Cluster Mysql8 InnoDB

Install, management & recovery

Document ID	**Date Effective**
MySQLINNODBC	*8 August 2018*
Author(s)	**Supercedes**
Thomas Guiseppin	*NA*
Peer reviewed by	*[Name] and [Date]*

Table of Contents

1 - Revision History

Rev	Date	Initials	Changes Made
1.0	8 August 2018	RC	Initial document created.
1.1	20 September	HA	Implementation Option2 and fixes

2 - Overview

Applies to:	Linux system administrator / Database administrator
Objective:	*Install, manage and recovery Mysql 8 InnoDB cluster*
Policy or Handbook Reference	
Pre-Requisites	Linux Knowledges Installation : Experimented Management & Recovery: Beginner / Experimented

3 - Introduction

This document explain the steps to follow for setting up a cluster Mysql InnoDB, how to manage it and the recovering procedure in case of failure. This document is originally made for Moodle CMS, but the main instructions should works for all Mysql InnoDB Cluster.

Before to apply some changes, read in first the part about the installation procedure can be useful to understand how this type of cluster work.

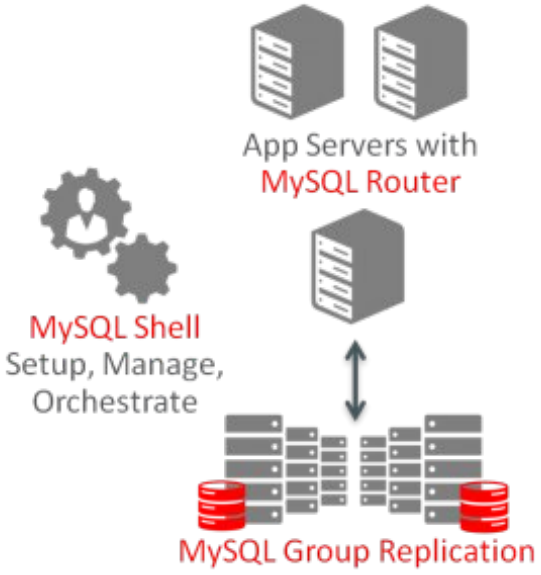
4 - General Information

InnoDB Cluster provides an out-of-the-box and easy to use built-in HA and Scaling solution for MySQL by tightly integrating the following GA components

MySQL Router 8.0.11+, to provide transparent routing between the client (application) requests and the InnoDB cluster server instances.

MySQL Shell 8.0.11+, the unified interface for MySQL developers and DBAs to create and manage InnoDB Clusters using the built-in AdminAPI.

MySQL 8.0.11+ Servers with Group Replication, to provide the data replication mechanism within InnoDB clusters, ensuring fault tolerance, automated failover, and elasticity.

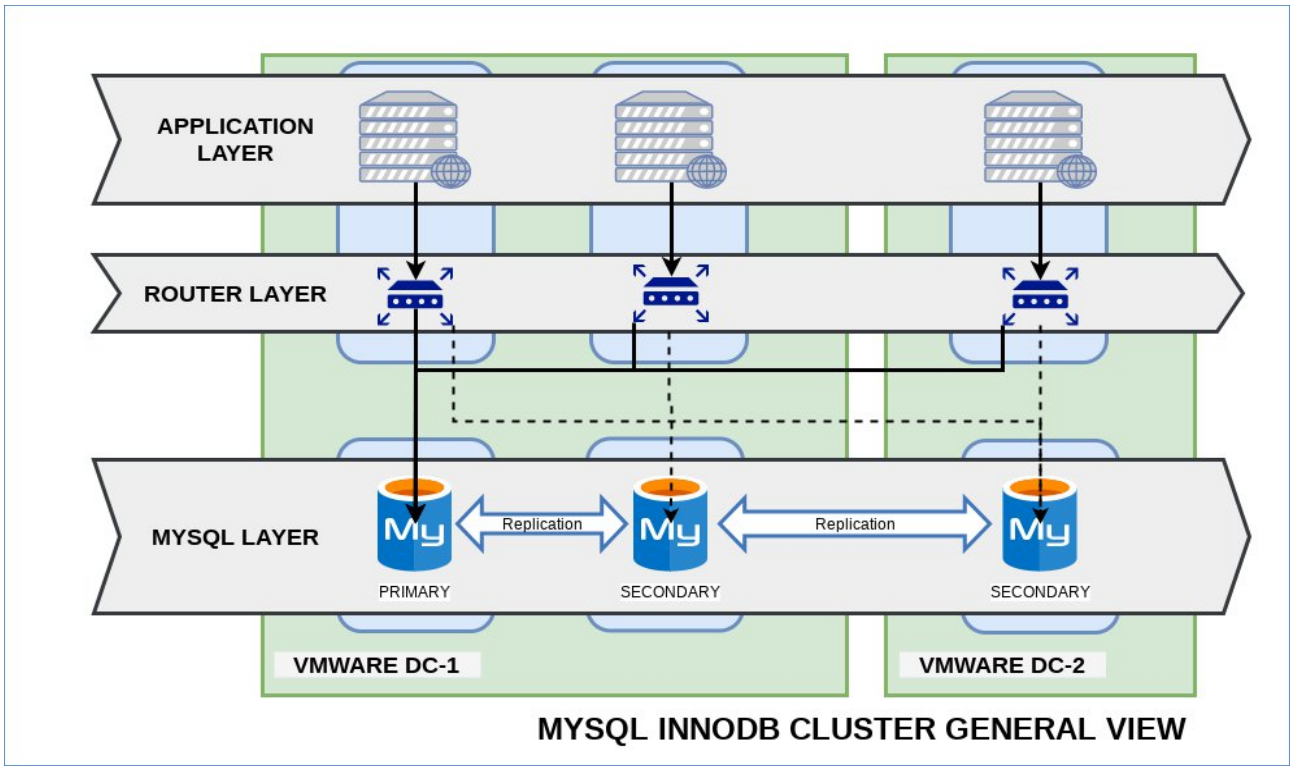


PART 1 : GENERAL SCHEMATIC VIEW AND DESCRIPTION

Each Mysql router have the possibility to read directly the cluster status and decide which server is the master. Theses configurations are **NOT** a multi-master system. In any case you can have just one master in R/W and the other nodes are set in secondary for data replication. Performance will be not improved.

Moodle doest not allow the possibility to route his different types of requests (SELECT / INSERT...) and can't use the possibility to write on the master and read on secondary. All requests arrive on the R/W instance. May be that this possibility will be added in the future...

1 - Schema (Option 1)



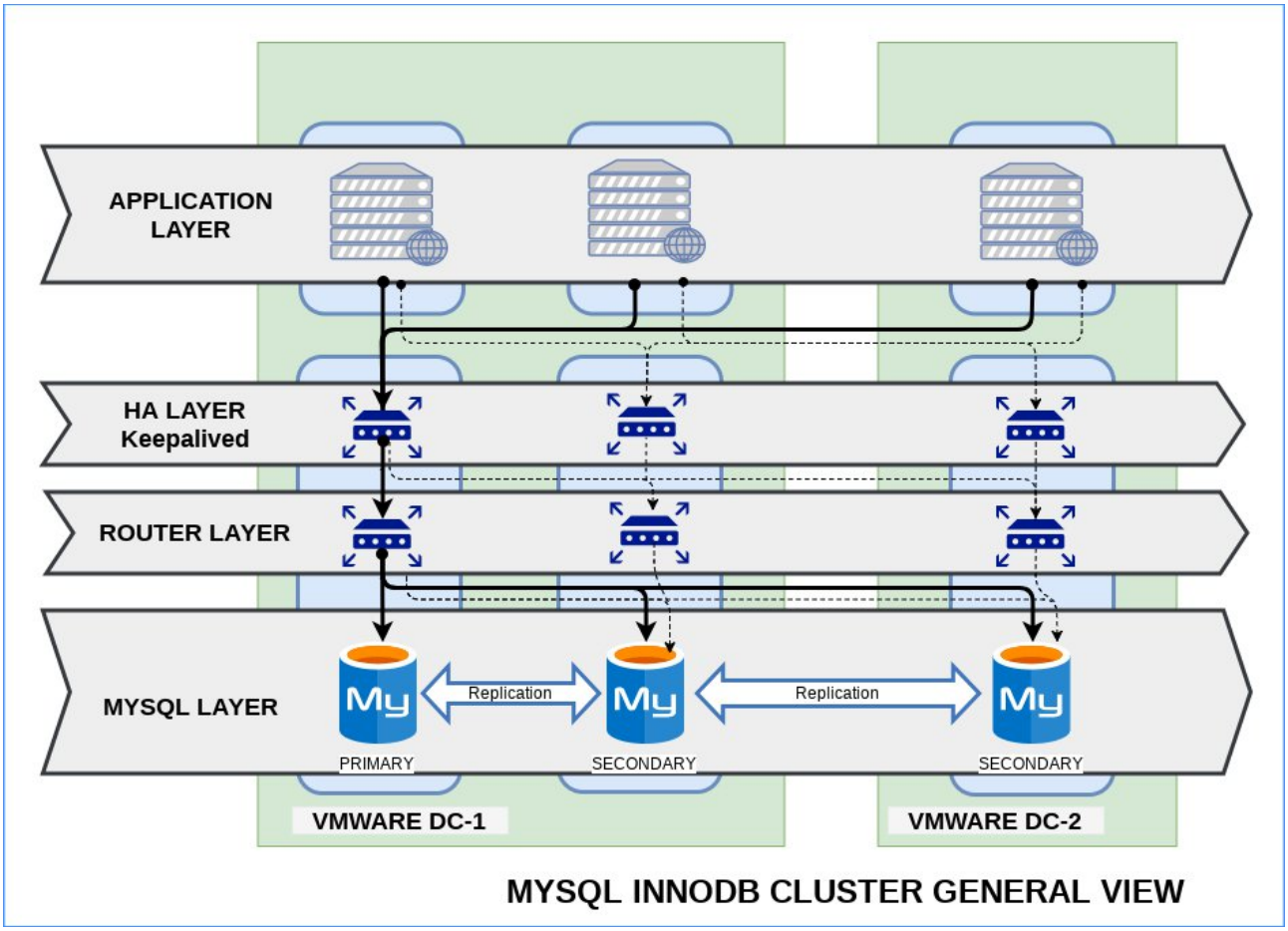
2 - Description (Option 1)

From the top to bottom:

- The first layer represent web server with applications and Mysql client requirement.
- The second layer represent Mysql router, one router is setting up per web server, usually **on the same server**, but it's not a requirement. Some other routers can be available and not represented on this schema (monitoring...)
- The third layer represent three Mysql servers. Two are available on the first DC and one on the second DC. It can work with just one DC, but you will loose redundancy.
- There are no load balancing system between a web server and his Mysql router. It's mean, that this type of infrastructure require to be monitored with specific scripts to exclude a pool (web + router) in case of failure.

There is no load balancing system between a web server and his Mysql router.

3 - Schema (Option 2)



4 - Description (Option 2)

From the top to bottom:

- The first layer represent web server with application and Mysql client's requirement.
- The second layer represent HA provided by Keepalived. In this configuration we have three routers, one on each database server. In fact, the secondary on DC-1 can be optional, but if you loose one server, your infrastructure loose all redundancy. And more than three with two DC is useless.
- The third layer represent Mysql router. One per keepalive server (on the same node).
- The fourth layer represent three Mysql server. Two are available on the first DC-1 and one on the second DC-2.

5 - Comparison

	Redundancy	Complexity	Performance	Management
Option 1	Good	Low	Good	Easy
Option 2	Very good	Hight	Good	Acceptable

6 - IP addresses / Hostname

We have three mysql server with theses configurations :

- 172.16.0.101 serverdb-one-dc-1
- 172.16.0.102 serverdb-two-dc-1
- 172.16.1.101 serverdb-one-dc-2

And six web servers in the same network and a minimum of six router associated.

PART 2: BOOTSTRAP YOUR CLUSTER

1 - Configure your MYSQL Servers

One of the most important things before start, is to have a perfect and stable system configuration. Mysql cluster will use theses configuration to work, and rename your instance, loose your DNS or any other base system change **can generate a general failure on your node.**

1.1 - Hostname configuration

```
S_MYSQL[*]> cat /etc/hostname
 serverdb-two-dc-1
```

1.2 - Hosts file configuration (DNS failure prevention)

```
S_MYSQL[*]> cat /etc/hosts
172.16.0.101 serverdb-one-dc-1
172.16.0.102 serverdb-two-dc-1
172.16.1.101 serverdb-one-dc-2
```

2 - Clean everything (if required) and install

(Optional) If one installation is existing and clean before to process:

```
S_MYSQL[*]>  apt-get purge mysql-commercial-server
S_MYSQL[*]>  dpkg -i mysql-commercial-server_8.0.11+commercial-1ubuntu16.04_amd64.deb
```

Wwe use the commercial Oracle mysql package, you can find theses packages on Oracle web site, accessible with your Oracle account(if you have). For one other installation, or just testing, you can use as well the classic distribution Mysql packages.
Installation order (important):

```
S_MYSQL[*]>  dpkg -i mysql-common_8.0.11+commercial-1ubuntu16.04_amd64.deb

S_MYSQL[*]>  dpkg -i mysql-commercial-client-core_8.0.11+commercial-1ubuntu16.04_amd64.deb
S_MYSQL[*]>  dpkg -i mysql-commercial-client_8.0.11+commercial-1ubuntu16.04_amd64.deb
S_MYSQL[*]>  dpkg -i mysql-client_8.0.11+commercial-1ubuntu16.04_amd64.deb


S_MYSQL[*]>  dpkg -i mysql-commercial-server-core_8.0.11+commercial-1ubuntu16.04_amd64.deb
S_MYSQL[*]>  dpkg -i mysql-commercial-server_8.0.11+commercial-1ubuntu16.04_amd64.deb
                # --> Ssetting up a root password, NOT a blank password.
S_MYSQL[*]>  dpkg -i mysql-server_8.0.11+commercial-1ubuntu16.04_amd64.deb
```

3 - Mysql configuration and optimisation (optional)

This part is optional for two reason: Your cluster will be able to work with the default mysql configuration and you can apply theses changes after.

Files configurations in appendix

4 - Instances Initialisation

Now that you have Mysql installed in all your servers, you have to prepare each mysql server to become a part of a cluster.

To do that, use Mysql Shell:

```
S_MYSQL[*]> mysqlsh --uri root@localhost:3306
MySQL  localhost:3306 ssl  JS > dba.configureInstance();
Please select an option [1]:  1
Account Host: 172.16.%
# Account Host: 172.16.% = Secure and production Network  !

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
```

5 - Init cluster on the first instance

The first instance will become automatically PRIMARY, if you have a preference for your R/W instance (physical localisation...) this parameter can be important, but you have the possibility to change (management part).

```
S_MYSQL[1]> mysqlsh --uri root@localhost:3306
S_MYSQL[1]> MySQL  localhost:3306 ssl  JS > \c root@serverdb-one-dc-2
#  \c root@serverdb-one-dc-2  = Connection to primary
S_MYSQL[1]>MySQL  serverdb-one-dc-2:33060+ ssl  JS > var cluster =
dba.createCluster('MyClusterName')
# MyClusterName= Cluster Name
S_MYSQL[1]>MySQL  serverdb-one-dc-2:33060+ ssl  JS > var cluster = dba.getCluster()
S_MYSQL[1]>MySQL  serverdb-one-dc-2:33060+ ssl  JS > cluster.status()
{
    "clusterName": "MyClusterName",
    "defaultReplicaSet": {
        "name": "default",
        "primary": "serverdb-one-dc-2:3306",
        "ssl": "REQUIRED",
        "status": "OK_NO_TOLERANCE",
        "statusText": "Cluster is NOT tolerant to any failures.",
        "topology": {
            "mysqlinstancesrv1:3306": {
                "address": "serverdb-one-dc-2:3306",
                "mode": "R/W",
                "readReplicas": {},
                "role": "HA",
                "status": "ONLINE"
            }
        }
    },
    "groupInformationSourceMember": "mysql://root@serverdb-one-dc-2:3306"
```

6 - Add other instances

At this step, you have your cluster with just one instance (primary), now we will join the secondary.

```
S_MYSQL[1]> MySQL  serverdb-one-dc-2:33060+ ssl  JS > cluster.addInstance('root@serverdb-one-dc-1:3306');
S_MYSQL[1]> MySQL  serverdb-one-dc-2:33060+ ssl  JS > cluster.addInstance('root@ serverdb-two-dc-1:3306');
```

7 - Check your cluster

```
S_MYSQL[1]> MySQL  serverdb-one-dc-2:33060+ ssl  JS > cluster.status()
MySQL  serverdb-one-dc-2:33060+ ssl  JS > cluster.status()
{
    "clusterName": "MyClusterName",
    "defaultReplicaSet": {
        "name": "default",
        "primary": "serverdb-one-dc-2:3306",
        "ssl": "REQUIRED",
        "status": "OK",
        "statusText": "Cluster is ONLINE and can tolerate up to ONE failure.",
        "topology": {
            "serverdb-one-dc-2:3306": {
                "address": "serverdb-one-dc-2:3306",
                "mode": "R/W",
                "readReplicas": {},
                "role": "HA",
                "status": "ONLINE"
            },
            "serverdb-one-dc-1:3306": {
                "address": "serverdb-one-dc-1:3306",
                "mode": "R/O",
                "readReplicas": {},
                "role": "HA",
                "status": "ONLINE"
            },
            " serverdb-two-dc-1:3306": {
                "address": " serverdb-two-dc-1:3306",
                "mode": "R/O",
                "readReplicas": {},
                "role": "HA",
                "status": "ONLINE"
            }
        }
    },
    "groupInformationSourceMember": "mysql://root@serverdb-one-dc-2:3306"
}
```

If everything is OK, you should have your cluster with all instances ONLINE and one R/W and two R/O.

PART 2: BOOTSTRAP YOUR ROUTER

1 - Prepare your environment

1.1 - Hostname

```
S_WEB[*]> cat /etc/hostname
serverweb-two-dc-1
```

1.2 - Hosts file

```
S_WEB[*] > cat /etc/hosts
172.16.0.101 serverdb-one-dc-1
172.16.0.102  serverdb-two-dc-1
172.16.1.101 serverdb-one-dc-2
...
```

2 - Install mysql_router

```
S_WEB[*] > dpkg -i mysql-router-commercial_8.0.11+commercial-1ubuntu16.04_amd64.deb
```

3 - Create a root user

Mysql router require a user with lot of access to be init (root type). This root user will be use to create a dedicated router user on your cluster during the initialisation.

```
mysql[PRIMARY]>
CREATE USER 'root'@'10.150.16.%' IDENTIFIED BY 'agoodpassword';
GRANT ALL PRIVILEGES ON *.* TO 'root'@'10.150.16.%' WITH GRANT OPTION;
FLUSH PRIVILEGES;
```

10.150.16.% = router network

4 - Generate on each server a mysql-router configuration

```
S_WEB[*] > /etc/mysqlrouter# mysqlrouter --bootstrap root@[RW-instance]:3306 --user=mysqlrouter --
name=serverweb-one-dc-1
....
Classic MySQL protocol connections to cluster 'MyClusterName':
- Read/Write Connections: localhost:6446
- Read/Only Connections: localhost:6447
X protocol connections to cluster 'MyClusterName':
- Read/Write Connections: localhost:64460
- Read/Only Connections: localhost:64470
```

4.1 - Change the bind address (Optional)

There are no reason to use 0.0.0.0 bind address for a router when the router is in the same server than your application.

```
S_WEB[*] > sed -i 's/0.0.0.0/127.0.0.1/g' /etc/mysqlrouter/mysqlrouter.conf
S_WEB[*] > restart mysql router
```

5 - Verify

```
root@serverweb-two-dc-1:/etc/mysqlrouter# netstat -lptn|grep router
tcp        0      0 127.0.0.1:64470         0.0.0.0:*               LISTEN      75763/mysqlrouter
tcp        0      0 127.0.0.1:64460         0.0.0.0:*               LISTEN      75763/mysqlrouter
tcp        0      0 127.0.0.1:6446          0.0.0.0:*               LISTEN      75763/mysqlrouter
tcp        0      0 127.0.0.1:6447          0.0.0.0:*               LISTEN      75763/mysqlrouter
```

PART 3: CONFIGURE KEEPALIVED (OPTION 2 ONLY)

1 - Prepare your environment

If you are following exactly the same implementation, your servers keepalived should be implemented on each mysql nodes. No more specifics configurations is required.

2 - Install Keepalived

```
S_HA[*] > apt-get install keepalived
```

3 - Configure Keepalived

Full configuration available in appendix !

Do not forget to change *router_id* in *global def* (SRV1,2,3...)

```
global_defs {
   router_id innodbrouter_SRV1
```

PART 4 : GRAPH AND SUPERVISION (Optional)

1 - Create a read only user

This user will be able to read the status on your database.

```
mysql[PRIMARY]>
CREATE USER 'cluster_ro'@'%' IDENTIFIED BY 'agoodpassword2';
GRANT SELECT ON mysql_innodb_cluster_metadata.* TO 'cluster_ro'@'%';
GRANT SELECT ON performance_schema.global_status TO 'cluster_ro'@'%';
GRANT SELECT ON performance_schema.replication_applier_configuration TO 'cluster_ro'@'%';
GRANT SELECT ON performance_schema.replication_applier_status TO 'cluster_ro'@'%';
GRANT SELECT ON performance_schema.replication_applier_status_by_coordinator TO
'cluster_ro'@'%';
GRANT SELECT ON performance_schema.replication_applier_status_by_worker TO 'cluster_ro'@'%';
GRANT SELECT ON performance_schema.replication_connection_configuration TO 'cluster_ro'@'%';
GRANT SELECT ON performance_schema.replication_connection_status TO 'cluster_ro'@'%';
GRANT SELECT ON performance_schema.replication_group_member_stats TO 'cluster_ro'@'%';
GRANT SELECT ON performance_schema.replication_group_members TO 'cluster_ro'@'%';
```

2 - Supervision

<To do>

3 - Graph

<To do>

4 - Script example

This script will be able to read directly in your database, the cluster status and show him.

vi /usr/local/bin/mysql_cluster_status

```
#!/usr/bin/env bash

# Bash script to view cluster status
# By Thomas Guiseppin
# Date 05/06/18

HOST="127.0.0.1"
USER="cluster_ro"
PW="agoodpassword2"
NBNODES=3

mysql -u${USER} -p${PW} -h${HOST} performance_schema -e 'select
MEMBER_HOST,MEMBER_STATE,MEMBER_ROLE from replication_group_members;' 2> /dev/null
NMNODESCLU=`mysql -u${USER} -p${PW} -h${HOST} performance_schema -ss -e 'select
count(MEMBER_HOST) from replication_group_members;' 2> /dev/null`
if ! [ "${NBNODES}" -eq "${NMNODESCLU}" ]; then
        echo -e "\e[1;31m/!\ NODES MISSING \e[0m"
fi
```

```
root@serverdb-one-dc-2:~# mysql_cluster_status
+--------------+--------------+-------------+
| MEMBER_HOST  | MEMBER_STATE | MEMBER_ROLE |
+--------------+--------------+-------------+
| serverdb-one-dc-1 | ONLINE       | SECONDARY   |
|  serverdb-two-dc-1 | ONLINE       | SECONDARY   |
| serverdb-one-dc-2 | ONLINE       | PRIMARY     |
+--------------+--------------+-------------+
```

PART 5 : BACKUP SYSTEM (Optional)

4 - Script example

This script will be able to read directly in your database, the cluster status and show him.

vi /usr/local/bin/mysql_cluster_status

```
# Connections settings
HOST=127.0.0.1
USER=databaseuser_in_ro
PW=password
PORT=6447 # Read port

# ENV Var
MYSQL=/usr/bin/mysql
MYSQLDUMP=/usr/bin/mysqldump

# General var
DATE=`date +%Y-%m-%d`
BACKUPDIR=/var/backups/databases
LOGS=/var/log/backups_sql.log

databases=`${MYSQL} -u${USER} -p${PW} -h${HOST} -P${PORT}  -e "SHOW DATABASES;" | grep -
Ev "(Database|information_schema|performance_schema)"`
mkdir -p ${BACKUPDIR}/${DATE}/ && chmod 600 ${BACKUPDIR}/${DATE}/

for db in ${databases}; do
    echo "DUMP: "  ${BACKUPDIR}/${DATE}/${db}.gz >> ${LOGS}
    ${MYSQLDUMP} -u${USER} -p${PW} -h${HOST} -P${PORT} --single-transaction --routines --triggers
--opt --databases ${db} | gzip > ${BACKUPDIR}/${DATE}/${db}.gz
done

# Purge OLD backup  +7jours
find ${BACKUPDIR}/ -name "*.gz" -mtime +7 -exec rm {} \;
```

PART 6 : MANAGEMENT

1 - Cluster status

```
S_MYSQL[PRIMARY]>mysqlsh --uri root@localhost:3306 -p
S_MYSQL[PRIMARY]>MySQL  localhost:3306 ssl  JS > \c root@serverdb-one-dc-2
S_MYSQL[PRIMARY]>MySQL  serverdb-one-dc-2:33060+ ssl  JS > var cluster = dba.getCluster()
S_MYSQL[PRIMARY]>MySQL  serverdb-one-dc-2:33060+ ssl  JS > cluster.status()
```

States as visible to the member itself:

- OFFLINE
- RECOVERING
- ERROR
- ONLINE

States as visible to other members:

- RECOVERING
- UNREACHABLE
- ONLINE

The group states shown by the cluster.status() command are:

- OK – is shown when all members belonging are ONLINE and there is enough redundancy to tolerate at least one failure.
- OK_PARTIAL – when one or more members are unavailable, but there's still enough redundancy to tolerate at least one failure.
- OK_NO_TOLERANCE – when there are enough ONLINE members for a quorum to be available, but there's no redundancy. A two member group has no tolerance, because if one of them becomes UNREACHABLE, the other member can't form a majority by itself; which means you will have a database outage. But unlike in a single member group, at least your data will still be safe on at least one of the nodes.
- NO_QUORUM – one or more members may still be ONLINE, but cannot form a quorum. In this state, your cluster is unavailable for writes, since transactions cannot be executed. However, read-only queries can still be executed and your data is intact and safe.
- UNKNOWN – this state is shown if you're executing the status() command from an instance that is not ONLINE or RECOVERING. In that case, try connecting to a different member.
- UNAVAILABLE – this state is shown in the diagram but will not be displayed by the cluster.status() command. In this state, ALL members of the group are OFFLINE. They may still be running, but they're not part of the group anymore. This can happen if all members restart without rejoining, for example.

2 - Insert a new instance

```
S_MYSQL[PRIMARY] > MySQL  serverdb-one-dc-2:33060+ ssl  JS > cluster.addInstance('root@ serverdb-two-dc-1:3306');
```

3 - Re-insert a MISSING instance

```
S_MYSQL[PRIMARY] > MySQL  serverdb-one-dc-2:33060+ ssl  JS > cluster.rejoinInstance('root@ serverdb-two-dc-1:3306');
```

If your instance was out of cluster for a while, this action can take long time or fail. In case of fail, please check your logs to determine the problem and fix-it.

4 - Force quorum

Typical error: (Master on cluster status =) "statusText": "Cluster has no quorum as visible from 'localhost:3306' and cannot process write transactions. 2 members are not active",

If so many members of your replica set become UNREACHABLE that it doesn't have a majority anymore, it will no longer have a quorum and can't take decisions on any changes. That includes user transactions, but also changes to the group's topology. That means that even if a member that became UNREACHABLE returns, it will be unable to rejoin the group for as long as the group is blocked.

Force the Quorum on your **last PRIMARY** available.

```
S_MYSQL[PRIMARY] > cluster.forceQuorumUsingPartitionOf("root@serverdb-one-dc-2:3306")
```

And rejoin the others instances:

```
S_MYSQL[PRIMARY] > cluster.rejoinInstance('root@ serverdb-two-dc-1:3306')
```

5 - Re-bootstrap your cluster after general fail

If somehow all your members are now OFFLINE, you can only recover the group if you "bootstrap" the group again, out of a single seed member. To perform that, you need to use the dba.restoreFroMyClusterNameompleteOutage() command on a designed seed instance, and then rejoinInstance() on the remaining members until you have your cluster fully restored.

This action must be on your **presumed last** master. ... Information that you can find with your mysql logs or by your supervision !

```
S_MYSQL[PRIMARY] > dba.rebootClusterFroMyClusterNameompleteOutage('MyClusterName')
```

5.1 - Other messages:

Dba.rebootClusterFroMyClusterNameompleteOutage: The active session instance isn't the most updated in comparison with the ONLINE instances of the Cluster's metadata. Please use the most up to date instance: ' serverdb-two-dc-1:3306'. (RuntimeError)

Mysql determine that the last PRIMARY before the general fail was **serverdb-two-dc-1** and you are working on **serverdb-one-dc-2**. If you are ***SURE*** that you want start with the data from **serverdb-one-dc-2,** stop mysql on **serverdb-two-dc-1** and process :

```
MySQL  serverdb-one-dc-2:33060+ ssl  JS >
dba.rebootClusterFroMyClusterNameompleteOutage('MyClusterName')

Reconfiguring the cluster 'MyClusterName' from complete outage...

The instance 'serverdb-one-dc-1:3306' was part of the cluster configuration.
Would you like to rejoin it to the cluster? [y/N]: y

Could not open a connection to ' serverdb-two-dc-1:3306': 'Can't connect to MySQL server on '
serverdb-two-dc-1' (111)'
Would you like to remove it from the cluster's metadata? [y/N]: y

The cluster was successfully rebooted.

<Cluster:MyClusterName>
```

After this process, probably that you will have to rebuilt your secondary...

6 - Remove one instance:

If these instance is still ONLINE:

```
S_MYSQL[PRIMARY] > cluster.removeInstance('root@serverdb-one-dc-1:3306', { force: true})
```

Any other status:

```
S_MYSQL[PRIMARY] > cluster.removeInstance('root@serverdb-one-dc-1:3306')
```

7 - View cluster status directly from mysql

Mysql shell is just a wrapper for mysql data. You can have a lot more information directly in mysql database.

7.1 - Cluster general status

```
use performance_schema;
select MEMBER_HOST,MEMBER_STATE,MEMBER_ROLE from replication_group_members;
```

7.2 - Secondary replication GTID

You have the possibility to follow the replication status directly on your mysql server.

Maybe that for you, it's working like a master-slave system, but in reality, it's a MASTER-MASTER system, with writing not allowed on some nodes. It's why you have to check the MASTER STATUS on your secondary and not the SLAVE STATUS .

```
mysql (PRIMARY) > SHOW MASTER STATUS;
+---------------+-----------+--------------+------------------+-------------------------------------------------------------------------------------------
-------------------+
| File          | Position  | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set
|
+---------------+-----------+--------------+------------------+-------------------------------------------------------------------------------------------
-------------------+
| binlog.000012 | 210175461 |              |                  | 81f7c9c0-6576-11e8-a1d0-0050568370b9:1-14,
c9e096d7-6576-11e8-afbb-0050568370b9:1-15090408 |
+---------------+-----------+--------------+------------------+-------------------------------------------------------------------------------------------
-------------------+
```

```
mysql (SECONDARY ) > SHOW MASTER STATUS;
+---------------+-----------+--------------+------------------+-------------------------------------------------------------------------------------------
-------------------+
| File          | Position  | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set
|
+---------------+-----------+--------------+------------------+-------------------------------------------------------------------------------------------
-------------------+
| binlog.000015 | 179055915 |              |                  | 81f7c9c0-6576-11e8-a1d0-0050568370b9:1-14,
c9e096d7-6576-11e8-afbb-0050568370b9:1-15090400 |
+---------------+-----------+--------------+------------------+-------------------------------------------------------------------------------------------
```

File = Binlog file currently used by your mysql server. It's a local binlog file, this value can change from a node to an other. This binlog file is generally situated in /var/lib/mysql.

Position = Cursor position in your binlog file.
Binlog_Do_DB = Not in use
Binlog_Ignore_DB = Not in use
Executed_Gtid_Set = pool of gtid, you can have multiples values (two or more).

How to determine if my replication is in late ? The most important value is the second part of your GTID: **1-15090408 and 1-15090400** . If this two value are close, probably that the synchronisation is ok. But if you have a large difference between your primary and secondary, it's mean that your secondary is in late, due to lot new transactions on the primary or the secondary is currently in recovering and trying to rejoin the primary.
If your **secondary** is in **recovering** and his value doest not change, your cluster is *just* broken.

GTID Processing details available in appendix.

PART 7 : RECOVERY FROM ZERO

Follow this part **if** :

- You have loose all data on a secondary server
- Your secondary is too in late and can't recovery from binary transaction logs
- You are working on a migration process
- You are adding a new secondary server

1 - Remove the bad instance if existing (failure...)

```
S_MYSQL[PRIMARY]> mysqlsh --uri root@localhost:3306
S_MYSQL[PRIMARY]>MySQL  localhost:3306 ssl  JS > \c root@serverdb-one-dc-2
S_MYSQL[PRIMARY]>MySQL  serverdb-one-dc-2:33060+ ssl  JS > var cluster = dba.getCluster()
S_MYSQL[PRIMARY]>MySQL  serverdb-one-dc-2:33060+ ssl  JS >
cluster.removeInstance('root@serverdb-one-dc-1:3306');
```

2 - From a ONLINE secondary or your master, dump your data (or use a recent backup)

```
S_MYSQL[PRIMARY]>   mysqldump -p --all-databases --triggers --routines  --single-transaction >
/tmp/all.sql
```

It's very important to dump your data with GTID transaction activate (default). When you will import this dump in your secondary, the transactions will start from the GTID saved in your dump.

3 - Transfer backup on secondary

As you want: rsync, scp, usbkey, pigeon

rsync -avzf /tmp/all.sql root@mysecondary:/home/

4 - Reset master on Secondary

*We assume that your new secondary is **ready** to become a mysql cluster member. If it's not the case, please apply **PART 2 - Section 1,2,3,4***

When you start your new secondary, probably that it did some transactions... You can't synchronise two server without clean the previous transactions.

THIS ACTION IS ON YOUR NEW SECONDARY

```
S_MYSQL[SECONDARY ]> mysql> SET GLOBAL super_read_only = 0;
S_MYSQL[SECONDARY ]> mysql> reset master;
```

Do not generate new transactions on your secondary (change/add user...)

5 - Import

```
S_MYSQL[SECONDARY ]>   mysql -p < /tmp/all.sql
```

6 - Insert your instance

Now you have the possibility to (re)insert your instance in the cluster.

Probably that this instance will stay on "RECOVERING" during some minutes, time to apply the modifications from your primary, between your dump and instance insertion.

```
S_MYSQL[PRIMARY]>MySQL  serverdb-one-dc-2:33060+ ssl  JS >
cluster.addInstance('root@serverdb-one-dc-1:3306');
```

PART 8 - LOGS : Read & Analyse

Logs are available on this file: /var/log/mysql/error.log

Warning : Timestamp

```
[Warning] [MY-010956] [Server] Invalid replication timestamps: original commit timestamp is more recent than the immediate commit timestamp. This may be an issue if delayed replication is active. Make sure that servers have their clocks set to the correct time. No further message will be emitted until after timestamps become valid again.
 [Warning] [MY-010957] [Server] The replication timestamps have returned to normal values.
```

A timestamp is a sequence of characters or encoded information identifying when a certain event occurred, usually giving date and time of day, sometimes accurate to a small fraction of a second. It's very important to keep your system with the good date, without this synchronisation, your cluster will fail.

Warning : member unreachable

```
[Warning] [MY-011493] [Repl] Plugin group_replication reported: 'Member with address serverdb-two-dc-1:3306 has become unreachable.'
```

Your member has become unreachable probably because you have : service mysql down, a network link down, set a new firewall rule or your VM is not available.

Warning : member removed

```
[Warning] [MY-011499] [Repl] Plugin group_replication reported: 'Members removed from the group: serverdb-one-dc-1:3306'
```

When a member become unreachable, it will be exclude. After have fixed your issue, use cluster.rejoinInstance("root@member") to come back in normal production state.

Warning : Ip address not resolved

```
[Warning] [MY-010055] [Server] IP address '10.210.3.11' could not be resolved: Name or service not known
```

When you create a MySQL user username@ serverdb-two-dc-1 MySQL has to do a reverse lookup on every IP address connecting to it to determine whether they are part of serverdb-two-dc-1

Error : group replication pushing message

```
[ERROR] [MY-011735] [Repl] Plugin group_replication reported: '[GCS] Error pushing message into group communication engine.
```

General transaction fail. Check your cluster status, it's probably broken.

Error : group replication reach majority (quorum)

```
[ERROR] [MY-011495] [Repl] Plugin group_replication reported: 'This server is not able to reach a majority of members in the group. This server will now block all updates. The server will remain blocked until contact with the majority is restored. It is possible to use group_replication_force_members to force a new group membership.'
```

You have loose your quorum, fix your cluster and follow the procedure Part 5 - 4.

Error : Maximum number of connection

```
[ERROR] [MY-011287] [Server] Plugin mysqlx reported: '25.1: Maximum number of authentication attempts reached, login failed.'
```

Main possibilities:

- Your application is not set correctly
- Your cluster is not set correctly (password, ip acl...)
- Someone try to brute force your database access

Error : Replication cannot replicate

```
[ERROR] [MY-011287] [Repl] Cannot replicate to server with server_uuid='f08fbf12-8b6e-11e8-9938-0050568343e5' because the present server has purged required binary logs. The connecting server needs to replicate the missing transactions from elsewhere, or be replaced by a new server created from a more recent backup. To prevent this error in the future, consider increasing the binary log expiration period on the present server. The missing transactions are '81f7c9c0-6576-11e8-a1d0-0050568370b9:1-14, c9e096d7-6576-11e8-afbb-0050568370b9:1-7414160'.
```

This message is explicit, your binary logs are too old. Apply Part 6 (Recovery from zero) to fix it.

Error : Failed open relay log

```
[ERROR] [MY-010544] [Repl] Failed to open the relay log './ serverdb-two-dc-1-relay-bin-group_replication_recovery.000001' (relay_log_pos 4).
```

Access problem to the binaries logs: Check your my.cnf configuration.

Error : Master initialization

```
[ERROR] [MY-010426] [Repl] Secondary: Failed to initialize the master info structure for channel 'group_replication_recovery'; its record may still be present in 'mysql.secondary_master_info' table, consider deleting it.
```

You have probably tried to configure again your cluster, with a previous configuration existing. If you want to reconfigure your cluster from zero, clean it before.

Error : Network failure

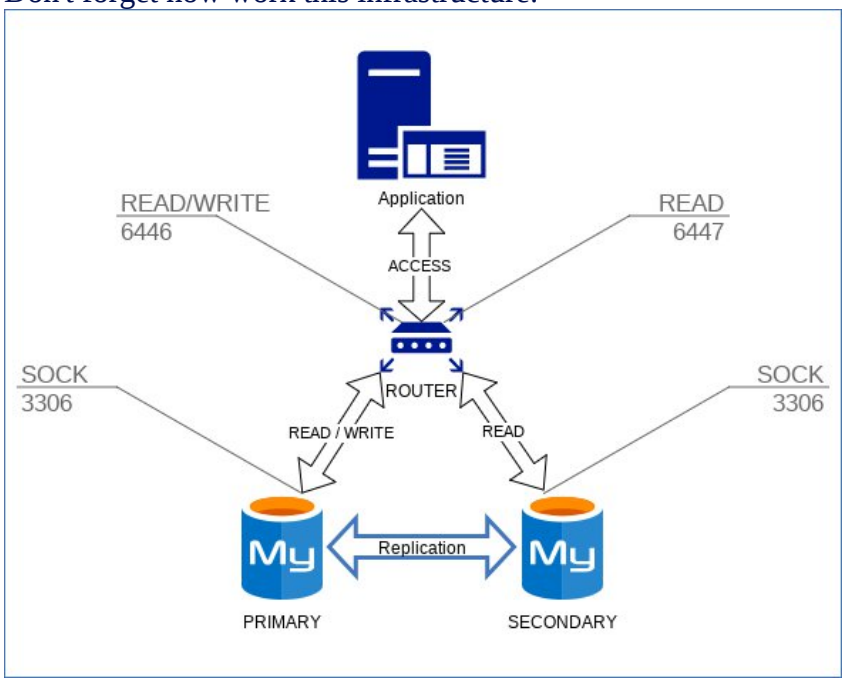
```
[ERROR] [MY-011505] [Repl] Plugin group_replication reported: 'Member was expelled from the group due to network failures, changing member status to ERROR.'
```

This message is explicit, your network link has failed.

PART 9 - HOW TO USE

1 - Connect your application

Don't forget how work this infrastructure:



Mysql router is NOT a SQL QUERY router, it's a basic router based on a port system. If you want a query sql routing system, this part have to be managed by your application with a specific driver (if existing). Moodle doest not provide this driver, it's why all requests will arrive on the same server (PRIMARY). To connect moodle to this server, use the **router** IP/PORT(127.0.0.1:6446).

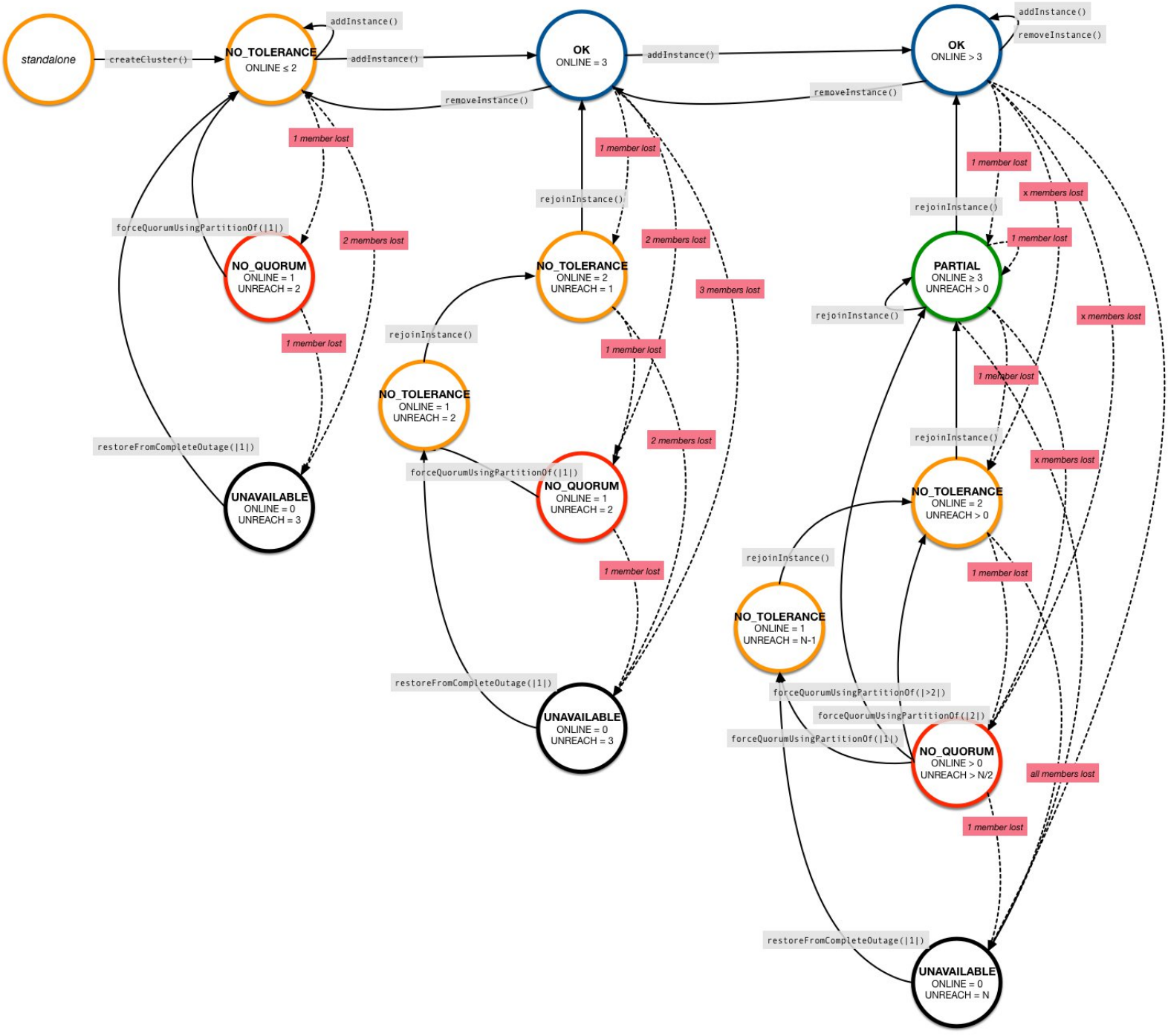
2 - Configuration example (Moodle)

```
$CFG->dbtype    = 'mysqli';
$CFG->dblibrary = 'native';
$CFG->dbhost    = '127.0.0.1';
$CFG->dbname    = 'moodle';
$CFG->dbuser    = 'myusermoodledb ';
$CFG->dbpass    = '**********';
$CFG->prefix    = 'mdl_';
$CFG->dboptions = array (
  'dbpersist' => 0,
  'dbport' => '6446',
  'dbsocket' => '',
  'dbcollation' => 'utf8mb4_unicode_ci',
);
```

This configuration is specific to moodle. If you use one other CMS, you will probably find something similar to this configuration in your config file.

PART 10 - APPENDIX

Cluster tolerance process



Mysql 8 - my.cnf configuration example

```
#
# The MySQL database server configuration file.
#

[client]
port                    = 3306
socket                  = /var/run/mysqld/mysqld.sock

[mysqld_safe]
socket                  = /var/run/mysqld/mysqld.sock
nice                    = 0

[mysqld]
user                    = mysql
pid-file                = /var/run/mysqld/mysqld.pid
socket                  = /var/run/mysqld/mysqld.sock
bind-address            = 0.0.0.0
log-error               = /var/log/mysql/error.log

port                    = 3306
basedir                 = /usr
datadir                 = /var/lib/mysql
tmpdir                  = /tmp

default_authentication_plugin  = mysql_native_password
binlog_expire_logs_seconds     = 691200
skip-external-locking
# UNIQUE
server_id               = 1
relay-log               = serverdb-one-dc-2-relay-bin

# Fine Tuning
sort_buffer_size        = 16M
tmp_table_size          = 512M
max_heap_table_size     = 128M
max_connections         = 300
connect_timeout         = 5
wait_timeout            = 200
max_allowed_packet      = 16M
bulk_insert_buffer_size = 16M
thread_stack            = 192K
thread_cache_size       = 2048

# MyISAM
key_buffer_size         = 128M
myisam_recover_options  = BACKUP
myisam_repair_threads   = 16
myisam_sort_buffer_size = 256M
concurrent_insert       = 2
read_buffer_size        = 2M
read_rnd_buffer_size    = 1M
```

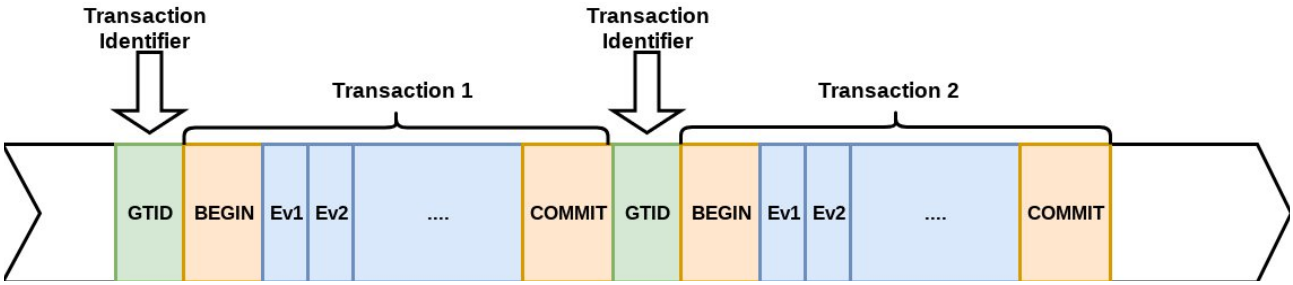
```
## InnoDB tuning
innodb_file_per_table          = 1
innodb_buffer_pool_size        = 4G
innodb_log_file_size           = 512M
innodb_log_buffer_size         = 512M
innodb_thread_concurrency      = 24
innodb_read_io_threads         = 12
innodb_write_io_threads        = 12
innodb_open_files              = 10000
innodb_io_capacity             = 1000
innodb_lock_wait_timeout       = 60
innodb_flush_method            = O_DIRECT
innodb_doublewrite             = 0
innodb_use_native_aio          = 0
innodb_flush_log_at_trx_commit = 0
default_storage_engine         = InnoDB

[mysqldump]
quick
quote-names
max_allowed_packet             = 16M

[isaMyClusterNamehk]
key_buffer                     = 16M

!includedir /etc/mysql/conf.d/
```

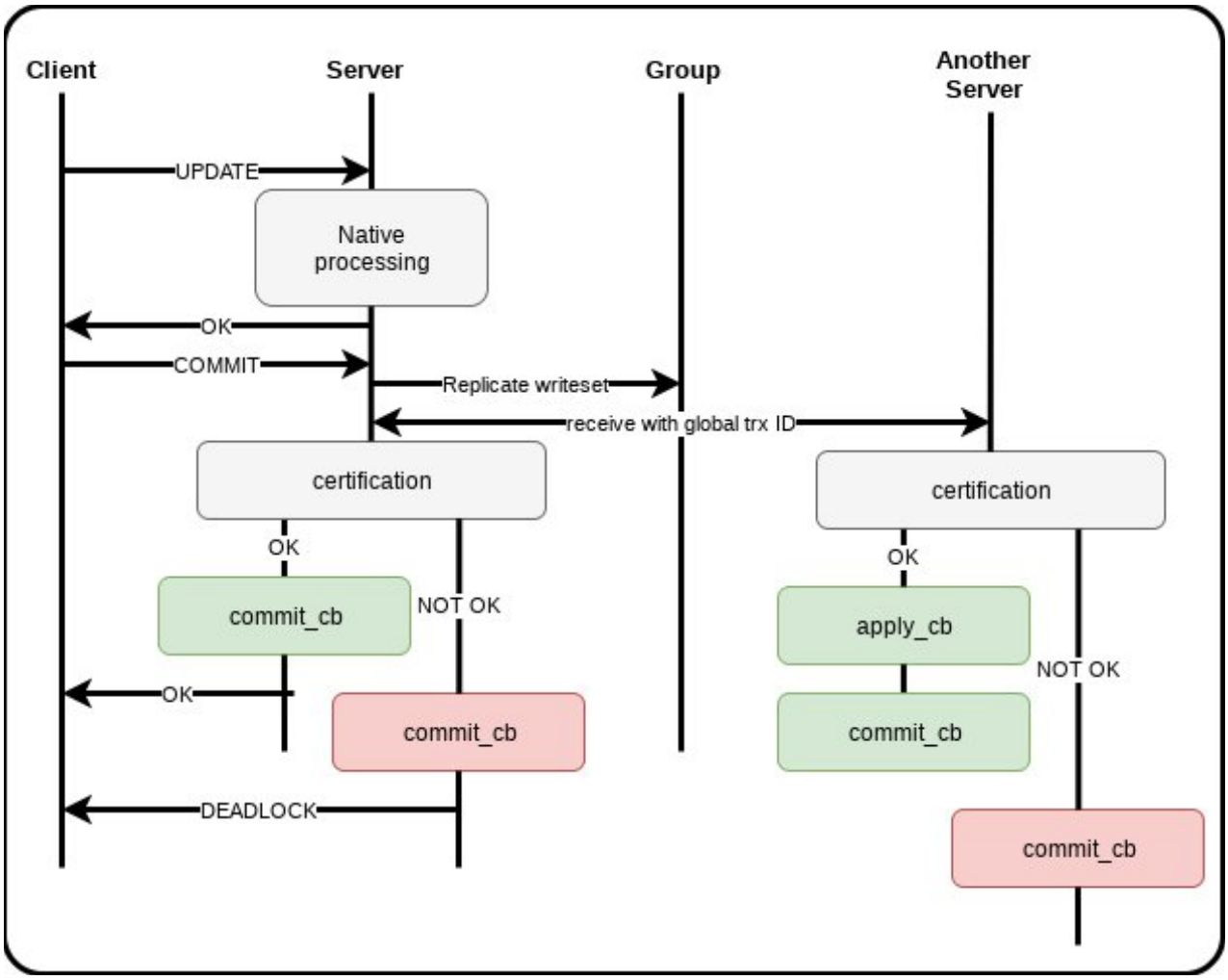
In bold = specific for one server, change it.

Keepalived configuration

```
global_defs {
   router_id innodbrouter_SRV1
   notification_email {
     tech@mydomain.org
   }
   notification_email_from keepalived@mydomain.org
   smtp_server localhost
   smtp_connect_timeout 30
}

vrrp_instance mysql {
  state BACKUP
  virtual_router_id 100
  interface enp0s8
  priority 100

   authentication {
       auth_type AH
       auth_pass 5d5d5df5d5
   }

  virtual_ipaddress {
    172.16.0.200/24 dev enp0s8 label enp0s8:MYSQL
  }

}

!! {{{ MYSQL-VIP-WRITE-READ
virtual_server 172.16.0.200 6446 {
    delay_loop 6
    lb_algo lc
    lb_kind NAT
    protocol TCP

    real_server 172.16.0.111 6446 {
        TCP_CHECK {
            connect_timeout 10
        }
    }

    real_server 172.16.0.112 6446 {
        TCP_CHECK {
            connect_timeout 10
        }
    }

    real_server 172.16.0.113 6446 {
        TCP_CHECK {
            connect_timeout 10
        }
    }

}
!! }}} MYSQL-VIP-WRITE-READ

!! {{{ MYSQL-VIP-READ-ONLY
virtual_server 172.16.0.200 6447 {
    delay_loop 6
    lb_algo lc
    lb_kind NAT
    protocol TCP

    real_server 172.16.0.111 6447 {
        TCP_CHECK {
            connect_timeout 10
        }
    }

    real_server 172.16.0.112 6447 {
        TCP_CHECK {
            connect_timeout 10
        }
    }

    real_server 172.16.0.113 6447 {
        TCP_CHECK {
            connect_timeout 10
        }
    }

}
!! }}} MYSQL-VIP-READ-ONLY
```

TRANSACTION PROCESS





PART 11 - REFERENCES

- https://dev.mysql.com/doc/refman/8.0/en/mysql-innodb-cluster-userguide.html
- https://mysqlserverteam.com/mysql-innodb-cluster-8-0-a-hands-on-tutorial
- https://www.percona.com/blog/2009/05/14/why-mysqls-binlog-do-db-option-is-dangerous/
- http://in355hz.iteye.com/blog/1770401 (Chinese)
- http://blog.51cto.com/16769017/1871828 (Chinese)