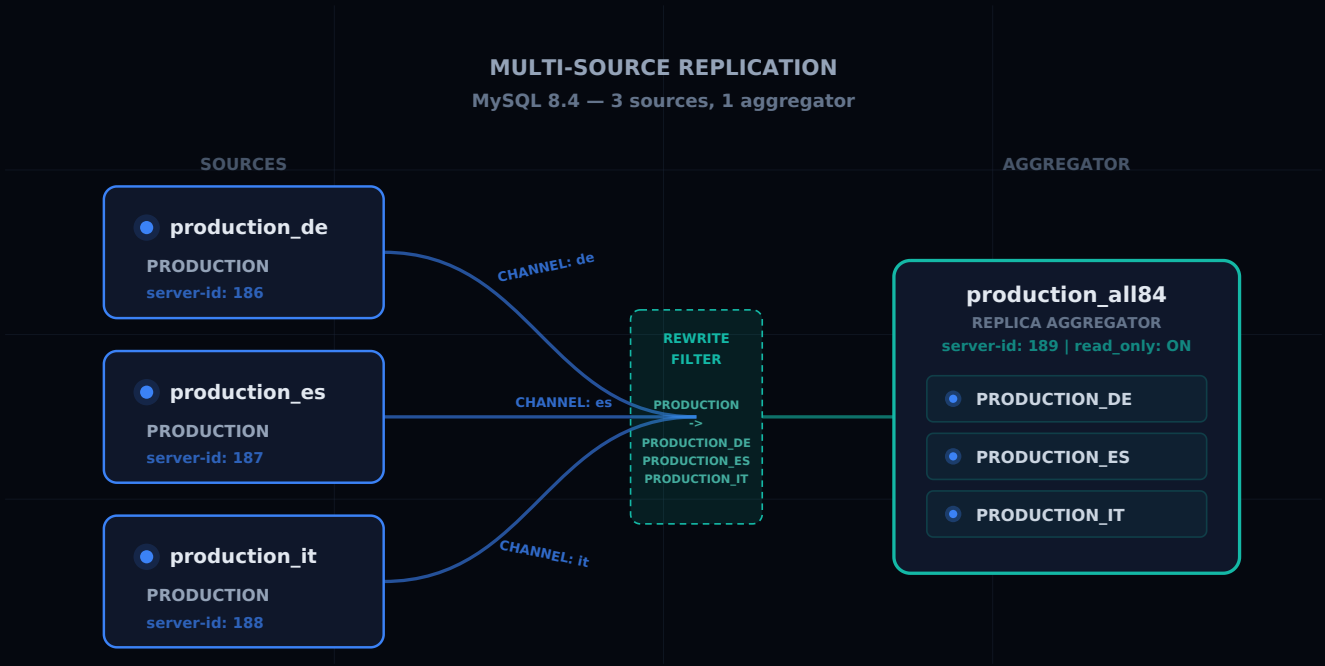
Multi-Source Replication avec MySQL 8.4

Aurélien LEQUOY · 12 avril 2026

MYSQL REPLICATION MULTI-SOURCE MYSQL-8.4



Introduction

La multi-source replication de MySQL 8.4 permet à un même replica de recevoir des transactions depuis plusieurs serveurs sources en parallèle. Chaque source est rattachée au replica via un canal de réplication distinct.

Ce mécanisme sert surtout à:

- consolider plusieurs serveurs dans un seul noeud
- agréger des flux provenant de plusieurs pays, sites ou applications
- centraliser des données en lecture
- préparer un noeud de reporting ou de réconciliation

En revanche, ce n'est pas un cluster d'écriture partagé. MySQL ne fait pas de résolution de conflit entre plusieurs sources. Si deux sources écrivent dans les mêmes objets logiques, la cohérence doit être pensée côté application ou via un cloisonnement strict des données.

Ce que MySQL 8.4 supporte réellement

En MySQL 8.4:

- un replica multi-source ouvre un canal par source
- chaque canal doit pointer vers une source différente
- la réplication peut être basée sur GTID ou sur positions binlog
- les filtres de réplication peuvent être appliqués par canal
- les repositories de métadonnées replica doivent être en mode `TABLE`, ce qui est le comportement par défaut en 8.4

Points importants:

- le multi-source sert à consolider, pas à faire du multi-primary avec arbitrage
- il n'y a pas de détection ou résolution de conflit intégrée
- un même replica ne peut pas ouvrir plusieurs canaux vers une même source

Topologie d'exemple

Exemple simple avec trois sources et un agrégateur:

```
production_de  ─┐
production_es  ─┼──> production_all84
production_it  ─┘
```

Dans cet exemple:

- `production_de` contient une base `PRODUCTION`
- `production_es` contient aussi une base `PRODUCTION`
- `production_it` contient aussi une base `PRODUCTION`
- `production_all84` reçoit les trois flux mais les remappe vers des bases différentes:
  - `PRODUCTION_DE`
  - `PRODUCTION_ES`
  - `PRODUCTION_IT`

Ce remappage évite que les trois flux écrivent dans la même base sur le replica.

Pré-requis

Sur chaque source:

- `server-id` unique
- binaire log activé
- accès TCP/IP au port MySQL
- utilisateur de réplication dédié

Sur le replica multi-source:

- `server-id` unique
- `relay_log` configuré
- MySQL 8.4
- restauration initiale des données avant démarrage de la réplication

Configuration type des sources

Exemple de configuration minimale sur une source:

```
[mysqld]
bind-address = 0.0.0.0
server-id = 186
log_bin = mysql-bin
binlog_format = ROW
binlog_row_image = FULL
sync_binlog = 1
innodb_flush_log_at_trx_commit = 1
```

Même logique sur les autres sources, avec un `server-id` différent sur chaque serveur.

Configuration type du replica agrégateur

Exemple:

```
[mysqld]
bind-address = 0.0.0.0
server-id = 189
```

```
log_bin = mysql-bin
relay_log = mysql-relay-bin
binlog_format = ROW
binlog_row_image = FULL
skip_replica_start = ON
read_only = ON
super_read_only = ON
```

`skip_replica_start=ON` est utile pendant la phase de montage ou de maintenance, car il évite un redémarrage automatique des canaux avant validation.

Création du compte de réplication

Sur chaque source:

```
CREATE USER 'repl'@'10.68.68.%' IDENTIFIED BY 'Repl84Geo2026x';
GRANT REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO 'repl'@'10.68.68.%';
```

Le privilège `REPLICATION SLAVE` reste celui documenté par MySQL pour ce type de montage en 8.4.

Chargement initial des données

Avant d'attacher un canal, il faut charger les données de départ sur le replica.

Exemple avec des dumps pris sur les sources:

```
mysqldump --single-transaction --set-gtid-purged=OFF PRODUCTION > PRODUCTION_de.sql
mysqldump --single-transaction --set-gtid-purged=OFF PRODUCTION > PRODUCTION_es.sql
mysqldump --single-transaction --set-gtid-purged=OFF PRODUCTION > PRODUCTION_it.sql
```

Puis sur le replica:

```
CREATE DATABASE PRODUCTION_DE;
CREATE DATABASE PRODUCTION_ES;
CREATE DATABASE PRODUCTION_IT;
```

Et import des trois dumps dans les trois bases cibles correspondantes.

Récupération des coordonnées binlog

Si on ne travaille pas en GTID, il faut récupérer pour chaque source:

- le nom du fichier binlog
- la position de départ

Exemple:

```
SHOW BINARY LOG STATUS;
```

Le replica démarrera ensuite à partir de ces coordonnées avec `SOURCE_LOG_FILE` et `SOURCE_LOG_POS`.

Création des canaux

En MySQL 8.4, chaque source se configure avec `CHANGE REPLICATION SOURCE TO ... FOR CHANNEL`.

Exemple pour la source allemande:

```
CHANGE REPLICATION SOURCE TO
  SOURCE_HOST='10.68.68.186',
  SOURCE_PORT=3306,
  SOURCE_USER='repl',
  SOURCE_PASSWORD='Repl84Geo2026x',
  SOURCE_LOG_FILE='mysql-bin.000001',
  SOURCE_LOG_POS=158,
  SOURCE_AUTO_POSITION=0,
  GET_SOURCE_PUBLIC_KEY=1
FOR CHANNEL 'production_de';
```

Même principe pour:

- `production_es`
- `production_it`

Filtres par canal

La force du multi-source vient du filtrage canal par canal.

Si les trois sources ont une base `PRODUCTION`, on peut les réécrire côté replica:

```
CHANGE REPLICATION FILTER
  REPLICATE_REWRITE_DB=((PRODUCTION,PRODUCTION_DE))
FOR CHANNEL 'production_de';

CHANGE REPLICATION FILTER
  REPLICATE_REWRITE_DB=((PRODUCTION,PRODUCTION_ES))
FOR CHANNEL 'production_es';

CHANGE REPLICATION FILTER
  REPLICATE_REWRITE_DB=((PRODUCTION,PRODUCTION_IT))
FOR CHANNEL 'production_it';
```

Ce point est important:

- le canal doit exister avant d'appliquer `CHANGE REPLICATION FILTER ... FOR CHANNEL`
- si le canal n'existe pas encore, MySQL renvoie une erreur

Démarrage des canaux

```
START REPLICA FOR CHANNEL 'production_de';
START REPLICA FOR CHANNEL 'production_es';
START REPLICA FOR CHANNEL 'production_it';
```

Ensuite, si le replica doit rester passif:

```
SET GLOBAL read_only = ON;
SET GLOBAL super_read_only = ON;
```

Vérification

Contrôle canal par canal:

```
SHOW REPLICA STATUS FOR CHANNEL 'production_de'\G
SHOW REPLICA STATUS FOR CHANNEL 'production_es'\G
SHOW REPLICA STATUS FOR CHANNEL 'production_it'\G
```

Les indicateurs attendus:

- `Replica_IO_Running: Yes`
- `Replica_SQL_Running: Yes`
- `Seconds_Behind_Source: 0` ou proche de `0`
- `Replicate_Rewrite_DB` correctement défini

Contrôle fonctionnel:

```
SELECT * FROM PRODUCTION_DE.germany_feed;
SELECT * FROM PRODUCTION_ES.spain_feed;
SELECT * FROM PRODUCTION_IT.italy_feed;
```

Opérations courantes

Arrêter un seul canal:

```
STOP REPLICA FOR CHANNEL 'production_es';
```

Redémarrer un seul canal:

```
START REPLICA FOR CHANNEL 'production_es';
```

Réinitialiser un canal:

```
RESET REPLICA ALL FOR CHANNEL 'production_es';
```

Supprimer un filtre de réécriture sur un canal:

```
CHANGE REPLICATION FILTER REPLICATE_REWRITE_DB=() FOR CHANNEL 'production_es';
```

Ce qu'il faut éviter

- faire converger plusieurs sources vers la même base cible sans cloisonnement
- supposer que MySQL va résoudre les conflits de clés ou d'ordonnancement
- utiliser plusieurs versions MySQL différentes sans contrôle strict de compatibilité
- oublier que `SOURCE_PASSWORD` dans `CHANGE REPLICATION SOURCE TO` est limité en longueur
- appliquer les filtres avant la création du canal

Positionnement technique

Le multi-source MySQL 8.4 est adapté pour:

- consolidation multi-pays
- reporting centralisé
- réconciliation
- reprise de plusieurs flux indépendants

Il n'est pas adapté, à lui seul, pour:

- un vrai multi-master d'écriture concurrente
- une architecture de consensus
- une résolution automatique de conflits

Conclusion

Avec MySQL 8.4, la multi-source replication est propre, mature et exploitable pour agréger plusieurs serveurs sources vers un seul replica.

Le schéma recommandé est simple:

1. séparer clairement les rôles source et agrégateur
2. imposer un `server-id` unique partout
3. charger un snapshot initial
4. créer un canal par source
5. appliquer des filtres de réécriture par canal
6. vérifier chaque canal indépendamment

Si les données des sources portent des noms de bases identiques, `REPLICATE_REWRITE_DB` par canal est le mécanisme le plus utile pour garder un replica final lisible et exploitable.

Sources officielles

- MySQL 8.4 — Multi-Source Replication
- Configuring Multi-Source Replication
- Adding Binary Log Based Sources

- Channel Based Filters
- CHANGE REPLICATION SOURCE TO