集群架构：单主模式：---基础篇

1.单主模式和多主模式之间的相互切换：

在组复制运行时，不能手动更改系统变量group_replication_single_primary_mode的值（也就是说不能在单主模式和多主模式之间动态切换）。

但从MySQL 8.0.13开始，可以使用group_replication_switch_to_single_primary_mode和group_replication_switch_to_multi_primary_mode自定义函数在组复制运行时将集群从一种模式切换到另一种模式（使用group_replication_switch_to_single_primary_mode函数指定切换到单主模式时，可以不指定组成员UUID，会自动选择一个健康的组成员作为新的主节点，也可以手工在集群中挑选一个健康节点的UUID作为参数，指定为主节点，该函数可以在单主模式下使用，但不会有任何效果也不会报错。只有在多主模式下使用时才有效，表示将多主模式切换为单主模式；group_replication_switch_to_multi_primary_mode函数指定切换到多主模式时，不需要指定UUID参数，否则会报错，在多主模式下也可以正常使用该函数，但也不会有任何效果也不会报错。只有在单主模式下使用才有效果，表示将单主模式切换为多主模式。使用这些自定义函数管理更改集群模式的过程，可以确保数据的安全性和一致性。在早期版本中，要更改集群的模式，必须先停止组复制并更改所有成员上的group_replication_single_primary_mode系统变量的值。然后将集群进行完全的重新引导(由设置了系统变量group_replication_bootstrap_group=ON的Server引导)，以使得新操作配置更改生效。注意：不需要重新启动MySQL Server进程，只需要将组复制重新引导即可。

无论部署的模式如何，组复制都不会处理客户端请求的故障转移。该工作必须由中间件框架(如MySQL Router 8.0、代理、连接器或应用程序本身)来处理。

PS：虽然从MySQL 8.0.13版本开始，从多主模式切换为单主模式时，不需要停止组复制，但是，需要将系统变量在所有节点上设置为ON（使用自定义函数做在线切换可以自动修改，不需要人工参与）

2.单主模式：

在单主模式下，组复制会强制只有一个节点作为可读写节点，因此与多主模式相比，一致性检查的严格性可能会降低，并且不需要特别小心地处理DDL语句。选项group_replication_enforce_update_everywhere_checks启用或禁用组的严格一致性检查。在单主模式下部署或要将组从多主模式更改为单主模式时，必须将此系统变量设置为OFF。

在单主模式下（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（同一个组中，不能将组的成员部署在不同的模式中，例如：一个成员配置为多主模式，而另一个成员配置为单主模式）。ON表示单主模式，也是默认模式，OFF表示多主模式。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。

被指定为主节点的成员可以通过以下方式进行角色的更改：

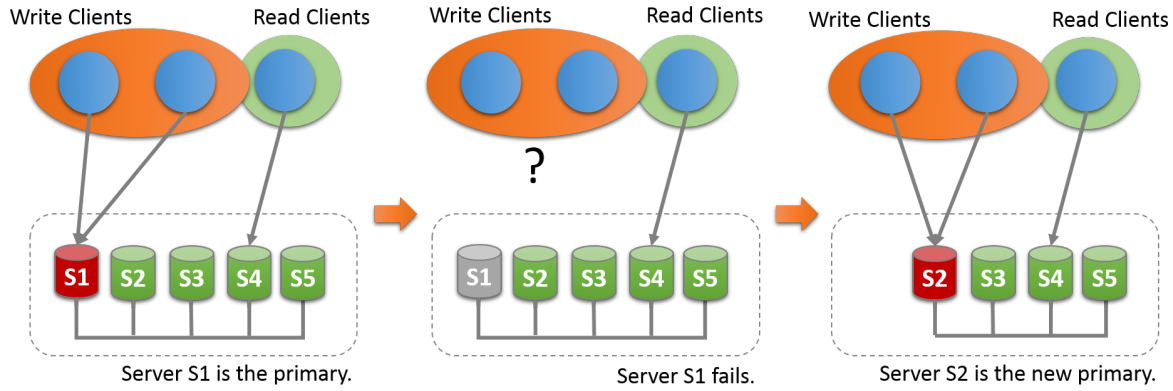
如果现有的主节点自愿或意外离开该组，则会自动选择一个新的主节点。

可以使用group_replication_set_as_primary函数来指定特定成员作为新的主节点。

如果使用group_replication_switch_to_single_primary_mode将以多主模式运行的组更改为以单主要模式运行，则将自动选举新的主或者可以通过该函数手工指定一个新的主节点。

自动选择新的主节点或手动指定新的主节点时，它会自动设置为可读写，其他组成员保留为只读服务器，下图显示了此过程：

ps:上述使用自定义函数的在线变更操作需要集群内所有节点都运行在MySQL 8.0.13或更高版本时才能使用。



当集群中一个节点被选为新的主节点时，它可能积压了已应用于旧主但尚未应用于新主节点已经变更了的数据。在这种情况下，在新的主节点追平旧的主节点的数据前。在新主节点上发起的读写事务可能会和之前的旧数据导致冲突并回滚，而在新主节点上发起只读事务可能导致读取到旧的数据。

组复制的流控制机制最大程度地减少了数据更新快成员和更新慢成员之间的数据差异，如果激活并适当调整了它，则可以减少这种情况的发生。有关流控制的更多信息，我们后面会深入讨论。从MySQL 8.0.14开始，可以使用group_replication_consistency系统变量来配置集群的事务一致性级别，以防止出现上述问题。设置BEFORE_ON_PRIMARY_FAILOVER（或更高的一致性级别）将在新选举的主节点上保留新事务，直到应用完成了积压的历史事务后才会接着应用在新主节点上发起的读写事务。

ps:如果一个集群中没有使用流控与事务一致性保证，那么在将应用程序重新路由到新的主节点之前建议等待新主节点应用完成与复制相关的中继日志（这里指的是已经通过冲突认证检测，但在relay log中还未来得及回放的日志）。

3.选主方式：

自动选主过程包括每个成员查看集群的新视图，确认当前集群内存活的成员，然后选择最合适的成员作为新主。每个成员都按照其MySQL Server版本中的选主算法在本地做出自己的决定。因为所有成员都必须做出相同的决定，所以如果其他组成员正在运行较低版本的MySQL Server，则成员将调整其主要选举算法，从而使其与该组中拥有最低MySQL Server版本的成员具有相同的行为。

选则新的主时，需要考虑的因素如下：

考虑的第一个因素是哪个或哪些成员运行最低版本的MySQL Server。如果所有组成员都运行MySQL 8.0.17或更高版本，则首先按其发行版的修补程序版本对成员进行排序。如果任何成员运行的是MySQL Server 5.7或MySQL 8.0.16或更低版本，则首先按其发行版的主要版本对成员进行排序，而忽略修补程序版本。

第二个考虑因素是如果有多个成员运行最低版本的MySQL Server，则考虑的第二个因素是每个成员的成员权重，具体由成员上的group_replication_member_weight系统变量决定。

如果组中的任何成员运行在MySQL Server 5.7上，此时对该组成员忽略第二个考虑因素（因为系统变量group_replication_member_weight是8.0版本引入，5.7版本不支持）。

如果该集群成员支持系统变量group_replication_member_weight，该系统变量指定一个介于0到100之间的数字。所有成员的默认权重均为50，因此将权重设置为低于该权重以降低其排序，将权重设置为该权重以增加其排序。可以使用此加权功能来优先使用更好的硬件，确保在计划的维护窗口期将故障转移到特定成员。

如果有多个成员正在运行最低的MySQL Server版本，并且多个成员中有一个具有最高成员权重（或成员权重无效），则考虑的第三个因素是每个成员生成的server UUID的顺序，由server_uuid系统变量指定。serverUUID最低的成员被选为主节点。

该因素是最后一个可靠的决定因素，因为它可以在第一和第二因素不生效时，使所有的集群成员达成最终一致的决策（按照相同的顺序排序UUID并选择最小值，所有组成员能够达成一致，因为MGR是从5.7版本引入的，且UUID是5.6就已经引入，所有组复制成员都支持UUID）。

4.确认集群中的主节点：

若要确定在单主模式下部署时当前节点是主节点，performance_schema.replication_group_members表中的MEMBER_ROLE列可以确认当前集群的主节点。例如：

```
mysql> SELECT MEMBER_HOST, MEMBER_ROLE FROM
performance_schema.replication_group_members; +--------------------------+-------
------+ | MEMBER_HOST              | MEMBER_ROLE | +--------------------------+----
---------+

+-------------------------+-------------+ | node1| PRIMARY            | +--------
-----------------+-------------+

+-------------------------+-------------+ | node2     | SECONDARY    |+-----------
--------------+-------------+

+-------------------------+-------------+| node3    | SECONDARY     | +------------
-------------+-------------+
```

也可以使用状态变量group_replication_primary_member值进行查看，该变量已经被弃用，计划在将来的版本中删除，所以，不再建议使用该方法来查看组中的主要节点。

```
mysql> SHOW STATUS LIKE 'group_replication_primary_member';
```