第六节：组复制常规操作：---基础篇

概要：在MGR集群运行时，可以使用依赖于集群操作协调器的UDF自定义函数来对集群做一些在线变更操作。这些自定义函数由8.0.13或更高版本的MGR插件提供。

注意：要使协调器能够在正在运行的集群中有效执行整个集群范围内的配置变更操作，所有节点必须运行在8.0.13或更高版本中，并且提前加载好了MGR插件。

要调用这些自定义函数时，直接用客户端链接到集群的任意一个节点上，执行select调用该UDF即可。

由于这些变更操作要在集群的所有节点上完成，因此有几点需要格外留意：

1.在集群内任意一个节点上执行函数调用就可以。

2.集群内的所有成员必须处于online状态。

3.在变更操作执行期间，不能有任何新节点尝试加入集群。

4.为防止脑裂，同一集群在同一事件只能接受一个在线变更操作，换句话说，在线变更操作在集群内只能串行执行

5.集群中的所有节点必须运行在MySQL 8.0.13或更高版本上。

1.在线变更集群primary节点：

在单主集群中，group_replication_set_as_primary自定义函数可以将集群中某个节点提升为primary节点。多主集群中执行该操作前后对集群无任何影响。

注意:单主模式下，建议整个集群中运行的MySQL server不低于8.0.17版本。

如果在调用变更过程中没有将要选举的server的server uuid作为该函数的参数传入，则集群按照单基数节点选举规则来选举出新的primary节点。（自动选举集群中最低版本的成员作为primary节点）。

如果在运行MySQL 8.0.17及其更高版本的集群节点上调用UDF，且所有节点都运行在MySQL 8.0.17或更高版本中，且集群中存在不同版本的节点时，则只能基于小版本指定集群中最低MySQL Server版本的节点做为primary节点。这样可以确保集群与新功能保持兼容性。

如果任何成员正在运行MySQL 8.0.13和MySQL 8.0.16之间的MySQL Server版本，则可以指定任何新的集群节点作为primary节点，但建议选择集群中最低MySQL Server版本的节点作为primary节点。如果没有指定新的primary节点，则选举最低主版本的节点作为primary节点。

该自定义函数需要将新primary节点的server uuid作为参数传递到该函数内部才能生效，如下所示：

查询组成员的状态信息，可以看到当前node1为primary节点（MEMBER_ROLE列值为PRIMARY）

```
select * from replication_group_members;

+---------------------------+--------------------------------------+-------------
+-------------+--------------+-------------+----------------+

| CHANNEL_NAME | MEMBER_ID | MEMBER_HOST | MEMBER_PORT | MEMBER_STATE |
MEMBER_ROLE | MEMBER_VERSION |

+---------------------------+--------------------------------------+-------------
+-------------+--------------+-------------+----------------+

| group_replication_applier | 2d283e92-de7b-11e9-a14d-525400c33752 | node2 |
3308 | ONLINE | SECONDARY | 8.0.21 |
```

```
| group_replication_applier | 2e33b2a7-de7b-11e9-9a21-525400bdd1f2 | node3 |
3307 | ONLINE | SECONDARY | 8.0.21 |

| group_replication_applier | 320675e6-de7b-11e9-b3a9-5254002a54f2 | node1 |
3306 | ONLINE | PRIMARY | 8.0.21 |

+---------------------------+--------------------------------------+-------------
+-------------+--------------+-------------+----------------+

3 rows in set (0.01 sec)
```

指定node2为新的primary节点

```
SELECT group_replication_set_as_primary('2d283e92-de7b-11e9-a14d-525400c33752');

+------------------------------------------------------------------------------+

| group_replication_set_as_primary('2d283e92-de7b-11e9-a14d-525400c33752') |

+------------------------------------------------------------------------------+

| Primary server switched to: 2d283e92-de7b-11e9-a14d-525400c33752 |

+------------------------------------------------------------------------------+

1 row in set (0.03 sec)
```

再次查看集群节点状态信息，可以发现node2已经被切换为了节点

```
select * from replication_group_members;

+---------------------------+--------------------------------------+-------------
+-------------+--------------+-------------+----------------+

| CHANNEL_NAME | MEMBER_ID | MEMBER_HOST | MEMBER_PORT | MEMBER_STATE |
MEMBER_ROLE | MEMBER_VERSION |

+---------------------------+--------------------------------------+-------------
+-------------+--------------+-------------+----------------+

| group_replication_applier | 2d283e92-de7b-11e9-a14d-525400c33752 | node2 |
3307 | ONLINE | PRIMARY | 8.0.21 |

| group_replication_applier | 2e33b2a7-de7b-11e9-9a21-525400bdd1f2 | node3 |
3308 | ONLINE | SECONDARY | 8.0.21 |

| group_replication_applier | 320675e6-de7b-11e9-b3a9-5254002a54f2 | node1 |
3306 | ONLINE | SECONDARY | 8.0.21 |

+---------------------------+--------------------------------------+-------------
+-------------+--------------+-------------+----------------+

3 rows in set (0.00 sec)
```

在集群中执行上述操作后，可以通过如下命令来检查该操作在集群内是否被成功执行：

注：events_stages_current表是记录线程当前正在执行的event信息，所以只有线程正在执行某个语句时才能够查询到该信息，一旦线程执行完成，event信息就会被清除：

```
SELECT event_name, work_completed, work_estimated FROM
performance_schema.events_stages_current WHERE event_name LIKE
"%stage/group_rpl%";

+-------------------------------------------------------------------------------
--+----------------+----------------+

| event_name | work_completed | work_estimated |

+-------------------------------------------------------------------------------
--+----------------+----------------+

| stage/group_rpl/Primary Election: Waiting for members to turn on
super_read_only | 3 | 3 |

+-------------------------------------------------------------------------------
--+----------------+----------------+
```

重要：如果该primary节点本身还作为主从复制中的slave节点，则在停止异步复制之前，集群不允许进行primary节点的切换操作。

2.集群模式变更：

无论是单主模式还是多主模式。用于修改集群运行模式的函数可以在集群中任意节点上执行该操作。

注意：单主模式中，调用 group_replication_switch_to_single_primary_mode()自定义函数可以成功执行，但是对于整个集群来说，并没有任何的作用。对于多主模式来讲，同样的道理。

多主---->单主：

如果线上运行的MGR集群目前处于多主模式，则使用group_replication_switch_to_single_primary_mode()自定义函数可将MGR集群由多主模式改为单主模式：

变更为单主模式时，还会在集群内所有节点上禁用多主模式下严格的一致性检查，在调用group_replication_switch_to_single_primary_mode() UDF时，可以为其指定一个节点的server_uuid字符串作为参数，这样指定的节点将成为新的主要节点，如果未指定，则将自动根据单基数集群选举策略选出新的primary节点。

示例如下：

先查看一下组成员的状态信息，可以发现此时3个成员都为primary节点

```
select * from replication_group_members;

+---------------------------+--------------------------------------+-------------
+-------------+--------------+-------------+----------------+

| CHANNEL_NAME | MEMBER_ID | MEMBER_HOST | MEMBER_PORT | MEMBER_STATE |
MEMBER_ROLE | MEMBER_VERSION |

+---------------------------+--------------------------------------+-------------
+-------------+--------------+-------------+----------------+
```

```
| group_replication_applier | 2d283e92-de7b-11e9-a14d-525400c33752 | node2 |
3308 | ONLINE | PRIMARY | 8.0.21 |

| group_replication_applier | 2e33b2a7-de7b-11e9-9a21-525400bdd1f2 | node3 |
3307 | ONLINE | PRIMARY | 8.0.21 |

| group_replication_applier | 320675e6-de7b-11e9-b3a9-5254002a54f2 | node1 |
3306 | ONLINE | PRIMARY | 8.0.21 |

+---------------------------+--------------------------------------+-------------
+-------------+--------------+-------------+----------------+

3 rows in set (0.00 sec)
```

查看集群模式系统变量，发现此时单主模式被关闭，集群处于多主模式下

```
show variables like '%group_replication_single_primary_mode%';

+---------------------------------------+-------+

| Variable_name | Value |

+---------------------------------------+-------+

| group_replication_single_primary_mode | OFF |

+---------------------------------------+-------+

1 row in set (0.00 sec)
```

执行模式切换，指定集群内一个节点的server_uuid作为group_replication_switch_to_single_primary_mode() UDF的参数

```
SELECT group_replication_switch_to_single_primary_mode('320675e6-de7b-11e9-b3a9-
5254002a54f2');

+-------------------------------------------------------------------------------
---------+

| group_replication_switch_to_single_primary_mode('320675e6-de7b-11e9-b3a9-
5254002a54f2') |

+-------------------------------------------------------------------------------
---------+

| Mode switched to single-primary successfully. |

+-------------------------------------------------------------------------------
---------+

1 row in set (0.02 sec)
```

再次查看集群内节点的状态信息，可以发现指定的server_uuid(node1)成员成为了primary节点

```
select * from replication_group_members;
```

```
+---------------------------+--------------------------------------+-------------
+-------------+--------------+-------------+----------------+

| CHANNEL_NAME | MEMBER_ID | MEMBER_HOST | MEMBER_PORT | MEMBER_STATE |
MEMBER_ROLE | MEMBER_VERSION |

+---------------------------+--------------------------------------+-------------
+-------------+--------------+-------------+----------------+

| group_replication_applier | 2d283e92-de7b-11e9-a14d-525400c33752 | node2 |
3307 | ONLINE | SECONDARY | 8.0.21 |

| group_replication_applier | 2e33b2a7-de7b-11e9-9a21-525400bdd1f2 | node3 |
3308 | ONLINE | SECONDARY | 8.0.21 |

| group_replication_applier | 320675e6-de7b-11e9-b3a9-5254002a54f2 | node1 |
3306 | ONLINE | PRIMARY | 8.0.21 |

+---------------------------+--------------------------------------+-------------
+-------------+--------------+-------------+----------------+

3 rows in set (0.00 sec)
```

再次查看组模式系统变量值，发现已经被启用，组处于单主模式下

```
show variables like '%group_replication_single_primary_mode%';

+---------------------------------------+-------+

| Variable_name | Value |

+---------------------------------------+-------+

| group_replication_single_primary_mode | ON |

+---------------------------------------+-------+

1 row in set (0.01 sec)
```

注意：如果在调用变更过程中没有将要选举的server的server uuid作为该函数的参数传入，则集群按照单基数节点选举规则来选举出新的primary节点。（自动选举集群中最低版本的成员作为primary节点）。

如果在运行MySQL 8.0.17及其更高版本的集群节点上调用UDF，且所有节点都运行在MySQL 8.0.17或更高版本中，且集群中存在不同版本的节点时，则只能基于小版本指定集群中最低MySQL Server版本的节点做为primary节点。这样可以确保集群与新功能保持兼容性。

如果任何成员正在运行MySQL 8.0.13和MySQL 8.0.16之间的MySQL Server版本，则可以指定任何新的集群节点作为primary节点，但建议选择集群中最低MySQL Server版本的节点作为primary节点。如果没有指定新的primary节点，则选举最低主版本的节点作为primary节点。

通过如下命令来检查该操作是否在集群内被正确执行：

```
SELECT event_name, work_completed, work_estimated FROM
performance_schema.events_stages_current WHERE event_name LIKE
"%stage/group_rpl%";

+-------------------------------------------------------------------------------+---
------------+----------------+

| event_name | work_completed | work_estimated |

+-------------------------------------------------------------------------------+---
------------+----------------+

| stage/group_rpl/Primary Switch: waiting for pending transactions to finish | 4
| 20 |

+-------------------------------------------------------------------------------+---
------------+----------------+
```

单主---->多主：

对于已经处于单主模式的线上集群，使用group_replication_switch_to_multi_primary_mode() UDF执行如下语句，将单主模式下运行的集群修改为多主模式：

查看集群节点的状态信息，可以发现目前集群处于单主模式（有一个PRIMARY成员，2个SECONDARY成员）

```
select * from replication_group_members;

+---------------------------+--------------------------------------+-------------
+-------------+--------------+-------------+----------------+

| CHANNEL_NAME | MEMBER_ID | MEMBER_HOST | MEMBER_PORT | MEMBER_STATE |
MEMBER_ROLE | MEMBER_VERSION |

+---------------------------+--------------------------------------+-------------
+-------------+--------------+-------------+----------------+

| group_replication_applier | 2d283e92-de7b-11e9-a14d-525400c33752 | node2 |
3307 | ONLINE | PRIMARY | 8.0.21 |

| group_replication_applier | 2e33b2a7-de7b-11e9-9a21-525400bdd1f2 | node3 |
3308 | ONLINE | SECONDARY | 8.0.21 |

| group_replication_applier | 320675e6-de7b-11e9-b3a9-5254002a54f2 | node1 |
3306 | ONLINE | SECONDARY | 8.0.21 |

+---------------------------+--------------------------------------+-------------
+-------------+--------------+-------------+----------------+

3 rows in set (0.00 sec)
```

查看集群模式系统变量值，发现已经被启用，集群处于单主模式下

```
show variables like '%group_replication_single_primary_mode%';

+---------------------------------------+-------+

| Variable_name | Value |

+---------------------------------------+-------+

| group_replication_single_primary_mode | ON |

+---------------------------------------+-------+

1 row in set (0.01 sec)
```

执行如下语句切换到多主模式（不能指定集群节点的系统变量server_uuid值作为参数，

```
否则报错："ERROR 1123 (HY000): Can't initialize function
'group_replication_switch_to_multi_primary_mode'; Wrong arguments: This function
takes no arguments."）

SELECT group_replication_switch_to_multi_primary_mode();

+-----------------------------------------------------+

| group_replication_switch_to_multi_primary_mode() |

+-----------------------------------------------------+

| Mode switched to multi-primary successfully. |

+-----------------------------------------------------+

1 row in set (1.02 sec)
```

再次查看集群节点状态信息，可以发现3个节点的MEMBER_ROLE列值都为PRIMARY，表示此时集群内3个节点都可读写

```
select * from replication_group_members;

+---------------------------+--------------------------------------+-------------
+-------------+--------------+-------------+----------------+

| CHANNEL_NAME | MEMBER_ID | MEMBER_HOST | MEMBER_PORT | MEMBER_STATE |
MEMBER_ROLE | MEMBER_VERSION |

+---------------------------+--------------------------------------+-------------
+-------------+--------------+-------------+----------------+

| group_replication_applier | 2d283e92-de7b-11e9-a14d-525400c33752 | node2 |
3307 | ONLINE | PRIMARY | 8.0.21 |

| group_replication_applier | 2e33b2a7-de7b-11e9-9a21-525400bdd1f2 | node3 |
3308 | ONLINE | PRIMARY | 8.0.21 |

| group_replication_applier | 320675e6-de7b-11e9-b3a9-5254002a54f2 | node1 |
3306 | ONLINE | PRIMARY | 8.0.21 |
```

```
+---------------------------+--------------------------------------+-------------
+-------------+--------------+-------------+----------------+

3 rows in set (0.01 sec)
```

再次查看集群模式系统变量，发现已经被关闭，集群处于多主模式下

```
show variables like '%group_replication_single_primary_mode%';

+---------------------------------------+-------+

| Variable_name | Value |

+---------------------------------------+-------+

| group_replication_single_primary_mode | OFF |

+---------------------------------------+-------+

1 row in set (0.00 sec)
```

调用group_replication_switch_to_multi_primary_mode() 函数之后，集群内经过一些能够确保数据的安全性和一致性的协调操作之后，集群中所有的节点都成为了primary节点。

将单主模式运行的集群更改为以多主模式运行时，如果集群中存在MySQL 8.0.17或更高版本的节点，且高于集群中所有节点的最低版本时，会自动将MySQL 8.0.17或更高版本的节点置于只读模式。运行在MySQL 8.0.16或更低版本的节点不执行此检查，且始终处于读写模式。所以，在MySQL 8.0.17或更高版本中，多主模式下不一定是所有节点都会处于读写模式，如果你的集群中使用了多个版本的MySQL Server，则在这个地方需要留意。

通过如下操作来检查确认变更操作在集群内是否被正确执行：

```
SELECT event_name, work_completed, work_estimated FROM
performance_schema.events_stages_current WHERE event_name LIKE
"%stage/group_rpl%";

+-----------------------------------------------------------------------+---------
-------+----------------+

| event_name | work_completed | work_estimated |

+-----------------------------------------------------------------------+---------
-------+----------------+

| stage/group_rpl/Multi-primary Switch: applying buffered transactions | 0 | 1 |

+-----------------------------------------------------------------------+---------
-------+----------------+
```

3.MGR集群并发可读写实例数量：

使用 group_replication_get_write_concurrency()函数来检查确认一个集群内最大并行可写实例数（并行在集群节点之间广播传输的最大并发可写实例数）这个最大值称为集群的事件范围，可以通过调整该值来优化组复制的性能。

检查集群的写并发性：使用group_replication_get_write_concurrency() UDF在运行时检查集群的事件范围值（注：该函数不需要传入参数），如下：

```
select group_replication_get_write_concurrency();

+-----------------------------------------+

| group_replication_get_write_concurrency() |

+-----------------------------------------+

| 10 |

+-----------------------------------------+

1 row in set (0.00 sec)
```

配置集群的写并发性：使用group_replication_set_write_concurrency() UDF设置集群可以并行执行的可写实例的最大数量，如下：

```
select group_replication_set_write_concurrency(100);

+-------------------------------------------------------------------------------
---+

| group_replication_set_write_concurrency(100) |

+-------------------------------------------------------------------------------
---+

| UDF is asynchronous, check log or call
group_replication_get_write_concurrency(). |

+-------------------------------------------------------------------------------
---+

1 row in set (0.00 sec)
```

使用该UDF自定义函数配置集群的写并发性（修改集群中允许并行执行的可写实例的最大数量）需要用户具有GROUP_REPLICATION_ADMIN权限。

默认的最大并发读写实例数为10，该函数的有效参数值为10~200。

4.集群通信协议版本设置：

从MySQL 8.0.16开始，MGR就有了组通信协议的概念。可以显式地管理组复制通信协议的版本，并将其设置为希望组支持的最老的MySQL Server版本号。这使得集群允许由运行在不同MySQL Server版本的节点组成，同时确保向后兼容性。

假如一个MySQL Server的版本为X，则它只能加入通讯协议版本小于等于X的集群。当有新的Server申请加入集群时，它将检查集群中声明的现有通讯协议版本，如果新加入节点支持该版本（符合版本规则），则该节点加入集群且使用集群中已声明的通讯协议版本，如果新加入成员不支持集群声明的通讯协议版本，则加入集群将会失败。

只有新加入集群的成员的通讯协议版本与集群中声明的通讯协议版本兼容时，才允许加入集群。如果存在多个成员同时加入集群时，且他们的通讯协议版本与集群中声明的通讯协议版本相同，则允许同时加入集群，否则，他们只能串行加入集群，具体的判断规则如下：

- 一个MySQL 8.0.16版本的Server可以成功加入使用通信协议版本为5.7.24的集群，因为集群的通讯协议版本为5.7.24，小于新加入成员的版本8.0.16。
- 一个MySQL 5.7.24版本的Server无法成功加入使用通讯协议版本为8.0.16的集群，因为集群的通讯协议版本为8.0.16，大于新加入成员的版本5.7.24。
- 两个MySQL 8.0.16版本的Server不能同时加入使用通讯协议版本为5.7.24的集群，因为集群的通讯协议版本为5.7.24，小于（不等于）新加入成员的版本8.0.16。
- 两个MySQL 8.0.16版本的Server可以同时加入使用通讯协议版本为8.0.16的集群，因为集群的通讯协议版本为8.0.16，等于新加入成员的版本8.0.16。

  为什么会存在这种版本限制呢？在单主模式下，集群中最小版本的成员通常会被选举为primary节点，相当于主从复制拓扑中的主库，而其他更高版本的成员通常会置为辅助节点，相当于主从复制拓扑中的从库。在主从复制拓扑中，为了保证从库回放主库的二进制日志时能够向下兼容主库，所以从库的版本必须大于等于主库的版本。在一个正常运行的集群中，一定存在着一个已经选举成功的写节点，而写节点通常是版本最低的（集群中的通讯协议版本通常是以组中最低版本的成员为准），新加入一个成员就相当于在主从复制拓扑中新加入一个从库，所以，同理，在组复制拓扑中，辅助节点为了保证版本能够向下兼容primary节点，新加入的节点版本必须大于等于集群中的通讯协议版本（集群中所有节点中最低的MySQL Server版本）。

  注意：通讯协议版本不意味着集群中存在该版本的节点，例如：默认情况下，MySQL 版本为8.0.17的Server使用的是8.0.16的通讯协议版本，该协议版本表示该集群当前声明的所能支持的最低MySQL Server版本号。

使用group_replication_get_communication_protocol() UDF检查组使用的通信协议，UDF返回组支持的最老的MySQL Server版本。组中所有现有成员中执行该UDF都会返回相同的通信协议版本。

如下：

```
SELECT group_replication_get_communication_protocol();

+-----------------------------------------------+

| group_replication_get_communication_protocol() |

+-----------------------------------------------+

| 8.0.20 |

+-----------------------------------------------+

1 row in set (0.00 sec)
```

如果要更改集群的通信协议版本，以便可以加入较早版本的节点，使用group_replication_set_communication_protocol（）UDF指定要允许的MySQL Server版本。这会使集群退回到兼容的通信协议版本。要使用此UDF，必须具有GROUP_REPLICATION_ADMIN特权，并且在执行该语句时，所有现有的组成员都必须处于online状态。例如：

注意：group_replication_get_communication_protocol () UDF返回的是集群中支持的最低MySQL Server版本（集群中声明的通讯协议版本），这可能与使用group_replication_set_communication_protocol () UDF设置组的通讯协议版本时传递给它的版本不同、也可能与组中的最低成员的MySQL Server版本不同。

```
SELECT group_replication_set_communication_protocol("5.7.25");

+-----------------------------------------------------------------------------------
---+

| group_replication_set_communication_protocol("5.7.25") |

+-----------------------------------------------------------------------------------
---+

| The operation group_replication_set_communication_protocol completed
successfully |

+-----------------------------------------------------------------------------------
---+

1 row in set (0.00 sec)

SELECT group_replication_get_communication_protocol();

+------------------------------------------------+

| group_replication_get_communication_protocol() |

+------------------------------------------------+

| 5.7.24 |

+------------------------------------------------+

1 row in set (0.00 sec)
```

如果将组中所有成员都升级到新的MySQL Server版本，则组不会自动将该组的通信协议版本升级到匹配最新的版本。必须使用group_replication_set_communication_protocol() UDF将通信协议版本设置为最新MySQL Server版本。

如下所示：

```
select version();

+-----------+

| version() |

+-----------+

| 8.0.21 |

+-----------+

1 row in set (0.00 sec)
```

```
SELECT group_replication_set_communication_protocol("8.0.21");

+-----------------------------------------------------------------------------------
---+

| group_replication_set_communication_protocol("8.0.21") |

+-----------------------------------------------------------------------------------
---+

| The operation group_replication_set_communication_protocol completed
successfully |

+-----------------------------------------------------------------------------------
---+

1 row in set (0.01 sec)

SELECT group_replication_get_communication_protocol();

+------------------------------------------------+

| group_replication_get_communication_protocol() |

+------------------------------------------------+

| 8.0.20 |

+------------------------------------------------+

1 row in set (0.00 sec)
```

group_replication_set_communication_protocol() UDF作为一个MGR操作实现，因此它同时在MGR集群的所有节点上执行。执行该函数过程中，集群操作会缓冲消息并等待通讯协议版本修改完成之后再将缓冲的消息发送出去。如果某个Server在更改通信协议版本后尝试加入集群，则集群中的节点将使用最新的通讯协议版本来决定是否允许该Server加入集群。