ORACLE

From Single MySQL Instance to HA

The Journey To MySQL InnoDB Cluster

Frédéric Descamps
Community Manager
MySQL
February 2021

Who am I ?

about.me/lefred

Frédéric Descamps

- @lefred
- MySQL Evangelist
- Managing MySQL since 3.20
- devops believer
- living in Belgium 🇧🇪
- https://lefred.be
- hairdressers are closed in .be

Evolution to HA

MySQL 8.0

MySQL™ High Availability Evolution

Single MySQL Instance

This is where it all begins...

Some tips:

- use exclusively InnoDB
- keep the durability defaults

Next level

The database becomes more important, loosing it might be an issue...

Next level

The database becomes more important, loosing it might be an issue...

RTO → hours

Next level

The database becomes more important, loosing it might be an issue...

RTO → hours

RPO → 1 day

Next level

The database becomes more important, loosing it might be an issue...

RTO → hours

RPO → 1 day

RTO: *Recovery Time Objective (how long to recover)*
RPO: *Recovery Point Objective (how much data can be lost)*

Backups

- Physical Backups
- Logical Backups

For logical backups, please use MySQL Shell Dump & Load Utility !

```
MySQL Shell 8.0.22

Copyright (c) 2016, 2020, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

Type '\help' or '\?' for help; '\quit' to exit.
Creating a session to 'root@localhost'
Fetching schema names for autocompletion... Press ^C to stop.
Your MySQL connection id is 14 (X protocol)
Server version: 8.0.22 MySQL Community Server - GPL
No default schema selected; type \use <schema> to set one.
 MySQL  localhost:33060+  2021-01-05 11:57:32
 JS  util.dumpInstance('/tmp/backup', {threads: 8})
```

Next level

RPO of 1 day ? Really ?? We want to reduce it to minutes at least !

Next level

RPO of 1 day ? Really ?? We want to reduce it to minutes at least !

RTO → hours

Next level

RPO of 1 day ? Really ?? We want to reduce it to minutes at least !

RTO → hours

RPO → minutes

Durable Binlogs

This is the default in MySQL 8.0:

```
MySQL  localhost:33060+  2021-01-05 12:14:57
SQL  show global variables like 'log_bin';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| log_bin       | ON    |
+---------------+-------+
1 row in set (0.0020 sec)
MySQL  localhost:33060+  2021-01-05 12:15:03
SQL  show global variables like 'sync_binlog';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| sync_binlog   | 1     |
+---------------+-------+
1 row in set (0.0020 sec)
```

Next level

My data is very important and I've a heavy workload... I would like to loose less than a second !

Next level

My data is very important and I've a heavy workload... I would like to loose less than a second !

RTO → hours

Next level

My data is very important and I've a heavy workload... I would like to loose less than a second !

RTO → hours

RPO → less than a second

Point-In-Time Recovery (PTR)

- Enable GTID (optional but so convenient)
- Off-load binlogs (in real time)

```
MySQL  localhost:33060+  2021-01-05 12:14:57
SQL  show global variables like 'log_bin';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| log_bin       | ON    |
+---------------+-------+
1 row in set (0.0020 sec)
MySQL  localhost:33060+  2021-01-05 12:15:03
SQL  show global variables like 'sync_binlog';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| sync_binlog   | 1     |
+---------------+-------+
1 row in set (0.0020 sec)
```

Enable GTID

with restart

```
MySQL  localhost:33060+  2021-01-05 12:27:49
SQL  show global variables like 'gtid_mode';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| gtid_mode     | OFF   |
+---------------+-------+
1 row in set (0.0040 sec)
MySQL  localhost:33060+  2021-01-05 12:28:16
SQL  set persist_only gtid_mode='on';
Query OK, 0 rows affected (0.0008 sec)
MySQL  localhost:33060+  2021-01-05 12:28:34
SQL  set persist_only enforce_gtid_consistency='on';
Query OK, 0 rows affected (0.0008 sec)
MySQL  localhost:33060+  2021-01-05 12:29:07
SQL  restart;
Query OK, 0 rows affected (0.0009 sec)
```

Enable GTID

with restart

```
MySQL  localhost:33060+  2021-01-05 12:27:49
SQL  show global variables like 'gtid_mode';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| gtid_mode     | OFF   |
+---------------+-------+
1 row in set (0.0040 sec)
MySQL  localhost:33060+  2021-01-05 12:28:16
SQL  set persist_only gtid_mode='on';
Query OK, 0 rows affected (0.0008 sec)
MySQL  localhost:33060+  2021-01-05 12:28:34
SQL  set persist_only enforce_gtid_consistency='on';
Query OK, 0 rows affected (0.0008 sec)
MySQL  localhost:33060+  2021-01-05 12:29:07
SQL  restart;
Query OK, 0 rows affected (0.0009 sec)
```

without restart

```
MySQL  localhost:33060+  2021-01-05 12:31:00
SQL  show global variables like 'gtid_mode';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| gtid_mode     | OFF   |
+---------------+-------+
1 row in set (0.0043 sec)
MySQL  localhost:33060+  2021-01-05 12:31:01
SQL  set persist enforce_gtid_consistency='on';
Query OK, 0 rows affected (0.0008 sec)
MySQL  localhost:33060+  2021-01-05 12:31:27
SQL  set persist gtid_mode='off_permissive';
Query OK, 0 rows affected (0.0101 sec)
MySQL  localhost:33060+  2021-01-05 12:32:00
SQL  set persist gtid_mode='on_permissive';
Query OK, 0 rows affected (0.0097 sec)
MySQL  localhost:33060+  2021-01-05 12:32:06
SQL  set persist gtid_mode='on';
Query OK, 0 rows affected (0.0138 sec)
MySQL  localhost:33060+  2021-01-05 12:32:10
SQL  show global variables like 'gtid_mode';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| gtid_mode     | ON    |
+---------------+-------+
1 row in set (0.0020 sec)
```

Off-load Binlogs

```
MySQL  localhost:33060+  2021-01-05 13:07:33
SQL  create user getbinlog identified by 'password' require ssl;
Query OK, 0 rows affected (0.0073 sec)
MySQL  localhost:33060+  2021-01-05 13:08:05
SQL  grant replication slave on *.* to getbinlog;
Query OK, 0 rows affected (0.0044 sec)
```

Off-load Binlogs

```
MySQL  localhost:33060+  2021-01-05 13:07:33
SQL  create user getbinlog identified by 'password' require ssl;
Query OK, 0 rows affected (0.0073 sec)
MySQL  localhost:33060+  2021-01-05 13:08:05
SQL  grant replication slave on *.* to getbinlog;
Query OK, 0 rows affected (0.0044 sec)
```

On another machine:

```
mysqlbinlog --raw --read-from-remote-server --stop-never --host 10.0.0.2 \
            --port 3306 -u getbinlog -ppassword \
            --ssl-mode='REQUIRED' binlog.xxxxxx
```

Off-load Binlogs

```
MySQL  localhost:33060+  2021-01-05 13:07:33
SQL  create user getbinlog identified by 'password' require ssl;
Query OK, 0 rows affected (0.0073 sec)
MySQL  localhost:33060+  2021-01-05 13:08:05
SQL  grant replication slave on *.* to getbinlog;
Query OK, 0 rows affected (0.0044 sec)
```

On another machine:

```
mysqlbinlog --raw --read-from-remote-server --stop-never --host 10.0.0.2 \
            --port 3306 -u getbinlog -ppassword \
            --ssl-mode='REQUIRED' binlog.xxxxxx
```

See https://lefred.be/content/howto-make-mysql-point-in-time-recovery-faster/

Next level

My service is important, I would like to be up again in some minutes

Next level

My service is important, I would like to be up again in some minutes

RTO → minutes

Next level

My service is important, I would like to be up again in some minutes

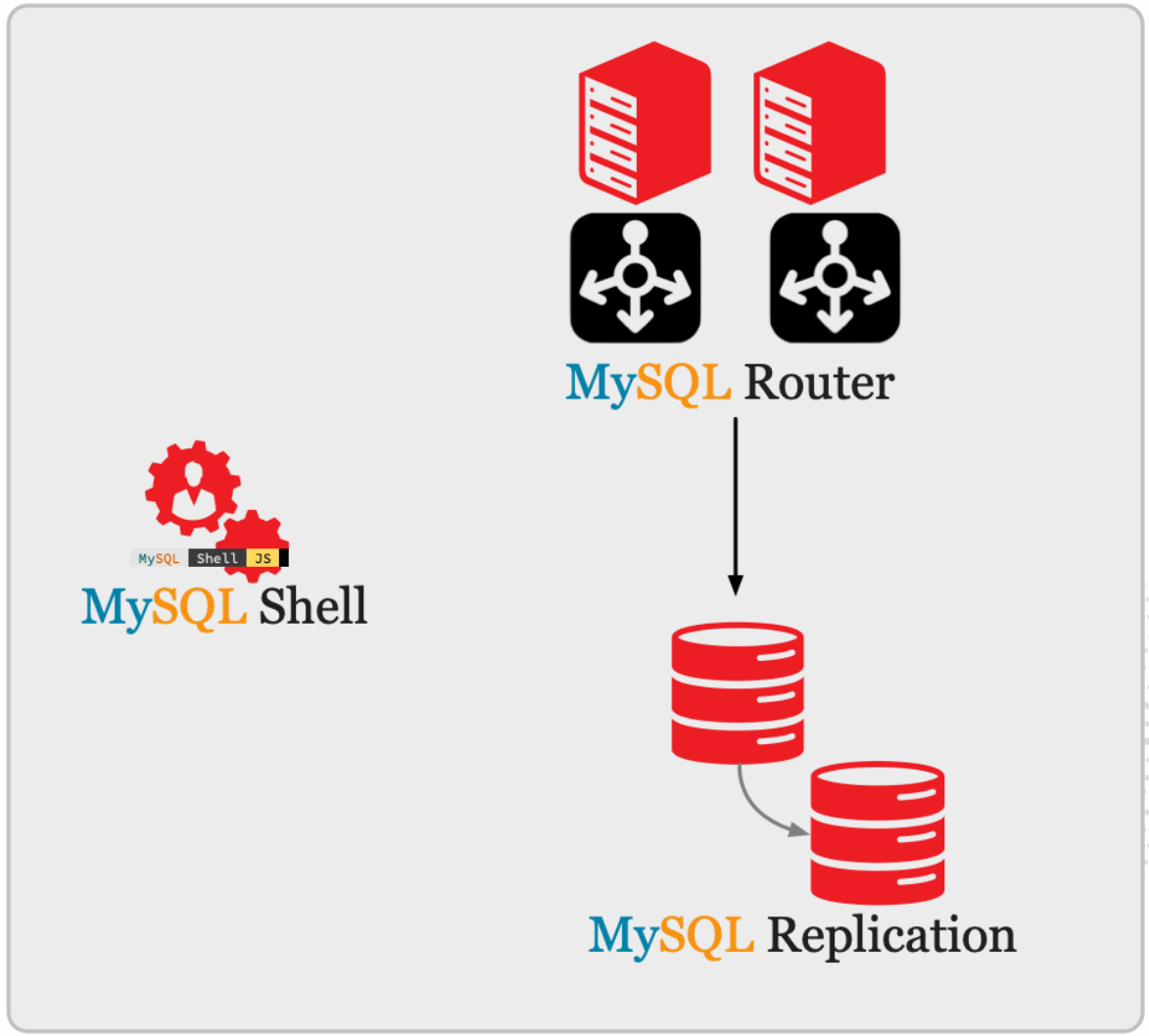
RTO → minutes

RPO → less than a second

MySQL InnoDB ReplicaSet

- Based on native Asynchronous Replication
- But easier…. easier is better !
- Data provisioning included (clone)
- 2 or more nodes
- Manual Failover
- Transparent/Automatic query routing with MySQL Router

MySQL InnoDB ReplicaSet



MySQL InnoDB ReplicaSet

On the current server:

```
JS > dba.configureReplicaSetInstance()

JS > rs=dba.createReplicaSet('myreplicaset')
```

On a new one where MySQL (server and shell) is just installed:

```
JS > dba.configureReplicaSetInstance()
```

Back on the first instance:

```
JS > rs.addInstance('10.0.1.2')
```

MySQL InnoDB ReplicaSet - examples

```
 MySQL  localhost:33060+ ssl  JS > dba.configureReplicaSetInstance()
Configuring local MySQL instance listening at port 3306 for use in an InnoDB ReplicaSet...

This instance reports its own address as mysql-2:3306
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.

ERROR: User 'root' can only connect from 'localhost'. New account(s) with proper source address specification to allow remote connection from all instances must
 be created to manage the cluster.

1) Create remotely usable account for 'root' with same grants and password
2) Create a new admin account for InnoDB ReplicaSet with minimal required grants
3) Ignore and continue
4) Cancel

Please select an option [1]: 2
Please provide an account name (e.g: icroot@%) to have it created with the necessary
privileges or leave empty and press Enter to cancel.
Account Name: clusteradmin
Password for new account: ********
Confirm password: ********

NOTE: Some configuration options need to be fixed:
+--------------------------+---------------+----------------+--------------------------------------------------+
| Variable                 | Current Value | Required Value | Note                                             |
+--------------------------+---------------+----------------+--------------------------------------------------+
| enforce_gtid_consistency | OFF           | ON             | Update read-only variable and restart the server |
| gtid_mode                | OFF           | ON             | Update read-only variable and restart the server |
| server_id                | 1             | <unique ID>    | Update read-only variable and restart the server |
+--------------------------+---------------+----------------+--------------------------------------------------+

Some variables need to be changed, but cannot be done dynamically on the server.
Do you want to perform the required configuration changes? [y/n]: y
Do you want to restart the instance after configuring it? [y/n]: y
Cluster admin user 'clusteradmin'@'%' created.
Configuring instance...
The instance 'mysql-2:3306' was configured to be used in an InnoDB ReplicaSet.
Restarting MySQL...
NOTE: MySQL server at mysql-2:3306 was restarted.
```

MySQL InnoDB ReplicaSet - examples

```
Please select a recovery method [C]lone/[I]ncremental recovery/[A]bort (default Clone): c
* Updating topology
Waiting for clone process of the new member to complete. Press ^C to abort the operation.
* Waiting for clone to finish...
NOTE: mysql-2:3306 is being cloned from single-mysql:3306
** Stage DROP DATA: Completed
** Clone Transfer
    FILE COPY  ############################################################  100%  Completed
    PAGE COPY  ############################################################  100%  Completed
    REDO COPY  ############################################################  100%  Completed

NOTE: mysql-2:3306 is shutting down...

* Waiting for server restart... ready
* mysql-2:3306 has restarted, waiting for clone to finish...
** Stage RESTART: Completed
* Clone process has finished: 365.82 MB transferred in 2 sec (182.91 MB/s)

** Configuring mysql-2:3306 to replicate from single-mysql:3306
** Waiting for new instance to synchronize with PRIMARY...

The instance 'mysql-2:3306' was added to the replicaset and is replicating from single-mysql:3306.
```

MySQL InnoDB ReplicaSet - examples

```
MySQL  localhost:33060+  2021-01-05 16:38:38
JS  rs.status()
{
    "replicaSet": {
        "name": "myreplicaset",
        "primary": "single-mysql:3306",
        "status": "AVAILABLE",
        "statusText": "All instances available.",
        "topology": {
            "mysql-2:3306": {
                "address": "mysql-2:3306",
                "instanceRole": "SECONDARY",
                "mode": "R/O",
                "replication": {
                    "applierStatus": "APPLIED_ALL",
                    "applierThreadState": "Slave has read all relay log; waiting for more updates",
                    "receiverStatus": "ON",
                    "receiverThreadState": "Waiting for master to send event",
                    "replicationLag": null
                },
                "status": "ONLINE"
            },
            "single-mysql:3306": {
                "address": "single-mysql:3306",
                "instanceRole": "PRIMARY",
                "mode": "R/W",
                "status": "ONLINE"
            }
        },
        "type": "ASYNC"
    }
}
```

MySQL Router

It's very easy to configure MySQL Router with the **`bootstrap`** command !

```
[root@single-mysql ~]# mysqlrouter --bootstrap clusteradmin@localhost:3306 --user=mysqlrouter
Please enter MySQL password for clusteradmin:
# Bootstrapping system MySQL Router instance...

- Creating account(s) (only those that are needed, if any)
- Verifying account (using it to run SQL queries that would be run by Router)
- Storing account in keyring
- Adjusting permissions of generated files
- Creating configuration /etc/mysqlrouter/mysqlrouter.conf

Existing configuration backed up to '/etc/mysqlrouter/mysqlrouter.conf.bak'

# MySQL Router configured for the InnoDB ReplicaSet 'myreplicaset'

After this MySQL Router has been started with the generated configuration

    $ /etc/init.d/mysqlrouter restart
or
    $ systemctl start mysqlrouter
or
    $ mysqlrouter -c /etc/mysqlrouter/mysqlrouter.conf

the cluster 'myreplicaset' can be reached by connecting to:

## MySQL Classic protocol

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

## MySQL X protocol

- Read/Write Connections: localhost:64460
- Read/Only Connections:  localhost:64470
```

MySQL Router

And as usual, visible in MySQL Shell

```
JS rs.listRouters()
{
    "replicaSetName": "myreplicaset",
    "routers": {
        "single-mysql.mysqlpub.mysqlvcn.oraclevcn.com::system": {
            "hostname": "single-mysql.mysqlpub.mysqlvcn.oraclevcn.com",
            "lastCheckIn": "2021-01-05 18:36:03",
            "roPort": 6447,
            "roXPort": 64470,
            "rwPort": 6446,
            "rwXPort": 64460,
            "version": "8.0.22"
        }
    }
}
```

MySQL InnoDB ReplicaSet - Manual Failover

```
MySQL  localhost:6447  2021-01-05 18:41:28
JS > rs.status()
ERROR: Unable to connect to the PRIMARY of the replicaset myreplicaset: MySQL Error 2003: Could not open connection to
 'single-mysql:3306': Can't connect to MySQL server on 'single-mysql' (111)
Cluster change operations will not be possible unless the PRIMARY can be reached.
If the PRIMARY is unavailable, you must either repair it or perform a forced failover.
See \help forcePrimaryInstance for more information.
WARNING: MYSQLSH 51118: PRIMARY instance is unavailable
{
    "replicaSet": {
        "name": "myreplicaset",
        "primary": "single-mysql:3306",
        "status": "UNAVAILABLE",
        "statusText": "PRIMARY instance is not available, but there is at least one SECONDARY that could be force-prom
oted.",
```

MySQL Shell is connected to MySQL InnoDB ReplicaSet using MySQL Router.

MySQL InnoDB ReplicaSet - Manual Failover

```
MySQL  localhost:6447  2021-01-05 18:41:32
JS  rs.forcePrimaryInstance()
* Connecting to replicaset instances
** Connecting to mysql-2:3306

* Waiting for all received transactions to be applied
** Waiting for received transactions to be applied at mysql-2:3306
* Searching instance with the most up-to-date transaction set
mysql-2:3306 has GTID set 5a47e14a-4f4b-11eb-97af-020017078dee:1-2639
mysql-2:3306 will be promoted to PRIMARY of the replicaset and the former PRIMARY will be invalidated.

* Checking status of last known PRIMARY
NOTE: single-mysql:3306 is UNREACHABLE
* Checking status of promoted instance
NOTE: mysql-2:3306 has status ERROR
* Checking transaction set status
* Promoting mysql-2:3306 to a PRIMARY...

* Updating metadata...

mysql-2:3306 was force-promoted to PRIMARY.
NOTE: Former PRIMARY single-mysql:3306 is now invalidated and must be removed from the replicaset.
* Updating source of remaining SECONDARY instances

Failover finished successfully.
```

Next level

Now my service is **very** important, I would like to be almost always up (automatic failover) and never loose data !

Next level

Now my service is **very** important, I would like to be almost always up (automatic failover) and never loose data !

RTO → seconds

Next level

Now my service is **very** important, I would like to be almost always up (automatic failover) and never loose data !

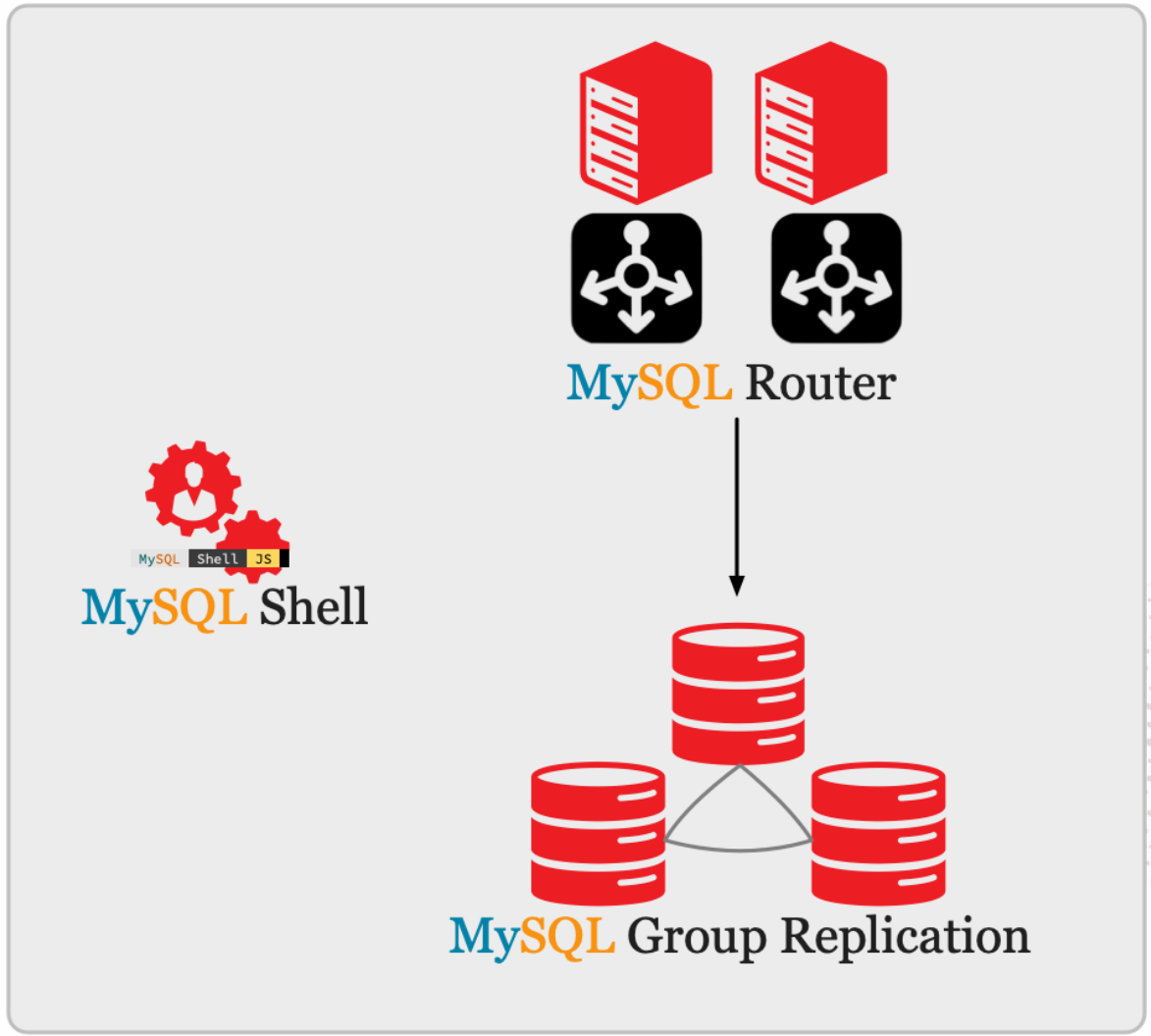
RTO → seconds

RPO → 0

MySQL InnoDB Cluster

- Based on native Group Replication
- But easier…. easier is still better !
- Data provisioning included (clone)
- 3 or more nodes (odd number)
- **Automatic Failover**
- Uses MySQL Router
- **Configurable Consistency Levels**

MySQL InnoDB Cluster



MySQL InnoDB Cluster

On the ReplicaSet Primary Server:

```
JS > dba.dropMetadataSchema()
JS > cluster=dba.createCluster('mycluster')
```

On the ReplicaSet Secondary instance:

```
SQL > STOP REPLICA;
SQL > RESET ALL REPLICA;
```

And back on the new Cluster Primary member:

```
JS > cluster.addInstance('mysql-2')
```

And we can add a third note for automatic fail-over, on that node:

MySQL InnoDB Cluster

On the ReplicaSet Primary Server:

```
JS > dba.dropMetadataSchema()
JS > cluster=dba.createCluster('mycluster')
```

On the ReplicaSet Secondary instance:

```
SQL > STOP REPLICA;
SQL > RESET ALL REPLICA;
```

And back on the new Cluster Primary member:

```
JS > cluster.addInstance('mysql-2')
```

And we can add a third note for automatic fail-over, on that node:

```
JS > dba.configureInstance()
```

From a member of the cluster:

```
JS > cluster=dba.getCluster()
JS > cluster.addInstance('mysql-3')
```

Finally, don't forget to bootstrap MySQL Router again:

```
# mysqlrouter --bootstrap \
  clusteradmin@single-mysql:3306 \
  --conf-use-gr-notifications \
  --user mysqlrouter --force
# systemctl restart mysqlrouter
```

MySQL InnoDB Cluster

```
JS cluster.status()
{
    "clusterName": "mycluster",
    "defaultReplicaSet": {
        "name": "default",
        "primary": "single-mysql:3306",
        "ssl": "REQUIRED",
        "status": "OK",
        "statusText": "Cluster is ONLINE and can tolerate up to ONE failure.",
        "topology": {
            "mysql-2:3306": {
                "address": "mysql-2:3306",
                "mode": "R/O",
                "readReplicas": {},
                "replicationLag": null,
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.22"
            },
            "mysql-3:3306": {
                "address": "mysql-3:3306",
                "mode": "R/O",
                "readReplicas": {},
                "replicationLag": null,
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.22"
            },
            "single-mysql:3306": {
                "address": "single-mysql:3306",
                "mode": "R/W",
                "readReplicas": {},
                "replicationLag": null,
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.22"
            }
        },
        "topologyMode": "Single-Primary"
    },
    "groupInformationSourceMember": "single-mysql:3306"
}
```

And what's next ?

- MySQL InnoDB Cluster with an asynchronous replica for DR:
  - asynchronous replication source connection failover (8.0.22)
  - with support for Group Replication (8.0.23)
- MySQL InnoDB Cluster with asynchronous Group Replication nodes
- and more to come...

Questions ?