主讲老师：Fox

> 【有道云笔记】Mysql8.0高可用集群架构实战
> https://note.youdao.com/s/2EjCTYwR

MySQL InnoDB Cluster

> 官方文档：https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-innodb-cluster.html

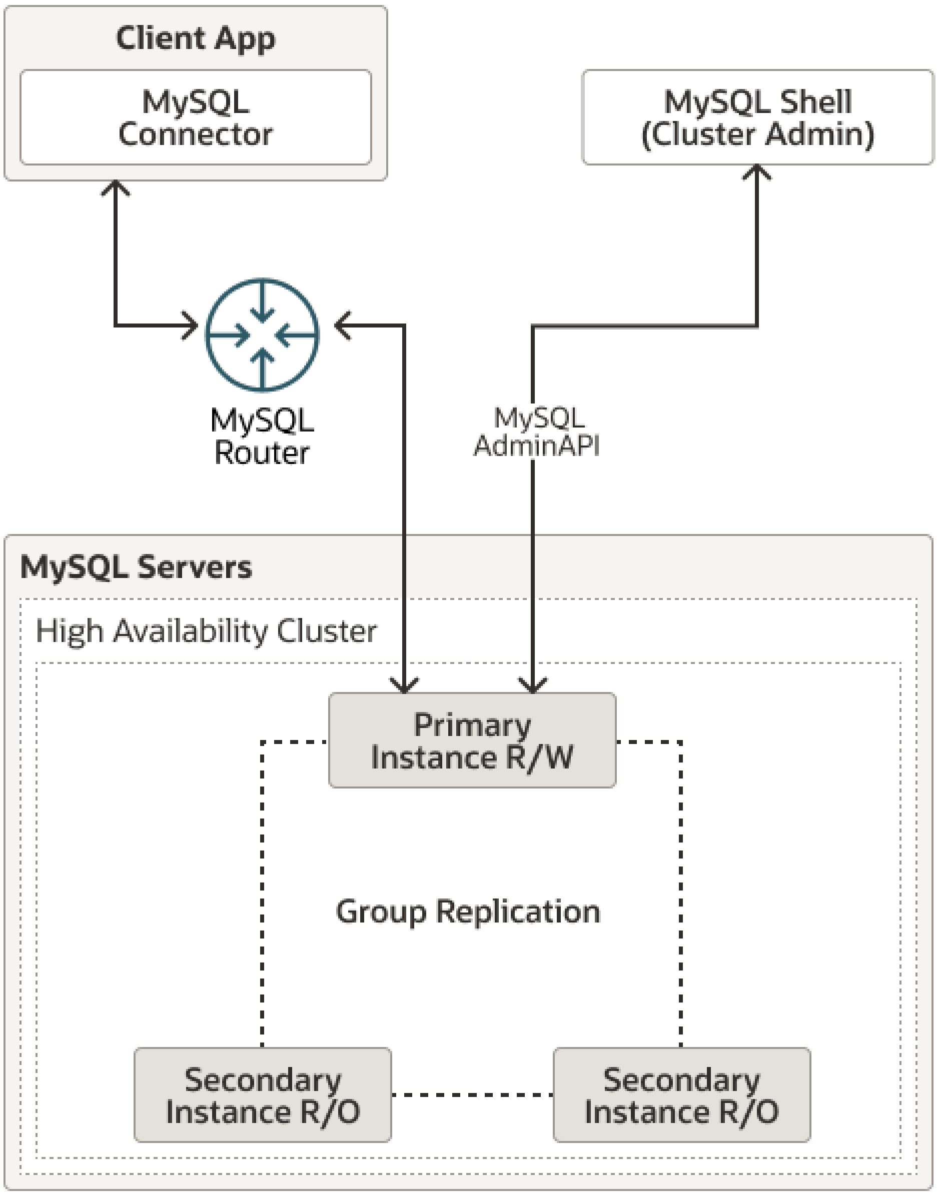
基本概述

InnoDB Cluster是MySQL官方实现高可用+读写分离的架构方案,其中包含以下组件

- **MySQL Group Replication**,简称MGR,是MySQL的主从同步高可用方案,包括数据同步及角色选举
- **Mysql Shell** 是InnoDB Cluster的管理工具,用来创建和管理集群
- **Mysql Router** 是业务流量入口,支持对MGR的主从角色判断,可以配置不同的端口分别对外提供读写服务,实现读写分离

MySQL Router与组复制和MySQL Shell高度整合，只有将其与组复制和MySQL Shell共同使用，才能够称为InnoDB Cluster。

集群架构



InnoDB Cluster将三个MySQL数据库实例构成一个高可用集群。其中一个实例是具有读/写能力的主要成员，其他两个实例是具有只读能力的次要成员。组复制将数据从主要成员复制到次要成员。MySQL Router将客户端应用程序连接到集群的主要成员。

搭建一主两从InnoDB集群

1. 安装3个数据库实例

> 参考：Docker 安装 MySQL8.0

可以利用docker快速部署3个MySQL实例

主机名（角色）	server_id	宿主机IP	容器固定IP	DB Port
mgr-node1(primary)	1	192.168.65.223	172.19.0.10	3321>3306
mgr-node2(Secondary)	2	192.168.65.223	172.19.0.11	3322>3306
mgr-	3	192.168.65.223	172.19.0.12	3323>3306

node3(Secondary)				

```
# 创建组复制的网络  保证三个mysql容器之间可以通过容器名访问
docker network create --driver bridge --subnet 172.19.0.0/24 --gateway 172.19.0.1 mgr-network

mkdir -p /mysql/mgr/node1/data /mysql/mgr/node1/conf /mysql/mgr/node1/log
mkdir -p /mysql/mgr/node2/data /mysql/mgr/node2/conf /mysql/mgr/node2/log
mkdir -p /mysql/mgr/node3/data /mysql/mgr/node3/conf /mysql/mgr/node3/log

#以mgr-node1配置为例，创建/mysql/mgr/node1/conf/custom.cnf，添加以下配置：
vim /mysql/mgr/node1/conf/custom.cnf
[mysql]
# 设置mysql客户端默认编码
default-character-set=utf8
[mysqld]
#指定sever_id，三个Mysql实例需要分别改为对应的sever_id
server_id=1
# 必须开启GTID支持
gtid_mode=ON
enforce_gtid_consistency=ON
# 启用二进制日志
log-bin=mysql-bin
#启用并行复制
binlog_transaction_dependency_tracking=WRITESET
replica_preserve_commit_order=ON
replica_parallel_type=LOGICAL_CLOCK
transaction_write_set_extraction=XXHASH64
#对于Group Replication，数据必须存储在InnoDB事务存储引擎中
disabled_storage_engines="MyISAM,BLACKHOLE,FEDERATED,ARCHIVE,MEMORY"

# mgr-node2和mgr-node3同上，注意配置文件路径和修改server_id
vim /mysql/mgr/node2/conf/custom.cnf
[mysql]
# 设置mysql客户端默认编码
default-character-set=utf8
[mysqld]
```

```
#指定sever_id，三个Mysql实例需要分别改为对应的sever_id
server_id=2
# 必须开启GTID支持
gtid_mode=ON
enforce_gtid_consistency=ON
# 启用二进制日志
log-bin=mysql-bin
#启用并行复制
binlog_transaction_dependency_tracking=WRITESET
replica_preserve_commit_order=ON
replica_parallel_type=LOGICAL_CLOCK
transaction_write_set_extraction=XXHASH64
#对于Group Replication，数据必须存储在InnoDB事务存储引擎中
disabled_storage_engines="MyISAM,BLACKHOLE,FEDERATED,ARCHIVE,MEMORY"

vim /mysql/mgr/node3/conf/custom.cnf
[mysql]
# 设置mysql客户端默认编码
default-character-set=utf8
[mysqld]
#指定sever_id，三个Mysql实例需要分别改为对应的sever_id
server_id=3
# 必须开启GTID支持
gtid_mode=ON
enforce_gtid_consistency=ON
# 启用二进制日志
log-bin=mysql-bin
#启用并行复制
binlog_transaction_dependency_tracking=WRITESET
replica_preserve_commit_order=ON
replica_parallel_type=LOGICAL_CLOCK
transaction_write_set_extraction=XXHASH64
#对于Group Replication，数据必须存储在InnoDB事务存储引擎中
disabled_storage_engines="MyISAM,BLACKHOLE,FEDERATED,ARCHIVE,MEMORY"

#运行mysql容器
# 为了便于测试，启动容器时指定好IP、hostname
docker run  -d  \
--name mgr-node1 \
--privileged=true \
--restart=always \
--ip 172.19.0.10 \
--hostname mgr-node1 \
--add-host  mgr-node2:172.19.0.11 \
--add-host  mgr-node3:172.19.0.12 \
--network  mgr-network \
-p 3321:3306 \
-v /mysql/mgr/node1/data:/var/lib/mysql \
-v /mysql/mgr/node1/conf:/etc/mysql/conf.d  \
-v /mysql/mgr/node1/log:/logs \
-e MYSQL_ROOT_PASSWORD=123456 \
-e TZ=Asia/Shanghai mysql:8.0.27 \
--lower_case_table_names=1


docker run  -d  \
--name mgr-node2 \
--privileged=true \
--restart=always \
--ip 172.19.0.11 \
--hostname mgr-node2 \
--add-host  mgr-node1:172.19.0.10 \
--add-host  mgr-node3:172.19.0.12 \
--network  mgr-network \
-p 3322:3306 \
-v /mysql/mgr/node2/data:/var/lib/mysql \
-v /mysql/mgr/node2/conf:/etc/mysql/conf.d  \
-v /mysql/mgr/node2/log:/logs \
-e MYSQL_ROOT_PASSWORD=123456 \
-e TZ=Asia/Shanghai mysql:8.0.27 \
--lower_case_table_names=1

docker run  -d  \
--name mgr-node3 \
--privileged=true \
--restart=always \
--ip 172.19.0.12 \
--hostname mgr-node3 \
--add-host  mgr-node1:172.19.0.10 \
--add-host  mgr-node2:172.19.0.11 \
--network  mgr-network \
-p 3323:3306 \
-v /mysql/mgr/node3/data:/var/lib/mysql \
-v /mysql/mgr/node3/conf:/etc/mysql/conf.d  \
-v /mysql/mgr/node3/log:/logs \
-e MYSQL_ROOT_PASSWORD=123456 \
-e TZ=Asia/Shanghai mysql:8.0.27 \
--lower_case_table_names=1


# 在宿主机上配置mysql容器的ip和host映射
vim /etc/hosts
172.19.0.10  mgr-node1
172.19.0.11  mgr-node2
172.19.0.12  mgr-node3
```

所有实例分别配置远程访问

```
# 以node1为例
docker exec -it mgr-node1 /bin/bash
mysql -u root -p123456
#进入mysql执行
ALTER USER 'root'@'%' IDENTIFIED WITH mysql_native_password BY '123456';
flush privileges;
```

2. 安装mysqlrouter和安装mysqlshell

MySQL Router是MySQL Proxy的后继产品，它提供了MySQL协议的路由器功能，可以用来实现读写分离、负载均衡和高可用性解决方案。

安装mysql-router

下载地址：https://downloads.mysql.com/archives/router/

```
# 以centos7为例
```

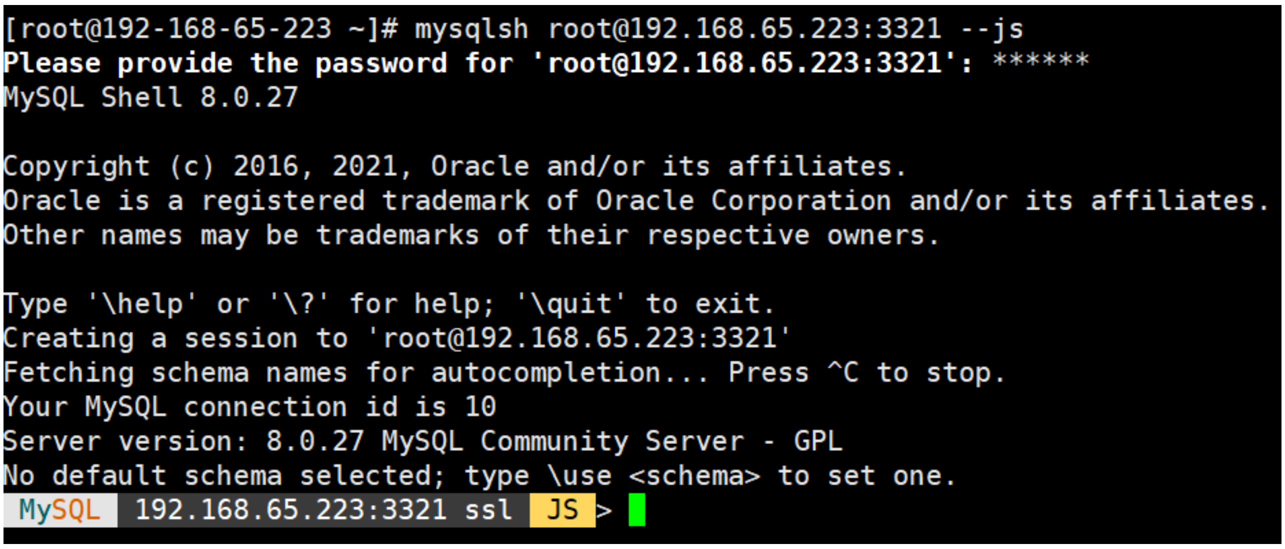
```
wget https://downloads.mysql.com/archives/get/p/41/file/mysql-router-community-8.0.27-1.el7.x86_64.rpm
rpm -ivh mysql-router-community-8.0.27-1.el7.x86_64.rpm
```

安装mysql-shell

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

```
# 以centos7为例
wget https://downloads.mysql.com/archives/get/p/43/file/mysql-shell-8.0.27-1.el7.x86_64.rpm
rpm -ivh mysql-shell-8.0.27-1.el7.x86_64.rpm
# 远程连接mysql
mysqlsh root@192.168.65.223:3321 --js
mysqlsh root@mgr-node1:3306 --js
```

> MySQL Shell 教程: https://note.youdao.com/s/IpZEoHG0



```
[root@192-168-65-223 ~]# mysqlsh root@192.168.65.223:3321 --js
Please provide the password for 'root@192.168.65.223:3321': ******
MySQL Shell 8.0.27

Copyright (c) 2016, 2021, 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@192.168.65.223:3321'
Fetching schema names for autocompletion... Press ^C to stop.
Your MySQL connection id is 10
Server version: 8.0.27 MySQL Community Server - GPL
No default schema selected; type \use <schema> to set one.
 MySQL  192.168.65.223:3321 ssl  JS >
```

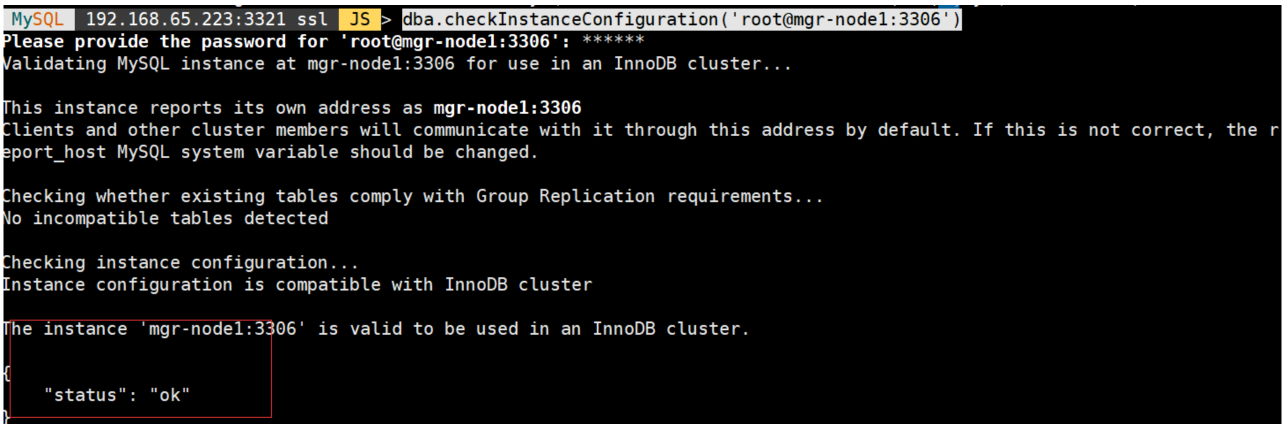
3. InnoDB Cluster 初始化

3.1 参数及权限配置预需求检测

在添加实例到集群中前，使用该方法检查实例配置是否满足InnoDB 集群要求。

```
mysqlsh root@192.168.65.223:3321 --js
// 检查实例是否符合InnoDB Cluster的参数及权限配置要求
dba.checkInstanceConfiguration('root@mgr-node1:3306')
dba.checkInstanceConfiguration('root@mgr-node2:3306')
dba.checkInstanceConfiguration('root@mgr-node3:3306')
```

如果验证通过返回ok。



```
MySQL  192.168.65.223:3321 ssl  JS > dba.checkInstanceConfiguration('root@mgr-node1:3306')
Please provide the password for 'root@mgr-node1:3306': ******
Validating MySQL instance at mgr-node1:3306 for use in an InnoDB cluster...

This instance reports its own address as mgr-node1:3306
Clients and other cluster members will communicate with it through this address by default. If this is not correct, the r
eport_host MySQL system variable should be changed.

Checking whether existing tables comply with Group Replication requirements...
No incompatible tables detected

Checking instance configuration...
Instance configuration is compatible with InnoDB cluster

The instance 'mgr-node1:3306' is valid to be used in an InnoDB cluster.

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

如果验证没通过，比如出现下面的日志提示，需要mysql实例开启gtid和指定server_id

搭建InnoDB Cluster需要满足的要求如下：

InnoDB集群使用了Group Replication，因此必须满足使用组复制的要求。具体可以参考https://dev.mysql.com/doc/refman/8.0/en/group-replication-requirements.html。其中比较重要的几点有：

- 必须开启二进制日志，并且日志格式为ROW，即--log-bin和binlog_format=row（默认）；
- 必须开启副本更新日志，即log_replica_updates=ON（默认）；
- 必须开启GTID，即gtid_mode=ON和enforce_gtid_consistency=ON。
- 存储引擎只能使用**InnoDB**。最好禁用其他存储引擎：

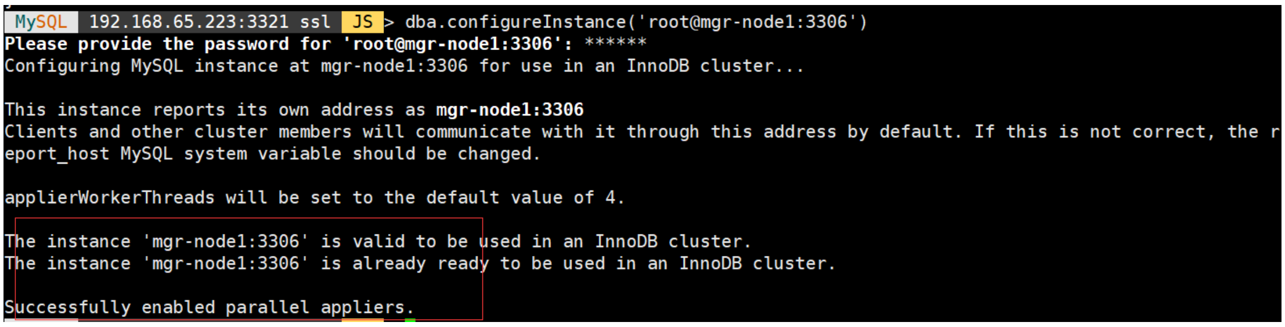
```
disabled_storage_engines="MyISAM,BLACKHOLE,FEDERATED,ARCHIVE,MEMORY"
```

- 从8.0.23开始，集群中的实例要启用并行复制。需要配置以下系统变量：

```
binlog_transaction_dependency_tracking=WRITESET
slave_preserve_commit_order=ON
slave_parallel_type=LOGICAL_CLOCK
transaction_write_set_extraction=XXHASH64
```

3.2 初始化InnoDB Cluster相关配置

```
// 对实例配置InnoDB Cluster相关参数
dba.configureInstance('root@mgr-node1:3306')
dba.configureInstance('root@mgr-node2:3306')
dba.configureInstance('root@mgr-node3:3306')
```



```
 MySQL  192.168.65.223:3321 ssl  JS > dba.configureInstance('root@mgr-node1:3306')
Please provide the password for 'root@mgr-node1:3306': ******
Configuring MySQL instance at mgr-node1:3306 for use in an InnoDB cluster...

This instance reports its own address as mgr-node1:3306
Clients and other cluster members will communicate with it through this address by default. If this is not correct, the r
eport_host MySQL system variable should be changed.

applierWorkerThreads will be set to the default value of 4.

The instance 'mgr-node1:3306' is valid to be used in an InnoDB cluster.
The instance 'mgr-node1:3306' is already ready to be used in an InnoDB cluster.

Successfully enabled parallel appliers.
```

4.创建一主两从InnoDB集群

集群常用命令

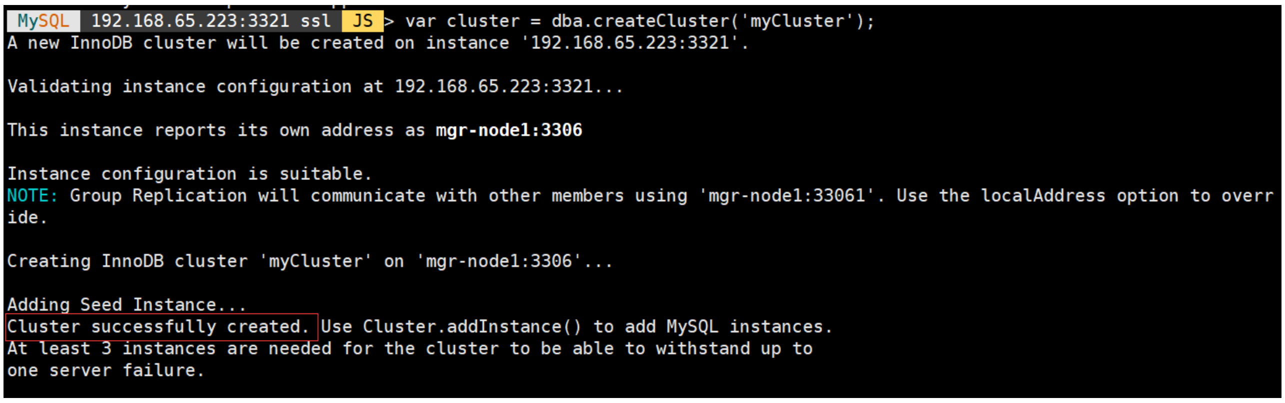
```
#会列出dba相关指令
dba.help();
#列出详细指令的用法
dba.help('deploySandboxInstance');
#检查节点配置实例，用于加入cluster之前
dba.checkInstanceConfiguration("root@hostname:3306");
#节点初始化
dba.configureInstance('root@hostname:3306');
#重启集群
dba.rebootClusterFromCompleteOutage('myCluster');
#会列出集群相关指令
cluster.help();
#创建集群
var cluster = dba.createCluster('myCluster');
#获取当前集群实例
var cluster = dba.getCluster('myCluster');
查看集群状态
cluster.status();
#检查cluster节点状态
```

```
cluster.checkInstanceState("root@hostname:3306") ;
#增加节点
cluster.addInstance("root@hostname:3306") ;
#删除节点
cluster.removeInstance("root@hostname:3306") ;
#强制删除节点
cluster.removeInstance('root@hostname:3306',{force:true});
#  状态为missing的节点可以重新加入集群
cluster.rejoinInstance("root@hostname:3306")
#解散集群
cluster.dissolve({force:true}) ;
#集群描述
cluster.describe();
```

进入主节点创建集群

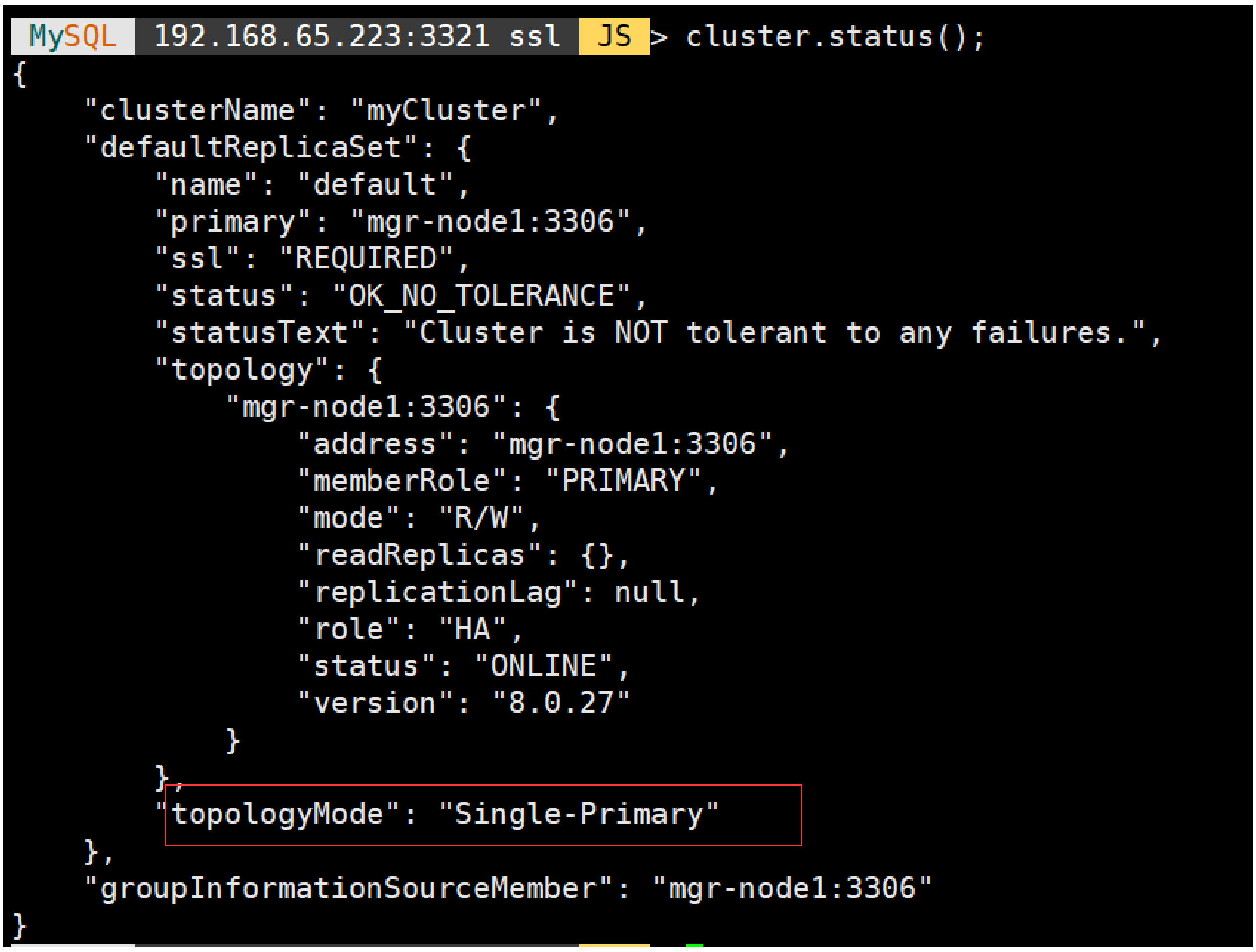
初始化完第一个实例后，就可以创建集群了。

```
# 进入主节点创建集群
mysqlsh root@192.168.65.223:3321 --js
# 创建一个 cluster，命名为 'myCluster'
var cluster = dba.createCluster('myCluster');
# 创建成功后，查看cluster状态
cluster.status();
```



```
 MySQL  192.168.65.223:3321 ssl  JS > var cluster = dba.createCluster('myCluster');
A new InnoDB cluster will be created on instance '192.168.65.223:3321'.

Validating instance configuration at 192.168.65.223:3321...

This instance reports its own address as mgr-node1:3306

Instance configuration is suitable.
NOTE: Group Replication will communicate with other members using 'mgr-node1:33061'. Use the localAddress option to overr
ide.

Creating InnoDB cluster 'myCluster' on 'mgr-node1:3306'...

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.
```

当前集群的状态如下

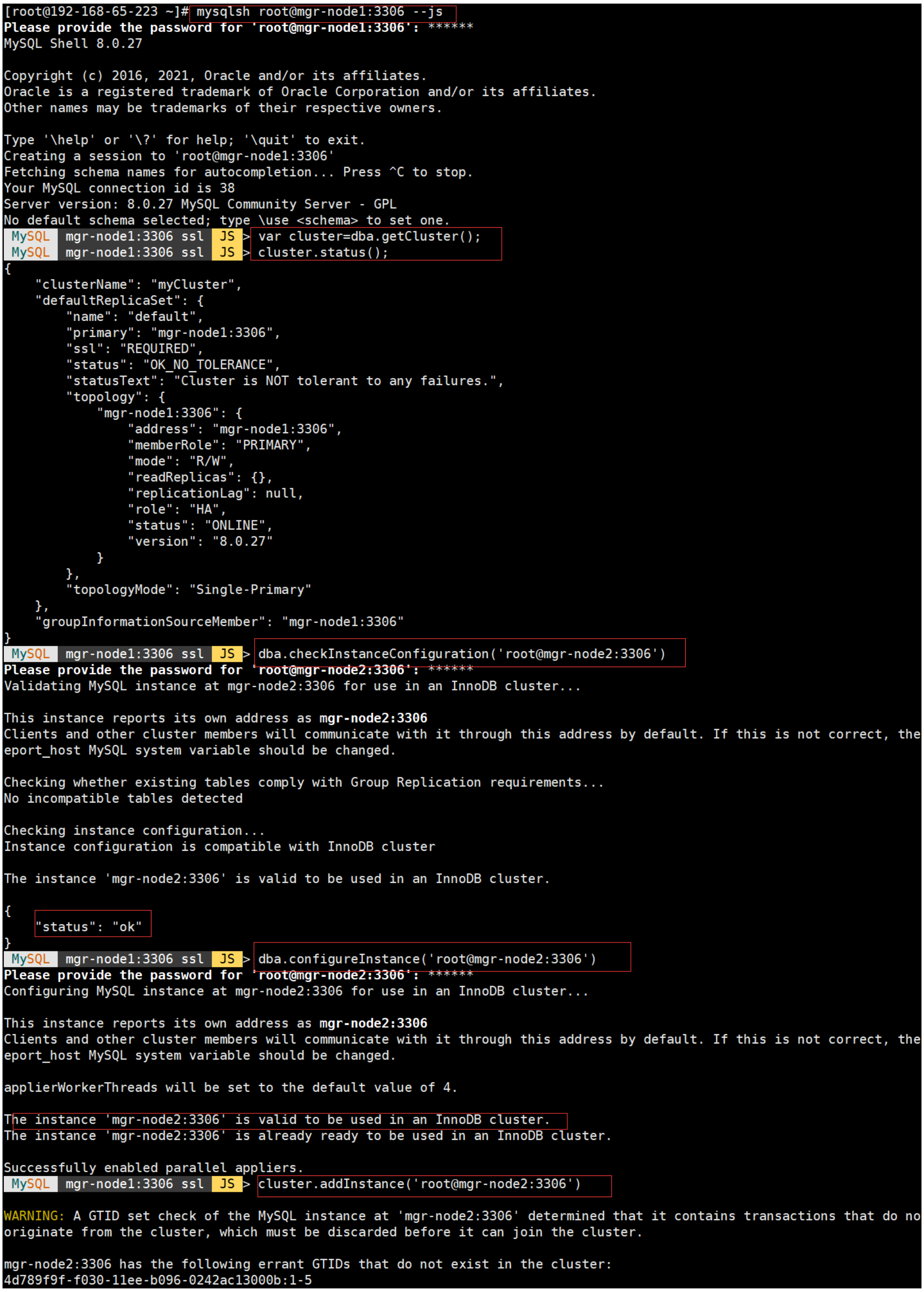


```
MySQL  192.168.65.223:3321 ssl  JS > cluster.status();
{
    "clusterName": "myCluster",
    "defaultReplicaSet": {
        "name": "default",
        "primary": "mgr-node1:3306",
        "ssl": "REQUIRED",
        "status": "OK_NO_TOLERANCE",
        "statusText": "Cluster is NOT tolerant to any failures.",
        "topology": {
            "mgr-node1:3306": {
                "address": "mgr-node1:3306",
                "memberRole": "PRIMARY",
                "mode": "R/W",
                "readReplicas": {},
                "replicationLag": null,
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.27"
            }
        },
        "topologyMode": "Single-Primary"
    },
    "groupInformationSourceMember": "mgr-node1:3306"
}
```

添加副本实例

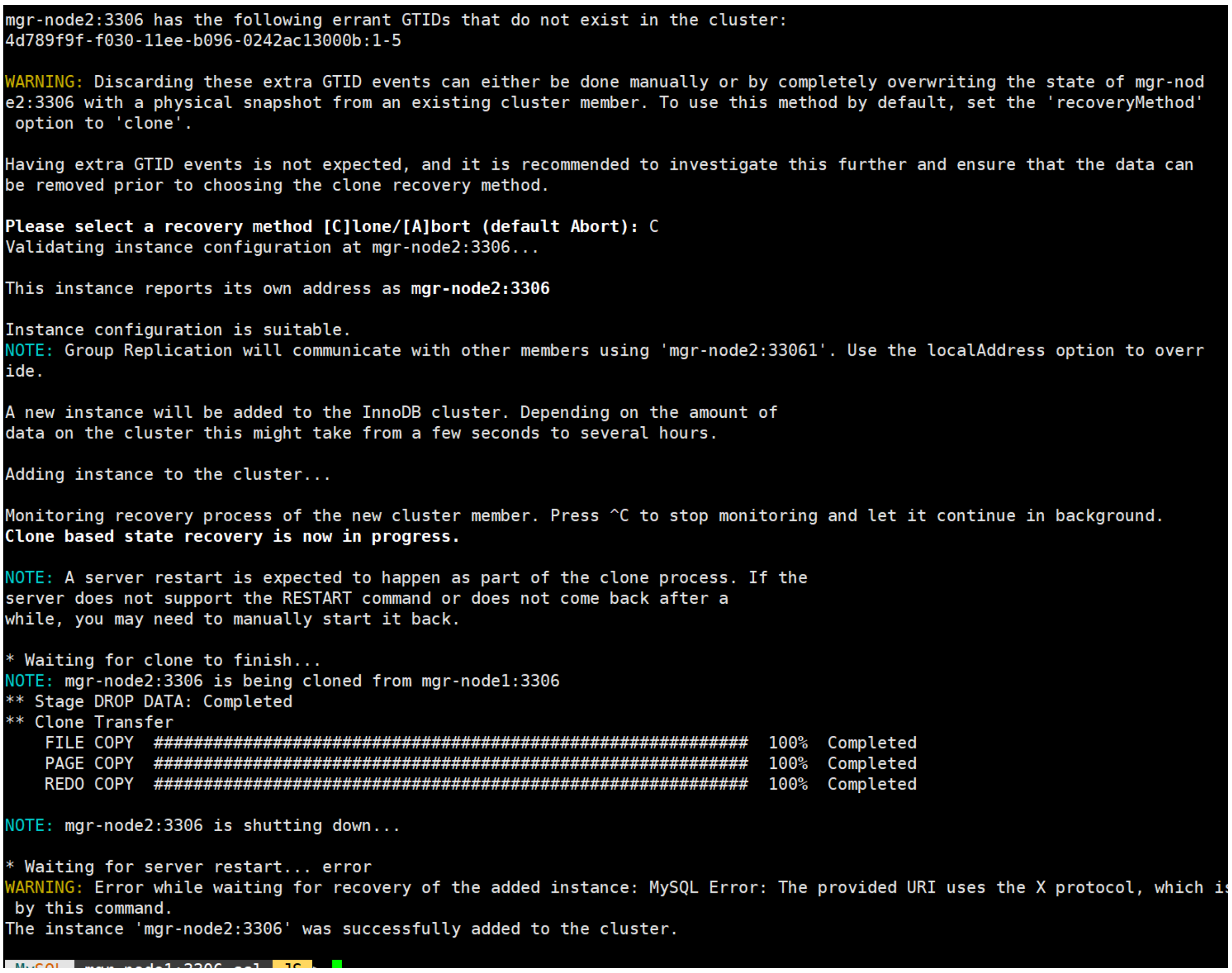
添加副本实例到创建好的集群。

```
#初始化第二个和第三个实例:
cluster.addInstance('root@mgr-node2:3306');
cluster.addInstance('root@mgr-node3:3306');
#查看cluster状态
cluster.status();
```



```
[root@192-168-65-223 ~]# mysqlsh root@mgr-node1:3306 --js
Please provide the password for 'root@mgr-node1:3306': ******
MySQL Shell 8.0.27

Copyright (c) 2016, 2021, 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@mgr-node1:3306'
Fetching schema names for autocompletion... Press ^C to stop.
Your MySQL connection id is 38
Server version: 8.0.27 MySQL Community Server - GPL
No default schema selected; type \use <schema> to set one.
 MySQL  mgr-node1:3306 ssl  JS > var cluster=dba.getCluster();
 MySQL  mgr-node1:3306 ssl  JS > cluster.status();
{
    "clusterName": "myCluster",
    "defaultReplicaSet": {
        "name": "default",
        "primary": "mgr-node1:3306",
        "ssl": "REQUIRED",
        "status": "OK_NO_TOLERANCE",
        "statusText": "Cluster is NOT tolerant to any failures.",
        "topology": {
            "mgr-node1:3306": {
                "address": "mgr-node1:3306",
                "memberRole": "PRIMARY",
                "mode": "R/W",
                "readReplicas": {},
                "replicationLag": null,
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.27"
            }
        },
        "topologyMode": "Single-Primary"
    },
    "groupInformationSourceMember": "mgr-node1:3306"
}
 MySQL  mgr-node1:3306 ssl  JS > dba.checkInstanceConfiguration('root@mgr-node2:3306')
Please provide the password for 'root@mgr-node2:3306': ******
Validating MySQL instance at mgr-node2:3306 for use in an InnoDB cluster...

This instance reports its own address as mgr-node2:3306
Clients and other cluster members will communicate with it through this address by default. If this is not correct, the
eport_host MySQL system variable should be changed.

Checking whether existing tables comply with Group Replication requirements...
No incompatible tables detected

Checking instance configuration...
Instance configuration is compatible with InnoDB cluster

The instance 'mgr-node2:3306' is valid to be used in an InnoDB cluster.

{
    "status": "ok"
}
 MySQL  mgr-node1:3306 ssl  JS > dba.configureInstance('root@mgr-node2:3306')
Please provide the password for 'root@mgr-node2:3306': ******
Configuring MySQL instance at mgr-node2:3306 for use in an InnoDB cluster...

This instance reports its own address as mgr-node2:3306
Clients and other cluster members will communicate with it through this address by default. If this is not correct, the
eport_host MySQL system variable should be changed.

applierWorkerThreads will be set to the default value of 4.

The instance 'mgr-node2:3306' is valid to be used in an InnoDB cluster.
The instance 'mgr-node2:3306' is already ready to be used in an InnoDB cluster.

Successfully enabled parallel appliers.
 MySQL  mgr-node1:3306 ssl  JS > cluster.addInstance('root@mgr-node2:3306')

WARNING: A GTID set check of the MySQL instance at 'mgr-node2:3306' determined that it contains transactions that do no
originate from the cluster, which must be discarded before it can join the cluster.

mgr-node2:3306 has the following errant GTIDs that do not exist in the cluster:
4d789f9f-f030-11ee-b096-0242ac13000b:1-5
```

如果提示副本实例的GTID与集群不一致，选择通过**Clone**方式覆盖副本实例上的数据即可。



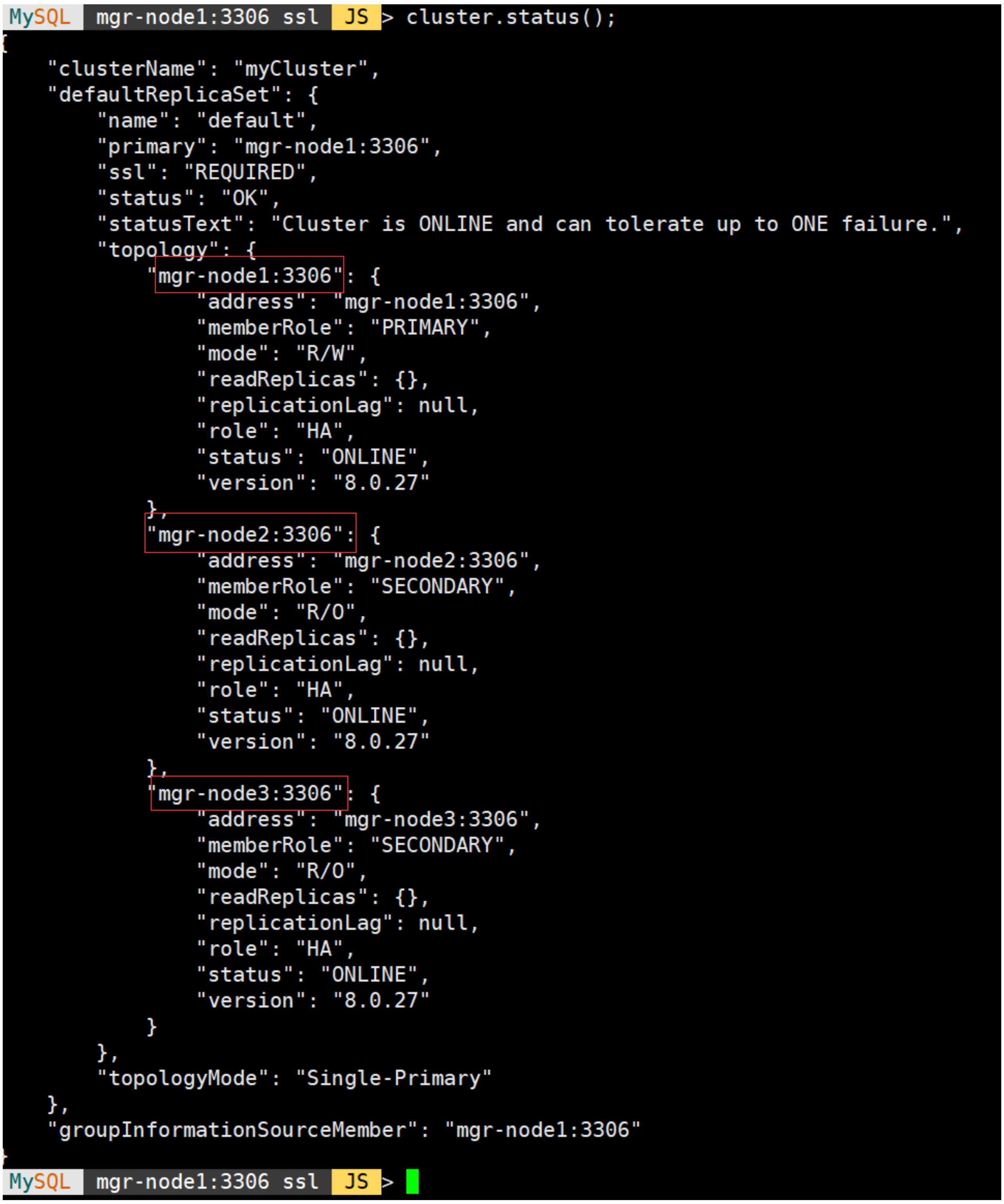
```
mgr-node2:3306 has the following errant GTIDs that do not exist in the cluster:
4d789f9f-f030-11ee-b096-0242ac13000b:1-5

WARNING: Discarding these extra GTID events can either be done manually or by completely overwriting the state of mgr-nod
e2:3306 with a physical snapshot from an existing cluster member. To use this method by default, set the 'recoveryMethod'
 option to 'clone'.

Having extra GTID events is not expected, and it is recommended to investigate this further and ensure that the data can
be removed prior to choosing the clone recovery method.

Please select a recovery method [C]lone/[A]bort (default Abort): C
Validating instance configuration at mgr-node2:3306...

This instance reports its own address as mgr-node2:3306

Instance configuration is suitable.
NOTE: Group Replication will communicate with other members using 'mgr-node2:33061'. Use the localAddress option to overr
ide.

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...

Monitoring recovery process of the new cluster member. Press ^C to stop monitoring and let it continue in background.
Clone based state recovery is now in progress.

NOTE: A server restart is expected to happen as part of the clone process. If the
server does not support the RESTART command or does not come back after a
while, you may need to manually start it back.

* Waiting for clone to finish...
NOTE: mgr-node2:3306 is being cloned from mgr-node1:3306
** Stage DROP DATA: Completed
** Clone Transfer
    FILE COPY  ############################################################  100%  Completed
    PAGE COPY  ############################################################  100%  Completed
    REDO COPY  ############################################################  100%  Completed

NOTE: mgr-node2:3306 is shutting down...

* Waiting for server restart... error
WARNING: Error while waiting for recovery of the added instance: MySQL Error: The provided URI uses the X protocol, which is
 by this command.
The instance 'mgr-node2:3306' was successfully added to the cluster.
```

小结：完整的集群创建步骤

```
#进入主节点
mysqlsh root@mgr-node1:3306 --js

#mgr-node1
# 参数权限检查
dba.checkInstanceConfiguration('root@mgr-node1:3306');
# 初始化
dba.configureInstance('root@mgr-node1:3306');
# 创建集群
var cluster = dba.createCluster('myCluster');
#查看cluster状态
cluster.status();

#mgr-node2
# 参数权限检查
```

```
dba.checkInstanceConfiguration('root@mgr-node2:3306');
# 初始化
dba.configureInstance('root@mgr-node2:3306');
# 添加副本
cluster.addInstance('root@mgr-node2:3306');

#mgr-node3
# 参数权限检查
dba.checkInstanceConfiguration('root@mgr-node3:3306');
# 初始化
dba.configureInstance('root@mgr-node3:3306');
# 添加副本
cluster.addInstance('root@mgr-node3:3306');

#查看cluster状态
cluster.status();

```

搭建一主两从集群架构最终效果如下：



```
MySQL  mgr-node1:3306 ssl  JS > cluster.status();
{
    "clusterName": "myCluster",
    "defaultReplicaSet": {
        "name": "default",
        "primary": "mgr-node1:3306",
        "ssl": "REQUIRED",
        "status": "OK",
        "statusText": "Cluster is ONLINE and can tolerate up to ONE failure.",
        "topology": {
            "mgr-node1:3306": {
                "address": "mgr-node1:3306",
                "memberRole": "PRIMARY",
                "mode": "R/W",
                "readReplicas": {},
                "replicationLag": null,
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.27"
            },
            "mgr-node2:3306": {
                "address": "mgr-node2:3306",
                "memberRole": "SECONDARY",
                "mode": "R/O",
                "readReplicas": {},
                "replicationLag": null,
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.27"
            },
            "mgr-node3:3306": {
                "address": "mgr-node3:3306",
                "memberRole": "SECONDARY",
                "mode": "R/O",
                "readReplicas": {},
                "replicationLag": null,
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.27"
            }
        },
        "topologyMode": "Single-Primary"
    },
    "groupInformationSourceMember": "mgr-node1:3306"
}
MySQL  mgr-node1:3306 ssl  JS >
```

注意到集群状态已变为"status": "OK"和"statusText": "Cluster is ONLINE and can tolerate up to ONE failure."。

集群节点状态：

- ONLINE - 节点状态正常。
- OFFLINE - 实例在运行，但没有加入任何Cluster。
- RECOVERING - 实例已加入Cluster，正在同步数据。
- ERROR - 同步数据发生异常。
- UNREACHABLE - 与其他节点通讯中断，可能是网络问题，可能是节点crash。
- MISSING - 节点已加入集群，但未启动group replication

测试数据是否同步

```
#主节点 mgr-node1
[root@192-168-65-223 ~]# docker exec -it mgr-node1 bash
root@mgr-node1:/# mysql -uroot -p123456
mysql>
create database test;
use test;
create table t(x int primary key auto_increment,y int);
insert into t(x,y) value(1,1);
```

注意：如果创建表没有设置主键，会抛出错误：ERROR 3098 (HY000): The table does not comply with the requirements by an external plugin.

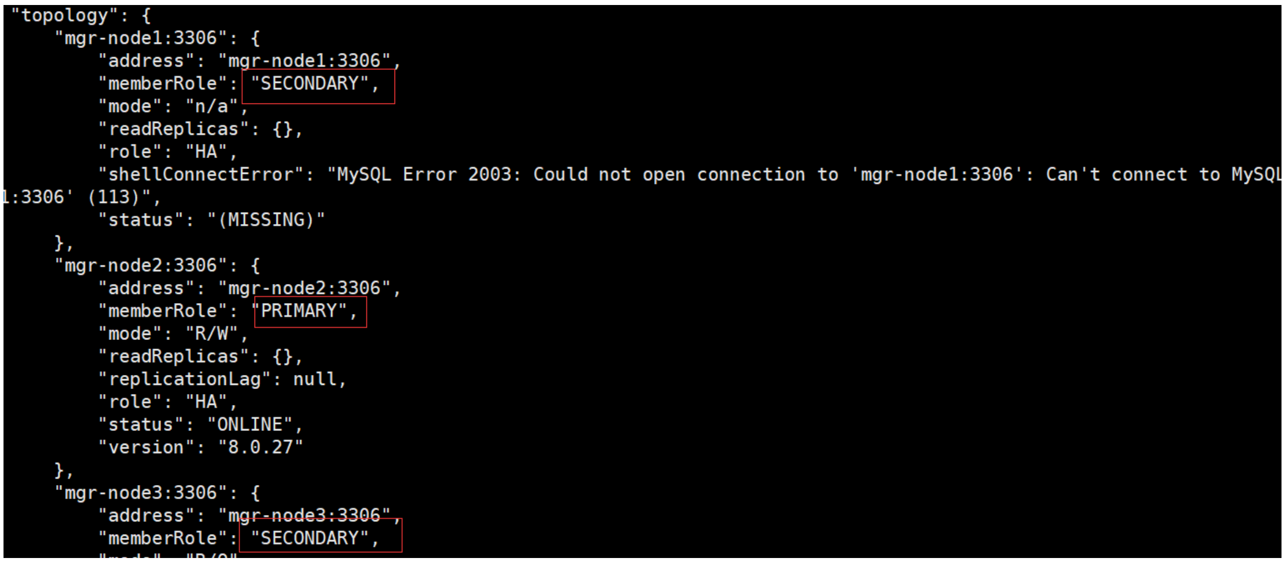
```
# 查看其他节点，数据是否同步
[root@192-168-65-223 ~]# docker exec -it mgr-node2 bash
root@mgr-node2:/# mysql -uroot -p123456
mysql> use test;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from t;
+---+------+
| x | y    |
+---+------+
| 1 |    1 |
+---+------+
1 row in set (0.00 sec)
```

测试主从切换

```
#停掉主节点
[root@192-168-65-223 ~]# docker stop mgr-node1

```

```
# 连接到mgr-node2
MySQL  mgr-node1:3306 ssl  JS > \connect root@mgr-node2:3306
# 获取集群实例
 MySQL  mgr-node2:3306 ssl  JS > var cluster=dba.getCluster();
# 查看集群状态
MySQL  mgr-node2:3306 ssl  JS > cluster.status()
{
    "clusterName": "myCluster",
    "defaultReplicaSet": {
        "name": "default",
        "primary": "mgr-node2:3306",
        "ssl": "REQUIRED",
        "status": "OK_NO_TOLERANCE",
        "statusText": "Cluster is NOT tolerant to any failures. 1 member is not active.",
        "topology": {
            "mgr-node1:3306": {
                "address": "mgr-node1:3306",
                "memberRole": "SECONDARY",
                "mode": "n/a",
                "readReplicas": {},
                "role": "HA",
                "shellConnectError": "MySQL Error 2003: Could not open connection to 'mgr-node1:3306': Can't connect to MySQL server on 'mgr-node1:3306' (113)",
                "status": "(MISSING)"
            },
            "mgr-node2:3306": {
                "address": "mgr-node2:3306",
                "memberRole": "PRIMARY",
                "mode": "R/W",
                "readReplicas": {},
                "replicationLag": null,
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.27"
            },
            "mgr-node3:3306": {
                "address": "mgr-node3:3306",
                "memberRole": "SECONDARY",
                "mode": "R/O",
                "readReplicas": {},
                "replicationLag": null,
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.27"
            }
        },
        "topologyMode": "Single-Primary"
    },
    "groupInformationSourceMember": "mgr-node2:3306"


```

可以看到mgr-node2升级为主节点



```
"topology": {
    "mgr-node1:3306": {
        "address": "mgr-node1:3306",
        "memberRole": "SECONDARY",
        "mode": "n/a",
        "readReplicas": {},
        "role": "HA",
        "shellConnectError": "MySQL Error 2003: Could not open connection to 'mgr-node1:3306': Can't connect to MySQL
1:3306' (113)",
        "status": "(MISSING)"
    },
    "mgr-node2:3306": {
        "address": "mgr-node2:3306",
        "memberRole": "PRIMARY",
        "mode": "R/W",
        "readReplicas": {},
        "replicationLag": null,
        "role": "HA",
        "status": "ONLINE",
        "version": "8.0.27"
    },
    "mgr-node3:3306": {
        "address": "mgr-node3:3306",
        "memberRole": "SECONDARY",
```

```
# 启动mgr-node1节点
[root@192-168-65-223 ~]# docker start mgr-node1
# 查看集群状态，发现mgr-node1正在恢复，最终正常
 MySQL  mgr-node2:3306 ssl  JS > cluster.status()
{
    "clusterName": "myCluster",
    "defaultReplicaSet": {
        "name": "default",
        "primary": "mgr-node2:3306",
        "ssl": "REQUIRED",
        "status": "OK_NO_TOLERANCE",
        "statusText": "Cluster is NOT tolerant to any failures. 1 member is not active.",
        "topology": {
            "mgr-node1:3306": {
                "address": "mgr-node1:3306",
                "instanceErrors": [
                    "NOTE: group_replication is stopped."
                ],
                "memberRole": "SECONDARY",
                "memberState": "OFFLINE",
                "mode": "R/O",
                "readReplicas": {},
                "role": "HA",
                "status": "(MISSING)",
                "version": "8.0.27"
            },
            "mgr-node2:3306": {
                "address": "mgr-node2:3306",
                "memberRole": "PRIMARY",
                "mode": "R/W",
                "readReplicas": {},
                "replicationLag": null,
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.27"
            },
            "mgr-node3:3306": {
                "address": "mgr-node3:3306",
                "memberRole": "SECONDARY",
                "mode": "R/O",
                "readReplicas": {},
                "replicationLag": null,
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.27"
            }
        },
        "topologyMode": "Single-Primary"
    },
    "groupInformationSourceMember": "mgr-node2:3306"
```

```
}
MySQL  mgr-node2:3306 ssl  JS > cluster.status()
{
    "clusterName": "myCluster",
    "defaultReplicaSet": {
        "name": "default",
        "primary": "mgr-node2:3306",
        "ssl": "REQUIRED",
        "status": "OK",
        "statusText": "Cluster is ONLINE and can tolerate up to ONE failure.",
        "topology": {
            "mgr-node1:3306": {
                "address": "mgr-node1:3306",
                "memberRole": "SECONDARY",
                "mode": "R/O",
                "readReplicas": {},
                "replicationLag": null,
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.27"
            },
            "mgr-node2:3306": {
                "address": "mgr-node2:3306",
                "memberRole": "PRIMARY",
                "mode": "R/W",
                "readReplicas": {},
                "replicationLag": null,
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.27"
            },
            "mgr-node3:3306": {
                "address": "mgr-node3:3306",
                "memberRole": "SECONDARY",
                "mode": "R/O",
                "readReplicas": {},
                "replicationLag": null,
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.27"
            }
        },
        "topologyMode": "Single-Primary"
    },
    "groupInformationSourceMember": "mgr-node2:3306"
}

```

更多的常见操作

参数配置

可以用cluster.options()查看当前集群的配置属性,集群参数配置分为两种方式:

- cluster.setOption() 用来设置所有节点的参数
- cluster.setInstanceOption() 用来对指定节点配置属性

```
# 将所有节点的权重都改为50
var cluster = dba.getCluster()
cluster.setOption("memberWeight",50)
# 重新加入集群重试次数改为5次
cluster.setOption("autoRejoinTries",5)

# 将其中一个节点的权重改为75
cluster.setInstanceOption("mgr-node2:3306","memberWeight",75)
# 重新加入集群重试次数改为10次
cluster.setInstanceOption("mgr-node2:3306","autoRejoinTries",10)

```

配置节点权重

memberWeight选项的值域为0到100之间的整数，缺省值为50。该值是故障转移时自动选举主节点的百分比权重,具有较高memberWeight值的实例更有可能在单主群集中被选为主节点

```
// 查看集群的参数配置(包括memberWeight优先级配置)
cluster.options()

// 在集群创建时配置
```

```
dba.createCluster('myCluster', {memberWeight:75}) // 第一个节点配置方式
var cluster = dba.getCluster()
cluster.addInstance('mgr-node2:3306',{memberWeight:50})
cluster.addInstance('mgr-node3:3306',{memberWeight:25})

// 在集群创建完成后修改权重
var cluster = dba.getCluster()
cluster.setInstanceOption('mgr-node1:3306','memberWeight',100)
cluster.setInstanceOption('mgr-node2:3306','memberWeight',50)
cluster.setInstanceOption('mgr-node3:3306','memberWeight',25)
```

将节点重新加入集群

状态为mssing的节点,通常是组复制关闭或中断状态,可以用cluster.rejoinInstance()重新加入集群,会重新对该节点设置MGR相关参数(持久化到mysqld-auto.conf中)

```
cluster.removeInstance('root@hostname:3306',{force:true});
```

如果一些参数做了修改,如server_uuid变更,导致rejoin失败,则需要将节点从集群中删除后重新加入

```
cluster.removeInstance("root@hostname:3306",{force:true})
cluster.rescan()
cluster.addInstance("root@hostname:3306")
```

集群多数节点异常,恢复

当集群多个节点异常,则失去了仲裁机制,剩下的一个节点

```
// 将集群剥离为单节点运行
JS > cluster.forceQuorumUsingPartitionOf("root@hostname:3306")

// 重新加另外2个节点加入
JS > cluster.rejoinInstance("root@hostname2:3306")
JS > cluster.rejoinInstance("root@hostname3:3306")
```

集群节点角色切换

在MGR的管理下提供了一下3种方式进行角色切换,mysqlsh对其进行了封装调用

- group_replication_set_as_primary(member_uuid);
  - cluster.setPrimaryInstance("homename:3306")
- group_replication_switch_to_single_primary_mode()
  - cluster.switchToSinglePrimaryMode("homename:3306")
- group_replication_switch_to_multi_primary_mode()
  - cluster.switchToMultiPrimaryMode()

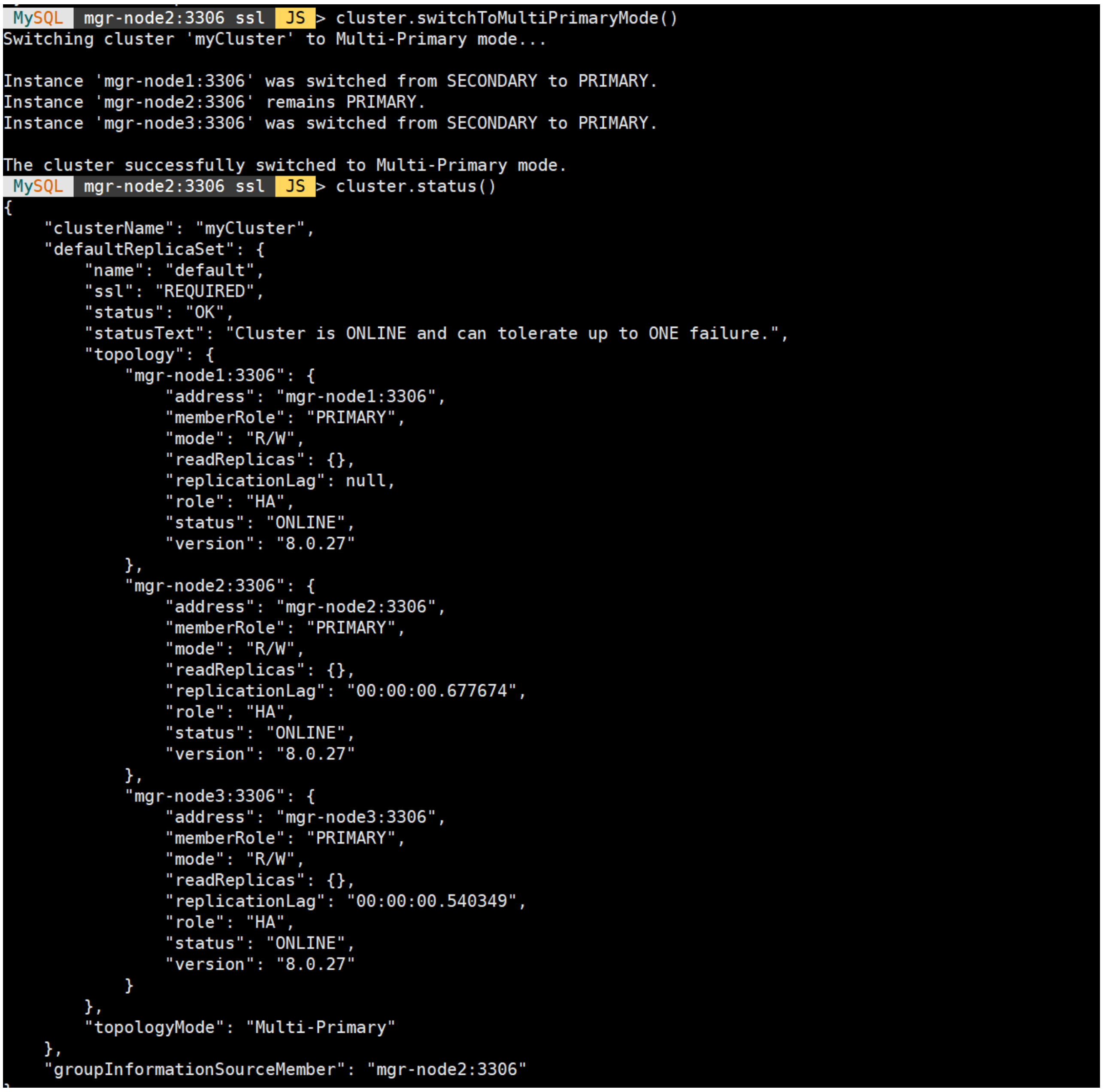
单主模式-指定主节点切换

```
var cluster = dba.getCluster()
cluster.setPrimaryInstance('homename:3306')
cluster.status()

```

单主模式和多主模式相互切换

```
// 切换为多主模式
var cluster = dba.getCluster()
cluster.switchToMultiPrimaryMode()

// 指定明确的主节点将多主模式切换为单主模式
cluster.switchToSinglePrimaryMode("homename:3306")

```

将单主模式切换为多主模式的效果



```
 MySQL  mgr-node2:3306 ssl  JS > cluster.switchToMultiPrimaryMode()
Switching cluster 'myCluster' to Multi-Primary mode...

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

The cluster successfully switched to Multi-Primary mode.
 MySQL  mgr-node2:3306 ssl  JS > cluster.status()
{
    "clusterName": "myCluster",
    "defaultReplicaSet": {
        "name": "default",
        "ssl": "REQUIRED",
        "status": "OK",
        "statusText": "Cluster is ONLINE and can tolerate up to ONE failure.",
        "topology": {
            "mgr-node1:3306": {
                "address": "mgr-node1:3306",
                "memberRole": "PRIMARY",
                "mode": "R/W",
                "readReplicas": {},
                "replicationLag": null,
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.27"
            },
            "mgr-node2:3306": {
                "address": "mgr-node2:3306",
                "memberRole": "PRIMARY",
                "mode": "R/W",
                "readReplicas": {},
                "replicationLag": "00:00:00.677674",
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.27"
            },
            "mgr-node3:3306": {
                "address": "mgr-node3:3306",
                "memberRole": "PRIMARY",
                "mode": "R/W",
                "readReplicas": {},
                "replicationLag": "00:00:00.540349",
                "role": "HA",
                "status": "ONLINE",
                "version": "8.0.27"
            }
        },
        "topologyMode": "Multi-Primary"
    },
    "groupInformationSourceMember": "mgr-node2:3306"
```

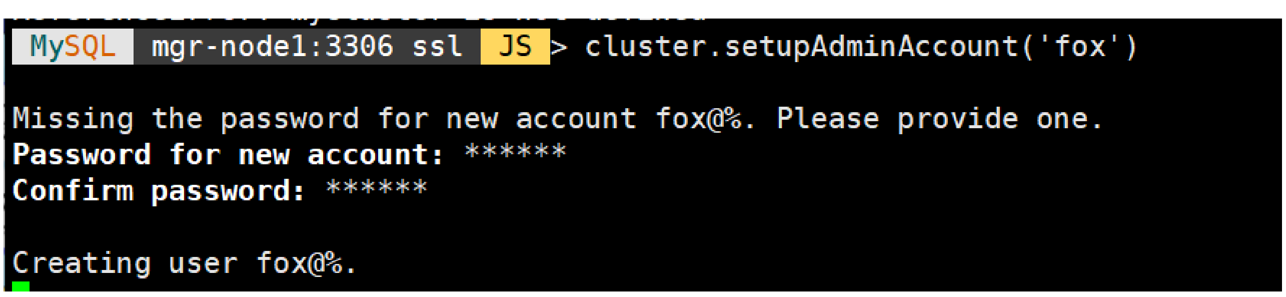
销毁集群

删除与群集关联的所有元数据和配置，并禁用实例上的组复制，但不会删除在实例之间复制的任何数据。要再次创建集群，使用dba.createCluster()

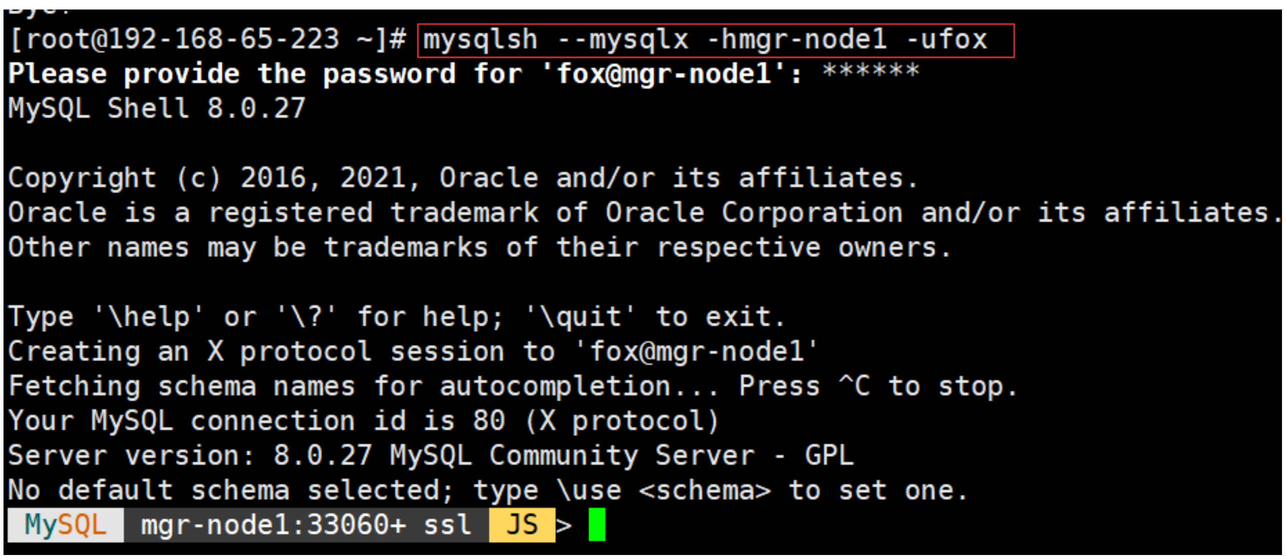
```
var cluster = dba.getCluster()
cluster.dissolve()
```

创建集群管理用户

```
cluster.setupAdminAccount('fox')
```


```
MySQL  mgr-node1:3306 ssl  JS > cluster.setupAdminAccount('fox')

Missing the password for new account fox@%. Please provide one.
Password for new account: ******
Confirm password: ******

Creating user fox@%.
```

```
# 经典MySQL协议连接
mysqlsh --mysql -hmgr-node1 -ufox
# X协议连接
mysqlsh --mysqlx -hmgr-node1 -ufox
```

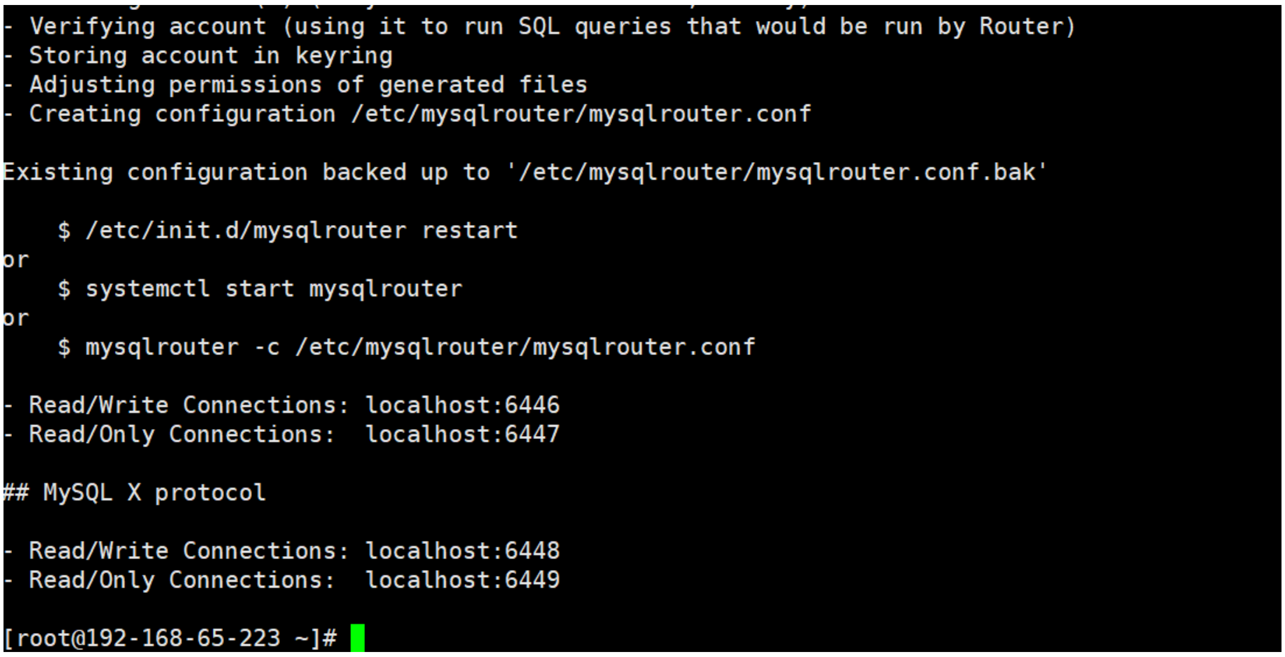

```
[root@192-168-65-223 ~]# mysqlsh --mysqlx -hmgr-node1 -ufox
Please provide the password for 'fox@mgr-node1': ******
MySQL Shell 8.0.27

Copyright (c) 2016, 2021, 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 an X protocol session to 'fox@mgr-node1'
Fetching schema names for autocompletion... Press ^C to stop.
Your MySQL connection id is 80 (X protocol)
Server version: 8.0.27 MySQL Community Server - GPL
No default schema selected; type \use <schema> to set one.
MySQL  mgr-node1:33060+ ssl  JS >
```

7.使用MySQL Router连接集群

配置路由器

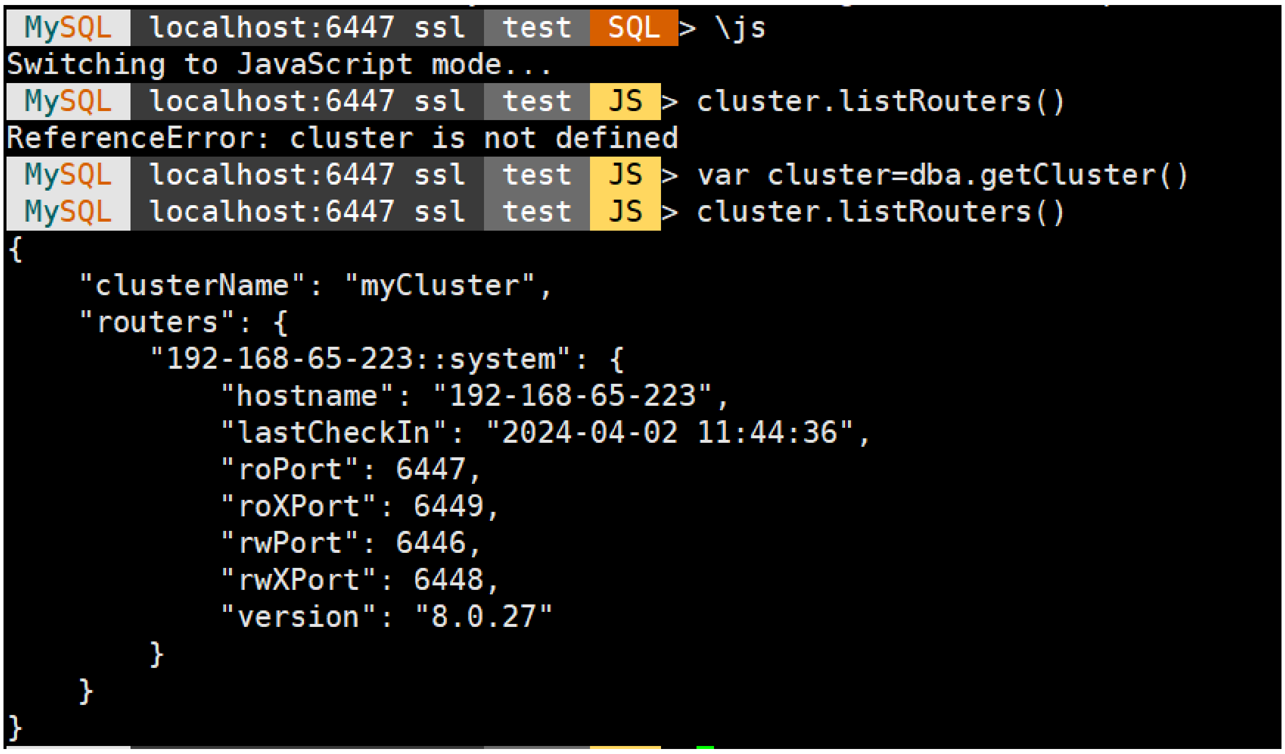
```
mysqlrouter --bootstrap root@mgr-node2:3306 --force --user=root
#或者指定host
mysqlrouter --bootstrap root@mgr-node2:3306 --force --user=root --report-host mgr
```

注意，如果用户之前为该实例配置过路由，则可以通过指定force选项强制引导启动。


```
- 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'

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

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

## MySQL X protocol

- Read/Write Connections: localhost:6448
- Read/Only Connections:  localhost:6449

[root@192-168-65-223 ~]#
```

上面的内容是引导启动路由器时输出的信息，信息提示，MySQL经典协议使用6446端口和6447端口，X协议使用6448端口和6449端口，每种协议使用的两个端口分别用于读写和只读。

在一个运行的集群中，AdminAPI 可以引导多个路由器。用户可以使用cluster.listRouters()方法显示所有注册的路由器列表。


```
 MySQL  localhost:6447 ssl  test  SQL > \js
Switching to JavaScript mode...
 MySQL  localhost:6447 ssl  test  JS > cluster.listRouters()
ReferenceError: cluster is not defined
 MySQL  localhost:6447 ssl  test  JS > var cluster=dba.getCluster()
 MySQL  localhost:6447 ssl  test  JS > cluster.listRouters()
{
    "clusterName": "myCluster",
    "routers": {
        "192-168-65-223::system": {
            "hostname": "192-168-65-223",
            "lastCheckIn": "2024-04-02 11:44:36",
            "roPort": 6447,
            "roXPort": 6449,
            "rwPort": 6446,
            "rwXPort": 6448,
            "version": "8.0.27"
        }
    }
}
```

启动路由器

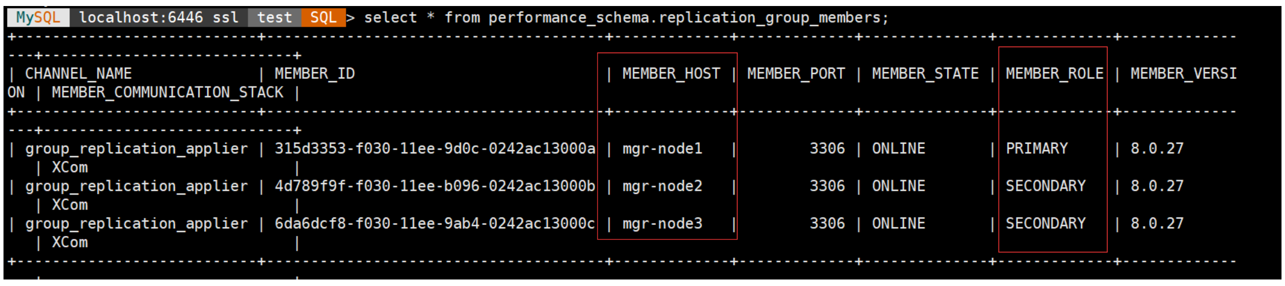
```
mysqlrouter &
```

路由器已经成功启动。现在，使用MySQL Shell连接路由器进行验证。

```
#连接mysqlrouter
[root@192-168-65-223 ~]# mysqlsh root@localhost:6446 --sql
MySQL  localhost:6446 ssl  SQL > use test;
MySQL  localhost:6446 ssl  test  SQL > select * from t;
```

查看集群成员信息

```
MySQL  localhost:6446 ssl  test  SQL > select * from
performance_schema.replication_group_members;
```

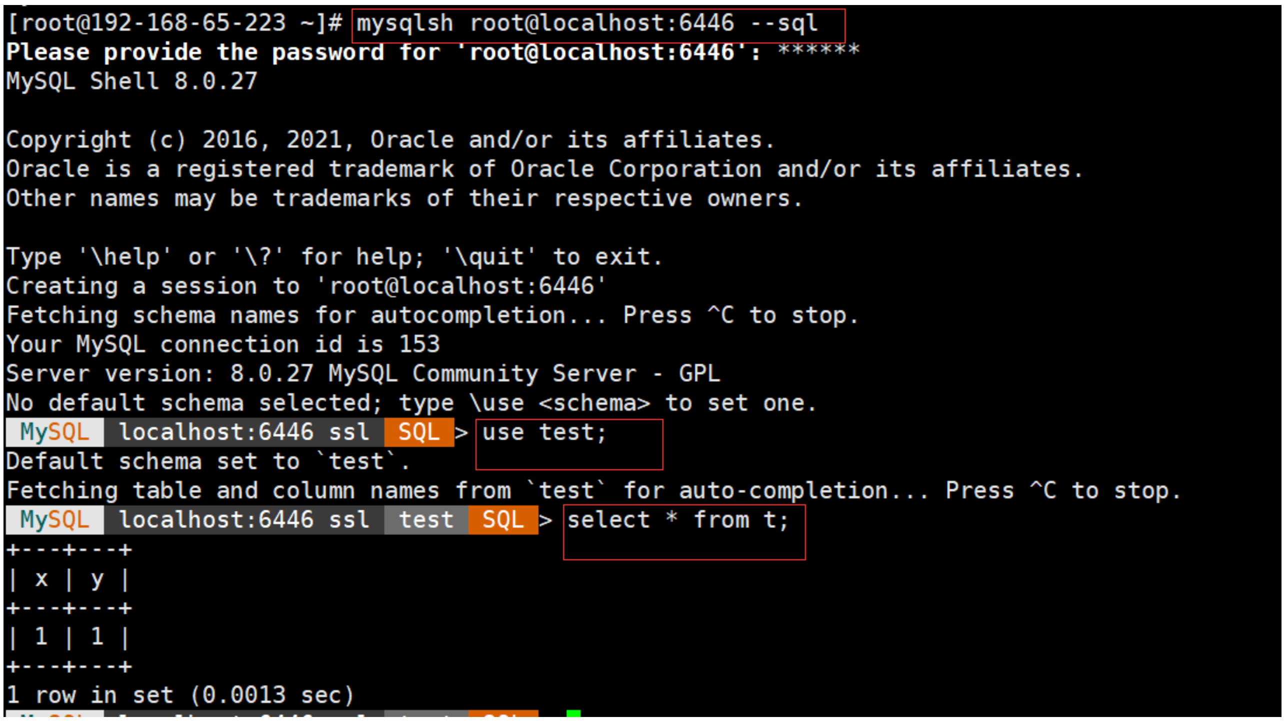


```
 MySQL  localhost:6446 ssl  test  SQL > select * from performance_schema.replication_group_members;
+---------------------------+--------------------------------------+-------------+-------------+--------------+-------------+-------------
---+----------------------------+
| CHANNEL_NAME              | MEMBER_ID                            | MEMBER_HOST | MEMBER_PORT | MEMBER_STATE | MEMBER_ROLE | MEMBER_VERSI
ON | MEMBER_COMMUNICATION_STACK |
+---------------------------+--------------------------------------+-------------+-------------+--------------+-------------+-------------
---+----------------------------+
| group_replication_applier | 315d3353-f030-11ee-9d0c-0242ac13000a | mgr-node1   |        3306 | ONLINE       | PRIMARY     | 8.0.27
   | XCom                       |
| group_replication_applier | 4d789f9f-f030-11ee-b096-0242ac13000b | mgr-node2   |        3306 | ONLINE       | SECONDARY   | 8.0.27
   | XCom                       |
| group_replication_applier | 6da6dcf8-f030-11ee-9ab4-0242ac13000c | mgr-node3   |        3306 | ONLINE       | SECONDARY   | 8.0.27
   | XCom                       |
+---------------------------+--------------------------------------+-------------+-------------+--------------+-------------+-------------
```

测试

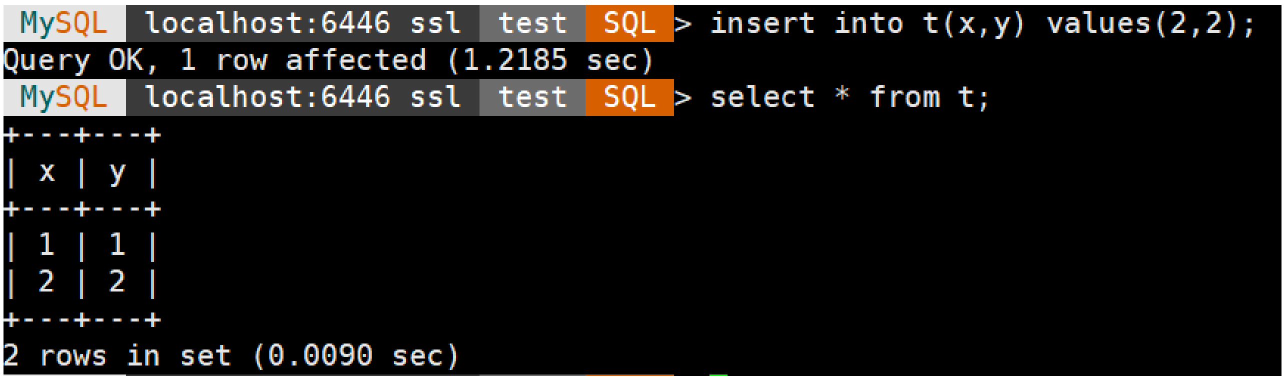
测试读写端口6446

用户可以通过连接本机的6446端口连接到MySQL实例



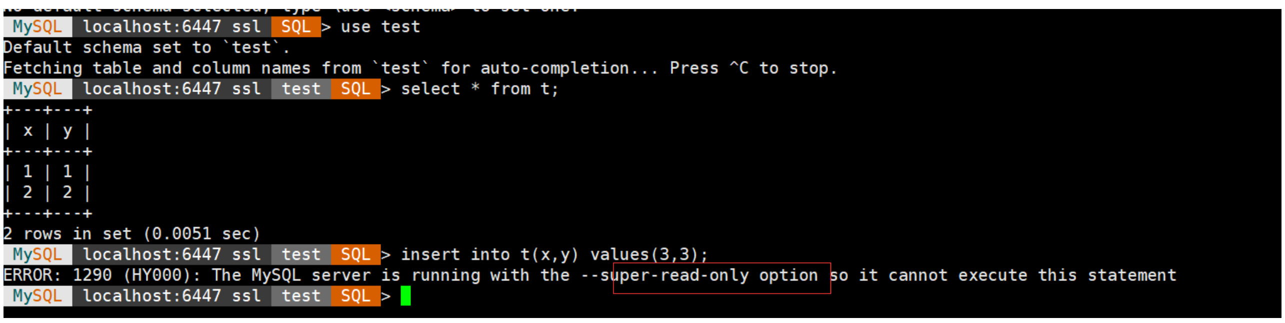
```
[root@192-168-65-223 ~]# mysqlsh root@localhost:6446 --sql
Please provide the password for 'root@localhost:6446': ******
MySQL Shell 8.0.27

Copyright (c) 2016, 2021, 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:6446'
Fetching schema names for autocompletion... Press ^C to stop.
Your MySQL connection id is 153
Server version: 8.0.27 MySQL Community Server - GPL
No default schema selected; type \use <schema> to set one.
 MySQL  localhost:6446 ssl  SQL > use test;
Default schema set to `test`.
Fetching table and column names from `test` for auto-completion... Press ^C to stop.
 MySQL  localhost:6446 ssl  test  SQL > select * from t;
+---+---+
| x | y |
+---+---+
| 1 | 1 |
+---+---+
1 row in set (0.0013 sec)
```

6446为读写端口，也可以插入数据



```
 MySQL  localhost:6446 ssl  test  SQL > insert into t(x,y) values(2,2);
Query OK, 1 row affected (1.2185 sec)
 MySQL  localhost:6446 ssl  test  SQL > select * from t;
+---+---+
| x | y |
+---+---+
| 1 | 1 |
| 2 | 2 |
+---+---+
2 rows in set (0.0090 sec)
```

测试只读端口6647：插入数据报错



```
 MySQL  localhost:6447 ssl  SQL > use test
Default schema set to `test`.
Fetching table and column names from `test` for auto-completion... Press ^C to stop.
 MySQL  localhost:6447 ssl  test  SQL > select * from t;
+---+---+
| x | y |
+---+---+
| 1 | 1 |
| 2 | 2 |
+---+---+
2 rows in set (0.0051 sec)
 MySQL  localhost:6447 ssl  test  SQL > insert into t(x,y) values(3,3);
ERROR: 1290 (HY000): The MySQL server is running with the --super-read-only option so it cannot execute this statement
 MySQL  localhost:6447 ssl  test  SQL >
```

MySQL InnoDB ReplicaSet

基本概述

MySQL Innodb Cluster = MySQL Shell + MySQL Router + MySQL Group Replication(MGR)，全程由 MySQL Shell 来管理操作 MGR 的聚合套件。MySQL 8.0.19 发布后，这种组合延伸到 MySQL Replication（主从复制），也就是 MySQL Shell + MySQL Router + MySQL Replication。

InnoDB ReplicaSet至少由两个MySQL服务器实例组成，并提供用户熟知的主从复制功能，例如读取横向扩展和数据安全性。InnoDB ReplicaSet使用以下MySQL技术。

- **MySQL Shell**：MySQL的高级客户端、管理工具，可以用来管理复制集。
- **MySQL复制**：一组MySQL实例，通过复制能够提供可用性和异步读取的横向扩展。
- **MySQL Router**：一种轻量级的中间件，可在应用程序和InnoDB ReplicaSet之间提供透明的路由。InnoDB ReplicaSet的接口类似于InnoDB Cluster，用户可以利用MySQL Shell使用MySQL实例和MySQL Router。

与InnoDB集群相比，InnoDB ReplicaSet具有多个限制，因此，官方建议尽可能部署InnoDB群集。通常，InnoDB ReplicaSet本身不能提供高可用性。InnoDB ReplicaSet的限制包括：

- **没有自动故障转移**。如果主服务器不可用，则需要使用AdminAPI手动触发故障转移，然后才能再次进行任何更改。但是，辅助实例仍然可用于读取。
- **无法防止因意外停止或不可用而导致部分数据丢失。**暂停之前尚未应用的事务可能会丢失。
- **无法防止崩溃或不可用后出现不一致情况。**如果故障转移在辅助节点仍可用的情况下提升了辅助节点（例如，由于网络分区），则可能会因脑裂而引起不一致。

搭建一主一从的复制集

1. 安装2个数据库实例

> 参考：Docker 安装 MySQL8.0

可以利用docker快速部署2个MySQL实例

主机名（角色）	server_id	宿主机IP	容器固定IP	DB Port
rs-node1(primary)	21	192.168.65.223	172.20.0.20	3331>3306
rs-node2(Secondary)	22	192.168.65.223	172.20.0.21	3332>3306

```
# 创建组复制的网络  保证三个mysql容器之间可以通过容器名访问
docker network create --driver bridge --subnet 172.20.0.0/24 --gateway 172.20.0.1 rs-network

mkdir -p /mysql/rs/node1/data /mysql/rs/node1/conf /mysql/rs/node1/log
mkdir -p /mysql/rs/node2/data /mysql/rs/node2/conf /mysql/rs/node2/log


#以rs-node1配置为例，创建/mysql/rs/node1/conf/custom.cnf，添加以下配置：
vim /mysql/rs/node1/conf/custom.cnf
[mysql]
# 设置mysql客户端默认编码
default-character-set=utf8
[mysqld]
#指定sever_id，多个Mysql实例需要分别改为对应的sever_id
server_id=21
# 必须开启GTID支持
gtid_mode=ON
enforce_gtid_consistency=ON
# 启用二进制日志
log-bin=mysql-bin
#启用并行复制
binlog_transaction_dependency_tracking=WRITESET
replica_preserve_commit_order=ON
replica_parallel_type=LOGICAL_CLOCK
transaction_write_set_extraction=XXHASH64
```

```

# rs-node2同上，注意配置文件路径和修改server_id
vim /mysql/rs/node2/conf/custom.cnf
[mysql]
# 设置mysql客户端默认编码
default-character-set=utf8
[mysqld]
#指定sever_id，多个Mysql实例需要分别改为对应的sever_id
server_id=22
# 必须开启GTID支持
gtid_mode=ON
enforce_gtid_consistency=ON
# 启用二进制日志
log-bin=mysql-bin
#启用并行复制
binlog_transaction_dependency_tracking=WRITESET
replica_preserve_commit_order=ON
replica_parallel_type=LOGICAL_CLOCK
transaction_write_set_extraction=XXHASH64


#运行mysql容器
# 为了便于测试，启动容器时指定好IP、hostname
docker run  -d  \
--name rs-node1 \
--privileged=true \
--restart=always \
--ip 172.20.0.20 \
--hostname rs-node1 \
--add-host  rs-node2:172.20.0.21 \
--network  rs-network \
-p 3331:3306 \
-v /mysql/rs/node1/data:/var/lib/mysql \
-v /mysql/rs/node1/conf:/etc/mysql/conf.d  \
-v /mysql/rs/node1/log:/logs \
-e MYSQL_ROOT_PASSWORD=123456 \
-e TZ=Asia/Shanghai mysql:8.0.27 \
--lower_case_table_names=1

```

```

docker run  -d  \
--name rs-node2 \
--privileged=true \
--restart=always \
--ip 172.20.0.21 \
--hostname rs-node2 \
--add-host  rs-node1:172.20.0.20 \
--network  rs-network \
-p 3332:3306 \
-v /mysql/rs/node2/data:/var/lib/mysql \
-v /mysql/rs/node2/conf:/etc/mysql/conf.d  \
-v /mysql/rs/node2/log:/logs \
-e MYSQL_ROOT_PASSWORD=123456 \
-e TZ=Asia/Shanghai mysql:8.0.27 \
--lower_case_table_names=1



# 在宿主机上配置mysql容器的ip和host映射
vim /etc/hosts
172.20.0.20  rs-node1
172.20.0.21  rs-node2
```

所有实例分别配置远程访问

```
# 以node1为例
docker exec -it rs-node1 /bin/bash
mysql -u root -p123456
#进入mysql执行
ALTER USER 'root'@'%' IDENTIFIED WITH mysql_native_password BY '123456';
flush privileges;
```

2. 配置复制集

1）初始化主节点，创建复制集

```
#进入主节点
mysqlsh root@rs-node1:3306 --js
# 初始化
dba.configureReplicaSetInstance('root@rs-node1:3306')
# 创建复制集，使用异步复制
var rs = dba.createReplicaSet("myrs")
#查看状态
rs.status()
```

2）添加副本节点

```
#将实例添加到复制集
rs.addInstance('root@rs-node2:3306')
```

测试数据是否同步

```
#主节点 rs-node1
[root@192-168-65-223 ~]# docker exec -it rs-node1 bash
root@mgr-node1:/# mysql -uroot -p123456
mysql>
create datebase test;
use test;
create table t(x int primary key auto_increment,y int);
insert into t(x,y) value(1,1);
select * from t;

# 进入从节点rs-node2，查看数据是否同步过来
[root@192-168-65-223 ~]# docker exec -it rs-node2 bash
root@rs-node2:/# mysql -uroot -p123456
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1596
Server version: 8.0.27 MySQL Community Server - GPL

Copyright (c) 2000, 2021, 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 '\h' for help. Type '\c' to clear the current input statement.

mysql> use test;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from t;
+---+------+
| x | y    |
+---+------+
| 1 |    1 |
+---+------+
1 row in set (0.00 sec)

```

3. 配置 MySQL Router 路由器

```
[root@192-168-65-223 ~]# mysqlrouter --bootstrap root@rs-node1:3306 --force --user=root
```

重启Mysql Router

```
[root@192-168-65-223 ~]# ps -ef|grep mysqlrouter
root     16993 14238  6 11:30 pts/1    00:18:01 mysqlrouter
root     21637 14178  0 16:22 pts/0    00:00:00 grep --color=auto mysqlrouter
[root@192-168-65-223 ~]# kill -9 16993
# 启动Mysql Router
[root@192-168-65-223 ~]# mysqlrouter &
```

4. 测试

用户可以通过连接本机的6446端口连接到MySQL实例

```
[root@192-168-65-223 ~]# mysqlsh root@localhost:6446 --sql
# 可以查询到插入的测试数据
 MySQL  localhost:6446 ssl  SQL > select * from test.t;
+---+---+
| x | y |
+---+---+
| 1 | 1 |
+---+---+

```