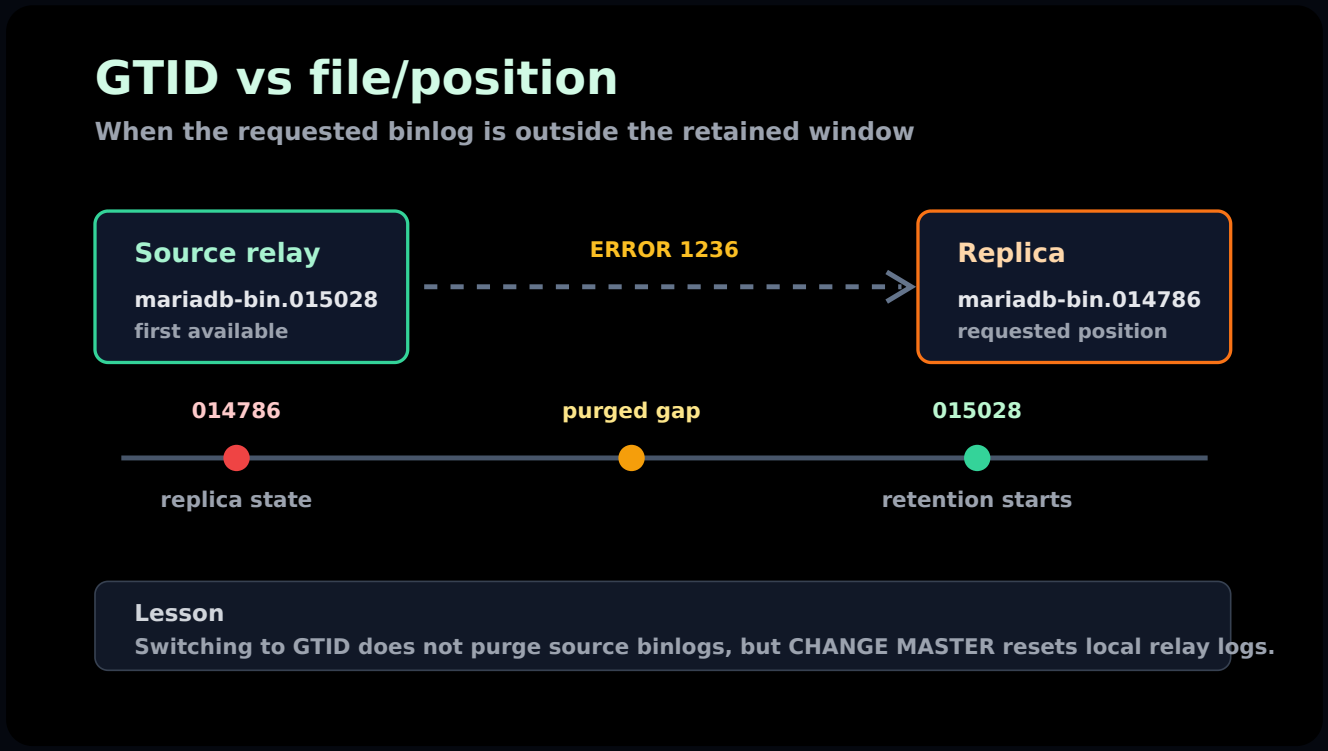
Quand le passage au GTID casse une réplication

Aurélien LEQUOY · 18 وياما 2026

MARIADB MYSQL REPLICATION GTID BINLOG PMACONTROL DBA INCIDENT



Le symptôme

Le scénario est classique : une réplication MariaDB fonctionne en mode fichier/position, on active GTID pour tester ou fiabiliser le rattachement, puis on revient en mode normal. Quelques secondes plus tard, le thread IO refuse de repartir :

```
SHOW SLAVE STATUS\G
```

```
Slave_IO_Running: No
Slave_SQL_Running: Yes
Using_Gtid: No
Last_IO_Errno: 1236
Last_IO_Error: Could not find first log file name in binary log index file
```

La tentation immédiate est de dire : "le passage en GTID a purgé les relay logs ou les binlogs". Dans l'incident analysé, ce n'était pas le cas.

Ce que fait réellement le bouton GTID

Dans PmaControl, l'action `GTID > Activate` sur une réplication MariaDB ne lance pas de purge. Elle ne lance pas non plus de `RESET SLAVE`.

Elle fait seulement :

```
STOP SLAVE;
CHANGE MASTER TO MASTER_USE_GTID = slave_pos;
START SLAVE;
```

Le retour arrière fait :

```
STOP SLAVE;
CHANGE MASTER TO MASTER_USE_GTID = no;
START SLAVE;
```

Ces commandes ne suppriment pas les binlogs du master. En revanche, le `CHANGE MASTER` recrée l'état local de réplication et réinitialise les relay logs du slave. C'est une nuance importante : le master ne perd rien, mais le slave jette ce qu'il avait déjà téléchargé localement.

Test grandeur nature

Pour vérifier le comportement, j'ai reproduit le cas sur deux serveurs MariaDB 11.8 de laboratoire :

```
source -> replica
file/position au départ
relay_log_purge=ON
```

Le test :

1. démarrer une réplication file/position propre ;
2. arrêter seulement le SQL thread du replica ;
3. générer 20 000 lignes sur la source ;

4. vérifier que l'IO thread télécharge bien les événements dans les relay logs ;
5. exécuter la séquence du bouton `GTID > Activate`.

Avant `CHANGE MASTER`, le replica avait des relay logs non appliqués :

```
Slave_IO_Running: Yes
Slave_SQL_Running: No
Read_Master_Log_Pos: 635180
Exec_Master_Log_Pos: 3464
Relay_Log_Space: 632576
relay-bin.000002: 632273 bytes
```

Après `STOP SLAVE`, rien n'était perdu :

```
Slave_IO_Running: No
Slave_SQL_Running: No
Relay_Log_Space: 632576
relay-bin.000002: 632273 bytes
```

Juste après :

```
CHANGE MASTER TO MASTER_USE_GTID=slave_pos;
```

les relay logs ont été réinitialisés :

```
Using_Gtid: Slave_Pos
Read_Master_Log_Pos: 3464
Exec_Master_Log_Pos: 3464
Relay_Log_Space: 256
relay-bin.000001: 256 bytes
```

Le test inverse donne le même résultat. En mode GTID, avec des relay logs non appliqués, un retour vers file/position via :

```
CHANGE MASTER TO MASTER_USE_GTID=no;
```

réinitialise aussi les relay logs locaux.

Conclusion du test : `STOP SLAVE` seul conserve les relay logs. `CHANGE MASTER`, même limité à `MASTER_USE_GTID`, les remet à zéro.

Ce qui s'est vraiment passé

Le slave est revenu en mode fichier/position avec une coordonnée persistée trop ancienne :

```
mariadb-bin.014786:35578691
```

Le relay-master ne possédait plus ce fichier dans son index binaire. Sa première entrée disponible était déjà plus récente :

```
first available: mariadb-bin.015028
last available:  mariadb-bin.015130
retained files:  103
retained size:   about 10 GiB
```

Le master était pourtant configuré avec :

```
binlog_expire_logs_seconds = 864000
expire_logs_days           = 10
binlog_space_limit         = 0
max_binlog_size            = 104857600
```

La rétention configurée disait donc "10 jours". Mais la position demandée par le slave était plus ancienne que la fenêtre effectivement disponible.

Pourquoi 10 jours ne garantissent pas ce fichier

Une rétention de 10 jours veut dire que le serveur peut purger les binlogs plus vieux que cette limite. Elle ne garantit pas qu'un slave pourra revenir sur n'importe quelle vieille coordonnée après un changement de mode.

Plusieurs cas rendent la situation piégeuse :

- la réplication était déjà en retard avant le clic ;
- le slave a conservé une ancienne coordonnée file/position pendant le passage GTID ;
- une purge antérieure avait déjà retiré les fichiers demandés ;
- la valeur de rétention a pu être changée dans le passé ;
- une maintenance ou un rebuild a pu recréer une fenêtre plus courte que prévu ;
- le master amont peut avoir une politique différente du relay-master.

Le point important est simple : MariaDB ne peut pas servir un fichier absent de `SHOW BINARY LOGS`, même si la configuration actuelle affiche 10 jours.

Pourquoi GTID a masqué le problème

En mode GTID MariaDB, le slave ne demande plus un couple `MASTER_LOG_FILE` / `MASTER_LOG_POS`. Il demande un ensemble logique de transactions via `gtid_slave_pos`.

Quand on repasse en file/position, on redonne de l'importance aux anciennes coordonnées physiques. Si ces coordonnées pointent vers un fichier purgé, le thread IO échoue immédiatement avec `1236`.

Le clic GTID n'a donc pas créé la purge des binlogs master. En revanche, la bascule via `CHANGE MASTER` a bien supprimé les relay logs locaux déjà téléchargés par le slave. Si les transactions présentes dans ces relay logs ne sont plus disponibles dans les binlogs du master, la reprise devient impossible sans resynchronisation.

Pourquoi un simple CHANGE MASTER ne suffit pas

On peut être tenté de repositionner le slave sur la position courante du master :

```
STOP SLAVE;
CHANGE MASTER TO
  MASTER_LOG_FILE='mariadb-bin.015130',
  MASTER_LOG_POS=97211185,
  MASTER_USE_GTID=no;
START SLAVE;
```

Le thread IO peut repartir. Mais le SQL thread risque ensuite de casser sur une contrainte, par exemple :

```
Last_SQL_Errno: 1452
Cannot add or update a child row:
foreign key constraint fails
```

C'est logique : en sautant directement à une position récente, on ignore une plage de transactions. La donnée du slave ne correspond plus à la suite d'événements qu'il reçoit.

Ce n'est pas une réparation, c'est un saut dans le journal.

La bonne correction

La correction saine est une resynchronisation complète du slave depuis une source cohérente :

1. arrêter la réplication ;
2. prendre un backup cohérent du master ou du relay-master ;
3. restaurer le slave ;
4. récupérer la position exacte du backup ;
5. reconfigurer la réplication en GTID ou en file/position ;
6. redémarrer la réplication ;
7. vérifier que les deux threads sont verts.

Le résultat attendu :

```
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
Seconds_Behind_Master: 0 ou en baisse
Last_IO_Error:
Last_SQL_Error:
```

Il faut éviter `sql_slave_skip_counter` dans ce cas. Sauter des erreurs FK masque la divergence et laisse une base incohérente.

Le contrôle à ajouter avant de cliquer

Avant d'activer ou de désactiver GTID, PmaControl doit pouvoir répondre à trois questions :

```
SHOW SLAVE STATUS\G
SHOW BINARY LOGS;
SHOW VARIABLES WHERE Variable_name IN (
  'expire_logs_days',
  'binlog_expire_logs_seconds',
  'binlog_space_limit',
  'max_binlog_size'
);
```

Ensuite :

- le `Relay_Master_Log_File` du slave existe-t-il encore sur le master ?
- le slave est-il déjà en retard ou en erreur ?
- la fenêtre de binlogs couvre-t-elle la position demandée ?

Si la réponse est non, le bouton ne doit pas juste envoyer `CHANGE MASTER`. Il doit afficher un avertissement et proposer un rebuild.

Conclusion

Le passage GTID n'a pas purgé les binlogs du master. Mais le `CHANGE MASTER` utilisé pour passer en GTID puis revenir en file/position a réinitialisé les relay logs locaux du slave.

Dans un environnement sain, le slave les retélécharge depuis le master. Dans l'incident analysé, la coordonnée demandée était déjà hors de la fenêtre de binlogs du master. Les relay logs locaux ayant été jetés, il ne restait plus de source pour rejouer la plage manquante.

La leçon opérationnelle est nette : avant de basculer entre GTID et file/position, il faut valider la fenêtre de binlogs du master. Sinon on transforme un test réversible en resynchronisation obligatoire.