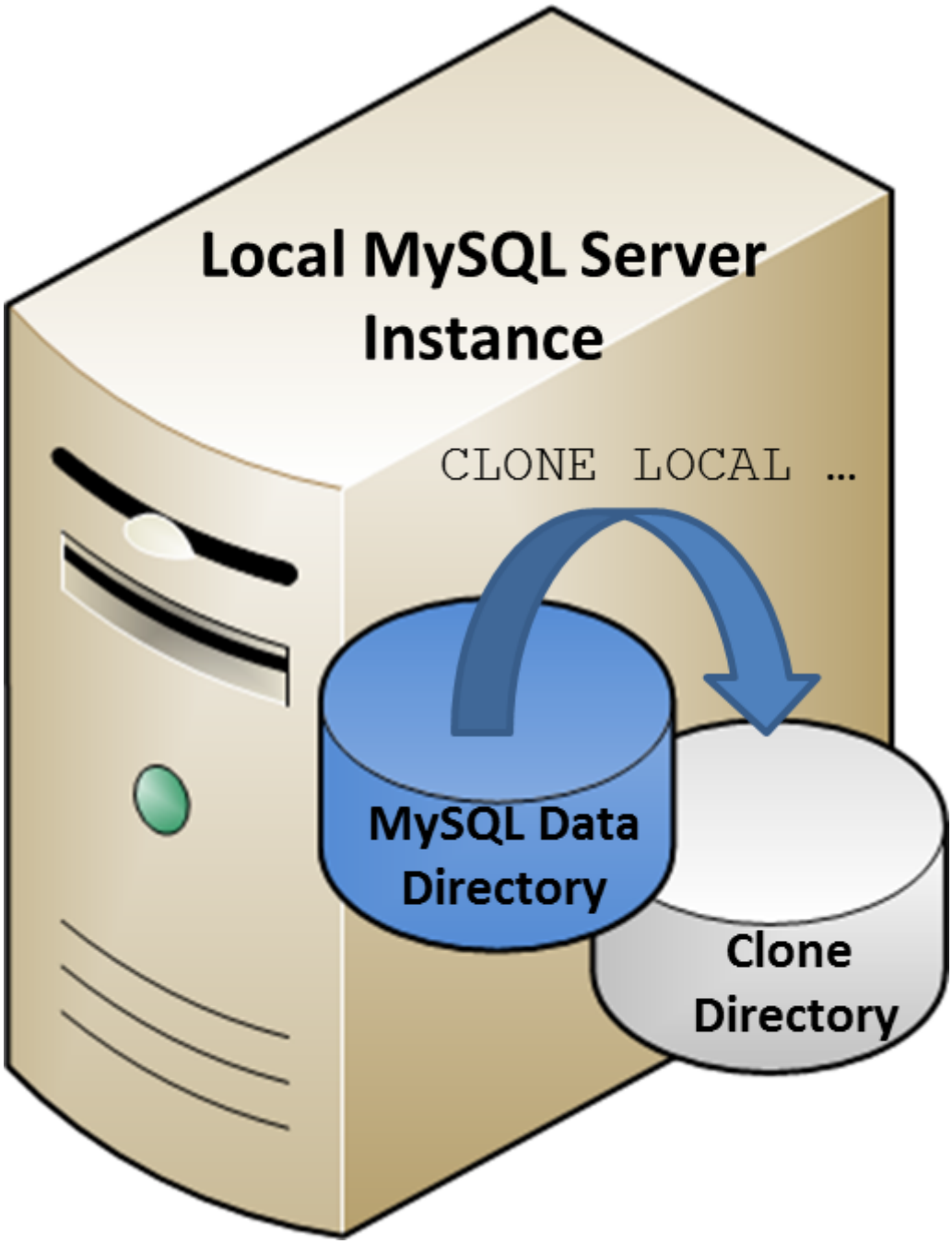
第二十一节：MySQL8.0 clone plugin---实战篇

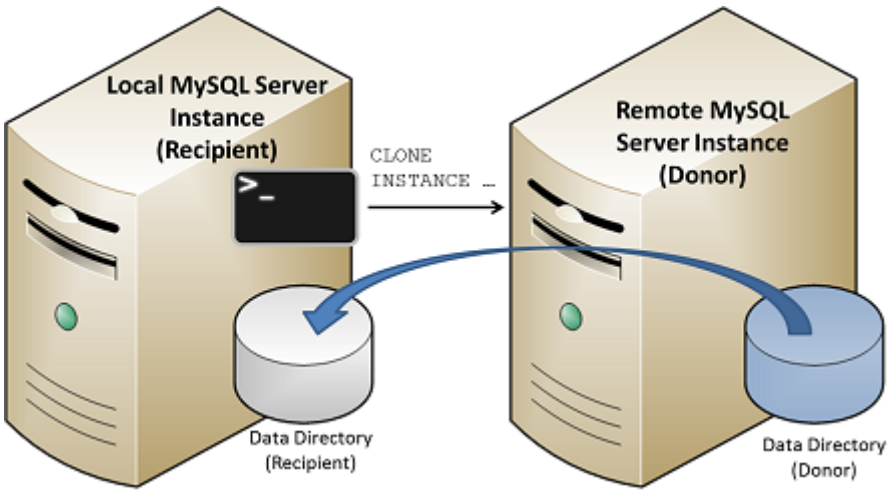
克隆插件允许从本地或远程的MySQL Server中克隆数据。克隆的数据是存储在InnoDB中的schema（database）、table（表）、tablespaces（表空间）和data dictionary metadata（数据字典元数据）的物理快照。该物理快照实际上是一个功能完整的数据目录，MySQL克隆插件可以使用该数据目录来配置并恢复一个MySQL Server。

本地克隆：指的是将数据从启动克隆操作的MySQL Server克隆到该MySQL Server的主机上的一个指定目录下。下图表示本地克隆操作的示意图：



远程克隆：涉及到启动克隆操作的本地MySQL Server（称为"recipient"，即，数据的接收者或接收方）和数据源所在的远程MySQL Server（称为"donor"，即，数据的提供者或发送方），在接收方上启动远程克隆操作时，克隆的数据会通过网络从发送方传输到接收方。默认情况下，远程克隆操作会删除接收方数据目录中的所有数据，并将其替换为克隆的新数据。如果不希望接收方中的现有数据被删除，也可以在接收方中执行克隆操作时将克隆数据指定存放在其他目录中。

下图表示远程克隆操作的示意图：



- 对于克隆的数据本身来说，本地克隆操作与远程克隆操作没有太大区别。
- 克隆插件支持在复制拓扑中使用。除了克隆数据外，克隆操作还能够从发送方中提取和传输复制坐标（二进制日志的位置），并将其应用于接收方，也就是说，我们可以使用克隆插件来在组复制中添加新的组成员，也可以在主从复制拓扑中添加新的从库。与通过二进制日志来复制大量事务相比，通过克隆插件要快得多，效率也更高。组复制成员还可以配置使用克隆插件来作为另一种恢复方法（如果不使用克隆插件，则必须使用基于二进制日志的状态传输进行数据恢复），当组成员和待加入组的Server都配置支持克隆插件时，待加入组的Server可以自行决定选择一个更加高效的方式从种子节点中获取数据。
    - 克隆插件支持克隆数据加密的和数据页压缩。
    - 要使用克隆功能，必须先安装克隆插件。
    - performance_schema中提供了用于监控克隆操作的一些性能事件采集器。
    - PS：在组复制拓扑中使用远程克隆操作时，为便于与非组复制拓扑做区分，我们这里也可以将"recipient"（即，接收方）称为"joiner"（即，加入方，表示将要加入组复制拓扑的MySQL Server）

1、安装克隆插件

- 本节介绍如何安装和配置克隆插件。对于远程克隆操作，克隆插件在发送方和接收方的MySQL Server上都必须安装
- 要使插件可被Server使用，插件库文件必须位于MySQL Server程序目录下的plugin目录中（由系统变量plugin_dir指定的目录）。如果需要修改默认值，则需要在MySQL Server启动时通过系统变量plugin_dir进行指定新的位置
- 插件库的基本名是mysql_clone.so。文件名后缀因平台而异（例如，对于Unix和类Unix系统下库文件名后缀为.so，Windows系统下库文件名后缀为.dll）。
- 要在MySQL Server启动时加载插件，可以使用--plugin-load-add选项来指定需要加载的库文件名。使用这种插件加载方法，每次MySQL Server启动之前都必须设置好该选项。例如，将其写入my.cnf配置文件中（根据平台的不同调整库文件名后缀为.so或者.dll）：

```
[mysqld]
plugin-load-add=mysql_clone.so
```

- 编辑好my.cnf之后，重新启动MySQL Server以使新设置生效。

- ps：在从旧版本升级到新版本过程中的重启操作，不能使用--plugin-load-add选项来加载克隆插件。例如，在将MySQL 5.7升级到MySQL 8.0的过程中，仅仅替换完成二进制程序文件之后，但还未完成其他升级步骤之前，尝试使用plugin-load-add=mysql_clone.so 重新启动MySQL Server。
- 会导致报错：
- ```
  [ERROR] [MY-013238] [Server] Error installing plugin 'clone': Cannot install during upgrade。
  ```
- 在尝试使用--plugin-load-add=mysql_clone .so选项启动MySQL Server之前完成所有的升级操作可以避免该报错。
- 或者，后续在需要时动态加载插件，可以使用以下语句（根据需要调整.so后缀）：
- ```
  mysql> INSTALL PLUGIN clone SONAME 'mysql_clone.so';
  ```
- INSTALL PLUGIN语句可以加载插件，并将其注册到mysql系统库下的mysql.plugins表中，这样在后续重启MySQL Server时不需要重复使用--plugin-load-add选项来加载插件库。
- 要验证插件是否安装成功，可以查看INFORMATION_SCHEMA.plugins表或者使用SHOW PLUGINS语句查看。例如:
- ```
  mysql> SELECT PLUGIN_NAME, PLUGIN_STATUS FROM
  INFORMATION_SCHEMA.PLUGINS WHERE PLUGIN_NAME = 'clone';
  +-----------------------+---------------+
  | PLUGIN_NAME | PLUGIN_STATUS |
  +-----------------------+---------------+
  | clone | ACTIVE |
  +-----------------------+---------------+
  ```
- - 如果插件初始化失败，请检查MySQL错误日志以获取克隆或插件相关的诊断消息。
  - 如果插件之前已经通过INSTALL PLUGIIN语句或者--plugin-load-add选项成功注册过了，则可以在MySQL Server启动时使用--clone选项来控制克隆插件的激活状态。例如，要在启动时加载插件并防止它在运行时被删除，可以使用以下选项:

    ```
    [mysqld]
    plugin-load-add=mysql_clone.so
    clone=FORCE_PLUS_PERMANENT
    ```

    - 如果想要阻止MySQL Server在没有克隆插件的情况下运行，那么在插件初始化失败时，可以使用--clone选项设置FORCE或FORCE_PLUS_PERMANENT值强制MySQL Server启动失败。

2、克隆本地数据

- 克隆插件支持用于在本地克隆数据的语法，即，将数据从本地（相同主机）的一个MySQL Server的数据目录克隆到本地MySQL Server所在主机的一个指定目录下，使用克隆插件执行克隆本地数据的操作语法如下：

  ```
  mysql> CLONE LOCAL DATA DIRECTORY [=] 'clone_dir';
  ```
  ```

要正确使用CLONE语法，必须先安装克隆插件。执行CLONE LOCAL DATA DIRECTORY语句需要用户具有BACKUP_ADMIN权限，因此需要先授予操作用户该权限，语句如下：

```
  #  其中，clone_user是用于执行克隆操作的MySQL用户。该用户可以是在"*.*"上（全局权限）具有BACKUP_ADMIN权限的任何MySQL用户。
  mysql> GRANT BACKUP_ADMIN ON *.* TO 'clone_user';
```

克隆本地数据示例：

```
  # 伪语句
  mysql> CLONE LOCAL DATA DIRECTORY = '/path/to/clone_dir';

  # 示例
  ## 执行克隆操作（这里是使用的root账号，具有所有权限，所以不用单独授予用户BACKUP_ADMIN权限）
  root@localhost : (none):31: > CLONE LOCAL DATA DIRECTORY = '/data/mysqldata1/mydata_clone';
  Query OK, 0 rows affected (6.21 sec)

  ## 查看克隆目录下的文件
  root@localhost : (none):34: > system ls -lh /data/mysqldata1/mydata_clone;  # 这是克隆的副本目录
  total 6.1G
  drwxr-x--- 2 mysql mysql 89 Feb 6 15:33 #clone
  -rw-r----- 1 mysql mysql 2.0G Feb 6 15:33 ibdata1
  -rw-r----- 1 mysql mysql 2.0G Feb 6 15:33 ib_logfile0
  -rw-r----- 1 mysql mysql 2.0G Feb 6 15:33 ib_logfile1
  drwxr-x--- 2 mysql mysql 6 Feb 6 15:33 mysql
  -rw-r----- 1 mysql mysql 24M Feb 6 15:33 mysql.ibd
  drwxr-x--- 2 mysql mysql 28 Feb 6 15:33 sys
  -rw-r----- 1 mysql mysql 10M Feb 6 15:33 undo_001
  -rw-r----- 1 mysql mysql 10M Feb 6 15:33 undo_002
  root@localhost : (none):34: > system ls -lh /data/mysqldata1/mydata;  # 这是原始目录
  total 25M
  -rw-r----- 1 mysql mysql 56 Feb 6 15:27 auto.cnf
  -rw------- 1 mysql mysql 1.7K Feb 6 15:27 ca-key.pem
  -rw-r--r-- 1 mysql mysql 1.1K Feb 6 15:27 ca.pem
  -rw-r--r-- 1 mysql mysql 1.1K Feb 6 15:27 client-cert.pem
  -rw------- 1 mysql mysql 1.7K Feb 6 15:27 client-key.pem
  drwxr-x--- 3 mysql mysql 22 Feb 6 15:33 #ib_archive
  drwxr-x--- 2 mysql mysql 187 Feb 6 15:27 #innodb_temp
  drwxr-x--- 2 mysql mysql 143 Feb 6 15:27 mysql
  -rw-r----- 1 mysql mysql 24M Feb 6 15:30 mysql.ibd
  drwxr-x--- 2 mysql mysql 8.0K Feb 6 15:30 performance_schema
  -rw------- 1 mysql mysql 1.7K Feb 6 15:27 private_key.pem
  -rw-r--r-- 1 mysql mysql 452 Feb 6 15:27 public_key.pem
  -rw-r--r-- 1 mysql mysql 1.1K Feb 6 15:27 server-cert.pem
  -rw------- 1 mysql mysql 1.7K Feb 6 15:27 server-key.pem
  drwxr-x--- 2 mysql mysql 28 Feb 6 15:27 sys
```

- 在上述操作中，"/path/to/clone_dir"是将数据克隆到本地目录的绝对路径。该路径中，"clone_dir"目录不能事先存在（事先存在会报错），但路径前缀"/path/to/"必须事先存在。另外，MySQL Server必须具有在文件系统中创建目录所需的写权限。
-
  - 注意：本地克隆操作不支持克隆位于数据目录外部的用户创建的表或表空间。尝试克隆此类表或表空间会导致报错：

    ```
      ERROR 1086 (HY000): File
    '/path/to/tablespace_name.ibd' already exists.。
    ```

  - 克隆操作时如果指定了一个与数据源表空间相同路径时会导致冲突，因此被禁止。
- 当执行克隆操作时，所有用户创建的InnoDB表和表空间，InnoDB系统表空间，redo log和undo log表空间都将被克隆到指定目录下（注意：克隆操作只会克隆数据文件，除了系统变量datadir之外，如果系统变量innodb_data_home_dir、innodb_data_file_path、innodb_log_group_home_dir、innodb_undo_directory单独指定了不同于datadir指定的路径，则也会被执行克隆，系统变量socket、pid-file、tmpdir、log-error、slow_query_log_file、log-bin、relay-log指定路径下的文件不会被克隆）
- 如果需要，可以在克隆操作完成后使用克隆的数据目录启动一个新的MySQL Server，例如：

  ```
    # 其中clone_dir是克隆操作完成之后的数据副本目录
    shell> mysqld_safe --datadir=clone_dir

    # 示例
    ## 先将第二个MySQL Server的配置文件设置好，不能与同一个主机中其他MySQL Server的配置文件存在路径冲突，也不能存在端口冲突
    ## 然后，使用mysqld_safe启动第二个MySQL Server，使用--datadir指定克隆的数据副本目录
    [root@physical-machine ~]# mysqld_safe --defaults-file=/etc/my.cnf_clone --datadir=/data/mysqldata1/mydata_clone &
    ......

    ## 尝试登陆第二个MySQL Server
    [root@physical-machine ~]# mysql -S /tmp/mysql.sock
    ......
    root@localhost : (none):18: > select @@datadir;
    +--------------------------------+
    | @@datadir |
    +--------------------------------+
    | /data/mysqldata1/mydata_clone/ |
    +--------------------------------+
    1 ro in set (0.00 sec)
  ```

  - PS：克隆数据副本目录下，有一个"#clone"目录，其下有4个文本文件，2个为空，2个有一些元数据，如下：

    ```
      [root@physical-machine ~]# ll /data/mysqldata1/mydata_clone/#clone/
      total 8
    ```

```
  -rw-r----- 1 mysql mysql 0 Feb 6 18:31
#replace_files
  -rw-r----- 1 mysql mysql 0 Feb 6 18:31
#status_fix
  -rw-r----- 1 mysql mysql 306 Feb 6 18:31
#view_progress
  -rw-r----- 1 mysql mysql 60 Feb 6 18:31
#view_status
  [root@physical-machine ~]# cd
/data/mysqldata1/mydata_clone/#clone/
  [root@physical-machine #clone]# ll
  total 8
  -rw-r----- 1 mysql mysql 0 Feb 6 18:31
#replace_files
  -rw-r----- 1 mysql mysql 0 Feb 6 18:31
#status_fix
  -rw-r----- 1 mysql mysql 306 Feb 6 18:31
#view_progress
  -rw-r----- 1 mysql mysql 60 Feb 6 18:31
#view_status
  [root@physical-machine #clone]# cat
\#replace_files
  [root@physical-machine #clone]# cat \#status_fix
  [root@physical-machine #clone]# cat
\#view_progress
  1
  2 1 1580985014237404 1580985014248529 0 0 0
  2 4 1580985014248632 1580985052940633 25115607040
25115607040 0
  2 4 1580985052940818 1580985057728398 1515782144
1515782144 0
  2 4 1580985057728613 1580985059941926 1505075200
1505075200 0
  2 4 1580985059942216 1580985066342384 0 0 0
  0 0 0 0 0 0 0
  0 0 0 0 0 0 0
  [root@physical-machine #clone]# cat \#view_status
  2 1
  1580985014237372 1580985066342589
  LOCAL INSTANCE
  0
  0
```

3、克隆远程数据

- 克隆插件支持以下语法来克隆远程数据，即，从远程MySQL Server（数据捐赠者，或称为donor节点）克隆数据并将其传输到执行克隆操作的MySQL Server（数据接收者，或称为recipient节点）

```
  CLONE INSTANCE FROM 'user'@'host':port IDENTIFIED BY
'password' [DATA DIRECTORY [=] 'clone_dir'] [REQUIRE [NO] SSL];
```

- 以上语法中的一些关键字解释：

        - "user"是donor MySQL Server上的用于执行克隆操作的用户，需要具有对所有库所有表的BACKUP_ADMIN权限
            - "host"是donor MySQL Server的主机名或IP地址。不支持IPV6地址，但支持IPV6地址别名与IPV4地址
            - "port"是donor MySQL Server的端口号。(不支持mysqlx_port指定的X协议端口。也不支持通过MySQL Router连接到donor MySQL Server)
            - "password"是"user"的用户密码
            - "DATA DIRECTORY [=] 'clone_dir'" 是一个可选子句，用于在recipient节点上为要克隆的数据副本指定一个本地存放目录。如果不希望删除recipient节点上数据目录中的现有数据，请使用此选项指定一个其他路径。但需要指定一个绝对路径，并且该目录不能事先存在、MySQL Server必须具有创建目录所需的写访问权限。如果在执行克隆操作时未使用可选的"DATA DIRECTORY [=] 'clone_dir'" 子句，则克隆操作将删除recipient节点数据目录中的现有数据，并用克隆数据副本来替换它，然后自动重新启动MySQL Server
            - "[REQUIRE [NO] SSL]" 用于显式指定在通过网络传输克隆数据时是否使用加密连接。如果使用了该子句但不能满足SSL使用条件，则返回一个错误。如果没有指定SSL子句，则克隆数据时默认会先尝试建立加密连接，但如果SSL连接尝试失败，则退回使用不安全连接。另外，无论是否指定此子句，如果要克隆加密数据，则必须使用安全连接。

    ps：

        - 默认情况下，驻留在发送方（donor节点）MySQL Server的数据目录中用户创建的InnoDB表和表空间，会被克隆到接收方（recipient节点）MySQL Server的数据目录中（接收方中与数据文件存放相关的系统变量指定的路径下）。如果指定了"DATA DIRECTORY [=] 'clone_dir'"子句，则在接收方中会将克隆数据存放到指定的目录下
            - 如果用户创建的InnoDB表和表空间位于发送方MySQL Server的数据目录之外，它们会被克隆到接收方MySQL Server的相同路径上。如果在接收方MySQL Server的相同路径上存在相同文件（表或表空间文件），则会报错。
            - 默认情况下，InnoDB的系统表空间、redo log和undo log表空间被克隆到与donor节点上的相关系统变量指定的相同位置（分别由系统变量innodb_data_home_dir和innodb_data_file_path、innodb_log_group_home_dir和innodb_undo_directory指定）。因此，如果未指定DATA DIRECTORY [=] 'clone_dir'子句，请确保donor节点和recipient节点中的相关系统变量设了为相同路径，如果指定了DATA DIRECTORY [=] 'clone_dir'子句，那么这些表空间和日志将被克隆到指定的目录下（但如果指定了克隆数据的存放目录，则所有的数据文件都会被存放到该目录下，因此，在使用这些集中存放的克隆数据文件来启动新的MySQL Server之前，你可能需要手动做一些路径调整）
    - 远程克隆的前提条件
    - 要执行远程克隆操作，克隆插件必须在发送方和接收方的MySQL Server上都是安装且都处于激活状态。
        - 执行远程克隆操作需要在发送方和接收方上都创建好用于克隆操作的MySQL用户（对于发送方和接收方上各自用户克隆的用户，其用户名和密码可以不相同），且授予足够的权限

    * 对于发送方，克隆用户需要BACKUP_ADMIN权限来访问和传输发送方的数据，并在克隆操作期间阻塞DDL操作

* 对于接收方，克隆用户需要CLONE_ADMIN权限来替换接收方的数据，并在克隆操作期间阻塞DDL操作，并自动重新启动MySQL Server。

注意：CLONE_ADMIN权限隐式地包含了BACKUP_ADMIN和SHUTDOWN权限

- 当执行远程克隆操作时（执行CLONE INSTANCE语句）会执行一些前提条件检查：

* 发送方和接收方必须拥有相同的MySQL Server版本。

注意：MySQL 8.0.17及更高版本支持克隆插件，低于MySQL 8.0.17版本不支持，可以使用：SELECT VERSION()语句查看版本号

* 发送方和接收方MySQL Server必须运行在相同的操作系统和平台上。例如，如果donor节点运行在一个Linux 64位平台上，那么recipient节点也必须运行在Linux 64位平台上。

* 接收方必须有足够的磁盘空间来存放克隆数据。默认情况下，在接收方中接收发送方的克隆数据之前会先删除接收方的数据，因此只需要按照发送方的数据大小来提供足够的磁盘空间即可。但如果使用了DATA DIRECTORY [=] 'clone_dir'子句将克隆数据存放到指定的目录下，则必须考虑接收方和发送方的数据总大小，以便提供足够存放两者数据大小的磁盘空间。可以通过检查文件系统上的数据目录大小和位于数据目录之外的任何表空间的大小来估计数据的大小。在估计发送方的数据大小时，请记住只有InnoDB数据是克隆的。如果在其他存储引擎中存储了数据，在估算克隆所需的磁盘空间时请注意排除这些数据文件的大小

* InnoDB允许在数据目录（datadir系统变量指定的目录）之外创建一些表空间类型。如果发送方MySQL Server有驻留在数据目录之外的表空间，则克隆操作必须能够访问这些表空间。可以通过查询INFORMATION_SCHEMA.FILES表来识别位于数据目录之外的数据表空间有哪些，驻留在数据目录之外的数据表空间文件是一个绝对路径。

查询语句：SELECT FILE_NAME FROM INFORMATION_SCHEMA.FILES

```
admin@localhost : (none):40: > SELECT FILE_NAME FROM INFORMATION_SCHEMA.FILES;
+--------------------------------------------+
| FILE_NAME                                  |
+--------------------------------------------+
| ./sbtest/sbtest1.ibd                       |
| ./sbtest/sbtest2.ibd                       |
| ./sbtest/sbtest3.ibd                       |
| ./sbtest/sbtest4.ibd                       |
| ./sbtest/sbtest5.ibd                       |
| ./sbtest/sbtest6.ibd                       |
| ./sbtest/sbtest7.ibd                       |
| ./sbtest/sbtest8.ibd                       |
| ./sys/sys_config.ibd                       |
| /home/mysql/data/mysqldata1/undo/undo_001  |
| /home/mysql/data/mysqldata1/undo/undo_002  |
| ./ibdata1                                  |
| ./ibtmp1                                   |
| ./mysql.ibd                                |
```

* 在发送方上处于激活状态的任何插件，也必须在接收方上处于激活状态。可以通过执行SHOW plugins语句或查询INFORMATION_SCHEMA.PLUGINS表来识别活跃状态的插件

* 发送方和接收方必须具有相同的MySQL Server字符集和排序规则。

* 发送方和接收方需要具有相同的innodb_page_size和innodb_data_file_path系统变量设置。在发送方和接收方上的innodb_data_file_path系统变量设置必须指定相同数量、相同大小的数据文件。可以使用SHOW VARIABLES语句检查各自的变量设置值。

例如：SHOW VARIABLES LIKE 'innodb_page_size';

SHOW VARIABLES LIKE 'innodb_data_file_path';

```
root@localhost : (none) 11:27:46> SHOW VARIABLES LIKE 'innodb_page_size';SHOW VARIABLES LIKE 'innodb_data_file_path';
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
| innodb_page_size | 16384 |
+------------------+-------+
1 row in set (0.01 sec)
```

```
+-----------------------+---------------------------+
| Variable_name         | Value                     |
+-----------------------+---------------------------+
| innodb_data_file_path | ibdata1:2048M:autoextend |
+-----------------------+---------------------------+
1 row in set (0.01 sec)
```

* 如果克隆加密数据或压缩页数据，则发送方和接收方必须具有相同的文件系统块大小。对于页压缩数据，接收方的文件系统必须支持稀疏文件和打孔（在接收方的文件系统上打孔。要确定文件系统块大小，请参阅操作系统相关的文档。）

* 如果要克隆加密数据，则需要启用安全连接。

* 接收方上的系统变量clone_valid_donor_list的设置必须包含donor MySQL Server的主机地址。因为只能从有效的接收方列表中的主机克隆数据。如果要设置该系统变量，则需要用户具有SYSTEM_VARIABLES_ADMIN权限。在本节后面的远程克隆示例中提供了设置clone_valid_donor_list系统变量的说明。可以使用SHOW VARIABLES语句检查clone_valid_donor_list系统变量的设置。例如：SHOW VARIABLES LIKE 'clone_valid_donor_list'

* 克隆操作只能串行执行，不能多个克隆操作并行执行，要确定是否有克隆操作正在运行，可以通过查询performance_schema.clone_status表进行确认。

* 克隆插件以1MB大小的数据包和1M大小的元数据的形式传输数据。因此，在发送方和接收方的MySQL Server上，max_allowed_packet变量的最小值要求为2MB。小于2MB的max_allowed_packet变量值会导致报错。可以使用查询语句来检查max_allowed_packet变量的设置：SHOW VARIABLES LIKE 'max_allowed_packet'

- 以下前提条件也适用于远程克隆操作：

* UNDO表空间文件名必须是唯一的。当数据被从发送方克隆到接收方时，无论UNDO表空间在发送方上的什么位置下存放着，都会被克隆到接收方上的innodb_undo_directory系统变量指定的位置，或者被克隆到DATA DIRECTORY [=] 'clone_dir'子句指定的目录下（如果使用该子句的话）。从MySQL 8.0.18开始，如果在克隆操作期间遇到重复的undo表空间文件名，就会报告错误。在MySQL 8.0.18之前，克隆具有相同文件名的undo表空间可能会导致接收方上的undo表空间文件被覆盖。

* 要查看UNDO表空间文件名以确保它们的名称是唯一的，可以通过查询INFORMATION_SCHEMA.FILES表，例如：SELECT TABLESPACE_NAME, FILE_NAME FROM INFORMATION_SCHEMA.FILES WHERE FILE_TYPE LIKE 'UNDO LOG';

```
admin@localhost : (none):39: > SELECT TABLESPACE_NAME, FILE_NAME FROM INFORMATION_SCHEMA.FILES  WHERE FILE_TYPE LIKE 'UNDO LOG';
+-----------------+-----------------------------------------+
| TABLESPACE_NAME | FILE_NAME                               |
+-----------------+-----------------------------------------+
| innodb_undo_001 | /home/mysql/data/mysqldata1/undo/undo_001 |
```

* 默认情况下，在克隆数据完成之后，将自动重新启动（停止和启动）接收方MySQL Server。要实现自动重启，必须在接收方上有一个监控进程来检测Server关闭。否则，在数据被克隆后、克隆操作停止、并关闭了接收方MySQL Server之后会报错：

```
ERROR 3707 (HY000): Restart server failed (mysqld is not managed by supervisor
process)。
```

此错误不表示克隆操作失败。这意味着在克隆数据之后，必须手动启动接收方MySQL Server。手动启动MySQL Server后，可以连接到接收方MySQL Server检查performance_schema下的clone_progress和clone_status表，以验证克隆操作是否成功完成。对于RESTART语句也会执行相同的监控。

如果使用DATA DIRECTORY [=] 'clone_dir'子句克隆到指定目录，则不适用此要求，因为在此情况下不会执行自动重新启动MySQL Server（指定克隆目录的情况下，所有数据文件都被拷贝到了该目录下，实际启动MySQL Server时，配置文件中可能将redo log、undo log和数据文件指向了不同的路径）

* 克隆插件支持多个系统变量用于控制远程克隆操作。在执行远程克隆操作之前，请检查系统变量并根据需要进行调整以适应你的运行环境。克隆相关的系统变量是在执行克隆操作的MySQL Server上设置的（接收方）。

克隆远程数据操作示例：默认情况下，远程克隆操作会删除接收方数据目录中的数据，用克隆的数据替换，然后重新启动MySQL Server（但指定了DATA DIRECTORY [=] 'clone_dir'子句时不会执行自动重启），该示例假设已经满足了所有的前提条件。

使用管理用户登录到donor MySQL Server中，创建一个克隆用户并赋予BACKUP_ADMIN权限

```
root@localhost : (none):08: > create user donor_clone_user@'%' identified by
'letsg0';
Query OK, 0 rows affected (0.02 sec)

root@localhost : (none):09: > grant backup_admin on *.* to donor_clone_user@'%';
Query OK, 0 rows affected (0.01 sec)

在donor MySQL Server中安装克隆插件:
root@localhost : (none):10: > INSTALL PLUGIN clone SONAME 'mysql_clone.so';
Query OK, 0 rows affected (0.01 sec)
```

使用管理用户登录到接收方MySQL Server，创建一个克隆用户并赋予

```
CLONE_ADMIN权限
root@localhost : (none):14: > create user recipient_clone_user@'%' identified by
'letsg0';
Query OK, 0 rows affected (0.03 sec)

root@localhost : (none):14: > grant clone_admin on *.* to
recipient_clone_user@'%';
Query OK, 0 rows affected (0.01 sec)
```

- 在接收方MySQL Server中安装克隆插件:

```
root@localhost : (none):14: > INSTALL PLUGIN clone SONAME 'mysql_clone.so';
Query OK, 0 rows affected (0.01 sec)
```

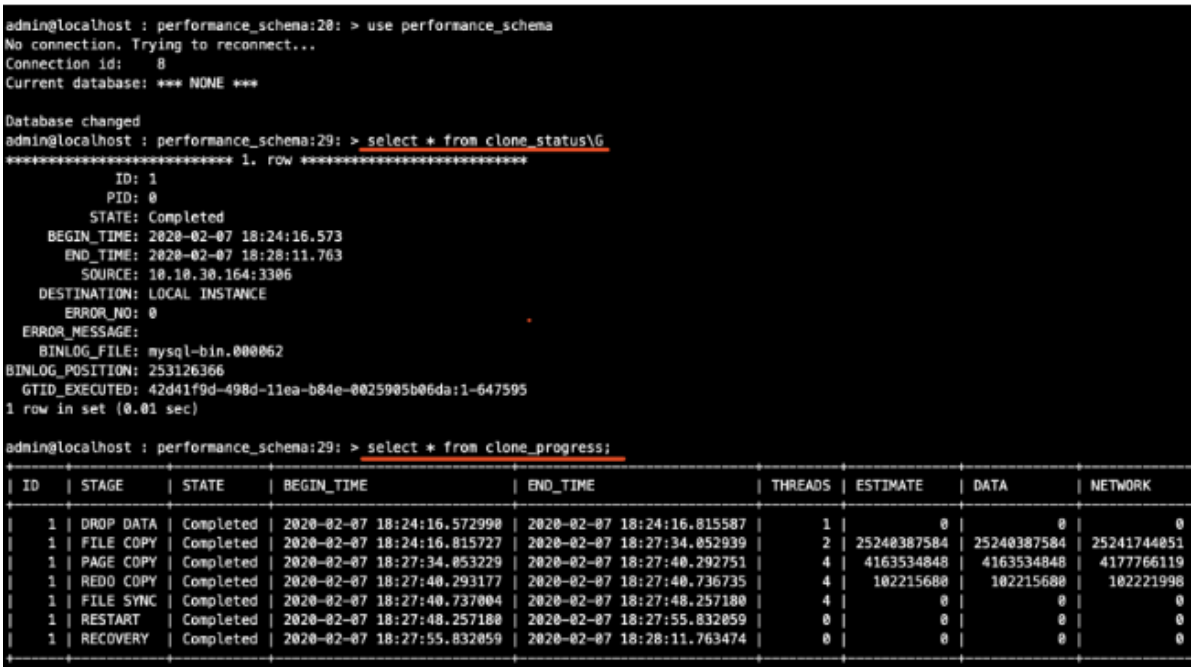
将donor MySQL Server的主机地址添加到接收方MySQL Server的clone_valid_donor_list变量中

```
root@localhost : (none):37: > SET GLOBAL clone_valid_donor_list =
'10.10.30.164:3306';
Query OK, 0 rows affected (0.00 sec)
```

以在donor MySQL Server中创建的克隆用户，登录到接收方MySQL Server中，执行如下克隆语句

```
mysql> CLONE INSTANCE FROM 'donor_clone_user'@'10.10.30.164':3306 IDENTIFIED BY
'letsg0' ;
```

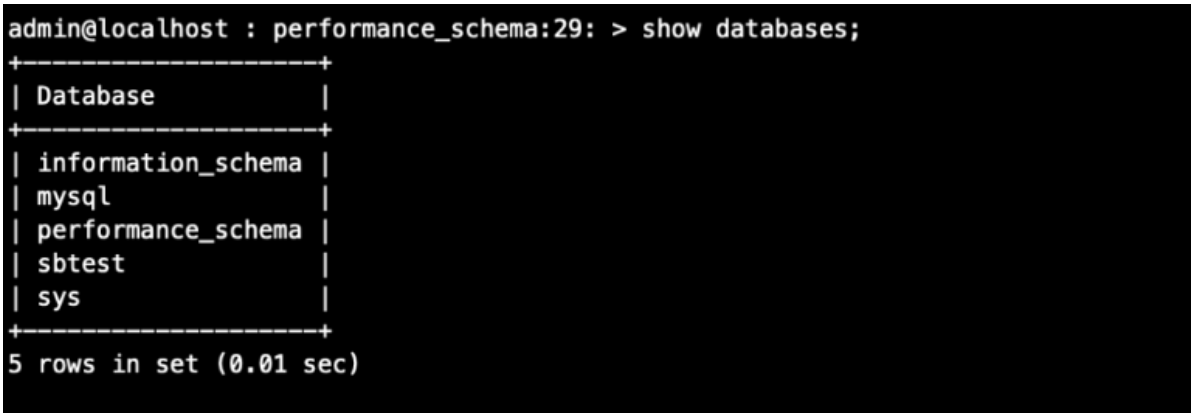
- 克隆数据完成后，将自动重新启动接收方的MySQL Server。重启之后可以登录到接收方的MySQL Server中查看performance_schema下的克隆状态信息表，可以查看到克隆操作的状态和进度的信息。
- - 简单查看克隆的进度和状态信息



```
admin@localhost : performance_schema:28: > use performance_schema
No connection. Trying to reconnect...
Connection id:    8
Current database: *** NONE ***

Database changed
admin@localhost : performance_schema:29: > select * from clone_status\G
*************************** 1. row ***************************
           ID: 1
          PID: 0
        STATE: Completed
   BEGIN_TIME: 2020-02-07 18:24:16.573
     END_TIME: 2020-02-07 18:28:11.763
       SOURCE: 10.10.30.164:3306
  DESTINATION: LOCAL INSTANCE
     ERROR_NO: 0
ERROR_MESSAGE:
   BINLOG_FILE: mysql-bin.000062
BINLOG_POSITION: 253126366
  GTID_EXECUTED: 42d41f9d-498d-11ea-b84e-0025905b06da:1-647595
1 row in set (0.01 sec)

admin@localhost : performance_schema:29: > select * from clone_progress;
+----+-----------+-----------+----------------------------+----------------------------+---------+-------------+-------------+-------------+
| ID | STAGE     | STATE     | BEGIN_TIME                 | END_TIME                   | THREADS | ESTIMATE    | DATA        | NETWORK     |
+----+-----------+-----------+----------------------------+----------------------------+---------+-------------+-------------+-------------+
|  1 | DROP DATA | Completed | 2020-02-07 18:24:16.572990 | 2020-02-07 18:24:16.815587 |       1 |           0 |           0 |           0 |
|  1 | FILE COPY | Completed | 2020-02-07 18:24:16.815727 | 2020-02-07 18:27:34.052939 |       2 | 25240387584 | 25240387584 | 25241744051 |
|  1 | PAGE COPY | Completed | 2020-02-07 18:27:34.053229 | 2020-02-07 18:27:40.292751 |       4 |  4163534848 |  4163534848 |  4177766119 |
|  1 | REDO COPY | Completed | 2020-02-07 18:27:40.293177 | 2020-02-07 18:27:40.736735 |       4 |   102215680 |   102215680 |   102221998 |
|  1 | FILE SYNC | Completed | 2020-02-07 18:27:40.737004 | 2020-02-07 18:27:48.257180 |       4 |           0 |           0 |           0 |
|  1 | RESTART   | Completed | 2020-02-07 18:27:48.257180 | 2020-02-07 18:27:55.832059 |       0 |           0 |           0 |           0 |
|  1 | RECOVERY  | Completed | 2020-02-07 18:27:55.832059 | 2020-02-07 18:28:11.763474 |       0 |           0 |           0 |           0 |
+----+-----------+-----------+----------------------------+----------------------------+---------+-------------+-------------+-------------+
```



```
admin@localhost : performance_schema:29: > show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sbtest             |
| sys                |
+--------------------+
5 rows in set (0.01 sec)
```

```
admin@localhost : performance_schema:32: > use sbtest
Database changed
admin@localhost : sbtest:32: > show tables;
+-----------------+
| Tables_in_sbtest |
+-----------------+
| sbtest1         |
| sbtest2         |
| sbtest3         |
| sbtest4         |
| sbtest5         |
| sbtest6         |
| sbtest7         |
| sbtest8         |
+-----------------+
8 rows in set (0.00 sec)
```

- 克隆到指定目录
- 默认情况下，远程克隆操作会删除接收方数据目录中的数据，并用克隆的数据替换它。通过克隆到指定目录，可以避免接收方数据目录中的现有数据被删除，也不会执行重启MySQL Server的操作。将数据克隆到指定目录的过程与不指定目录的克隆远程数据过程相同，但有一点区别，前者的克隆语句必须包含DATA DIRECTORY [=] 'clone_dir'子句。
- 例如：

```
CLONE INSTANCE FROM 'donor_clone_user'@'10.10.30.164':3306 IDENTIFIED BY
'letsg0' DATA DIRECTORY = '/data/mysqldata1/mydata_clone';
```

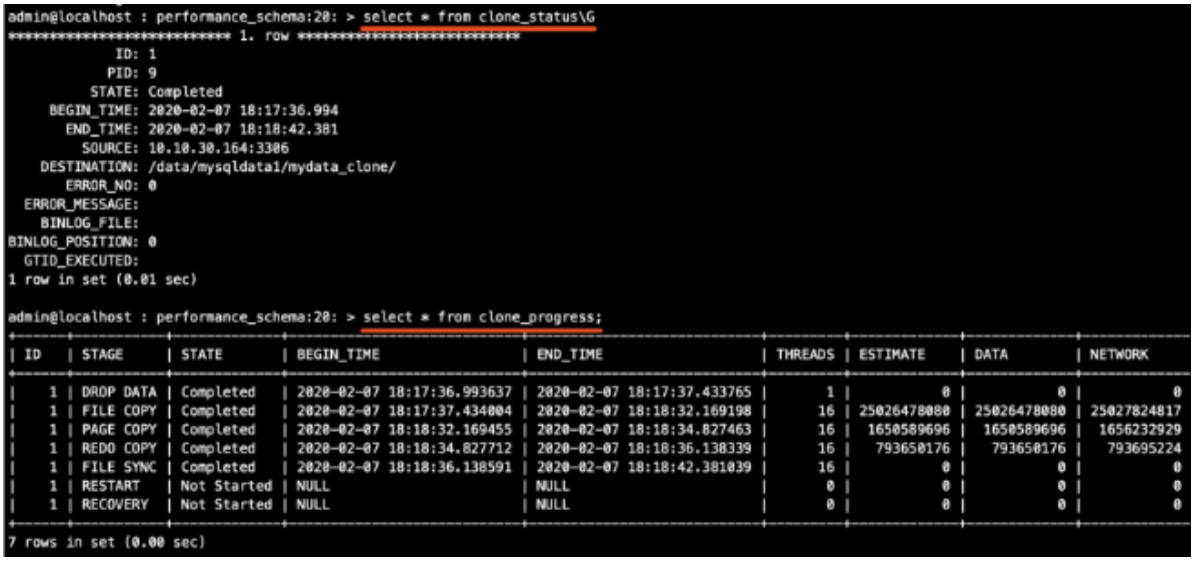
- 其中，DATA DIRECTORY子句需要指定一个绝对路径，且该目录不能事先存在，MySQL Server必须具有创建目录所需的写访问权限

```
admin@localhost : (none):08: >  CLONE INSTANCE FROM 'donor_clone_user'@'10.10.30.164':3306 IDENTIFIED BY 'letsg0' DATA DIRECTORY = '/data/mysqldata1/mydata_clone';
Query OK, 0 rows affected (1 min 5.39 sec)
```

克隆到指定目录时，在克隆数据完成之后，不会自动重新启动接收方MySQL Server。如果想使用指定目录启动MySQL Server，必须手动执行一些文件的目录调整之后再执行启动，或者直接使用新的my.cnf配置文件，在启动时将--datadir选项指定到克隆数据所在的目录，例如：

```
mysqld_safe --datadir=/path/to/clone_dir
```

指定了DATA DIRECTORY [=] 'clone_dir'子句的克隆状态和进度信息类似如下:



```
admin@localhost : performance_schema:20: > select * from clone_status\G
*************************** 1. row ***************************
           ID: 1
          PID: 9
        STATE: Completed
    BEGIN_TIME: 2020-02-07 18:17:36.994
      END_TIME: 2020-02-07 18:18:42.381
       SOURCE: 10.10.30.164:3306
  DESTINATION: /data/mysqldata1/mydata_clone/
     ERROR_NO: 0
ERROR_MESSAGE:
   BINLOG_FILE:
BINLOG_POSITION: 0
 GTID_EXECUTED:
1 row in set (0.01 sec)

admin@localhost : performance_schema:20: > select * from clone_progress;
+----+------------+-------------+----------------------------+----------------------------+---------+--------------+--------------+--------------+
| ID | STAGE      | STATE       | BEGIN_TIME                 | END_TIME                   | THREADS | ESTIMATE     | DATA         | NETWORK      |
+----+------------+-------------+----------------------------+----------------------------+---------+--------------+--------------+--------------+
|  1 | DROP DATA  | Completed   | 2020-02-07 18:17:36.993637 | 2020-02-07 18:17:37.433765 |       1 |            0 |            0 |            0 |
|  1 | FILE COPY  | Completed   | 2020-02-07 18:17:37.434004 | 2020-02-07 18:18:32.169198 |      16 | 25026478080 | 25026478080 | 25027824817 |
|  1 | PAGE COPY  | Completed   | 2020-02-07 18:18:32.169455 | 2020-02-07 18:18:34.827463 |      16 |   1650589696 |   1650589696 |   1656232929 |
|  1 | REDO COPY  | Completed   | 2020-02-07 18:18:34.827712 | 2020-02-07 18:18:36.138339 |      16 |    793650176 |    793650176 |    793695224 |
|  1 | FILE SYNC  | Completed   | 2020-02-07 18:18:36.138591 | 2020-02-07 18:18:42.381039 |      16 |            0 |            0 |            0 |
|  1 | RESTART    | Not Started | NULL                       | NULL                       |       0 |            0 |            0 |            0 |
|  1 | RECOVERY   | Not Started | NULL                       | NULL                       |       0 |            0 |            0 |            0 |
+----+------------+-------------+----------------------------+----------------------------+---------+--------------+--------------+--------------+
7 rows in set (0.00 sec)
```

- 为克隆配置加密连接
- - 可以为远程克隆操作配置加密连接，以保护通过网络克隆的数据安全。在克隆加密数据时，默认情况下需要使用加密连接。

- 下面的说明描述了如何配置接收方MySQL Server来使用加密连接。
- - PS：假设已将donor MySQL Server配置为使用加密连接。
- 配置接收方MySQL Server使用加密连接，使发送方MySQL Server的客户端证书和密钥文件适配接收方主机。可以使用安全通道将证书文件分发到接收方MySQL Server所在的主机。这些客户端证书和密钥文件包括如下一些（在8.0中，这些证书和秘钥文件默认情况下在datadir下会自动生成，即使被删除，重启MySQL Server时也会自动生成，因此，直接从发送方的MySQL Server的datadir下拷贝客户端证书和客户端秘钥文件到接收方MySQL Server所在的主机即可）：
- - ca.pem：自签名证书颁发机构(CA)文件
  - client-cert.pem：客户端公钥证书文件
  - client-key.pem：客户端私钥文件

- 在接收方MySQL Server上配置以下SSL选项，使用这些选项分别指定从发送方拷贝过来的客户端证书和秘钥文件所在的绝对路径即可
- - clone_ssl_ca：指定自签名证书颁发机构(CA)文件的路径
  - clone_ssl_cert：指定客户端公钥证书文件的路径
  - clone_ssl_key：指定客户端私有密钥文件的路径

- 示例

```
clone_ssl_ca=/path/to/ca.pem
clone_ssl_cert=/path/to/client-cert.pem
clone_ssl_key=/path/to/client-key.pem
```

若要求使用加密连接，需要在接收方中执行克隆语句时包含REQUIRE SSL子句，如下：

```
mysql> CLONE INSTANCE FROM 'user'@'example.donor.host.com':3306 IDENTIFIED BY
'password' DATA DIRECTORY = '/path/to/clone_dir' REQUIRE SSL;
```

- 如果没有指定REQUIRE SSL子句，克隆插件将默认先尝试建立加密连接，如果加密连接尝试失败，则回退到未加密连接。
- 请注意：如果要克隆加密的数据，则无论是否指定了REQUIRE SSL子句，默认情况下都必须使用加密连接。如果在试图克隆加密的数据时使用了REQUIRE NO SSL子句来强行跳过使用加密连接，会导致报错。

4、克隆加密数据

- 克隆插件支持克隆加密数据。适用下列规定:
- - 在克隆远程数据时，为了确保在网络上安全传输未加密的表空间密钥（表空间密钥指的是用于数据加密的密钥），需要一个安全连接，表空间密钥在发送方传输之前解密，然后，在接收方中使用接收方的主密钥重新加密。如果在CLONE INSTANCE语句执行时加密连接不可用或使用了REQUIRE NO SSL子句，则会报错。
- - 当将数据克隆到使用本地托管的keyring的本地数据目录时（克隆本地数据），则在使用克隆数据所在的目录启动MySQL Server时必须使用相同的keyring（keyring，即密钥环，密钥环是一个以加密方式存储登录信息的本地数据库。各种应用（如浏览器、电子邮件客户端）使用密钥环来安全地存储并管理登录凭证、密码、证书或密钥。对于那些需要检索存储在密钥环中的信息的应用程序，需要解锁该密钥环。）

    - 当将数据克隆到使用本地管理的keyring的远程数据目录（接收方目录，即克隆远程数据）时，则在使用克隆数据所在的目录启动MySQL Server时必须使用接收方的keyring
    - PS：注意，在进行克隆操作时，无法动态修改innodb_redo_log_encrypt和innodb_undo_log_encrypt变量设置

  - 有关数据加密特性的信息，请参阅：
  - - https://dev.mysql.com/doc/refman/8.0/en/innodb-data-encryption.html
    - https://dev.mysql.com/doc/refman/8.0/en/keyring.html

5、克隆压缩数据

- 克隆插件支持克隆压缩数据。克隆远程数据时需要满足以下要求:
- - 接收方文件系统必须支持稀疏文件和打孔，以便在接收方上进行打孔
  - 发送方和接收方文件系统必须具有相同的块大小。如果文件系统块大小不同，将报错，类似如下：

    ```
    ERROR 3868 (HY000): Clone Configuration FS Block Size: Donor value:
    114688 is different from Recipient value: 4096.
    ```

- 有关页压缩特性的信息，请参阅：https://dev.mysql.com/doc/refman/8.0/en/innodb-page-compression.html

6、在复制拓扑中使用克隆

- 克隆插件支持复制拓扑。除了克隆数据外，克隆操作还可以从发送方提取复制（二进制日志或GTID SET）坐标并将其传输给接收方，这就使得可以使用克隆插件来提供（搭建）组复制的成员和主从复制拓扑中的从库。与通过逻辑备份导出导入的方式来复制大量事务到从库或组复制成员相比，使用克隆插件来备份数据要快得多，效率也高得多（注：即使使用percona 的xtrabackup备份工具进行物理文件备份，也仍然不如克隆插件方便快捷，因为，使用克隆插件不需要像xtrabackup工具那样，备份之后还需要执行恢复等繁琐的手工操作，且，如果克隆数据的发送方存在负载的情况下，相比于使用xtrabackup备份来说，使用克隆插件备份的优势更加明显，另外，使用克隆插件在

从库中克隆数据时，如果从库负载较高且都是DML语句，则不会像xtrabackup那样非常容易发生锁死问题）。

- 组复制成员还可以配置使用克隆插件作为分布式恢复的选项，在这种情况下，joiner节点会自动选择最有效的方式从现有组成员中检索组数据。
- 在克隆操作期间，会提取二进制日志位置（文件名、偏移量）和从gtid_executed系统变量中提取GTID SET，并将它们从donor MySQL Server传输给接收方。此位置数据是在复制流中获取的一致位置，因此可以使用该一致位置来启动复制。由于二进制日志和中继日志保存在文件中，因此不会从发送方复制到接收方。要正常启动复制，在发送方上不能清除在数据克隆操作完成之后、复制启动之前这段时间间隔内新产生的二进制日志，这些二进制日志是接收方用于追赶发送方最新数据所必须的。如果所需的二进制日志不可用，则会报一个handshake错误。因此，要避免接收方的数据落后太多（克隆插件的数据快照是克隆结束时的数据，增量拷贝数据主要体现在FILE COPY、PAGE COPY、REDO COPY几个阶段。在FILE COPY期间，先通过Page Tracking机制追踪克隆操作开始时的CHECKPOINT LSN号来识别后续新增的脏页，当FILE COPY节点完成时，记录完成时的CHECKPOINT LSN，从内存中直接拷贝FILE COPY期间新增的数据页。同时，从FILE COPY结束时记录的CHECKPOINT LSN位置开始，使用Redo Archiving机制追踪在之后新产生的redo log，当PAGE COPY结束时，就停止redo log追踪，并记录停止PAGE COPY时的LSN，该LSN被称为clone lsn，即，克隆数据最终的最新数据对应的lsn，进入REDO COPY阶段将PAGE COPY阶段新产生的redo log进行归档，当REDO COPY阶段完成时，会用全新的加锁机制获取二进制日志位置和GTID SET，然后将和redo log归档一并发送给接收方，结束克隆操作。其效率比xtrabackup拷贝增量的方式高效得多）。
- 克隆操作结束之后，可在执行克隆操作的MySQL Server上执行如下查询语句，以检查发送给接收方的二进制日志位置：

```
root@localhost : (none):31: > SELECT BINLOG_FILE, BINLOG_POSITION FROM
performance_schema.clone_status;
+------------------+-----------------+
| BINLOG_FILE | BINLOG_POSITION |
+------------------+-----------------+
| mysql-bin.000062 | 253126366 |
+------------------+-----------------+
1 row in set (0.01 sec)
```

在克隆操作结束之后，可在执行克隆操作的MySQL Server上执行如下查询，以检查传输给接收方的GTID SET：

```
root@localhost : (none):49: > SELECT @@GLOBAL.GTID_EXECUTED;
+-------------------------------------------+
| @@GLOBAL.GTID_EXECUTED |
+-------------------------------------------+
| 42d41f9d-498d-11ea-b84e-0025905b06da:1-647595 |
+-------------------------------------------+
1 row in set (0.00 sec)
```

- 当在发送方上设置master_info_repository=TABLE和relay_log_info_repository=TABLE时（这是MySQL 8.0的默认设置），如果发送方是一个从库角色，那么从库的状态日志会被保存在表中，这些表在克隆操作期间会从发送方当做数据复制到接收方。从库的状态日志中保存了与复制相关的配置设置，这些设置可用于在克隆操作成功之后正确地自动恢复复制（如果配置文件中没有设置skip_slave_start参数，则在克隆操作完成之后会自动启动复制线程，另外，如果发送方不是从库而是主库，那么从库的状态日志表中并不存在复制配置信息，因此不适用，恢复复制过程中需要手动执行CHANGE MASTER语句进行配置）。
-
    - 在MySQL 8.0.17和8.0.18中，只有mysql.slave_master_info表才会被复制到接收方（主库信息日志）

- 从MySQL 8.0.19开始，mysql.slave_relay_log_info（中继日志信息日志）和mysql.slave_worker_info（从库worker线程日志）也会被复制到接收方
- PS：如果在发送方上设置了master_info_repository=FILE和relay_log_info_repository=FILE（这不是MySQL 8.0的默认设置，并且是不推荐的），则不会克隆从库的状态日志，只有在发送方设置master_info_repository=TABLE和relay_log_info_repository=TABLE时才会克隆从库的状态日志信息。

- 要在复制拓扑中使用克隆，请按照以下步骤执行：
-
  - 对于用于组复制的新组成员，请先按照链接：https://dev.mysql.com/doc/refman/8.0/en/group-replication-adding-instances.html 中的说明为组复制配置好MySQL Server环境，且按照链接：https://dev.mysql.com/doc/refman/8.0/en/group-replication-cloning.html的说明设置好克隆功能所需的前提条件。然后，在joiner节点上执行START GROUP_REPLICATION语句时，克隆操作由组复制自动管理，因此不需要手动执行joiner节点加入集群的操作，也不需要在joiner节点上执行任何进一步的设置步骤。

  - 对于主从复制拓扑中的从库，首先在从库（接收方）中手动执行远程克隆操作语句，将数据从donor MySQL Server克隆到接收方。在复制拓扑中，发送方必须是主库或从库。如果发送方是主库的，则后续需要手动执行CHANGE MASTER语句来配置复制，如果发送方是从库的，则不需要手动执行复制配置操作，复制能够通过克隆数据进行自动恢复复制（配置文件中指定了skip_slave_start参数的情况除外）。

  - 克隆操作成功完成后，如果要在接收方MySQL Server上使用与发送方相同的复制通道，请验证其中哪些设置或配置可以在主从复制拓扑中自动恢复，哪些需要手动设置才能恢复。

* 对于基于GTID的复制，如果在接收方配置了gtid_mode=ON，并且在发送方设置了gtid_mode=ON、ON_PERMISSIVE或 OFF_PERMISSIVE值，则接收方的gtid_executed系统变量中的GTID SET会作为接收方的GTID SET。如果接收方的数据是从拓扑中已经存在的一个从库中克隆出来的，则在启用了GTID自动定位（由CHANGE MASTER TO语句上的MASTER_AUTO_POSITION选项指定）的接收方上的复制通道可以在自动重启MySQL Server之后自动恢复，不需要执行任何手动设置（如果需要修改复制通道，则自行调整，这里不做赘述）。

* 对于基于二进制日志文件位置的复制，如果接收方使用的是MySQL 8.0.17或8.0.18，则来自发送方的二进制日志位置不应用于接收方，仅记录在performance_schema.clone_status表中。因此，在自动重启实例之后，必须在接收方中手动为复制通道设置使用基于二进制日志文件位置的复制，以便恢复复制。在这种情况下，需要确保在重启实例时启用skip_slave_start来避免复制自动启动，因为此时并未设置二进制日志位置，自动启动复制将尝试从头开始复制。

* 对于基于二进制日志文件位置的复制，如果接收方使用的是MySQL 8.0.19或以上版本，则来自发送方的二进制日志位置将应用于接收方。接收方上使用基于二进制日志文件位置的复制，将会在自动重启复制通道之前，使用中继日志信息自动尝试执行中继日志的恢复过程。注意：对于单线程的从库（slave_parallel_workers设置为0），在没有任何其他意外发生的情况下，复制通道无需进一步设置即可成功恢复复制。对于多线程的从库（slave_parallel_workers大于0），中继日志恢复可能会失败，因为它通常不能自动完成。在这种情况下，会发出一条错误消息，必须手动设置通道。

- 如果需要手动设置复制通道，或者希望在接收方上使用不同的复制通道，以下说明提供了将接收方MySQL Server添加到复制拓扑的简短示例：

* 若要将接收方MySQL Server添加到使用基于GTID的事务作为复制数据源的MySQL复制拓扑中，请按照链接：https://dev.mysql.com/doc/refman/8.0/en/replication-gtids-howto.html中的说明按需配置实例。为实例添加复制通道，如下面的简短示例所示。CHANGE MASTER TO语句必须定义主库的主机地址（IP）和端口号，并且应该启用MASTER_AUTO_POSITION选项，如下所示:

```
mysql> CHANGE MASTER TO MASTER_HOST = 'source_host_name', MASTER_PORT =
source_port_num,
       ...
       MASTER_AUTO_POSITION = 1,
       FOR CHANNEL 'setup_channel';
mysql> START SLAVE USER = 'user_name' PASSWORD = 'password' FOR CHANNEL
'setup_channel';
Or from MySQL 8.0.22:
mysql> START REPLICA USER = 'user_name' PASSWORD = 'password' FOR CHANNEL
'setup_channel';
```

* 若要将接收方MySQL Server添加到使用基于二进制日志文件位置复制的MySQL复制拓扑中，请按照链接：https://dev.mysql.com/doc/refman/8.0/en/replication-howto.html中的说明对实例进行必要的配置。使用在克隆操作期间传递给接收方的二进制日志位置（记录在performance_schema.clone_status表中），为实例添加复制通道，如下面的简短示例所示:

```
mysql> SELECT BINLOG_FILE, BINLOG_POSITION FROM performance_schema.clone_status;
mysql> CHANGE MASTER TO MASTER_HOST = 'source_host_name', MASTER_PORT =
source_port_num,
       ...
       MASTER_LOG_FILE = 'source_log_name',
       MASTER_LOG_POS = source_log_pos,
       FOR CHANNEL 'setup_channel';
mysql> START SLAVE USER = 'user_name' PASSWORD = 'password' FOR CHANNEL
'setup_channel';
Or from MySQL 8.0.22:
mysql> START REPLICA USER = 'user_name' PASSWORD = 'password' FOR CHANNEL
'setup_channel';
```

7、克隆操作期间创建的目录和文件

- 当执行克隆操作时，将创建以下目录和文件供内部使用。它们不应该被修改。
-
  - "#clone"：这是一个目录，位于接收方克隆目录下，其中包含了克隆操作使用的内部克隆文件。如果克隆语句指定了一个目录，则该目录会在指定的克隆目录下创建，如果克隆语句未指定目录，则该目录在接收方的datadir系统变量指定的目录下创建
  - "#ib_archive"：这是一个目录，位于发送方的数据目录下，其中包含内部归档的日志文件，这些文件在克隆操作期间在发送方上进行归档。
  - "*.#clone文件"：当接收方中的现有数据目录被远程克隆操作的数据替换时，在接收方上创建的临时数据文件（在克隆操作完成之后会被删除）

8、远程克隆操作的失败处理

- 本节描述克隆操作的不同阶段的失败处理
- 1）、检查克隆操作的前提条件是否满足

* 如果在前提条件检查期间发生故障，则CLONE INSTANCE语句操作将报错

- 2）、克隆操作期间采用备份锁来阻止DDL操作

* 如果克隆操作无法在clone_ddl_timeout系统变量指定的时间限制内获得DDL锁，则将报错

- 3）、在将数据克隆到接收方数据目录之前，将删除接收方上用户创建的数据（schema、table、tablespaces）和二进制日志

* 在远程克隆操作期间在接收方删除用户创建的数据时，不会保存接收方数据目录中的现有数据（不会先备份），如果发生故障，可能会丢失这些数据。如果这些数据很重要，则应在启动远程克隆操作之前先进行备份

* 出于参考的目的，警告信息会被打印到MySQL Server的错误日志中，以便记录删除接收方数据目录下数据的操作是何时开始和结束的：

```
2020-02-07T18:24:16.581742+08:00 9 [Warning] [MY-013460] [InnoDB] Clone removing all user data for provisioning: Started
2020-02-07T18:24:16.707929+08:00 9 [Warning] [MY-013460] [InnoDB] Clone removing all user data for provisioning: Finished
```

* 如果在删除数据时发生故障，则接收方中可能会留下一组在克隆操作之前存在的schema、table和tablespaces（数据删除失败的残留文件）

- 4）、数据是从发送方克隆出来的，其中包括：用户创建的数据、字典元数据和其他系统数据

* 如果在克隆数据时发生故障，将回滚克隆操作并删除所有克隆数据。在这个阶段，接收方之前存在的数据也已经被删除了，这会使得接收方不能正常启动实例，因为没有任何用户数据可用

* 如果出现这种情况，可以先找出失败的原因，并纠正它，然后重新执行克隆操作，或者放弃克隆操作并从克隆操作之前的备份中恢复接收方数据

- 5）、接收方的MySQL Server将在克隆操数据完成之后自动重新启动（适用于不克隆到指定目录的远程克隆操作）。在启动期间，执行典型的Server启动任务

* 如果MySQL Server自动重启失败，则可以手动重启MySQL Server来完成克隆操作（此时数据已经被完整克隆了，只差重启操作即可收尾）。

- 如果在克隆操作期间发生网络错误，那么，如果在五分钟内解决了错误，则克隆操作将继续执行。否则，克隆操作将中止并报错。

9、监控克隆操作

- 本节描述如何监控克隆操作的一些方法
- - 使用performance_schema下的克隆表对克隆操作进行监控：clone_status和clonne_progress表
  - 使用performance_schema下的阶段事件来监控克隆操作
  - 使用performance_schema下的克隆事件采集器来监控克隆操作
  - Com_clone状态变量

9.1. 使用performance_schema下的克隆表来监控克隆操作

- 克隆操作可能需要一些时间才能完成，这取决于具体的数据量和与数据传输相关的其他因素。可以在接收方的MySQL Server中使用performance_schema下的clone_status和clone_progress表来监控克隆操作的状态和进度，这两张表从MySQL 8.0.17版本开始可用
- - clone_status表：提供当前或最后执行的克隆操作的状态

- clone_progress表：按执行阶段提供当前或最后执行的克隆操作的进度信息
- 注意：
- performance_schema下的clone_status和clone_progress表只能用于监控接收方MySQL Server上的克隆操作。要监控发送方MySQL Server上的克隆操作，请使用克隆阶段事件进行监控，详见"9.2. 使用performance_schema下的阶段事件来监控克隆操作"
- 访问performance_schema下的克隆表需要用户在performance_schema上具有SELECT和EXECUTE权限
- clone_status和clone_progress表，需要加载了克隆插件且克隆插件处于激活状态时才可用，否则，这两张表不会被创建。表中的数据，在Server关闭或者重启时，其中的状态和进度数据会被持久化。

9.1.1. clone_status表

- clone_status表仅显示当前或最后执行的克隆操作的状态。表只包含一行数据，或者没数据（是空的）

- clone_status表有以下列：
-
  - ID：当前MySQL Server中的唯一标识克隆操作的标识符。
  - PID：执行克隆操作的会话进程列表ID。注：为什么是一个会话列表？因为克隆操作会涉及到很多个阶段，每个阶段具体执行工作的线程可能不同
  - STATE：克隆操作的当前状态。有效状态值包括Not Started（未启动）、In Progress（正在执行）、Completed（已完成）、Failed（失败）
  - BEGIN_TIME：'YYYY-MM-DD hh:mm:ss[.fraction]'格式的时间戳，显示克隆操作的开始时间
  - END_TIME：'YYYY-MM-DD hh:mm:ss[.fraction]'格式的时间戳，显示克隆操作的完成时间。如果操作正在执行（未结束），则该字段报告NULL值
  - SOURCE：donor节点（发送方）MySQL Server的地址信息，执行远程克隆操作时，值格式为'主机:端口'。执行本地克隆操作时，值为"LOCAL INSTANCE"
  - DESTINATION：克隆数据的存放目录路径，如果是远程克隆操作且未指定存放路径的，则值为"LOCAL INSTANCE"，如果是本地克隆操作或远程克隆操作指定了克隆数据的存放路径的，则值为具体的目录路径
  - ERROR_NO：克隆操作失败时报告的错误号
  - ERROR_MESSAGE：克隆操作失败时报告的错误消息字符串
  - BINLOG_FILE：复制数据对应的一致性二进制日志文件的名称
  - BINLOG_POSITION：复制数据对应的一致性二进制日志文件偏移量
  - GTID_EXECUTED：最后一个克隆事务的GTID值（该GTID SET与克隆数据保持一致）

- clone_status表是只读的。不允许使用DDL、TRUNCATE语句操作表
- 表结构定义

```
CREATE TABLE `clone_status` (
 `ID` int DEFAULT NULL,
 `PID` int DEFAULT NULL,
 `STATE` char(16) DEFAULT NULL,
 `BEGIN_TIME` timestamp(3) NULL DEFAULT NULL,
 `END_TIME` timestamp(3) NULL DEFAULT NULL,
```

```
  `SOURCE` varchar(512) DEFAULT NULL,
  `DESTINATION` varchar(512) DEFAULT NULL,
  `ERROR_NO` int DEFAULT NULL,
  `ERROR_MESSAGE` varchar(512) DEFAULT NULL,
  `BINLOG_FILE` varchar(512) DEFAULT NULL,
  `BINLOG_POSITION` bigint DEFAULT NULL,
  `GTID_EXECUTED` varchar(4096) DEFAULT NULL
) ENGINE=PERFORMANCE_SCHEMA DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

9.1.2. clone_progress表

- clone_progress表只显示当前或最后执行的克隆操作的进度信息（阶段信息）。克隆操作的阶段包括：DROP DATA（删除数据）、FILE COPY（拷贝文件）、PAGE_COPY（拷贝数据页）、REDO_COPY（拷贝redo log）、FILE_SYNC（SYNC文件）、RESTART（重启）、RECOVERY（恢复）。克隆操作会为每个阶段生成一条记录。因此，该表只包含7行数据，或没任何数据（为空）

- clone_progress表包含以下列：
- - ID：当前MySQL Sever中的唯一标识克隆操作标识符
  - STAGE：当前克隆阶段的名称。克隆操作的阶段包括：DROP DATA（删除数据）、FILE COPY（拷贝文件）、PAGE_COPY（拷贝数据页）、REDO_COPY（拷贝redo log）、FILE_SYNC（SYNC文件）、RESTART（重启）、RECOVERY（恢复）
  - STATE：克隆阶段的当前状态。状态包括：Not Started（未开始）、In Progress（正在执行）、Completed（已完成）
  - BEGIN_TIME：'YYYY-MM-DD hh:mm:ss[.fraction]'格式的时间戳，显示当前克隆阶段的开始时间。如果该阶段尚未启动，则报告NULL值
  - END_TIME：'YYYY-MM-DD hh:mm:ss[.fraction]'格式的时间戳，显示当前克隆阶段的完成时间。如果该阶段尚未结束，则报告NULL值
  - THREADS：该克隆阶段中使用的并发线程数
  - ESTIMATE：当前克隆阶段的估算数据量，以字节为单位
  - DATA：当前克隆阶段已传输的数据量，以字节为单位。
  - NETWORK：当前克隆阶段通过网络传输的数据量，以字节为单位
  - DATA_SPEED：当前数据传输的实际速度，以字节每秒为单位。此值可能与clone_max_data_bandwidth定义的请求最大数据传输速率不同
  - NETWORK_SPEED：当前网络传输的速度，以每秒字节为单位

- clone_progress表是只读的。不允许使用DDL、TRUNCATE语句操作表
- 表结构定义

```
CREATE TABLE `clone_progress` (
 `ID` int DEFAULT NULL,
 `STAGE` char(32) DEFAULT NULL,
 `STATE` char(16) DEFAULT NULL,
 `BEGIN_TIME` timestamp(6) NULL DEFAULT NULL,
 `END_TIME` timestamp(6) NULL DEFAULT NULL,
 `THREADS` int DEFAULT NULL,
 `ESTIMATE` bigint DEFAULT NULL,
 `DATA` bigint DEFAULT NULL,
 `NETWORK` bigint DEFAULT NULL,
 `DATA_SPEED` int DEFAULT NULL,
 `NETWORK_SPEED` int DEFAULT NULL
) ENGINE=PERFORMANCE_SCHEMA DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

9.1.3. 检查克隆操作状态的步骤

- 连接到接收方MySQL Server
- 查询performance_schema.clone_status表，类似如下：

```
mysql> SELECT STATE FROM performance_schema.clone_status;
+-----------+
| STATE |
+-----------+
| Completed |
+-----------+
```

如果克隆操作失败，可以查询performance_schema.clone_status表中的错误信息，类似如下:

```
mysql> SELECT STATE, ERROR_NO, ERROR_MESSAGE FROM
performance_schema.clone_status;
+-----------+----------+---------------+
| STATE | ERROR_NO | ERROR_MESSAGE |
+-----------+----------+---------------+
| Failed | xxx | "xxxxxxxxxxx" |
+-----------+----------+---------------+
```

- 回顾克隆操作的每个阶段的细节，可按照如下步骤操作：
-
  - 连接到接收方MySQL Server
  - 查询performance_schema.clone_progress表。类似如下，返回的结果包括克隆操作的每个阶段的状态和结束时间数据:

```
mysql> SELECT STAGE, STATE, END_TIME FROM performance_schema.clone_progress;
+-----------+-----------+----------------------------+
| stage | state | end_time |
+-----------+-----------+----------------------------+
| DROP DATA | Completed | 2019-01-27 22:45:43.141261 |
| FILE COPY | Completed | 2019-01-27 22:45:44.457572 |
| PAGE COPY | Completed | 2019-01-27 22:45:44.577330 |
| REDO COPY | Completed | 2019-01-27 22:45:44.679570 |
| FILE SYNC | Completed | 2019-01-27 22:45:44.918547 |
| RESTART | Completed | 2019-01-27 22:45:48.583565 |
| RECOVERY | Completed | 2019-01-27 22:45:49.626595 |
+-----------+-----------+----------------------------+
```

9.2. 使用performance_schema下的等待事件来监控克隆操作

- 克隆操作可能需要一些时间才能完成，这取决于数据量和与数据传输相关的其他因素。在performance_schema中，有三个阶段事件可用于监控克隆操作的进展。每个阶段事件都会报告WORK_COMPLETED和WORK_ESTIMATED值。报告的值随着操作的进度更新而同步更新。这种监控克隆操作的方法可以用于发送方或接受方的MySQL Server（对于发送方，只能使用这种方法来监控克隆操作）

- 克隆操作的阶段事件发生的顺序如下：
-
    - stage/innodb/clone (file copy)：表示克隆操作进度为文件拷贝阶段。WORK_ESTIMATED和WORK_COMPLETED列值的单位在该事件下为文件块。在文件拷贝阶段开始时就已经知道了要传输的文件数量，并根据文件数量来估算文件块的数量。WORK_ESTIMATED列值被设置为估算的文件块总数量。WORK_COMPLETED列值会在每个块发送后更新。

    - stage/innodb/clone (page copy)：表示克隆操作进度为数据页拷贝阶段。WORK_ESTIMATED和WORK_COMPLETED列值的单位在该事件下为页面数量。一旦完成了文件拷贝阶段，就知道了要传输的页面数量，并将WORK_ESTIMATED列值设置为估算的页面总数量值。WORK_COMPLETED列值在每个页面发送后更新。

    - stage/innodb/clone (redo copy)：表示克隆操作进度为redo 拷贝阶段。WORK_ESTIMATED和WORK_COMPLETED列值的单位在该事件下为redo log。一旦页面拷贝阶段完成，就知道了要传输的redo log块的数量，并将WORK_ESTIMATED列值设置为估算的redo log块的总数量值。WORK_COMPLETED列值在每个redo log块发送后更新。

- 下面将对如何启用监控克隆的事件采集器"stage/innodb/clone%" 和相关的consumer表进行一个简单的演示。

```
# 启用stage/innodb/clone% instruments:
mysql> UPDATE performance_schema.setup_instruments SET ENABLED = 'YES'  WHERE
NAME LIKE 'stage/innodb/clone%';


# 启用阶段事件消费者表，其中包括events_stages_current、events_stages_history和
events_stages_history_long表
mysql> UPDATE performance_schema.setup_consumers SET ENABLED = 'YES' WHERE NAME
LIKE '%stages%';


# 执行克隆操作。在本示例中，执行了一个克隆本地数据的操作，将一个本地数据目录克隆到一个名为
cloned_dir的目录中，如下
mysql> CLONE LOCAL DATA DIRECTORY = '/path/to/cloned_dir';

# 通过查询performance_schema.events_stages_current表来检查克隆操作的进度。查询结果会随着
克隆操作执行的阶段不同而不断更新（因为该表中每个线程只会记录当前正在执行的操作相关的阶段事件，一
旦该阶段事件对应的操作结束，该阶段事件就会从该表中删除，被下一个正在执行的阶段事件覆盖）。
WORK_COMPLETED列值显示完成的工作。WORK_ESTIMATED列显示所需的工作总数
mysql> SELECT EVENT_NAME, WORK_COMPLETED, WORK_ESTIMATED FROM
performance_schema.events_stages_current  WHERE EVENT_NAME LIKE
'stage/innodb/clone%';
+------------------------------+----------------+----------------+
| EVENT_NAME | WORK_COMPLETED | WORK_ESTIMATED |
```

```
+--------------------------------+---------------+---------------+
| stage/innodb/clone (redo copy) | 1 | 1 |
+--------------------------------+---------------+---------------+


# 如果克隆操作已经完成，此时查询performance_schema.events_stages_current表将返回一个空集。在这种情况下，可以查询performance_schema.events_stages_history表来查看已完成的克隆操作相关的阶段事件数据。例如：
mysql> SELECT EVENT_NAME, WORK_COMPLETED, WORK_ESTIMATED FROM
events_stages_history  WHERE EVENT_NAME LIKE 'stage/innodb/clone%';
+--------------------------------+---------------+---------------+
| EVENT_NAME | WORK_COMPLETED | WORK_ESTIMATED |
+--------------------------------+---------------+---------------+
| stage/innodb/clone (file copy) | 301 | 301 |
| stage/innodb/clone (page copy) | 0 | 0 |
| stage/innodb/clone (redo copy) | 1 | 1 |
+--------------------------------+---------------+---------------+
```

9.3. 使用performance_schema下克隆相关的事件采集器来监控克隆操作

- performance_schema为克隆操作的高级性能监控提供了许多采集器。要查看可用的克隆相关的事件采集器有哪些，可以通过如下语句查询：

```
root@localhost : performance_schema:56: > SELECT * FROM
performance_schema.setup_instruments WHERE NAME LIKE '%clone%';
+-----------------------------------------------+---------+-------+------------+--
----------+---------------+
| NAME | ENABLED | TIMED | PROPERTIES | VOLATILITY | DOCUMENTATION |
+-----------------------------------------------+---------+-------+------------+--
----------+---------------+
| wait/synch/mutex/innodb/clone_snapshot_mutex | NO | NO | | 0 | NULL |
| wait/synch/mutex/innodb/clone_sys_mutex | NO | NO | | 0 | NULL |
| wait/synch/mutex/innodb/clone_task_mutex | NO | NO | | 0 | NULL |
| wait/io/file/innodb/innodb_clone_file | YES | YES | | 0 | NULL |
| stage/innodb/clone (file copy) | YES | YES | progress | 0 | NULL |
| stage/innodb/clone (redo copy) | YES | YES | progress | 0 | NULL |
| stage/innodb/clone (page copy) | YES | YES | progress | 0 | NULL |
| statement/abstract/clone | YES | YES | mutable | 0 | NULL |
| statement/clone/local | YES | YES | | 0 | NULL |
| statement/clone/client | YES | YES | | 0 | NULL |
| statement/clone/server | YES | YES | | 0 | NULL |
| memory/innodb/clone | YES | NULL | | 0 | NULL |
| memory/clone/data | YES | NULL | | 0 | NULL |
+-----------------------------------------------+---------+-------+------------+--
----------+---------------+
13 rows in set (0.00 sec)
```

- 等待事件采集器(Wait Instruments)：performance_schema等待事件采集器跟踪需要时间开销的事件。克隆等待事件采集器包括如下几个：
- - wait/synch/mutex/innodb/clone_snapshot_mutex：跟踪克隆快照互斥锁的等待事件，它在多个克隆线程之间同步对动态快照对象（在发送方和接收方上）的访问
  - wait/synch/mutex/innodb/clone_sys_mutex：跟踪克隆互斥锁的等待事件。MySQL Server中有一个克隆系统对象。这个互斥锁同步对发送方和接收方的克隆系统对象的访问。它是由克隆线程和其他前台线程和后台线程获取的

    - wait/synch/mutex/innodb/clone_task_mutex：跟踪用于克隆任务互斥锁的等待时间内，用于克隆任务管理。clone_task_mutex由克隆线程获取
    - wait/io/file/innodb/innodb_clone_file：跟踪克隆操作的文件的所有I/O等待操作

- 阶段事件采集器(Stage Instruments)：performance_schema 阶段事件跟踪在语句执行过程中发生的步骤。克隆操作相关的阶段事件采集器包括如下几个：
-
    - stage/innodb/clone (file copy)：详见"9.2. 使用performance_schema下的阶段事件来监控克隆操作"
    - stage/innodb/clone (redo copy)：详见"9.2. 使用performance_schema下的阶段事件来监控克隆操作"
    - stage/innodb/clone (page copy)：详见"9.2. 使用performance_schema下的阶段事件来监控克隆操作"

- 语句事件采集器(Statement Instruments)：performance_schema语句事件跟踪语句的执行。在启动克隆操作时，可以并行执行克隆语句事件采集器来跟踪的不同的语句类型。可以在performance_schema下的语句事件记录表中观察这些语句的事件。执行的语句数量取决于克隆系统变量clone_max_concurrency（默认为16）和clone_autotune_concurrency（默认为ON）的设置。克隆语句相关的事件采集器包括如下几个：
-
    - statement/abstract/clone：在将任何克隆操作归类为本地、客户端或服务端三种操作类型之前，该事件跟踪所有克隆操作的语句事件
    - statement/clone/local：跟踪本地克隆操作的克隆语句事件，在执行CLONE LOCAL语句时生成
    - statement/clone/client：跟踪发生在接收方MySQL Server上的远程克隆语句事件，在接收方上执行CLONE INSTANCE语句时生成
    - statement/clone/server：跟踪远程克隆语句事件，发生在接收方的MySQL Server，在接收方上执行CLONE INSTANCE语句时生成

    - PS：

    * statement/clone/local、statement/clone/client、statement/clone/server这三个语句事件采集器需要加载了mysql_clone.so插件库之后才存在

- 内存事件采集器(Memory Instruments)：performance_schema内存事件采集器跟踪内存使用情况。克隆相关的内存事件采集器包括如下几个：
-
    - memory/innodb/clone：跟踪innodb为动态快照分配的内存
    - memory/clone/data：在克隆操作期间跟踪克隆插件分配的内存

    - PS：

    * memory/clone/data 采集器需要加载了mysql_clone.so插件库之后才存在

- Com_clone状态变量：
-
    - 该状态变量提供克隆操作语句执行的计数

10、停止克隆操作

- 如果有需要，可以使用 KILL QUERY processlist_id 语句停止克隆操作。在接收方MySQL Server上，您可以从performance_schema.clone_status表的PID列中查看到用于克隆操作的线程的任务列表标识符（processlist identifier，即PID），如下:

```
mysql> SELECT * FROM performance_schema.clone_status\G
*************************** 1. row ***************************
           ID: 1
          PID: 8  # 这里就是可用于kill语句的processlist id
        STATE: In Progress
   BEGIN_TIME: 2019-07-15 11:58:36.767
     END_TIME: NULL
       SOURCE: LOCAL INSTANCE
  DESTINATION: /path/to/clone_dir/
     ERROR_NO: 0
ERROR_MESSAGE:
  BINLOG_FILE:
BINLOG_POSITION: 0
 GTID_EXECUTED:
```

- 除此之外，还可以从INFORMATION_SCHEMA.processlist表的ID列、SHOW processlist语句输出的ID列或performance_schema.threads表的PROCESSLIST_ID列中查看克隆操作线程的PID。这些获取PID信息的方法在发送方或接收方的MySQL Server中都适用。

11、克隆插件系统变量

- 当前MySQL 8.0.19版本中的克隆插件支持如下系统变量，这些系统变量都具有命令行选项、也都支持写入配置文件中、都是全局作用范围、都可动态修改：
- - clone_autotune_concurrency
  - clone_buffer_size
  - clone_ddl_timeout
  - clone_enable_compression
  - clone_max_concurrency
  - clone_max_data_bandwidth
  - clone_max_network_bandwidth
  - clone_ssl_ca
  - clone_ssl_cert
  - clone_ssl_key
  - clone_valid_donor_list

- 下面将对这些系统变量进行详细解释（注意：如果在启动MySQL Server时克隆系统变量指定了非法制，则克隆插件可能无法正确初始化，Server也无法正常加载，在错误日志中可能会为其生成错误信息，以提示无法识别这些设置。

- PS：克隆插件系统变量是在执行克隆操作的MySQL Server上进行配置的。所有的系统变量都需要在加载了mysql_clone.so插件库之后才存在。

11.1. clone_autotune_concurrency

- 启用远程克隆操作线程的动态生成机制。此设置仅适用于接收方MySQL Server。可以生成的最大线程数数量由系统变量clone_max_concurrency定义
- - 全局变量，动态变量，布尔类型，默认值为ON。MySQL 8.0.17版本引入

11.2. clone_buffer_size

- 定义在本地克隆操作期间，其传输数据时使用的中间缓冲区的大小。此设置不适用于远程克隆操作。默认值是4 mebibytes (MiB)。较大的缓冲区可能允许I/O设备并行地获取数据，这可以提高克隆性能。
- - 全局变量，动态变量，整型类型，默认值为4194304，取值范围为1048576~268435456（1MB~256MB），单位为字节：。MySQL 8.0.17版本引入

11.3. clone_ddl_timeout

- 执行克隆操作时等待获取备份锁的时间（单位为秒）。此设置同时应用于发送方和接收方的MySQL Server。克隆操作不能与DDL操作并行运行。因此在发送方和接收方的MySQL Server上需要使用备份锁。当执行克隆操作时，会等待当前存在的DDL操作完成。一旦获得了备份锁，则后续的DDL操作必须等待克隆操作完成。0值表示克隆操作不需要采取备份锁。在这种情况下，如果在克隆操作期间内尝试同时执行DDL操作，克隆操作将失败并报错
- - 全局变量，动态变量，整形类型，默认值为300，取值范围为：0~2592000，单位为秒。MySQL 8.0.17版本引入

11.4. clone_enable_compression

- 允许在远程克隆操作期间在网络层压缩数据。压缩以CPU为代价来节省网络带宽。启用压缩可以提高数据传输速率。此设置仅应用于接收方MySQL Server
- - 全局变量，动态变量，布尔类型，默认值为OFF。MySQL 8.0.17版本引入

11.5. clone_max_concurrency

- 定义远程克隆操作的最大并发线程数。默认值是16。设置更多的线程可以提高克隆性能，但也会占用更多的客户端连接数，从而减少了客户端可用的连接数量，这会影响现有客户端连接的性能（因此需要留意系统变量max_connections的设置值是否足够以及你的MySQL Server的负载情况）。此设置仅应用于接收方MySQL Server

- 如果启用了系统变量clone_autotune_concurrency（默认启用），则系统变量clone_max_concurrency设置值为远程克隆操作中可动态生成的最大线程数。如果禁用了系统变量clone_autotune_concurrency，则系统变量clone_max_concurrency设置值为远程克隆操作中生成的实际线程数。

- 对于远程克隆操作，建议每个线程的最小数据传输速率为1 mebibyte (MiB)。远程克隆操作的数据传输速率由系统变量clone_max_data_bandwidth控制

- 全局变量，动态变量，整型类型，默认值为16，取值范围为：1~128。MySQL 8.0.17版本引入

11.6. clone_max_data_bandwidth

- 定义远程克隆操作的每秒最大数据传输速率（以MiB为单位）。通过此变量可以方便地管理传输率大小对克隆操作的性能影响。只有当发送方磁盘I/O带宽饱和，并影响性能时，才应该设置限制。0值表示“无限制”，这允许克隆操作以尽可能高的数据传输速率运行。此设置仅适用于接收方MySQL Server

- 每个线程的最小数据传输速率为每秒1 MiB。例如，如果有8个线程，那么最小的传输速率是每秒8 MiB。为远程克隆操作生成的最大线程数量由系统变量clone_max_concurrency控制

- 系统变量clone_max_data_bandwidth指定的请求数据传输速率可能与performance_schema.clone_progress表中DATA_SPEED列报告的实际数据传输速率不同。当克隆操作没有达到所需的数据传输速率（系统变量指定的传输率），并且存在空闲你的带宽，那么请检查接收方和提供方的磁盘I/O使用情况。如果存在未充分利用的带宽，则磁盘I/O是最可能的瓶颈点
- 全局变量，动态变量，整型类型，默认值为0，取值范围为：0~1048576，单位为字节。MySQL 8.0.17版本引入

11.7. clone_max_network_bandwidth

- 指定远程克隆操作的最大近似网络传输速率（以每秒MiB为单位）。通过此变量可以方便地管理网络传输带宽大小对克隆操作的性能影响。但它应该只在网络带宽饱和时设置（例如：网络带宽饱和会影响发送方MySQL Server的性能时）。0值表示“无限制”，这允许以尽可能高的网络数据传输速率来进行克隆，从而提供最佳克隆性能。此设置仅适用于接收方MySQL Server
- 全局变量，动态变量，整型类型，默认值为0，取值范围为：0~1048576，单位为字节。MySQL 8.0.17版本引入

11.8. clone_ssl_ca

- 指定证书颁发机构(CA)文件的路径。用于为远程克隆操作配置加密连接。此系统变量在接收方上配置，并在连接到发送方时使用。
- 全局变量，动态变量，文件名称类型，默认值为空串。MySQL 8.0.14版本引入

11.9. clone_ssl_cert

- 指定公钥证书的路径。用于为远程克隆操作配置加密连接。此系统变量在接收方上配置，并在连接到发送方时使用
- 全局变量，动态变量，文件名称类型，默认值为空串。MySQL 8.0.14版本引入

11.10. clone_ssl_key

- 指定私钥文件的路径。用于为远程克隆操作配置加密连接。此系统变量在接收方上配置，并在连接到发送方时使用
- - 全局变量，动态变量，文件名称类型，默认值为空串。MySQL 8.0.14版本引入

11.11. clone_valid_donor_list

- 为远程克隆操作定义有效的发送方(donor)主机地址。此设置应用于接收方MySQL Server。多个donor主机地址之间使用逗号分隔，例如：“HOST1:PORT1,HOST2:PORT2,HOST3:PORT3”。注意：不允许有空格。

- 通过提供对克隆数据源的控制，系统变量clone_valid_donor_list增加了一层安全性。配置系统变量clone_valid_donor_list所需的权限与执行远程克隆操作所需的权限不同，后者允许将这些职责分配给不同的角色。配置系统变量clone_valid_donor_list需要用户具有SYSTEM_VARIABLES_ADMIN权限，而执行远程克隆操作需要用用户具有CLONE_ADMIN权限

- 不支持 IPv6地址格式，但可以使用IPv6地址的别名。支持使用IPv4地址
- - 全局变量，动态变量，字符串类型，默认值为空串。MySQL 8.0.17版本引入

12、克隆插件的限制

- 当前版本的克隆插件受以下限制：
- - DDL语句（包括TRUNCATE表）在克隆操作期间是不允许并行执行的。在选择数据源时应该考虑到这个限制。一种解决方法是使用专用的donor 节点，这样，就可以避开在同一个节点上同时执行克隆操作与DDL语句（DML语句与克隆操作不冲突）
  - 无法从不同的MySQL Server版本克隆数据，即，克隆操作的数据发送方和接收方，必须使用相同的MySQL Server版本。例如，不能在MySQL 5.7和MySQL 8.0之间进行克隆操作。且，克隆插件只支持MySQL 8.0.17或更高版本。

  - 一个克隆操作一次只能克隆一个MySQL Server。不支持在单个克隆操作中同时克隆多个MySQL Server的数据
  - 远程克隆操作（在CLONE INSTANCE语句中指定接收方MySQL Server的端口号时）不支持mysqlx_port指定的X协议端口
  - 克隆插件不支持克隆MySQL Server的配置信息（例如：my.cnf文件）。接收方MySQL Server的配置信息需要自己保留，其中也包括持久化的系统变量设置等。

  - 克隆插件不支持克隆二进制日志
  - 克隆插件仅克隆存储在InnoDB中的数据。其他存储引擎中的数据不会克隆（但会克隆表结构）。存储在任何schema（包括sys schema）中的MyISAM和CSV引擎表都会被克隆为空表（即，不克隆数据）
  - 不支持通过MySQL Router连接到donor MySQL Server
  - 本地克隆操作不支持克隆使用绝对路径创建的常规表空间。因为使用与源表空间文件相同的路径克隆表空间文件会导致冲突。

13、附录与参考文献

- 远程克隆与本地克隆：如果指定了目录，则目录会自动创建（不允许事先存在，否则报错）。

```
root@localhost : (none):15: > CLONE INSTANCE FROM 'repl'@'10.10.30.162':3306
IDENTIFIED BY 'letsg0' DATA DIRECTORY = '/data/backup/clone_data';
ERROR 1007 (HY000): Can't create database '/data/backup/clone_data'; database
exists
```

远程克隆时，clone instance语句指定的donor节点的用户中的主机地址，必须与clone_valid_donor_list系统变量中指定的用户中的主机地址完全一致，否则报错:

```
root@localhost : (none):03: > set global
clone_valid_donor_list='10.10.30.162:3306';
Query OK, 0 rows affected (0.00 sec)


root@localhost : (none):04: > CLONE INSTANCE FROM 'repl'@'%':3306 IDENTIFIED BY
'letsg0' DATA DIRECTORY = '/data/backup/clone_data';
ERROR 3869 (HY000): Clone system configuration: %:3306 is not found in
clone_valid_donor_list: 10.10.30.162:3306
```

- 在MGR组复制拓扑中，执行克隆操作的节点不能是MGR组中的成员，否则报错:

```
# 远程克隆
root@localhost : (none):50: > set global
clone_valid_donor_list='10.10.30.162:3306';
Query OK, 0 rows affected (0.00 sec)


root@localhost : (none):51: > CLONE INSTANCE FROM 'repl'@10.10.30.162':3306
IDENTIFIED BY 'letsg0' DATA DIRECTORY = '/data/backup/clone_data';
ERROR 3875 (HY000): The clone operation cannot be executed when Group Replication
is running.


# 本地克隆
root@localhost : (none):57: > CLONE LOCAL DATA DIRECTORY =
'/data/backup/clone_data';
ERROR 3875 (HY000): The clone operation cannot be executed when Group Replication
is running.
```

在MGR组复制拓扑中，执行远程克隆，虽然不需要将执行远程克隆的节点加入到MGR中，但是，必须要加载MGR插件才允许执行克隆操作，否则报错:

```
root@localhost : (none):07: > CLONE INSTANCE FROM 'repl'@'10.10.30.162':3306
IDENTIFIED BY 'letsg0' DATA DIRECTORY = '/data/backup/clone_data';
ERROR 3870 (HY000): Clone Donor plugin group_replication is not active in
Recipient.


root@localhost : (none):15: > INSTALL PLUGIN group_replication SONAME
'group_replication.so';
Query OK, 0 rows affected (0.01 sec)
```

```
root@localhost : (none):16: > CLONE INSTANCE FROM 'repl'@'10.10.30.162':3306
IDENTIFIED BY 'letsg0' DATA DIRECTORY = '/data/backup/clone_data';
Query OK, 0 rows affected (27.71 sec)


# 执行克隆12G的数据量不到30秒就传输完成了
[root@physical-machine clone_data]# du -sh /data/backup/clone_data/
12G /data/backup/clone_data/
[root@physical-machine clone_data]# ll
total 6355972
drwxr-x--- 2 mysql mysql 89 Oct 8 16:17 #clone
-rw-r----- 1 mysql mysql 2147483648 Oct 8 16:17 ibdata1
-rw-r----- 1 mysql mysql 2147483648 Oct 8 16:17 ib_logfile0
-rw-r----- 1 mysql mysql 2147483648 Oct 8 16:17 ib_logfile1
drwxr-x--- 2 mysql mysql 6 Oct 8 16:17 mysql
-rw-r----- 1 mysql mysql 24117248 Oct 8 16:17 mysql.ibd
drwxr-x--- 2 mysql mysql 44 Oct 8 16:17 sbtest
drwxr-x--- 2 mysql mysql 28 Oct 8 16:17 sys
drwxr-x--- 2 mysql mysql 20 Oct 8 16:17 test
-rw-r----- 1 mysql mysql 20971520 Oct 8 16:17 undo_001
-rw-r----- 1 mysql mysql 20971520 Oct 8 16:17 undo_002
```

- 参考文献
-
  - 官方手册：https://dev.mysql.com/doc/refman/8.0/en/clone-plugin.html
  - 温正湖知乎专栏：https://zhuanlan.zhihu.com/p/76255304
  - https://dev.mysql.com/worklog/task/?id=9211