MySQL InnoDB Cluster

Introduction

An InnoDB Cluster consists of at least three MySQL Server instances, and it provides high-availability and scaling features.

In this lab you will create 3 instances MySQL InnoDB Cluster as Single Primary and have a trial on the MySQL Shell to configure and operate. And you will be using MySQL Router to test for Server routing and test for Failover.

Estimated Lab Time: 45 minutes

Objectives

In this lab, you will:

- Check and, if required, fix data model for InnoDB Cluster compatibility
- Configure InnoDB Cluster in single primary (mysql1 as primary)
- Configure mysql router
- Test client connection with MySQL Router, read/write and read only
- Simulate a crash of primary instance

> **Note:**

- MySQL InnoDB cluster require at least 3 instances (refer to schema in the first part of the lab guide)
  - mysql1 will be the primary
  - mysql2 and mysql3 will be first secondaries
- We use here different ways to configure the secondaries
- We don't need a direct access to the instance servers

Task 1: Check data model and prepare instances

1. From **app-srv**, connect the client to mysql1 and verify data model compatibility with Group replication requirements. If needed, fix the errors

```
<span style="color:green">shell-app-srv$</span> <copy>mysqlsh
admin@mysql1:3307</copy>
```

2. Search non InnoDB tables and if there are you must change them. For this lab just drop them

```
<span style="color:blue">mysql-primary></span> <copy>SELECT table_schema,
table_name, engine, table_rows, (index_length+data_length)/1024/1024 AS
sizeMB
FROM information_schema.tables
WHERE engine != 'innodb'
```

```
AND table_schema NOT IN ('information_schema', 'mysql',
'performance_schema');</copy>
```

3. Search InnoDB tables without primary or unique key. In production you must fix.

```
<span style="color:blue">mysql-primary></span> <copy>SELECT
tables.table_schema, tables.table_name, tables.engine, tables.table_rows
FROM information_schema.tables
    LEFT JOIN (select table_schema, table_name
    FROM information_schema.statistics
    GROUP BY table_schema, table_name, index_name HAVING
    SUM(CASE
            WHEN non_unique = 0
            AND nullable != 'YES' then 1 ELSE 0
            END
        ) = count(*)
    ) puks
    ON tables.table_schema = puks.table_schema
    AND tables.table_name = puks.table_name
WHERE puks.table_name is null
    AND tables.table_type = 'BASE TABLE'
    AND engine='InnoDB'
    AND tables.table_schema NOT IN ('information_schema', 'mysql',
'performance_schema');</copy>
```

4. Most probably we have the table world.city_part without primary key. So let's fix it

```
<span style="color:blue">mysql-primary></span> <copy>desc world.city_part;
</copy>
```

```
<span style="color:blue">mysql-primary></span> <copy>alter table
world.city_part add column added_pk bigint unsigned NOT NULL primary key
auto_increment invisible;</copy>
```

```
<span style="color:blue">mysql-primary></span> <copy>desc world.city_part;
</copy>
```

```
<span style="color:blue">mysql-primary></span> <copy>show create table
world.city_part\G</copy>
```

```
<span style="color:blue">mysql-primary></span> <copy>\q</copy>
```

5. We use mysql1 as primary, but we need a second instance. For this we can use mysql2. Just stop replication

   - Connect your client to mysql2 with administrative account

```
<span style="color:green">shell-app-srv$</span> <copy>mysqlsh
admin@mysql2:3307</copy>
```

   - Stop and remove the replication settings

```
<span style="color:blue">mysql-secondary-1></span> <copy>stop replica;
</copy>
```

```
<span style="color:blue">mysql-secondary-1></span> <copy>reset replica all;
</copy>
```

```
<span style="color:blue">mysql-secondary-1></span> <copy>SHOW REPLICA
STATUS;</copy>
```

```
<span style="color:blue">mysql-secondary-1></span> <copy>\q</copy>
```

6. Now we need a third instance. We install now a fresh one on mysql3

   - Connect to mysql3 through app-srv

```
<span style="color:green">shell-app-srv$</span> <copy>ssh -i
$HOME/sshkeys/id_rsa_mysql3 opc@mysql3</copy>
```

   - Execute below script that replicate what we did in manual installation lab (create mysqluser/mysqlgrp, folders and install binaries and enterprise plugins, create the admin user)

```
<span style="color:green">shell-mysql3></span>
<copy>/workshop/support/MySQL_InnoDB_Cluster___secondary_on_mysql3.sh</copy>
```

7. The instance on mysql3 is new, so let's verify that everything is fine

   - Connect to the instance

```
<span style="color:green">shell-mysql3></span> <copy>mysqlsh
admin@mysql3:3307</copy>
```

   - Verify that the instance is empty

```
<span style="color:blue">mysql3></span> <copy>SHOW DATABASES;</copy>
```

8. Now exit from mysql2 and mysql3 shells, we don't need them anymore.

```
<span style="color:blue">mysql3></span> <copy>\q</copy>
```

```
<span style="color:green">shell-mysql3></span> <copy>exit</copy>
```

Task 2: Create Cluster

1. We can do it from any server, we use here app-srv

```
<span style="color:green">shell-app-srv$ </span> <copy>mysqlsh</copy>
```

2. Check the instance configuration using javascipt command mode

```
<span style="color:blue">My</span><span style="color: orange">SQL </span>
<span style="background-color:orange">SQL</span>> <copy>\js</copy>
```

```
<span style="color:blue">My</span><span style="color: orange">SQL </span>
<span style="background-color:yellow">JS</span>>
<copy>dba.checkInstanceConfiguration('admin@mysql1:3307')</copy>
```

OUTPUT EXAMPLE: CONFIGURATION ERRORS

```
...
NOTE: Some configuration options need to be fixed:
+----------------------------------------+--------------+----------------+-
```

```
---------------------------+
| Variable                               | Current Value | Required Value |
Note                         |
+----------------------------------------+---------------+----------------+-
---------------------------+
| binlog_transaction_dependency_tracking | COMMIT_ORDER  | WRITESET       |
Update the server variable |
+----------------------------------------+---------------+----------------+-
---------------------------+

NOTE: Please use the dba.configureInstance() command to repair these issues.

{
    "config_errors": [
        {
            "action": "server_update",
            "current": "COMMIT_ORDER",
            "option": "binlog_transaction_dependency_tracking",
            "required": "WRITESET"
        }
    ],
    "status": "error"
}
```

3. We use now MySQL Shell to fix these errors. Confirm when MySQL Shell ask to cahnge or reboot

```
<span style="color:blue">My</span><span style="color: orange">SQL </span>
<span style="background-color:yellow">JS</span>>
<copy>dba.configureInstance('admin@mysql1:3307')</copy>
```

OUTPUT EXAMPLE: FIX ERRORS

```
Configuring local MySQL instance listening at port 3307 for use in an InnoDB
cluster...

This instance reports its own address as mysql1:3307
Clients and other cluster members will communicate with it through this
address by default. If this is not correct, the report_host MySQL system
variable should be changed.

applierWorkerThreads will be set to the default value of 4.

NOTE: Some configuration options need to be fixed:
+----------------------------------------+---------------+----------------+-
---------------------------+
| Variable                               | Current Value | Required Value |
Note                         |
+----------------------------------------+---------------+----------------+-
---------------------------+
| binlog_transaction_dependency_tracking | COMMIT_ORDER  | WRITESET       |
Update the server variable |
+----------------------------------------+--------------+---------------+-
---------------------------+

Do you want to perform the required configuration changes? [y/n]: y
Configuring instance...

WARNING: '@@binlog_transaction_dependency_tracking' is deprecated and will
be removed in a future release. (Code 1287).
The instance 'mysql1:3307' was configured to be used in an InnoDB cluster.
```

4. Just to be sure, re-check the instance configuration and verify that you receive an "ok" message

```
<span style="color:blue">My</span><span style="color: orange">SQL </span>
<span style="background-color:yellow">JS</span>
<copy>dba.checkInstanceConfiguration('admin@mysql1:3307')</copy>
```

OUTPUT EXAMPLE: NO CONFIGURATION ERRORS

```
...
The instance 'mysql1:3307' is valid to be used in an InnoDB cluster.

{
    "status": "ok"
}
```

5. Now it's time to create the cluster.

```
<span style="color:blue">My</span><span style="color: orange">SQL </span>
<span style="background-color:yellow">JS</span>> <copy>\connect
admin@mysql1:3307</copy>
```

```
<span style="color:blue">My</span><span style="color: orange">SQL </span>
<span style="background-color:yellow">JS</span>> <copy>var cluster =
dba.createCluster('testCluster')</copy>
```

OUTPUT EXAMPLE: CLUSTER CREATED

```
A new InnoDB Cluster will be created on instance 'mysql1:3307'.

Validating instance configuration at mysql1:3307...
```

```
This instance reports its own address as mysql1:3307

Instance configuration is suitable.
NOTE: Group Replication will communicate with other members using
'mysql1:3307'. Use the localAddress option to override.

* Checking connectivity and SSL configuration...

Creating InnoDB Cluster 'testCluster' on 'mysql1:3307'...

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.
```

6. Verify cluster status (why "Cluster is NOT tolerant to any failures" ?)

```
<span style="color:blue">My</span><span style="color: orange">SQL </span>
<span style="background-color:yellow">JS</span>> <copy>cluster.status()
</copy>
```

OUTPUT EXAMPLE: CLUSTER STATUS WITH ONE INSTANCE

```
{
   "clusterName": "testCluster",
   "defaultReplicaSet": {
       "name": "default",
       "primary": "mysql1:3307",
       "ssl": "REQUIRED",
       "status": "OK_NO_TOLERANCE",
       "statusText": "Cluster is NOT tolerant to any failures.",
       "topology": {
           "mysql1:3307": {
               "address": "mysql1:3307",
               "memberRole": "PRIMARY",
               "mode": "R/W",
               "readReplicas": {},
               "replicationLag": "applier_queue_applied",
               "role": "HA",
               "status": "ONLINE",
               "version": "8.0.36"
           }
       },
       "topologyMode": "Single-Primary"
   },
    "groupInformationSourceMember": "mysql1:3307"
}
```

Task 3: Add a second and third instance to the cluster

1. Check the instance configuration on mysql2

```
<span style="color:blue">My</span><span style="color: orange">SQL </span>
<span style="background-color:yellow">JS</span>>
<copy>dba.checkInstanceConfiguration('admin@mysql2:3307')</copy>
```

2. Use MySQL Shell to fix issues (confirm required changes)

```
<span style="color:blue">My</span><span style="color: orange">SQL </span>
<span style="background-color:yellow">JS</span>>
<copy>dba.configureInstance('admin@mysql2:3307')</copy>
```

3. Add the instance to the cluster

```
<span style="color:blue">My</span><span style="color: orange">SQL </span>
<span style="background-color:yellow">JS</span>>
<copy>cluster.addInstance('admin@mysql2:3307')</copy>
```

OUTPUT EXAMPLE: ADD SECOND INSTANCE (EXTRACT) ``` The safest and most convenient way to provision a new instance is through automatic clone provisioning, which will completely overwrite the state of 'mysql2:3307' with a physical snapshot from an existing cluster member. To use this method by default, set the 'recoveryMethod' option to 'clone'.

```
The incremental state recovery may be safely used if you are sure all updates ever
executed in the cluster were done with GTIDs enabled, there are no purged
transactions and the new instance contains the same GTID set as the cluster or a
subset of it. To use this method by default, set the 'recoveryMethod' option to
'incremental'.

Incremental state recovery was selected because it seems to be safely usable.

...

The instance 'mysql2:3307' was successfully added to the cluster.
```
```

4. Verify cluster status
```

```
<span style="color:blue">My</span><span style="color: orange">SQL </span>
<span style="background-color:yellow">JS</span>> <copy>cluster.status()
</copy>
```

OUTPUT EXAMPLE: CLUSTER STATUS WITH 2 INSTANCES

```
{
    "clusterName": "testCluster",
    "defaultReplicaSet": {
        "name": "default",
        "primary": "mysql1:3307",
        "ssl": "REQUIRED",
        "status": "OK_NO_TOLERANCE",
        "statusText": "Cluster is NOT tolerant to any failures.",
        "topology": {
            "mysql1:3307": {
                "address": "mysql1:3307",
                "memberRole": "PRIMARY",
                "mode": "R/W",
                "readReplicas": {},
                "replicationLag": "applier_queue_applied",
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.36"
            },
            "mysql2:3307": {
                "address": "mysql2:3307",
                "memberRole": "SECONDARY",
                "mode": "R/O",
                "readReplicas": {},
                "replicationLag": "applier_queue_applied",
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.36"
            }
        },
        "topologyMode": "Single-Primary"
    },
    "groupInformationSourceMember": "mysql1:3307"
}
```

5. Now we add the third instance to cluster, check the instance configuration

```
<span style="color:blue">My</span><span style="color: orange">SQL </span>
<span style="background-color:yellow">JS</span>>
<copy>dba.checkInstanceConfiguration('admin@mysql3:3307')</copy>
```

6. If there are issues fix them

```
<span style="color:blue">My</span><span style="color: orange">SQL </span>
<span style="background-color:yellow">JS</span>>
<copy>dba.configureInstance('admin@mysql3:3307')</copy>
```

7. Add now the instance on mysql3 to the cluster

   - Use the same command as before to add the instance. But now we received an alert about incorrect GTID set (please remember that the this instance is empty).

```
<span style="color:blue">My</span><span style="color: orange">SQL </span>
<span style="background-color:yellow">JS</span>>
<copy>cluster.addInstance('admin@mysql3:3307')</copy>
```

OUTPUT EXAMPLE: ADD 3 INSTANCE GTID WARNING (EXTRACT) ``` WARNING: A GTID set check of the MySQL instance at 'mysql3:3307' determined that it contains transactions that do not originate from the cluster, which must be discarded before it can join the cluster.

```
 mysql3:3307 has the following errant GTIDs that do not exist in the
cluster:
 ...
 ```
```

   - To complete this step choose **[C]lone** as recovery mode to use the MySQL clone plugin

**OUTPUT EXAMPLE: ADD 3 INSTANCE WITH CLONE (EXTRACT)** ``` Please select a recovery method [C]lone/[A]bort (default Abort): C Validating instance configuration at mysql3:3307...

```
 ...

 The instance 'mysql3:3307' was successfully added to the cluster.
 ```
```

8. Verify cluster status, now with all three servers. Please check that **"status": "OK"** and **"primary": "mysql1:3307"**

```
<span style="color:blue">My</span><span style="color: orange">SQL </span>
<span style="background-color:yellow">JS</span>> <copy>cluster.status()
</copy>
```

OUTPUT EXAMPLE: CLUSTER STATUS WITH 2 INSTANCES

```
{
    "clusterName": "testCluster",
    "defaultReplicaSet": {
        "name": "default",
        "primary": "mysql1:3307",
        "ssl": "REQUIRED",
        "status": "OK",
        "statusText": "Cluster is ONLINE and can tolerate up to ONE 
failure.",
        "topology": {
            "mysql1:3307": {
                "address": "mysql1:3307",
                "memberRole": "PRIMARY",
                "mode": "R/W",
                "readReplicas": {},
                "replicationLag": "applier_queue_applied",
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.36"
            },
            "mysql2:3307": {
                "address": "mysql2:3307",
                "memberRole": "SECONDARY",
                "mode": "R/O",
                "readReplicas": {},
                "replicationLag": "applier_queue_applied",
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.36"
            },
            "mysql3:3307": {
                "address": "mysql3:3307",
                "memberRole": "SECONDARY",
                "mode": "R/O",
                "readReplicas": {},
                "replicationLag": "applier_queue_applied",
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.36"
            }
        },
        "topologyMode": "Single-Primary"
    },
    "groupInformationSourceMember": "mysql1:3307"
}
```

9. Now cluster is up and running, and we can exit from MySQL Shell

```
<span style="color:blue">My</span><span style="color: orange">SQL </span>
<span style="background-color:yellow">JS</span>> <copy>\q</copy>
```

Task 4: Deploy MySQL Router

1. Install MySQL Router via rpm package

    ```
    <span style="color:green">shell-app-srv$</span> <copy>sudo yum -y install
    /workshop/linux/mysql-router-commercial-8.*.x86_64.rpm</copy>
    ```

2. Configure MySQL Router

    ```
    <span style="color:green">shell-app-srv$</span> <copy>sudo mysqlrouter --
    bootstrap admin@mysql1:3307 --user=mysqlrouter</copy>
    ```

    Have a look on the output, note the following:

    - Read/Write Connections port: **localhost:6446**
    - Read/Only Connections: **localhost:6447**

3. Start MySQL Router

    ```
    <span style="color:green">shell-app-srv$</span> <copy>sudo systemctl start
    mysqlrouter</copy>
    ```

4. Test the connection with a mysql client connect to 6446 port (read/write). To which server are you currently connected? Can you change the content?

    ```
    <span style="color:green">shell-app-srv$</span> <copy>mysqlsh
    admin@127.0.0.1:6446</copy>
    ```

    ```
    <span style="color:blue">mysql-rw></span> <copy>SELECT @@hostname, @@port;
    </copy>
    ```

    ```
    <span style="color:blue">mysql-rw></span> <copy>use newdb;</copy>
    ```

    ```
    <span style="color:blue">mysql-rw></span> <copy>SHOW TABLES;</copy>
    ```

```
<span style="color:blue">mysql-rw></span> <copy>CREATE TABLE newtable (c1
int primary key);</copy>
```

```
<span style="color:blue">mysql-rw></span> <copy>INSERT INTO newtable
VALUES(1);</copy>
```

```
<span style="color:blue">mysql-rw></span> <copy>INSERT INTO newtable
VALUES(2);</copy>
```

```
<span style="color:blue">mysql-rw></span> <copy>SELECT * from newtable;
</copy>
```

```
<span style="color:blue">mysql-rw></span> <copy>\q</copy>
```

5. The second port of MySQL Router is used for read only sessions. Close session and re open on port 6447 (read only port).

   To which server are you currently connected? Can you change the content?

```
<span style="color:green">shell-app-srv$</span> <copy>mysqlsh
admin@127.0.0.1:6447</copy>
```

```
<span style="color:blue">mysql-ro></span> <copy>SELECT @@hostname, @@port;
</copy>
```

```
<span style="color:blue">mysql-ro></span> <copy>use newdb;</copy>
```

```
<span style="color:blue">mysql-ro></span> <copy>SELECT * from newtable;
</copy>
```

```
<span style="color:blue">mysql-ro></span> <copy>INSERT INTO newtable
VALUES(3);</copy>
```

```
<span style="color:blue">mysql-ro></span> <copy>show variables where
Variable_name in ('read_only','super_read_only');</copy>
```

```
<span style="color:blue">mysql-ro></span> <copy>\q</copy>
```

6. Reconnect to primary instance through the router. Please keep this session open!.

   We use here the mysql client to use its reconnect feature.

```
<span style="color:green">shell-app-srv$</span> <copy>mysql -uadmin -p -
h127.0.0.1 -P6446</copy>
```

```
<span style="color:blue">mysql-rw></span> <copy>SELECT @@hostname, @@port;
</copy>
```

7. Open a second SSH connection to mysql1 and simulate a crash killing primary instance.

   - Retrieve process ID

```
<span style="color:green">shell-app-srv$</span> <copy>ssh -i
$HOME/sshkeys/id_rsa_mysql1 opc@mysql1</copy>
```

```
<span style="color:green">shell-mysql1></span> <copy>cat
/mysql/data/*.pid</copy>
```

   - then kill the process of primary instance to simulate a crash

```
<span style="color:green">shell-mysql1></span> <copy>sudo kill -9 <process
id from previous step></copy>
```

8. Now return to mysql connection previously opened and verify if it works. It may requires 15/20 seconds to be online.

```
<span style="color:blue">mysql-rw></span> <copy>SELECT @@hostname, @@port;
</copy>
```

```
<span style="color:blue">mysql-rw></span> <copy>SELECT @@hostname, @@port;
</copy>
```

```
<span style="color:blue">mysql-rw></span> <copy>INSERT INTO newdb.newtable
VALUES(30);</copy>
```

```
<span style="color:blue">mysql-rw></span> <copy>SELECT * from
newdb.newtable;</copy>
```

9. From the shell where you killed the instance use MySQL Shell to verify cluster status (of course connect to a living instance) using javascipt command mode

```
<span style="color:green">shell-app-srv$</span> <copy>mysqlsh</copy>
```

```
<span style="color:blue">My</span><span style="color: orange">SQL </span>
<span style="background-color:orange">SQL</span>> <copy>\js</copy>
```

```
<span style="color:blue">My</span><span style="color: orange">SQL </span>
<span style="background-color:yellow">JS</span>> <copy>\c
admin@mysql1:3307</copy>
```

```
<span style="color:blue">My</span><span style="color: orange">SQL </span>
<span style="background-color:yellow">JS</span>> <copy>var cluster =
dba.getCluster()</copy>
```

```
<span style="color:blue">My</span><span style="color: orange">SQL </span>
<span style="background-color:yellow">JS</span>> <copy>cluster.status()
</copy>
```

Acknowledgements

- **Author** - Marco Carlessi, Principal Sales Consultant
- **Contributors** - Perside Foster, Principal Sales Consultant, Selena Sánchez, MySQL Solutions Engineer
- **Last Updated By/Date** - Perside Foster, Partner Solutions Engineer, November 2024