MySQL8.0.40 MGR集群安装部署及管理

原创 飞天 2024-12-25

MySQL MGR集群介绍

1、MGR集群是在Paxos分布式算法基础上实现的，以提供不同server之间的分布式协调。
2、是一种基于share-nothing的复制方案，每个server节点都有完整的副本，最少需要3个节点才能组成集群。它要求组中大多数节点在线才能达到法定票数，从而对一个决策做出一致的决定。大多数指的是N/2+1(N是组中目前节点总数)，例如目前组中有3个节点，则需要2个节点才能达到大多数的要求。
3、自带故障自动检测机制，发生故障时能自动切换到新的主节点。
4、支持单节点、多节点写入两种模式，强烈建议选用单主模式。

环境说明

主机名	ip地址	OS版本	内存、CPU	角色
node1	192.*.*.60	Centos7.9	2G 、 1个双核	主节点
node2	192.*.*.62	Centos7.9	2G 、 1个双核	从节点
node3	192.*.*.64	Centos7.9	2G 、 1个双核	从节点

数据库版本：8.0.40
Mysqlsh版本：8.0.40

安装部署

安装前准备

查看glibc版本

```
[root@node1 ~]# ldd --version
ldd (GNU libc) 2.17
Copyright (C) 2012 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
Written by Roland McGrath and Ulrich Drepper.
[root@node1 ~]#
```

下载mysql和mysqlsh

mysql8.0.40下载
下载地址：https://dev.mysql.com/downloads/mysql/

General Availability (GA) Releases | Archives

MySQL Community Server 8.0.40

Select Version: 8.0.40
Select Operating System: Linux - Generic
Select OS Version: Linux - Generic (glibc 2.17) (x86, 64-bit)

Compressed TAR Archive (mysql-8.0.40-linux-glibc2.17-x86_64.tar.xz)	8.0.40	819.2M	Download MD5: 8848417f38d75c21bc4ce778eb9d4377 | Signature
Compressed TAR Archive, Minimal Install (mysql-8.0.40-linux-glibc2.17-x86_64-minimal.tar.xz)	8.0.40	58.1M	Download MD5: 230b11ab4dbd84f1a3349559e6ed6bec | Signature
Compressed TAR Archive, Test Suite (mysql-test-8.0.40-linux-glibc2.17-x86_64.tar.xz)	8.0.40	423.1M	Download MD5: e58e8803d7e008ae4bad96e37debdc18 | Signature
Compressed TAR Archive, Minimal Install Test Suite (mysql-test-8.0.40-linux-glibc2.17-x86_64-minimal.tar.xz)	8.0.40	382.5M	Download MD5: 0e092cee5b3dcbc7790eec220fb3c80a | Signature
TAR (mysql-8.0.40-linux-glibc2.17-x86_64.tar)	8.0.40	1306.2M	Download MD5: 3c9169dded874b19e92b33eeb794167d | Signature
TAR, Minimal Install (mysql-8.0.40-linux-glibc2.17-x86_64-minimal.tar)	8.0.40	448.2M	Download MD5: 1eccbb2e7d212ce0b225cc1389224ba3 | Signature

mysqlsh8.0.40下载

下载地址：https://dev.mysql.com/downloads/shell/

General Availability (GA) Releases | Archives

MySQL Shell 8.0.40

Select Version: 8.0.40
Select Operating System: Linux - Generic
Select OS Version: Linux - Generic (glibc 2.17) (x86, 64-bit)

Compressed TAR Archive (mysql-shell-8.0.40-linux-glibc2.17-x86-64bit.tar.gz)	8.0.40	91.1M	Download MD5: 32ecfbf0bf900091911ab4b97d95135b | Signature

We suggest that you use the MD5 checksums and GnuPG signatures to verify the integrity of the packages you download.

软件下载后上传到服务器的/soft目录

```
[root@node1 ~]# cd /soft
[root@node1 soft]# ll
total 932100
-rw-r--r-- 1 root root 858950708 Dec 23 20:46 mysql-8.0.40-linux-glibc2.17-x86_64.tar.xz
-rw-r--r-- 1 root root  95517054 Dec 23 20:46 mysql-shell-8.0.40-linux-glibc2.17-x86-64bit.
[root@node1 soft]#
```

开始安装mysql（在所有节点操作）

下面以node1节点为例，node2、node3节点安装方法相同。

配置/etc/hosts

```
cat >> /etc/hosts <<EOF
192.*.*.60 node1
192.*.*.62 node2
192.*.*.64 node3
EOF
```

Oracle海量数据导出效率翻倍–sqluldr2从安装到实战全攻略
2025–07–27 125浏览

Oracle LogMiner解析重做日志操作指南
2025–07–08 65浏览

Oracle19c单机版 for centos7 图形化部署
2025–06–26 204浏览

目录

调整资源限制

```
cat >> /etc/security/limits.conf << EOF
#added by 20241224
*  soft  nofile  65535
*  hard  nofile  65535
*  soft  nproc  2048
*  hard  nproc  16384
*  soft  stack  1024
*  hard  stack  10240
EOF
```

关闭SELINUX

```
vi /etc/selinux/config
修改SELINUX=disabled
或者
sed -i 's/^SELINUX=enforcing/SELINUX=disabled/g' /etc/selinux/config
```

解压安装包

```
[root@node1 soft]# tar -xf mysql-8.0.40-linux-glibc2.17-x86_64.tar.xz -C /usr/local/
[root@node1 soft]# cd /usr/local
[root@node1 local]# ln -s mysql-8.0.40-linux-glibc2.17-x86_64 mysql
```

创建组和用户

```
groupadd mysql
useradd -g mysql -s /sbin/nologin  mysql
```

创建目录

```
mkdir -p /data/mysql3306/
chown -R mysql:mysql /data/mysql3306
```

编辑my.cnf文件

这里只是1个示例，按需修改/etc/my.cnf文件。

```
cat > /etc/my.cnf <<EOF
[client]
socket          = /data/mysql3306/mysql.sock

[mysqld]
server_id=60
log_bin=mysql-bin
log-bin-index = mysql-bin.index
enforce_gtid_consistency=ON
gtid_mode=ON

#large_pages=0

datadir=/data/mysql3306
max_connections=1000
init-connect='SET NAMES utf8mb4'
character-set-server=utf8mb4
port            = 3306
socket          = /data/mysql3306/mysql.sock
skip-external-locking
explicit_defaults_for_timestamp=true

transaction_isolation = READ-COMMITTED
max_allowed_packet = 1073741824

sort_buffer_size=524288
join_buffer_size=524288
read_buffer_size=524288
read_rnd_buffer_size=524288

internal_tmp_mem_storage_engine=MEMORY
innodb_io_capacity=10000
innodb_lru_scan_depth=100
table_definition_cache=32768
table_open_cache = 32768

innodb_read_io_threads=8
innodb_write_io_threads=8

skip_name_resolve

innodb_use_native_aio = 1
innodb_flush_method=O_DIRECT_NO_FSYNC
#innodb_buffer_pool_size = 16G
innodb_buffer_pool_size = 1G
innodb_file_per_table = 1
event_scheduler = 1

lower_case_table_names=1
slow_query_log=on
slow_query_log_file=slowquery.log
long_query_time=2

innodb_redo_log_capacity = 1G
innodb_log_buffer_size = 512M
binlog_expire_logs_auto_purge=OFF

innodb_rollback_on_timeout = on

log_bin_trust_function_creators = 1

cte_max_recursion_depth=4294967295
group_concat_max_len = 4294967295

max_prepared_stmt_count=100000

log_timestamps=SYSTEM
log_error_suppression_list='MY-013360'
log-error=/data/mysql3306/mysqld.log
pid-file=/data/mysql3306/mysqld.pid
innodb_adaptive_hash_index=OFF
```

```
transaction_write_set_extraction=XXHASH64
loose-group_replication_group_name="abcaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa"
loose-group_replication_start_on_boot=OFF
loose-group_replication_local_address= "192.*.*.60:33061"
loose-group_replication_group_seeds= "192.*.*.60:33061,192.*.*.62:33061,192.*.*.64:33061"
loose-group_replication_bootstrap_group=OFF
EOF
```

注意：node2、node3节点的/etc/my.cnf文件和node1节点的区别：

```
# node2的/etc/my.cnf文件
[mysqld]
server_id=62
loose-group_replication_local_address= "192.*.*.62:33061"
# node3的/etc/my.cnf文件
[mysqld]
server_id=64
loose-group_replication_local_address= "192.*.*.64:33061"
```

初始化MySQL

```
[root@node1 ~]# /usr/local/mysql/bin/mysqld --defaults-file=/etc/my.cnf --user=mysql --init
```

初始化后，从error.log找到root用户的临时密码

```
[root@node1 ~]# grep 'temporary password' /data/mysql3306/mysqld.log
2024-12-24T15:08:12.873076+08:00 6 [Note] [MY-010454] [Server] A temporary password is gene
```

启动mysql

```
/usr/local/mysql/bin/mysqld --defaults-file=/etc/my.cnf --user=mysql&
```

配置当前用户的环境变量

```
[root@node1 ~]# vi ~/.bash_profile
加入：
export PATH=$PATH:/usr/local/mysql/bin

source  ~/.bash_profile
```

修改root用户的密码

```
#在mysql客户端里执行
alter user 'root'@'localhost' identified by '******';
flush privileges;
```

创建管理账号

```
CREATE USER admin@'%' IDENTIFIED with caching_sha2_password BY 'XXXXXXXXXX';
GRANT  all  PRIVILEGES ON *.* TO admin@'%' WITH GRANT OPTION;
FLUSH PRIVILEGES;
```

创建mysql服务

参考之前写的《创建mysql服务》文章。

开始安装MGR

安装MGR插件，设置复制账号（所有节点操作）

```
#在mysql客户端里执行
INSTALL PLUGIN group_replication SONAME 'group_replication.so';
SET SQL_LOG_BIN=0;
CREATE USER repl@'%'  IDENTIFIED with mysql_native_password BY '******';
GRANT REPLICATION SLAVE ON *.* TO repl@'%';
FLUSH PRIVILEGES;
SET SQL_LOG_BIN=1;
#配置恢复通道
CHANGE MASTER TO MASTER_USER='repl', MASTER_PASSWORD='******' FOR CHANNEL 'group_replicatic
```

启动MGR单主模式（在主节点node1上操作）

单主模式：
group_replication_single_primary_mode = ON ，该变量在所有组成员中必须设置为相同的值。
• 该集群具有一个设置为读写模式的主节点。组中的所有其他成员都设置为只读模式（super-read-only = ON）。
• 读写节点通常是引导该组的第一个节点。加入该集群的所有其他只读节点均需要从读写节点同步数据，并自动设置为只读模式。

```
#在mysql客户端里执行
SET GLOBAL group_replication_bootstrap_group=ON;
START GROUP_REPLICATION;
SET GLOBAL group_replication_bootstrap_group=OFF;
#查看MGR集群状态
SELECT * FROM performance_schema.replication_group_members;
```

命令回显如下：

```
mysql> SET GLOBAL group_replication_bootstrap_group=ON;
Query OK, 0 rows affected (0.01 sec)

mysql> START GROUP_REPLICATION;
Query OK, 0 rows affected (1.52 sec)

mysql> SET GLOBAL group_replication_bootstrap_group=OFF;
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT * FROM performance_schema.replication_group_members;
+---------------------------+--------------------------------------+-------------+-------------+--------------+-------------+----------------+----------------------------+
| CHANNEL_NAME              | MEMBER_ID                            | MEMBER_HOST | MEMBER_PORT | MEMBER_STATE | MEMBER_ROLE | MEMBER_VERSION | MEMBER_COMMUNICATION_STACK |
+---------------------------+--------------------------------------+-------------+-------------+--------------+-------------+----------------+----------------------------+
| group_replication_applier | 9655b7d0-c1ca-11ef-98e4-00505621a1a4 | node1       |        3306 | ONLINE       | PRIMARY     | 8.0.40         | XCom                       |
+---------------------------+--------------------------------------+-------------+-------------+--------------+-------------+----------------+----------------------------+
1 row in set (0.05 sec)

mysql>
```

加入其他节点（在节点node2、node3上操作）

```
#在mysql客户端里执行
START GROUP_REPLICATION;
这一步可能会报错：
[ERROR] [MY-011526] [Repl] Plugin group_replication reported: 'This member has more execute
初次搭建集群环境，如果报错，建议执行：
reset  master ; #注意：有业务数据的时候执行要慎重

#查看MGR集群状态
SELECT * FROM performance_schema.replication_group_members;
```

命令回显如下：

```
mysql> reset master;
Query OK, 0 rows affected (0.02 sec)

mysql> START GROUP_REPLICATION;
Query OK, 0 rows affected (2.23 sec)

mysql> SELECT * FROM performance_schema.replication_group_members;
+---------------------------+--------------------------------------+-------------+-------------+--------------+-------------+----------------+----------------------------+
| CHANNEL_NAME              | MEMBER_ID                            | MEMBER_HOST | MEMBER_PORT | MEMBER_STATE | MEMBER_ROLE | MEMBER_VERSION | MEMBER_COMMUNICATION_STACK |
+---------------------------+--------------------------------------+-------------+-------------+--------------+-------------+----------------+----------------------------+
| group_replication_applier | 7b247d00-c2a9-11ef-b286-00505628bfbc | node3       |        3306 | ONLINE       | SECONDARY   | 8.0.40         | XCom                       |
| group_replication_applier | 803d29d4-c2a9-11ef-96f1-0050563b2ef8 | node1       |        3306 | ONLINE       | PRIMARY     | 8.0.40         | XCom                       |
| group_replication_applier | 831550eb-c2a9-11ef-b4f8-0050563a6146 | node2       |        3306 | ONLINE       | SECONDARY   | 8.0.40         | XCom                       |
+---------------------------+--------------------------------------+-------------+-------------+--------------+-------------+----------------+----------------------------+
3 rows in set (0.01 sec)
```

至此，MySQL8.0.40 MGR集群部署完成。

接下来，使用mysqlshell管理 MGR 集群。

开始安装mysqlshell

解压mysqlshell安装包（在所有节点操作）

下面以node1节点为例，node2、node3节点安装方法相同。

```
[root@node1 ~]# cd /usr/local
[root@node1 local]# tar -zxvf /soft/mysql-shell-8.0.40-linux-glibc2.17-x86-64bit.tar.gz
[root@node1 local]# ln -s mysql-shell-8.0.40-linux-glibc2.17-x86-64bit/ mysqlsh
```

配置当前用户的环境变量

```
vi ~/.bash_profile
加入：
export PATH=$PATH:/usr/local/mysql/bin:/usr/local/mysqlsh/bin
# 环境变量生效
source  ~/.bash_profile
```

使用 mysqlsh 管理集群

创建cluster

```
mysqlsh
shell.connect('admin@node1:3306');
var cluster = dba.createCluster('myCluster')
```

```
[root@node1 ~]# mysqlsh
MySQL Shell 8.0.40

Copyright (c) 2016, 2024, 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 Classic session to 'root@/data%2Fmysql3306%2Fmysql.sock'
Fetching schema names for auto-completion... Press ^C to stop.
Your MySQL connection id is 68
Server version: 8.0.40 MySQL Community Server - GPL
No default schema selected; type \use <schema> to set one.
 MySQL  localhost  JS > shell.connect('admin@node1:3306');
Creating a session to 'admin@node1:3306'
Fetching schema names for auto-completion... Press ^C to stop.
Closing old connection...
Your MySQL connection id is 70
Server version: 8.0.40 MySQL Community Server - GPL
No default schema selected; type \use <schema> to set one.
<ClassicSession:admin@node1:3306>
 MySQL  node1:3306 ssl  JS > var cluster = dba.createCluster('myCluster')
A new InnoDB Cluster will be created on instance 'node1:3306'.

You are connected to an instance that belongs to an unmanaged replication group.
Do you want to setup an InnoDB Cluster based on this replication group? [Y/n]: n
Dba.createCluster: Creating a cluster on an unmanaged replication group requires adoptFromGR option to be true (MYSQLSH 51315)
 MySQL  node1:3306 ssl  JS > var cluster = dba.createCluster('myCluster', {adoptFromGR: true});
A new InnoDB Cluster will be created based on the existing replication group on instance 'node1:3306'.

Creating InnoDB Cluster 'myCluster' on 'node1:3306'...

Adding Seed Instance...
Adding Instance 'node3:3306'...
Adding Instance 'node1:3306'...
Adding Instance 'node2:3306'...
Resetting distributed recovery credentials across the cluster...
Cluster successfully created based on existing replication group.

 MySQL  node1:3306 ssl  JS >
```

查看MGR集群状态

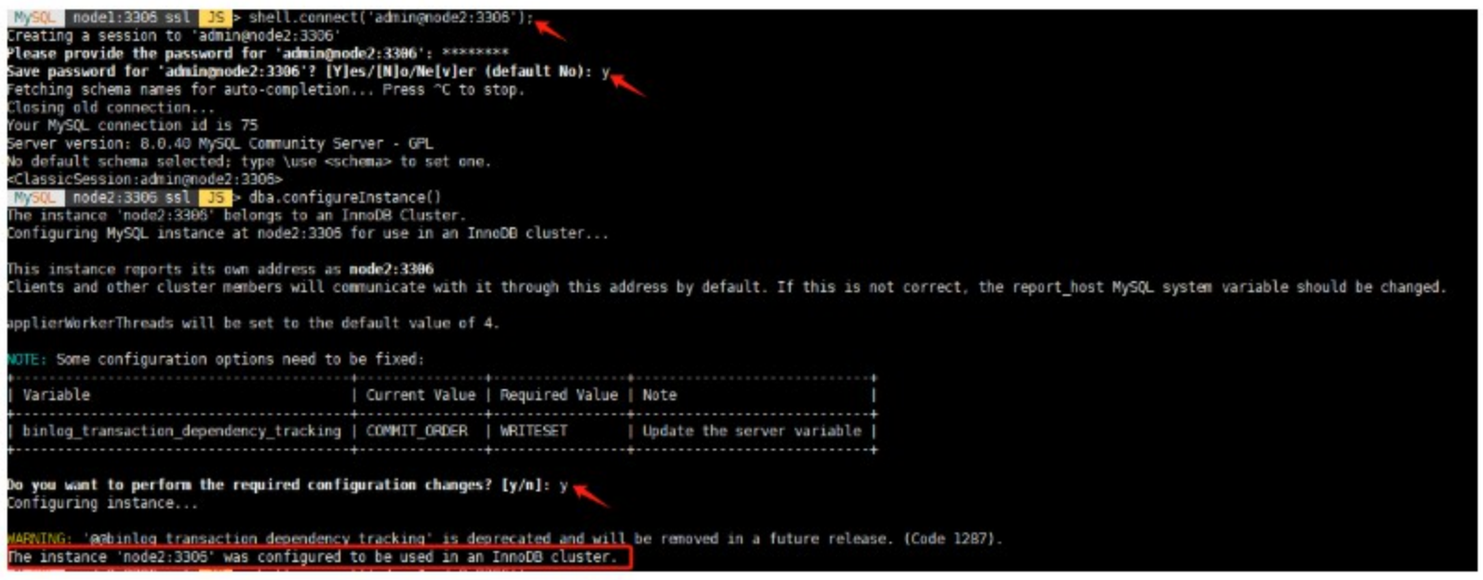
```
 MySQL  node1:3306 ssl  JS > var cluster = dba.getCluster()
 MySQL  node1:3306 ssl  JS > cluster.status();
{
    "clusterName": "myCluster",
    "defaultReplicaSet": {
        "name": "default",
        "primary": "node1:3306",
        "ssl": "DISABLED",
        "status": "OK",
        "statusText": "Cluster is ONLINE and can tolerate up to ONE failure.",
        "topology": {
            "node1:3306": {
                "address": "node1:3306",
                "instanceErrors": [
                    "NOTE: The required parallel-appliers settings are not enabled on the i
                ],
                "memberRole": "PRIMARY",
                "mode": "R/W",
                "readReplicas": {},
                "replicationLag": "applier_queue_applied",
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.40"
            },
            "node2:3306": {
                "address": "node2:3306",
                "instanceErrors": [
                    "NOTE: The required parallel-appliers settings are not enabled on the i
                ],
                "memberRole": "SECONDARY",
                "mode": "R/O",
                "readReplicas": {},
                "replicationLag": "applier_queue_applied",
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.40"
            },
            "node3:3306": {
                "address": "node3:3306",
                "instanceErrors": [
                    "NOTE: The required parallel-appliers settings are not enabled on the i
                ],
                "memberRole": "SECONDARY",
                "mode": "R/O",
                "readReplicas": {},
                "replicationLag": "applier_queue_applied",
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.40"
            }
        },
        "topologyMode": "Single-Primary"
    },
    "groupInformationSourceMember": "node1:3306"
}
```

修复问题

“instanceErrors”: [
“NOTE: The required parallel-appliers settings are not enabled on the instance. Use dba.configu reInstance() to fix it.”
]
执行下面命令：

```
shell.connect('admin@node2:3306');
dba.configureInstance()
shell.connect('admin@node3:3306');
dba.configureInstance()
```

命令回显如下：



再次查看MGR集群状态

```
 MySQL  node3:3306 ssl  JS > cluster.status()
{
    "clusterName": "myCluster",
    "defaultReplicaSet": {
        "name": "default",
        "primary": "node1:3306",
        "ssl": "DISABLED",
        "status": "OK",
        "statusText": "Cluster is ONLINE and can tolerate up to ONE failure.",
        "topology": {
            "node1:3306": {
                "address": "node1:3306",
                "memberRole": "PRIMARY",
                "mode": "R/W",
                "readReplicas": {},
                "replicationLag": "applier_queue_applied",
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.40"
            },
            "node2:3306": {
                "address": "node2:3306",
                "memberRole": "SECONDARY",
                "mode": "R/O",
                "readReplicas": {},
                "replicationLag": "applier_queue_applied",
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.40"
            },
            "node3:3306": {
                "address": "node3:3306",
                "memberRole": "SECONDARY",
                "mode": "R/O",
                "readReplicas": {},
                "replicationLag": "applier_queue_applied",
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.40"
            }
        },
        "topologyMode": "Single-Primary"
    },
    "groupInformationSourceMember": "node1:3306"
}
 MySQL  node3:3306 ssl  JS >
```

MGR 集群状态正常。

添加新节点

接下来将node4节点（ip：192.*.*.66） 加入集群。

安装并启动mysql数据库

在node4节点操作：方法同其他节点。

安装MGR插件，设置复制账号

方法同其他节点。
注意：在所有节点修改my.cnf：

```
#加入192.*.*.66:33061
loose-group_replication_group_seeds= "192.*.*.60:33061,192.*.*.62:33061,192.*.*.64:33061,19
```

```
shell.connect('admin@node4:3306');
dba.checkInstanceConfiguration('admin@node4:3306');
dba.configureInstance()
cluster.addInstance('node4:3306',{'recoveryMethod': "clone"})
#shell.options['dba.restartWaitTimeout']=50000
cluster.rescan()
```

```
MySQL node4:3306 ssl JS > dba.configureInstance()
Configuring MySQL instance at node4:3306 for use in an InnoDB cluster...

This instance reports its own address as node4:3306
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.

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 'node4:3306' was configured to be used in an InnoDB cluster.
MySQL node4:3306 ssl JS >
```

```
MySQL node4:3306 ssl JS > cluster.addInstance('node4:3306',{'recoveryMethod': "clone"})
ERROR: Instance 'node4:3306' is part of the Group Replication group but is not in the metadata. Please use <Cluster>.rescan() to update the metadata.

Clone based recovery selected through the recoveryMethod option

Validating instance configuration at node4:3306...

This instance reports its own address as node4:3306

Instance configuration is suitable.
NOTE: Group Replication will communicate with other members using 'node4:33061'. Use the localAddress option to override.

* Checking connectivity and SSL configuration...
A new instance will be added to the InnoDB Cluster. Depending on the amount of
data on the cluster this might take from a few seconds to several hours.

Adding instance to the cluster...

ERROR: MySQL Error 3190 (HY000): RESET SOURCE is not allowed because Group Replication is running.
Cluster.addInstance: RESET SOURCE is not allowed because Group Replication is running. (MYSQLSH 3190)
MySQL node4:3306 ssl JS > cluster.rescan()
Rescanning the cluster...

Result of the rescanning operation for the 'myCluster' cluster:
{
    "metadataConsistent": true,
    "name": "myCluster",
    "newTopologyMode": null,
    "newlyDiscoveredInstances": [
        {
            "host": "node4:3306",
            "member_id": "61765dab-c2c3-11ef-b27c-00505636f506",
            "name": null,
            "version": "8.0.40"
        }
    ],
    "unavailableInstances": [],
    "updatedInstances": []
}

A new instance 'node4:3306' was discovered in the cluster.
Would you like to add it to the cluster metadata? [Y/n]: y
Adding instance to the cluster metadata...
The instance 'node4:3306' was successfully added to the cluster metadata.
```

node4节点成功加入集群。

输出状态
cluster.status()

```
 MySQL  node4:3306 ssl  JS > cluster.status()
{
    "clusterName": "myCluster",
    "defaultReplicaSet": {
        "name": "default",
        "primary": "node4:3306",
        "ssl": "DISABLED",
        "status": "OK",
        "statusText": "Cluster is ONLINE and can tolerate up to ONE failure.",
        "topology": {
            "node1:3306": {
                "address": "node1:3306",
                "memberRole": "PRIMARY",
                "mode": "R/O",
                "readReplicas": {},
                "replicationLag": "applier_queue_applied",
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.40"
            },
            "node2:3306": {
                "address": "node2:3306",
                "memberRole": "SECONDARY",
                "mode": "R/O",
                "readReplicas": {},
                "replicationLag": "applier_queue_applied",
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.40"
            },
            "node3:3306": {
                "address": "node3:3306",
                "memberRole": "SECONDARY",
                "mode": "R/O",
                "readReplicas": {},
                "replicationLag": "applier_queue_applied",
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.40"
            },
            "node4:3306": {
                "address": "node4:3306",
                "memberRole": "SECONDARY",
                "mode": "R/W",
                "readReplicas": {},
                "replicationLag": "applier_queue_applied",
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.40"
            }
        },
        "topologyMode": "Single-Primary"
    },
    "groupInformationSourceMember": "node4:3306"
}
 MySQL  node4:3306 ssl  JS >
```

删除老节点

接下来将node3节点（ip：192...64） 踢出集群。

```
cluster.removeInstance('admin@node3:3306');
```

```
MySQL  node4:3306 ssl  JS > cluster.removeInstance('admin@node3:3306');
The instance will be removed from the InnoDB Cluster.

* Waiting for instance 'node3:3306' to synchronize with the primary...
** Transactions replicated  ############################################################  100%

* Instance 'node3:3306' is attempting to leave the cluster...

The instance 'node3:3306' was successfully removed from the cluster.

MySQL  node4:3306 ssl  JS >
```

查看集群状态

```
cluster.status()
```

回显如下：

```
 MySQL  node4:3306 ssl  JS > cluster.status()
{
    "clusterName": "myCluster",
    "defaultReplicaSet": {
        "name": "default",
        "primary": "node4:3306",
        "ssl": "DISABLED",
        "status": "OK",
        "statusText": "Cluster is ONLINE and can tolerate up to ONE failure.",
        "topology": {
            "node1:3306": {
                "address": "node1:3306",
                "memberRole": "PRIMARY",
                "mode": "R/O",
                "readReplicas": {},
                "replicationLag": "applier_queue_applied",
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.40"
            },
            "node2:3306": {
                "address": "node2:3306",
                "memberRole": "SECONDARY",
                "mode": "R/O",
                "readReplicas": {},
                "replicationLag": "applier_queue_applied",
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.40"
            },
            "node4:3306": {
                "address": "node4:3306",
                "memberRole": "SECONDARY",
                "mode": "R/W",
                "readReplicas": {},
                "replicationLag": "applier_queue_applied",
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.40"
            }
        },
        "topologyMode": "Single-Primary"
    },
    "groupInformationSourceMember": "node4:3306"
```

切换主节点

把主节点切换成node4

```
cluster.setPrimaryInstance("node4:3306");
```

```
MySQL  node4:3306 ssl  JS > cluster.setPrimaryInstance("node4:3306");
Setting instance 'node4:3306' as the primary instance of cluster 'myCluster'...

Instance 'node3:3306' remains SECONDARY.
Instance 'node2:3306' remains SECONDARY.
Instance 'node4:3306' was switched from SECONDARY to PRIMARY.
Instance 'node1:3306' was switched from PRIMARY to SECONDARY.

The instance 'node4:3306' was successfully elected as primary.
MySQL  node4:3306 ssl  JS >
```

碰到的问题

1、ERROR 2059 (HY000): Authentication plugin ‘caching_sha2_password’ cannot be loaded

```
[root@node1 mysql3306]# mysql -uroot -p
Enter password:
ERROR 2059 (HY000): Authentication plugin 'caching_sha2_password' cannot be loaded: /usr/li
```

原因：使用了系统自带的mysql客户端
解决办法：

```
# 先把/usr/bin/mysql*移走
cd /usr/bin
mkdir bak
mv mysql* bak/
# 修改当前用户的bash_profile文件
vi ~/.bash_profile
加入：
export PATH=$PATH:/usr/local/mysql/bin
# 环境变量生效
source  ~/.bash_profile
```

2、成功删除节点node2，然后重新把节点node2加入集群时报错：The instance ‘node2:3306’ is alrea dy part of another InnoDB Cluster

```
MySQL  node4:3306 ssl  JS > cluster.addInstance('node2:3306',{'recoveryMethod': "clone"})
ERROR: RuntimeError: The instance 'node2:3306' is already part of another InnoDB Cluster
Cluster.addInstance: The instance 'node2:3306' is already part of another InnoDB Cluster (R
```

原因：bug (参考文档：https://blog.51cto.com/u_15338523/11169415)
解决办法：通过mysql shell连接到这个实例(加入cluster遇到问题的实例，此案例为:node2)，执行：

```
shell.connect('admin@node2:3306');
shell.options.verbose=3
shell.options["dba.logSql"]=2
shell.options["logLevel"]=8
\sql
stop group_replication;
\js
dba.dropMetadataSchema();
```

如果上面命令没有成功，那么我们就必须连接到数据库，手工执行下面命令:

```
stop group_replication;
drop schema mysql_innodb_cluster_metadata;
```

然后在主节点执行下面命令，就可以重新将实例加入MySQL InnoDB Cluster。

```
var cluster=dba.getCluster()
cluster.addInstance('node2:3306',{'recoveryMethod': "clone"})
cluster.status()
```

总结

本文主要是手工安装了mysql数据库并配置了mgr集群，然后使用安装mysqlshell添加新节点、删除老节点操作。也可以在安装好mysql数据库后直接使用mysqlshell添加、删除节点，非常方便快捷。大家可以试一试~

关于作者：
专注于Oracle、MySQL、PG、OpenGauss和国产数据库的研究，热爱生活，热衷于分享数据库技术。
微信公众号：飞天online
墨天轮：https://www.modb.pro/u/15197
如有任何疑问，欢迎大家留言，共同进步~~~

墨力计划 mysql mgr mgr集群

最后修改时间：2024-12-27 21:44:45

文章被以下合辑收录

MySql数据库（共24篇）
MySql数据库
收藏合辑

评论

分享你的看法，一起交流吧～