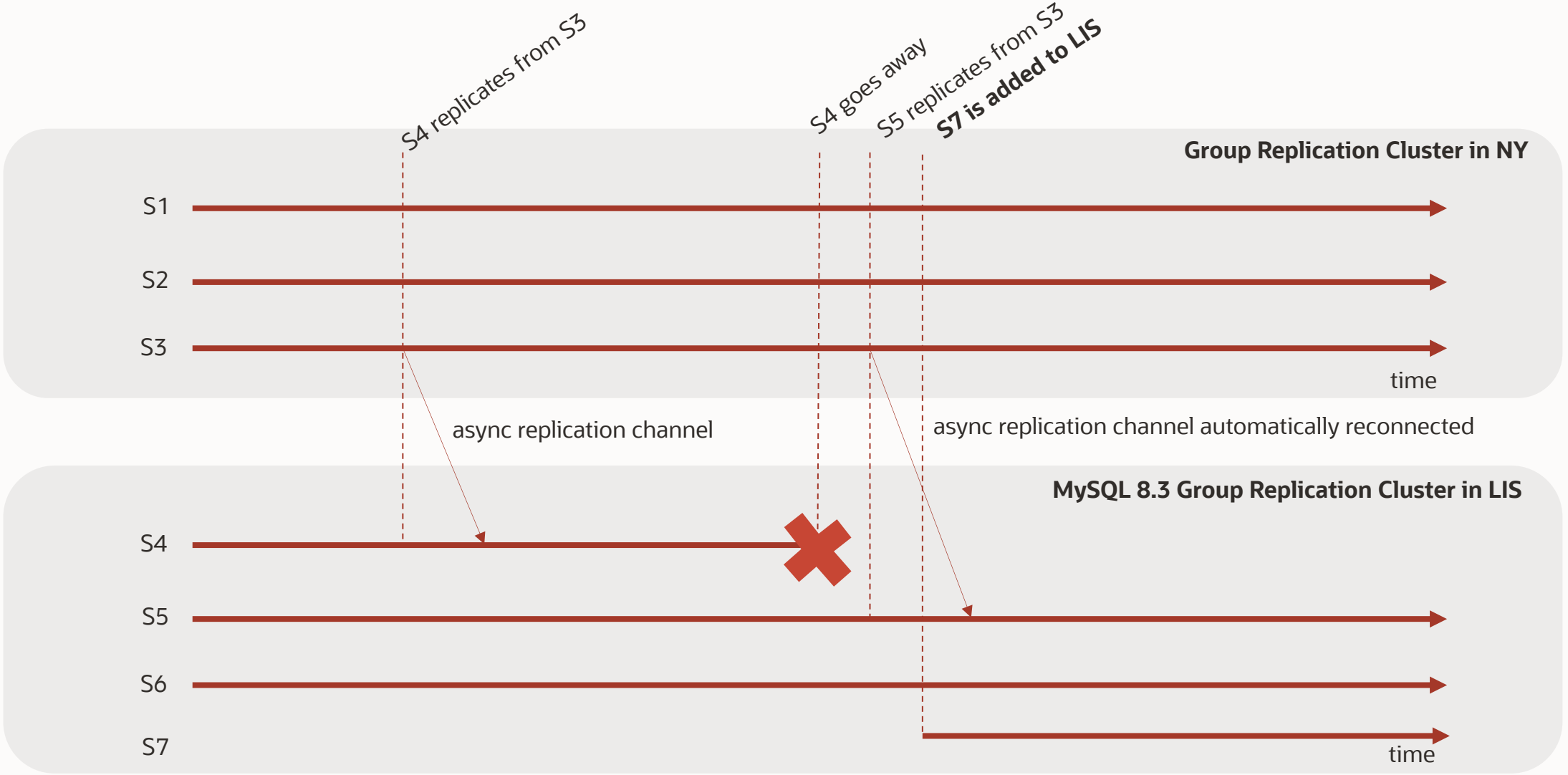
In the MySQL Heatwave Service

- Create an inbound replication channel
- Configure how to handle tables without primary key
- Replicate



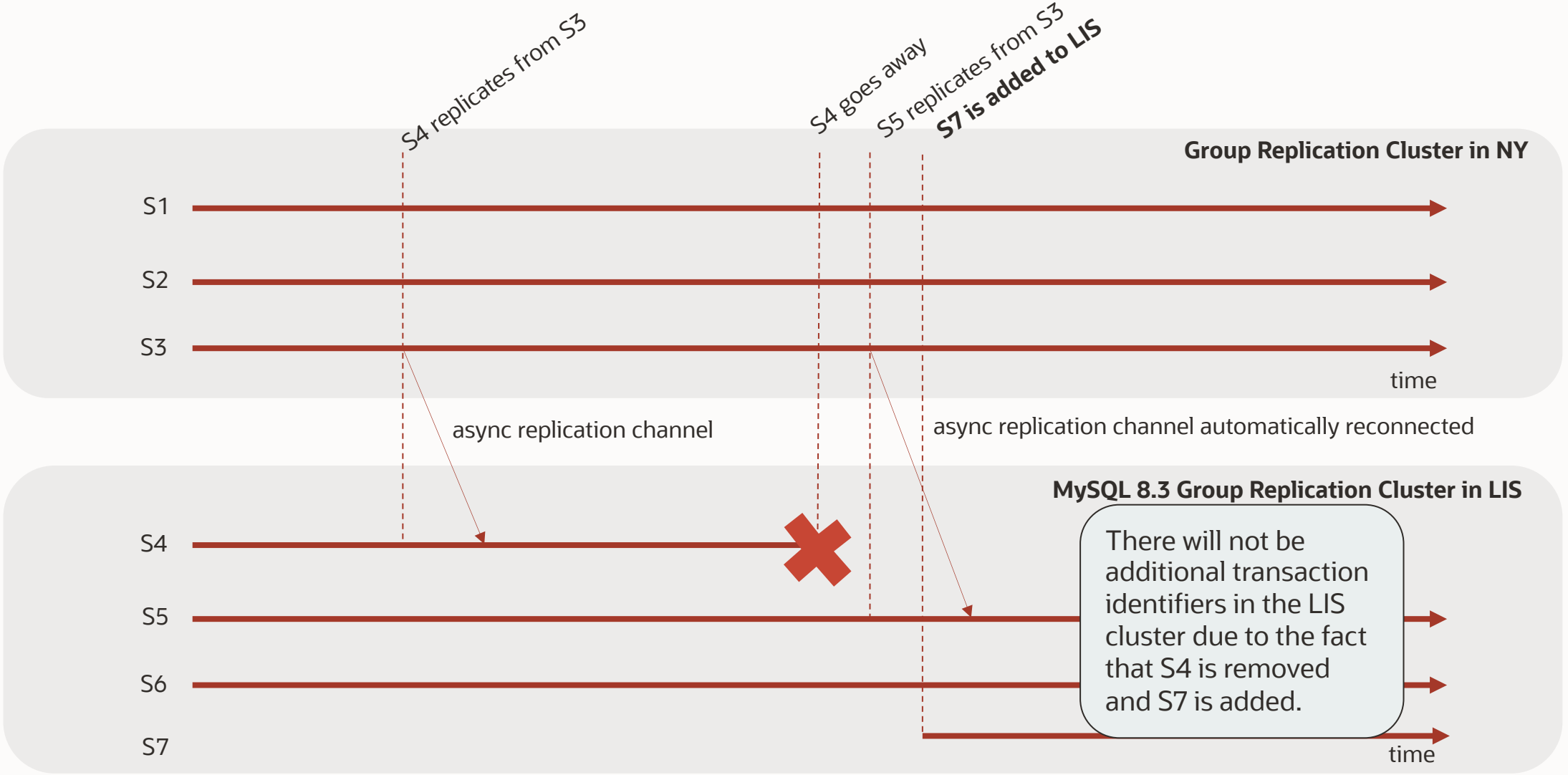
No System Transactions On Cluster Reconfiguration

Further automation for membership changes – view change transactions go away



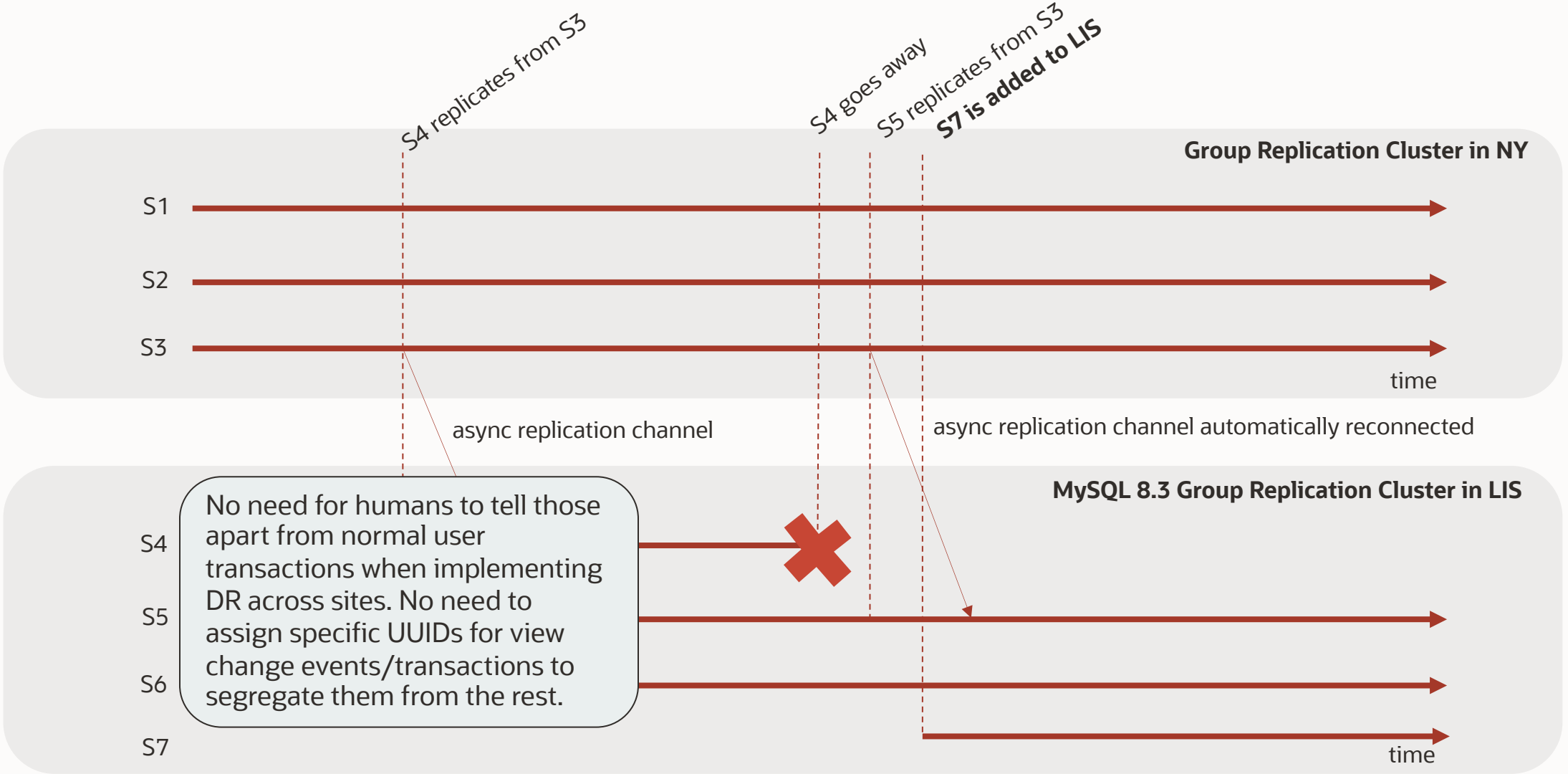
No System Transactions On Cluster Reconfiguration

Further automation for membership changes – view change transactions go away



No System Transactions On Cluster Reconfiguration

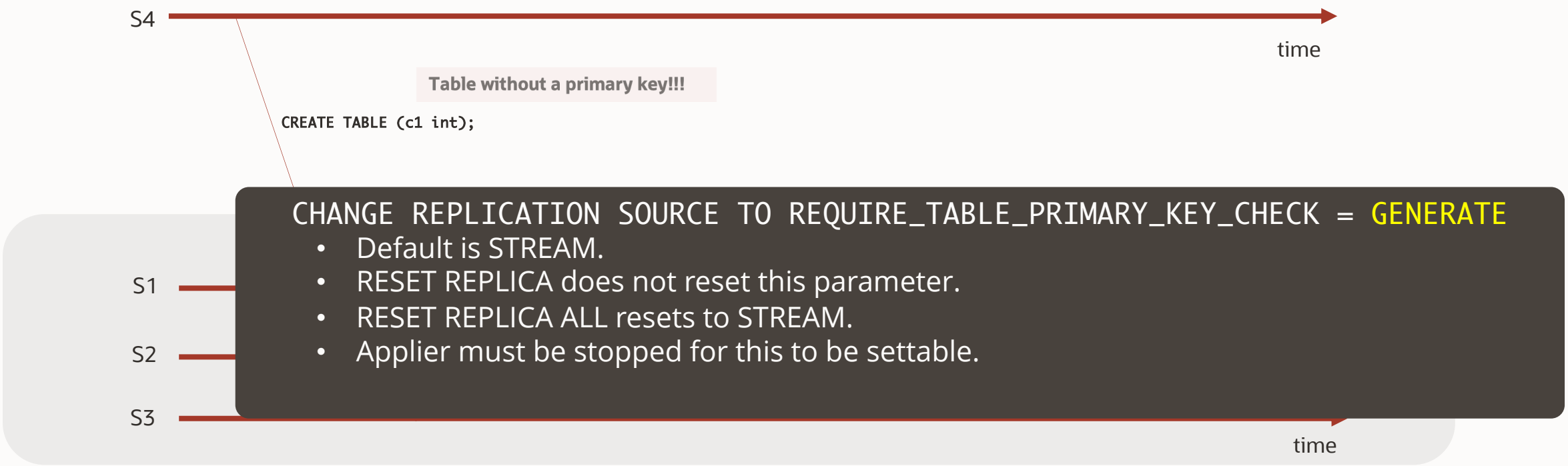
Further automation for membership changes – view change transactions go away



Operate

Control Assignment of Automatic Primary Key Per Channel

Control Assignment of Automatic Primary Key Per Channel



New Syntax

New batch of terminology changes

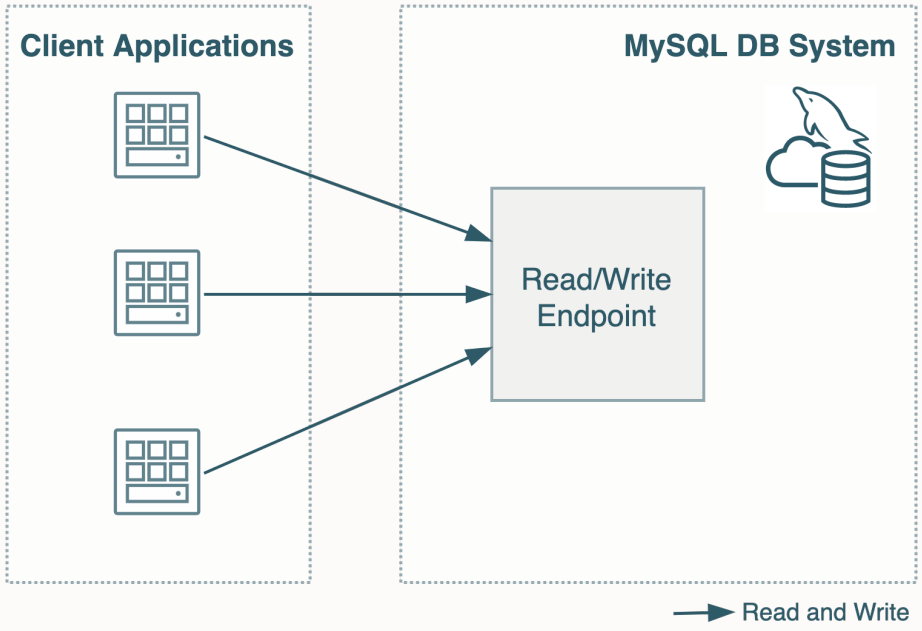
- MySQL continues to improve on the terminology front too.
- Operations may be affected if relying on old and deprecated syntax.
- Note the new syntax.

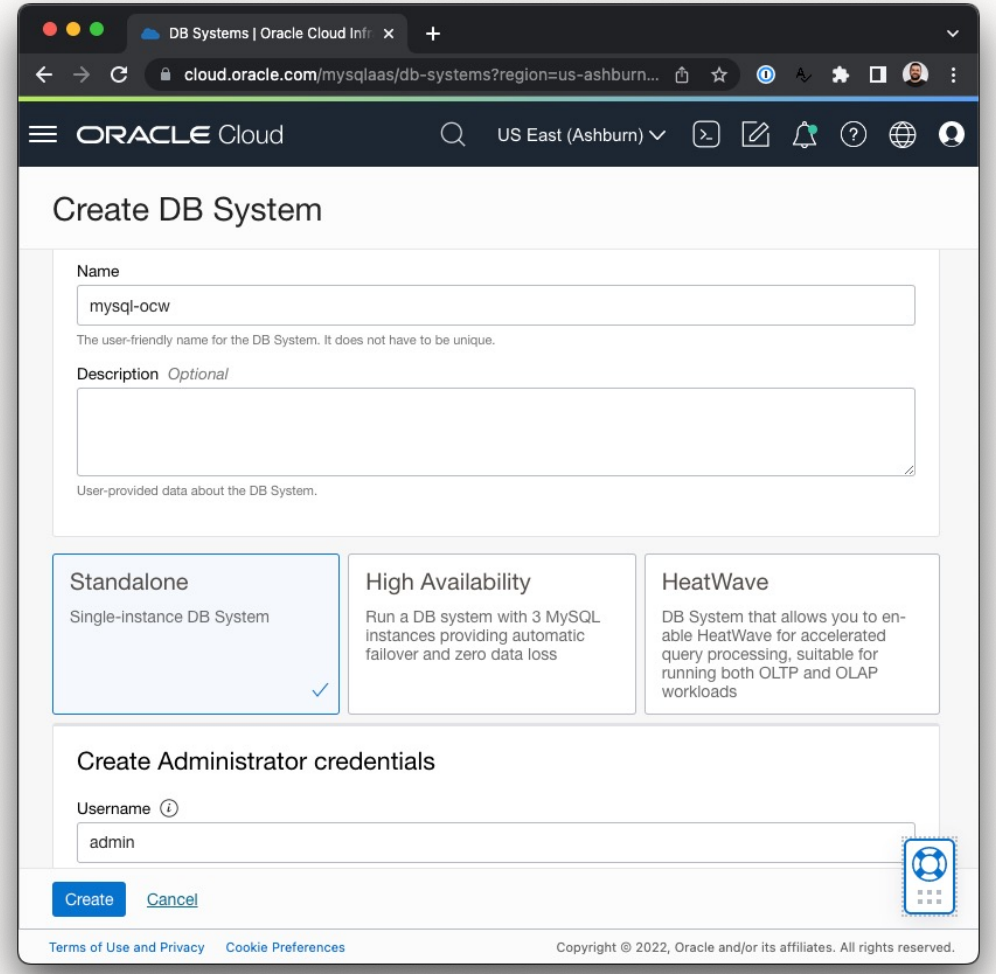
```
mysql> PURGE BINARY LOGS;
-- replaces PURGE MASTER LOGS

mysql> SHOW BINARY LOGS;
-- replaces SHOW MASTER LOGS

mysql> RESET BINARY LOGS AND GTIDS;
-- replaces RESET MASTER

mysql> [CREATE|ALTER] EVENT ... DISABLE ON REPLICA;
-- replaces [CREATE|ALTER] EVENT ... DISABLE ON SLAVE
```

MySQL Heatwave Service

Database
DB System

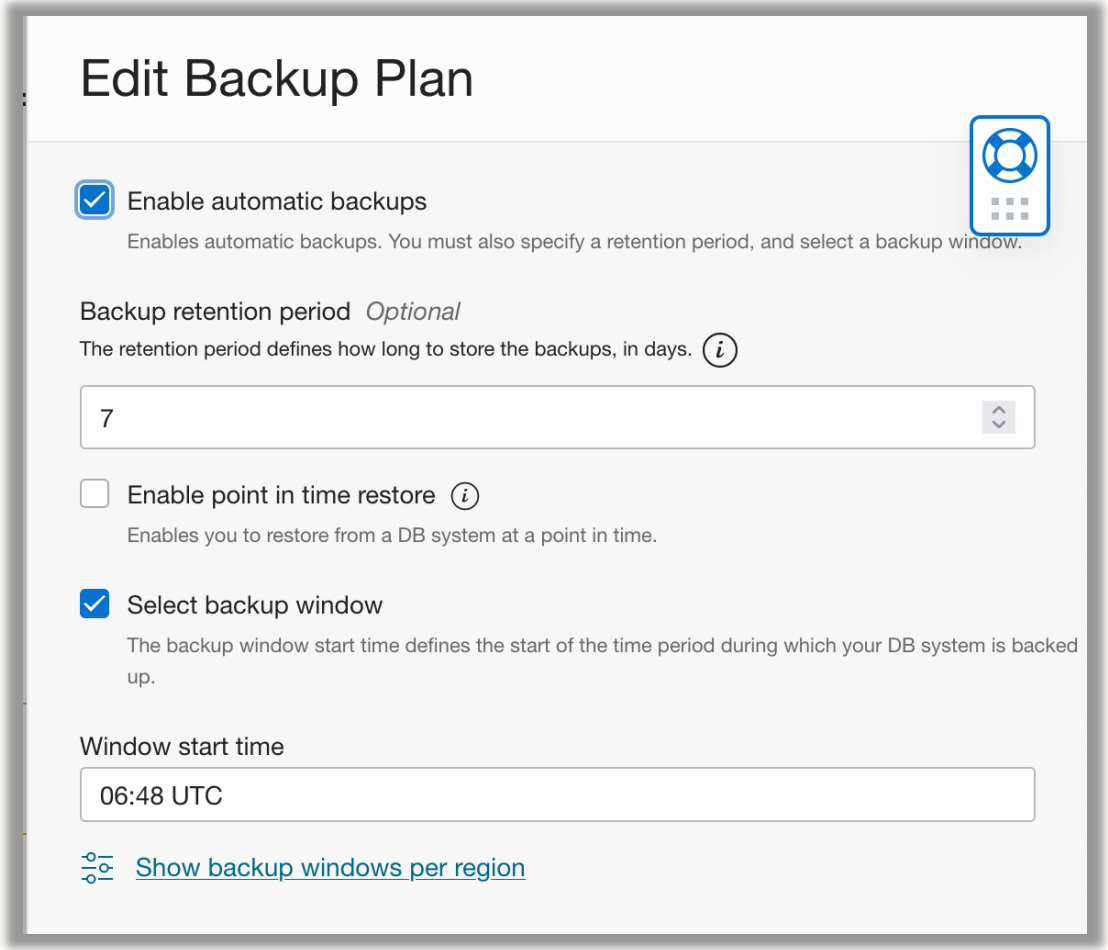
One-click DB System creation





Backups
Manual or Automatic

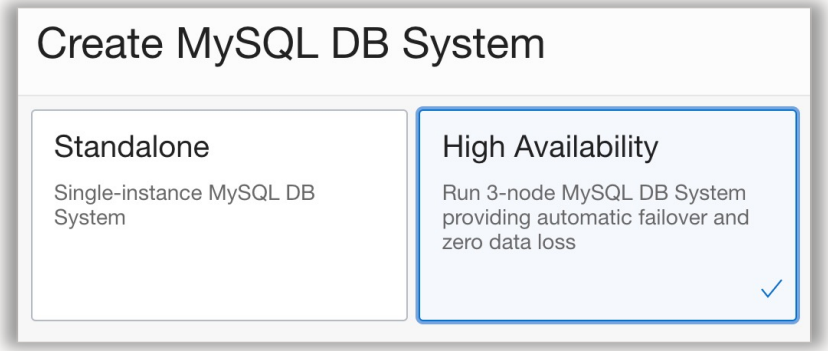
- Retention Period
- When to Backup
- Full or Incremental
- Point-in-Time Recovery (only non-HA DB Systems)

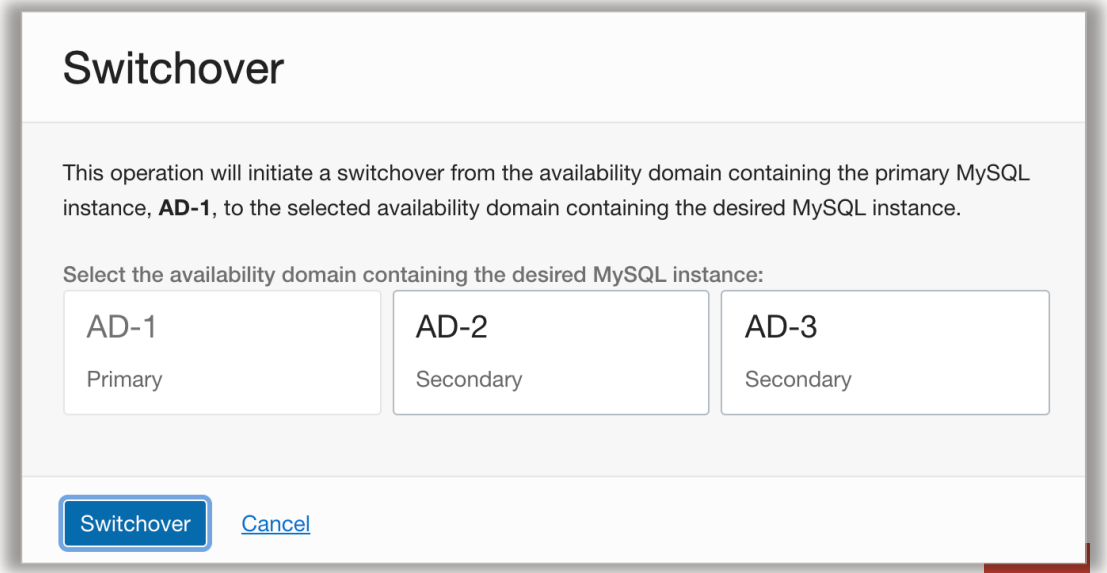


High Availability

RTO and RPO

- Single click High Availability
- Automatic Failover
- Planned Switchover
- Increase Uptime
- Reduce Downtime during a failure event (RTO: Minutes)
- Zero Data Loss during a failure event (RPO: Zero)





Inbound and Outbound Replication

Hybrid Deployments and Migrations

Hybrid deployments

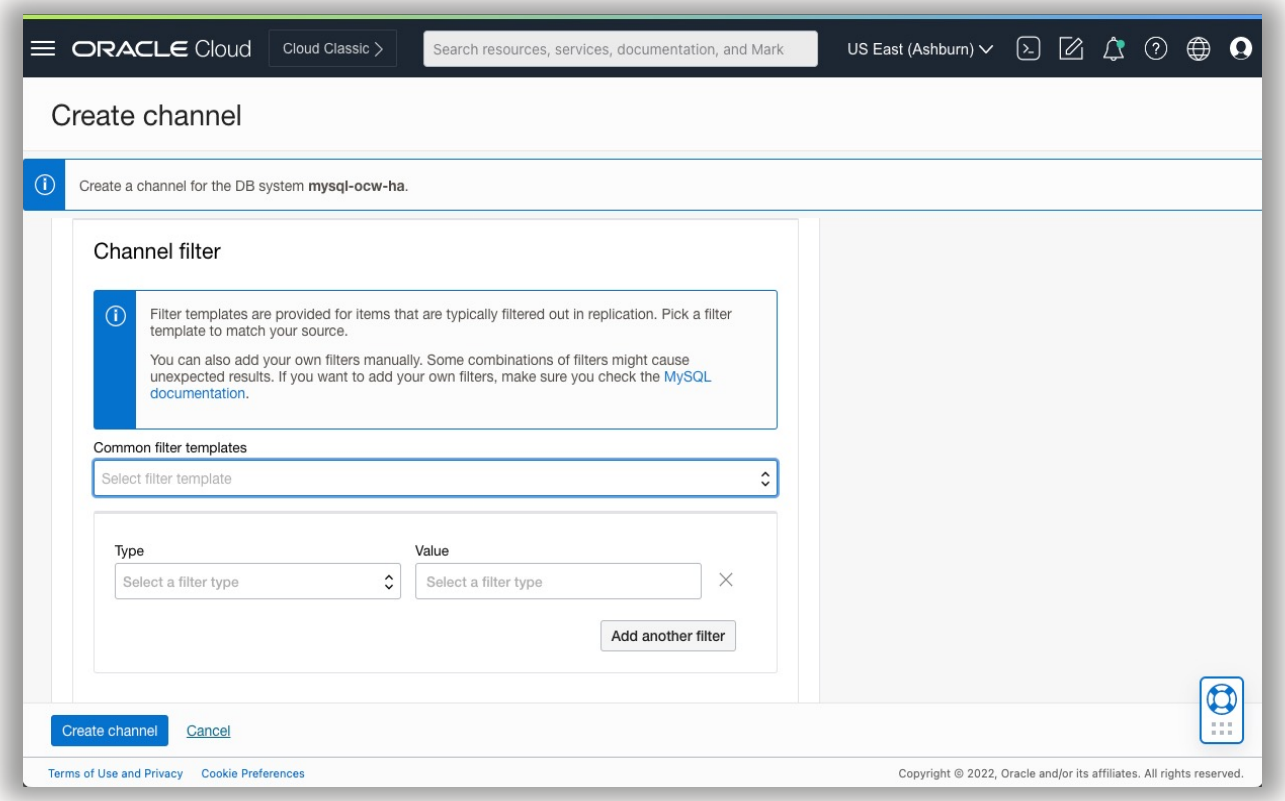
- On-premise and multi-cloud
- OCI as your main site
- OCI as your Disaster Recovery site
- OCI for capacity bursting
- HeatWave for Analytics

Live Migrations

- Minimize downtime

Cross-region replication

- DB System to DB System



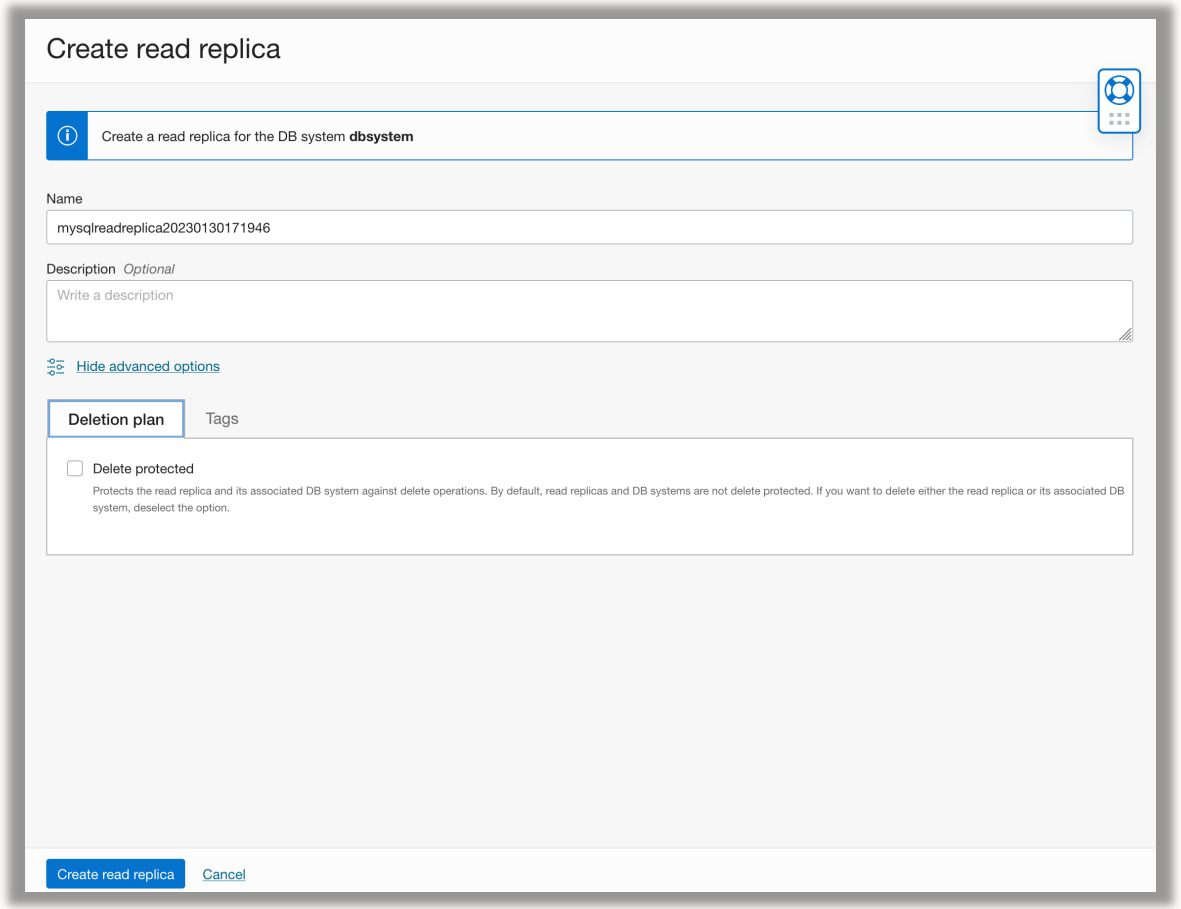
Managed Read Replicas

Grow and Shrink Read Capacity Seamlessly

High performance by scaling your reads.

- A single click creates a Read Replica
  - Provision
  - Launch
  - Setup Replication
  - Monitor and Manage

- Read Replicas are associated with a DB System
  - RO endpoints in the DB System
  - Up to 18 max per DB System
  - Requires a shape of 4 OCPUs or larger
  - CLI, SDK and Terraform support



The MySQL Heatwave Service

What?

- The most up to date MySQL, delivered by the MySQL team
- OLTP and OLAP in one database
- The technology you know, feels right at home for a traditional MySQL user
- Straightforward and simple for a new adopters
- Integrated with other Oracle services
- Powerful and secure infrastructure – The Oracle Cloud Infrastructure

- **Observed, Automated and Operated by MySQL Experts**

Conclusion

Conclusion

- Monitor, Observe
  - MySQL Replication enhancements makes tuning, troubleshooting and root cause analysis easier
- Automate, Automate
  - MySQL Replication has built-in automation that makes it is a great foundation for advanced system architectures
- Control and Operate
  - Overriding automation may be necessary during a maintenance event.
  - Emergency stopping a procedure is sometimes required.

Replication in MySQL 8.x continues to improve usability and operability.

The MySQL Heatwave Service runs and operates the database for you, on the Oracle Cloud Infrastructure. Go and give it a try if you are not using it already!


ORACLE