第十二节：使用xtrabackup为集群新增节点---进阶篇

1.备份组复制节点数据：

假设集群中已经有3个节点(node1、node2和node3)，通过集群中任意节点的performance_schema.replication_group_members表可以查询到集群中的成员资格与状态信息，如下：

```
SELECT * FROM performance_schema.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 |
3306 | ONLINE | SECONDARY | 8.0.17 |
| group_replication_applier | 2e33b2a7-de7b-11e9-9a21-525400bdd1f2 | node3 |
3306 | ONLINE | SECONDARY | 8.0.17 |
| group_replication_applier | 320675e6-de7b-11e9-b3a9-5254002a54f2 | node1 |
3306 | ONLINE | PRIMARY | 8.0.17 |
+---------------------------+--------------------------------------+-------------
+-------------+--------------+-------------+----------------+
3 rows in set (0.00 sec)
```

使用percona-xtrabackup工具执行备份（假设这里在node3节点上执行备份，注意，除非必须，否则不要在primary节点上执行备份--这里primary节点为node1）。

在node3中执行备份命令

```
[root@node3 ~]# xtrabackup --defaults-file=/etc/my.cnf --user=admin --
password='letsg0' --backup --target-dir=/data//backup/
......
191012 17:19:37 Backup created in directory '/data/backup/'
MySQL binlog position: filename 'mysql-bin.000023', position '231', GTID of the
last change '320675e6-de7b-11e9-b3a9-5254002a54f2:1-4,aaaaaaaa-aaaa-aaaa-aaaa-
aaaaaaaaaaaa:1-2771494'
191012 17:19:37 [00] Writing /data/backup/backup-my.cnf
191012 17:19:37 [00] ...done
191012 17:19:37 [00] Writing /data/backup/xtrabackup_info
191012 17:19:37 [00] ...done
xtrabackup: Transaction log of lsn (11495841139) to (11495843979) was copied.
191012 17:19:37 completed OK!
```

备份完成之后查看备份数据目录中的数据文件

```
[root@node3 ~]# ll /data//backup/
total 2147376
-rw-r----- 1 root root 515 Oct 12 17:19 backup-my.cnf
-rw-r----- 1 root root 2147483648 Oct 12 17:19 ibdata1
drwxr-x--- 2 root root 143 Oct 12 17:19 mysql
-rw-r----- 1 root root 231 Oct 12 17:19 mysql-bin.000023
-rw-r----- 1 root root 52 Oct 12 17:19 mysql-bin.index
```

```
-rw-r----- 1 root root 24117248 Oct 12 17:19 mysql.ibd
drwxr-x--- 2 root root 8192 Oct 12 17:19 performance_schema
drwxr-x--- 2 root root 44 Oct 12 17:19 sbtest
drwxr-x--- 2 root root 28 Oct 12 17:19 sys
drwxr-x--- 2 root root 20 Oct 12 17:19 test
-rw-r----- 1 root root 13631488 Oct 12 17:19 undo_001
-rw-r----- 1 root root 13631488 Oct 12 17:19 undo_002
-rw-r----- 1 root root 109 Oct 12 17:19 xtrabackup_binlog_info
-rw-r----- 1 root root 101 Oct 12 17:19 xtrabackup_checkpoints
-rw-r----- 1 root root 622 Oct 12 17:19 xtrabackup_info
-rw-r----- 1 root root 5632 Oct 12 17:19 xtrabackup_logfile
-rw-r----- 1 root root 262 Oct 12 17:19 xtrabackup_tablespaces
```

2.将备份数据传输到新的等待加入集群的Server node4的节点上：

在这个例子中，我们假设主机都是Linux服务器，并使用SCP在它们之间复制文件
在node4中创建用于存放备份数据的目录

```
[root@node4 ~]# mkdir /data/backup/xtrabackup_dir
```

在node3中将备份数据使用SCP传送到node4中

```
[root@node3 ~]# scp -r /data//backup/ 10.10.30.16:/data/backup/xtrabackup_dir
```

接下来，我们需要使用备份数据恢复目标实例（这里我们是在node4主机上执行恢复操作），先停止目标server的数据库进程，如下：

首先停止目标服务器的数据库进程

```
[root@physical-machine ~]# service mysqld stop
Shutting down MySQL..... SUCCESS!
[root@physical-machine ~]# ps aux |grep mysqld |grep -v grep
[root@physical-machine ~]#
```

注意：如果是用备份数据来恢复发生故障的节点，且故障节点的主机未宕机，则建议将故障节点的datadir下的auto.cnf和mysqld-auto.cnf文件先拷贝出来，稍后使用备份数据恢复完成之后，再拷贝回原目录（auto.cnf中记录着源实例的UUID，使用该UUID来对实例进行恢复时，可确保故障实例可以以原来的身份重新加入集群，而不是重新生成一个UUID以新的身份加入集群；mysqld-auto.cnf记录了Server本地的一些系统变量的临时持久化设置），拷贝命令类似如下

```
[root@node2 ~]# cp -ar /data//mysqldata1/mydata/{auto.cnf,mysqld-auto.cnf}
/data/backup/
[root@node2 ~]# ll /data//backup/*.cnf
-rw-r----- 1 mysql mysql 56 Sep 24 11:27 /data//backup/auto.cnf
-rw-r----- 1 mysql mysql 414 Sep 25 17:12 /data//backup/mysqld-auto.cnf
```

3.清空目标主机中的数据目录（node4），如下：

必须清空共享表空间(innodb_data_home_dir)、独立表空间(datadir)、独立undo表空间(innodb_undo_directory)、redo log日志(innodb_log_group_home_dir)数据文件所在的目录，否则后续恢复将无法拷贝文件（如果可以，中继日志、二进制日志所在的目录也一并清空），类似如下

```
[root@node4 ~]# rm -rf
/data/mysqldata1/{binlog,innodb_log,innodb_ts,mydata,relaylog,undo}/*
```

4.对备份数据执行redo log恢复：

（应用在执行备份过程中生成的redo日志，这些redo 日志是xtrabackup工具自己拷贝的且是存放在xtrabackup_logfile文件中的，而不是存放在InnoDB存储引擎层自己的redo log文件里，另外，xtrabackup备份工具执行备份时，也不会备份存储引擎层的redo log文件，需要执行该步骤来生成全新的redo log文件）。

执行恢复语句

```
[root@node4 ~]# xtrabackup --prepare --target-
dir=/data/backup/xtrabackup_dir/backup/
......
FTS optimize thread exiting.
Starting shutdown...
Log background threads are being closed...
Shutdown completed; log sequence number 11495844364
191014 15:32:39 completed OK!
```

执行完成恢复之后，查看备份数据文件所在目录，可以发现多了一些文件和目录

```
[root@physical-machine ~]# ll /data/backup/xtrabackup_dir/backup/
total 6362156
-rw-r----- 1 root root 515 Oct 12 18:01 backup-my.cnf
-rw-r----- 1 root root 2147483648 Oct 14 15:32 ibdata1
-rw-r----- 1 root root 2147483648 Oct 14 15:32 ib_logfile0 # 发现多了ib_logfile0和
ib_logfile1文件
-rw-r----- 1 root root 2147483648 Oct 14 15:32 ib_logfile1
-rw-r----- 1 root root 12582912 Oct 14 15:32 ibtmp1 # 多了ibtmp1文件
drwxr-x--- 2 root root 6 Oct 14 15:32 #innodb_temp # 多了一个目录
drwxr-x--- 2 root root 143 Oct 12 18:01 mysql
-rw-r----- 1 root root 231 Oct 12 18:01 mysql-bin.000023
-rw-r----- 1 root root 52 Oct 12 18:01 mysql-bin.index
-rw-r----- 1 root root 24117248 Oct 14 15:32 mysql.ibd
drwxr-x--- 2 root root 8192 Oct 12 18:01 performance_schema
drwxr-x--- 2 root root 44 Oct 12 18:00 sbtest
drwxr-x--- 2 root root 28 Oct 12 18:00 sys
drwxr-x--- 2 root root 20 Oct 12 18:00 test
-rw-r----- 1 root root 13631488 Oct 14 15:32 undo_001
-rw-r----- 1 root root 13631488 Oct 14 15:32 undo_002
-rw-r----- 1 root root 109 Oct 12 18:01 xtrabackup_binlog_info
-rw-r----- 1 root root 101 Oct 14 15:32 xtrabackup_checkpoints
-rw-r----- 1 root root 622 Oct 12 18:01 xtrabackup_info
-rw-r----- 1 root root 8388608 Oct 14 15:32 xtrabackup_logfile
-rw-r--r-- 1 root root 1 Oct 14 15:32 xtrabackup_master_key_id # 多了
xtrabackup_master_key_id文件
-rw-r----- 1 root root 262 Oct 14 15:32 xtrabackup_tablespaces
```

5.将恢复完成的备份数据文件拷贝（或移动）到目标数据库Server的数据目录下：

```
[root@physical-machine ~]# xtrabackup --defaults-file=/etc/my.cnf --copy-back --
target-dir=/data/backup/xtrabackup_dir/backup/
......
191014 16:26:10 [01] Creating directory ./#innodb_temp
191014 16:26:10 [01] ...done.
191014 16:26:10 completed OK!
```

6.修改目标数据库的数据目录权限，并配置好my.cnf：

修改目录权限

```
[root@node4 backup]# chown mysql.mysql /data/mysqldata1/ -R
```

在my.cnf中配置好组复制的系统变量（如果是恢复故障Server而不是新增Server，这些配置都在，无需重复配置）

```
[root@node4 backup]# cat /etc/my.cnf
......
disabled_storage_engines="MyISAM,BLACKHOLE,FEDERATED,ARCHIVE,MEMORY"
binlog_checksum=NONE
plugin_load_add='group_replication.so'
group_replication_group_name="aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa"
group_replication_start_on_boot=off
group_replication_local_address= "10.10.30.16:33061"
group_replication_group_seeds=
"10.10.30.162:33061,10.10.30.163:33061,10.10.30.164:33061"
group_replication_bootstrap_group=off
report_host=node4
```

配置好hosts解析记录，或者将report_host指定为IP地址也可以不用配置解析记录（将node4的解析记录同时配置到其他组成员的/etc/hosts文件中）（如果是恢复故障Server而不是新增Server，这些配置都在，无需重复配置）

```
[root@physical-machine ~]# cat /etc/hosts
......
10.10.30.16 node4
10.10.30.162 node1
10.10.30.163 node2
10.10.30.164 node3
```

如果有需要，且允许，请将auto.cnf与mysqld-auto.cnf文件拷贝到目标数据库Server的datadir目录下（故障Server恢复才需要此步骤，新加入组的Server不需要此步骤）。

操作命令类似如下

```
[root@node2 ~]# cp -ar /data/backup/{auto.cnf,mysqld-auto.cnf}
/data/mysqldata1/mydata/
```

7.启动目标数据库Server进程（node4）：

```
[root@physical-machine backup]# service mysqld start
Starting MySQL......... SUCCESS!
[root@node4 backup]# ps aux |grep mysqld |grep -v grep
root 27818 0.7 0.0 11760 1640 pts/3 S 17:23 0:00 /bin/sh
/usr/local/mysql/bin/mysqld_safe --datadir=/home/mysql/data/mysqldata1/mydata --
pid-file=/home/mysql/data/mysqldata1/sock/mysql.pid
mysql 29388 102 3.4 107051036 8436648 pts/3 Sl 17:23 0:16
/usr/local/mysql/bin/mysqld --basedir=/usr/local/mysql --
datadir=/home/mysql/data/mysqldata1/mydata --plugin-
dir=/usr/local/mysql/lib/plugin --user=mysql --log-
error=/home/mysql/data/mysqldata1/log/error.log --open-files-limit=65535 --pid-
file=/home/mysql/data/mysqldata1/sock/mysql.pid --
socket=/home/mysql/data/mysqldata1/sock/mysql.sock --port=3306
```

8.查看备份目录下的xtrabackup_binlog_info文件中的GTID信息：

```
[root@physical-machine ~]# cd /data/backup/xtrabackup_dir/backup/
\# 记录下320675e6-de7b-11e9-b3a9-5254002a54f2:1-4,aaaaaaaa-aaaa-aaaa-aaaa-
aaaaaaaaaaaa:1-2771494字符串，稍后登录数据库中需要使用
[root@physical-machine backup]# cat xtrabackup_binlog_info
mysql-bin.000023 231 320675e6-de7b-11e9-b3a9-5254002a54f2:1-4,aaaaaaaa-aaaa-
aaaa-aaaa-aaaaaaaaaaaa:1-2771494
```

9.登录到node4的数据库Server中，重设GTID SET信息，如下：

先查看一下当前的GTID SET信息，如果与xtrabackup_binlog_info文件中记录的一致，就不需要后续步骤了（在一个在线的集群中，这里GTID经常会不一致，这里我们的示例中GTID SET碰巧一致是因为在执行备份期间，集群并没有新的请求进入）

```
admin@localhost : (none):37: > show master status;
+------------------+----------+--------------+------------------+----------------
-------------------------------------------------------------------------+
| File | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
+------------------+----------+--------------+------------------+----------------
-------------------------------------------------------------------------+
| mysql-bin.000024 | 231 | | | 320675e6-de7b-11e9-b3a9-5254002a54f2:1-4,
aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa:1-2771494 |
+------------------+----------+--------------+------------------+----------------
-------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

如果发现GTID SET信息与xtrabackup_binlog_info文件中记录的不一样，则需要执行后续步骤，重设GTID SET为xtrabackup_binlog_info文件中记录的GTID SET:

```
admin@localhost : (none):37: > reset master;
Query OK, 0 rows affected (0.01 sec)
admin@localhost : (none):38: > set global gtid_purged='320675e6-de7b-11e9-b3a9-
5254002a54f2:1-4,aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa:1-2771494';
Query OK, 0 rows affected (0.00 sec)
admin@localhost : (none):38: > show master status;
+-----------------+----------+--------------+------------------+---------------
-------------------------------------------------------------------------+
| File | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
+-----------------+----------+--------------+------------------+---------------
-------------------------------------------------------------------------+
| mysql-bin.000001 | 151 | | | 320675e6-de7b-11e9-b3a9-5254002a54f2:1-4,
aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa:1-2771494 |
+-----------------+----------+--------------+------------------+---------------
-------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

10.登录到node4的数据库Server中，启动组复制：

使用如下语句使node4加入集群

```
admin@localhost : (none):38: > START GROUP_REPLICATION;
Query OK, 0 rows affected (3.77 sec)
```

稍后查看成员状态信息，可以看到node4节点已经处于ONLINE状态了（如果在node4操作过程中，组持续不断有新的事务写入，那么，在其申请加入组的过程中，会有一部分事务需要追赶，所以node4可能有一段时间处于RECOVERING状态，等待追赶事务完成之后，才会变更ONLINE状态）

```
admin@localhost : (none):02: > select * from
performance_schema.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 |
3306 | ONLINE | SECONDARY | 8.0.17 |
| group_replication_applier | 2e33b2a7-de7b-11e9-9a21-525400bdd1f2 | node3 |
3306 | ONLINE | SECONDARY | 8.0.17 |
| group_replication_applier | 320675e6-de7b-11e9-b3a9-5254002a54f2 | node1 |
3306 | ONLINE | PRIMARY | 8.0.17 |
| group_replication_applier | 54b857f4-ee64-11e9-ada9-0025905b06da | node4 |
3306 | ONLINE | SECONDARY | 8.0.17 |
+---------------------------+--------------------------------------+-------------
+-------------+--------------+-------------+----------------+
4 rows in set (0.01 sec)
```

将系统变量group_replication_start_on_boot=ON加入到my.cnf中，以避免node4重启之后无法自动加入组。

注意：如果集群中的主要节点发生故障，在应用程序端如果未做主要节点的切换操作（如果在主要节点发生故障之后，组内新选举出了主要节点，且应用程序已经切换到了新的主要节点，则无此问题。但凡是总有例外，风险意识必须有），那么，在使用备份恢复主要节点时，需要将如下一些参数在数据库Server启动之前配置到my.cnf配置文件中，以防止在恢复主要节点的过程中，应用程序写入新的数据，

造成数据与组中的数据不一致而导致加入集群失败。

```
[root@node4 ~]# cat my.cnf
......
\# 不自动启动MGR插件
group_replication_start_on_boot=OFF
\# 在启动故障主要节点的数据库进程之前，设置只读，防止在执行分布式恢复过程中，应用程序写入新的数据到该节点中，造成与组中的数据不一致而导致加入组失败
super_read_only=ON
\# 如果有使用事件调度器做一些数据操作，也需要在启动之前关闭，以防止造成数据不一致而加入组失败
event_scheduler=OFF

当发生故障的主要节点重新成功加入集群之后，再将my.cnf配置文件中MGR插件和事件调度器的系统变量修改为ON，删除只读操作参数，如下：

\# 动态启用事件调度器
mysql> SET global event_scheduler=ON;
\# 在my.cnf配置文件中启用MGR插件和事件调度器，删除只读操作参数（注意，这里不是将只读参数设置为OFF，组复制会在组中根据单主和多主模式自动调整只读参数的设置，前面添加该参数到配置文件中的目的是为了避免发生意外写入，一旦加入组之后，就不需要人为干预了）
group_replication_start_on_boot=ON
event_scheduler=ON
```

PS：

xtrabackup 8.0版本支持备份时不加全局读锁（不执行FLUSH TABLE WITH READ LOCK语句），这就避免了在组复制中启用多线程回放的组成员上执行备份时造成锁死的现象，但是，为了保证一致性，是通过执行FLUSH NO_WRITE_TO_BINLOG BINARY LOGS语句来切换二进制日志，并拷贝最后一个二进制日志，在执行恢复时使用最后一个二进制日志来恢复一致性位置的方式来变相实现的。如果有大事务，则存在无法切换二进制日志的风险（执行切换二进制日志的语句可能会被长时间阻塞），具体情况请大家自行斟酌，做好充足的测试。

在xtrabackup 8.0版本废弃了innobackupex命令，统一使用xtrabackup命令，这样一来，就可以使用xtrabackup命令的--lock-ddl和--lock-ddl-per-table选项在备份过程中阻塞DDL但并不阻塞DML操作。从而很好地防止了在备份期间发生DDL修改表结构导致备份失败的问题。

在xtrabackup 8.0版本中，当发现有非InnoDB或RocksDB引擎时，Oracle MySQL版本则仍然会加全局读锁（由于Percona Server支持备份锁，因此在这种情况下，Percona Server并不是加全局读锁，而是使用LOCK TABLES FOR BACKUP语句加备份锁），当发现不存在非InnoDB或RocksDB引擎时，则如果无需考虑在备份期间存在DDL操作的情况，无需使用--lock-ddl和--lock-ddl-per-table选项，且备份过程中不会对数据库执行加锁操作。