Replikacja w MySQL i MariaDB

Poziom: średni **Dotyczy: MySQL 5.7–8.0, MariaDB 10.x–11.x** **Czas: ~25 min**

Replika MySQL to nie kopia bazy odświeżana co jakiś czas — to osobna instancja, która bez przerwy odtwarza u siebie ten sam strumień zmian, jaki źródło zapisało do własnego binloga.

Skróty użyte na tej stronie (2)

Skrót	Co znaczy
GTID	Global Transaction Identifier: unikalny w całym klastrze numer nadawany każdej transakcji w MySQL i MariaDB, ułatwia przełączenie repliki na nowe źródło bez ręcznego szukania pozycji
SQL	Structured Query Language: język zapytań do relacyjnych baz danych

Fiszka

ZMIENIA SYSTEM

Utworzenie konta do replikacji: CREATE USER + GRANT REPLICATION SLAVE

```
CREATE USER 'NazwaUzytkownikaReplikacja'@'10.20.10.%' IDENTIFIED BY 'TrudneHaslo';
GRANT REPLICATION SLAVE ON *.* TO 'NazwaUzytkownikaReplikacja'@'10.20.10.%';
```

Na źródle tworzymy konto dedykowane wyłącznie do replikacji, z jedynym potrzebnym uprawnieniem.

TYLKO ODCZYT

Wskazanie źródła replikacji: CHANGE REPLICATION SOURCE TO

```
CHANGE REPLICATION SOURCE TO
  SOURCE_HOST = 'NazwaSerwera',
  SOURCE_PORT = 3306,
  SOURCE_USER = 'NazwaUzytkownikaReplikacja',
  SOURCE_PASSWORD = 'TrudneHaslo',
  SOURCE_AUTO_POSITION = 1;
```

`SOURCE_AUTO_POSITION = 1` włącza tryb GTID — replika sama ustala, od którego momentu zacząć, bez podawania pliku i pozycji. Bez GTID w to samo miejsce trafiają `SOURCE_LOG_FILE` i `SOURCE_LOG_POS`, odczytane zwykle z pozycji zapisanej przez `mysqldump --master-data` w chwili wykonania backupu źródłowego, opisanego na stronie mysqldump i backup logiczny.

GTID = Global Transaction Identifier: unikalny w całym klastrze numer nadawany każdej transakcji w MySQL i MariaDB, ułatwia przełączenie repliki na nowe źródło bez ręcznego szukania pozycji

TYLKO ODCZYT

Uruchomienie repliki: START REPLICA

```
START REPLICA;
```

TYLKO ODCZYT

Zatrzymanie repliki: STOP REPLICA

```
STOP REPLICA;
```

Nazwy obowiązujące od tych samych wersji co `CHANGE REPLICATION SOURCE TO` — starsze `START SLAVE` / `STOP SLAVE` nadal działają jako aliasy. `STOP REPLICA` zatrzymuje oba wątki repliki (`IO` i `SQL`) naraz; warianty `STOP REPLICA IO_THREAD` i `STOP REPLICA SQL_THREAD` pozwalają zatrzymać tylko jeden z nich, na przykład żeby zatrzymać aplikowanie zmian, jednocześnie dalej ściągając je do relay logu.

SQL = Structured Query Language: język zapytań do relacyjnych baz danych

TYLKO ODCZYT

Sprawdzenie stanu replikacji: SHOW REPLICA STATUS

```
SHOW REPLICA STATUS\G
```

Przykład (fragment):

```
*************************** 1. row ***************************
             Replica_IO_State: Waiting for source to send event
                  Source_Host: my01
                  Source_User: NazwaUzytkownikaReplikacja
                  Source_Port: 3306
             Replica_IO_Running: Yes
            Replica_SQL_Running: Yes
        Seconds_Behind_Source: 0
    Last_IO_Error:
   Last_SQL_Error:
       Retrieved_Gtid_Set: a1b2c3d4-...:1-48213
        Executed_Gtid_Set: a1b2c3d4-...:1-48213
```

`Replica_IO_Running` i `Replica_SQL_Running` na `Yes` obok siebie to warunek konieczny, żeby uznać replikację za działającą — `Seconds_Behind_Source` równe `0` oznacza, że replika w pełni dogoniła źródło. Wartość `NULL` w tym polu, przy jednoczesnym `Yes`/`Yes`, oznacza zwykle chwilowy brak nowych zdarzeń do przetworzenia, nie awarię.

□□ **show** = FortiOS: wypisuje konfigurację, w podstawowej postaci tylko ustawienia różniące się od domyślnych, a z wariantem full-configuration także wartości domyślne

□□ ZMIENIA SYSTEM

Pominięcie jednej transakcji na replice: SET GLOBAL sql_slave_skip_counter

```
SET GLOBAL sql_slave_skip_counter = 1;
START REPLICA;
```

Doraźnym (i ryzykownym) rozwiązaniem bywa pominięcie jednej, konkretnie tej transakcji. Polecenie działa wyłącznie bez GTID — z włączonym GTID pomijamy transakcję wstrzykując pusty GTID w jej miejsce, co wymaga dokładnej znajomości identyfikatora pomijanej transakcji. Pominięcie transakcji zawsze oznacza rozjazd danych między źródłem a repliką w tym jednym miejscu — właściwym rozwiązaniem długoterminowym jest znalezienie przyczyny rozjazdu, nie stałe pomijanie kolejnych błędów.

SET = Switch Embedded Teaming: zespół kart sieciowych zbudowany w przełączniku wirtualnym Hyper-V; łączy dwie karty w jedno łącze bez ustawień po stronie przełącznika fizycznego, więc bez LACP · **GTID** = Global Transaction Identifier: unikalny w całym klastrze numer nadawany każdej transakcji w MySQL i MariaDB, ułatwia przełączenie repliki na nowe źródło bez ręcznego szukania pozycji

Z życia wzięte

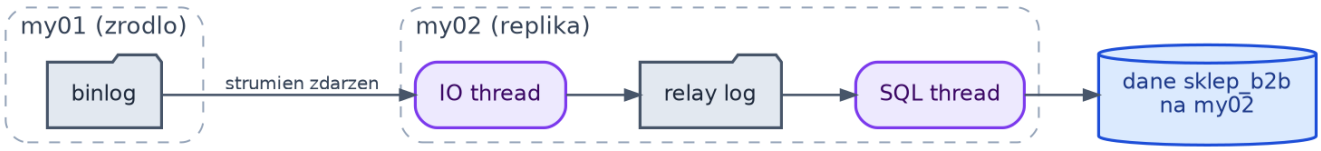
Z ŻYCIA WZIĘTE

Stawiamy replikę my02 dla `sklep_b2b` na my01, żeby raporty księgowości nie obciążały instancji, z której korzystają klienci hurtowi. Po tygodniu `Seconds_Behind_Source` zaczyna rosnąć w godzinach szczytu zamówień, bo pojedynczy SQL thread na replice nie nadąża za wieloma równoległymi zapisami na źródle. Włączamy replikację równoległą po stronie repliki, żeby SQL thread nadążał za tempem źródła.

Replikacja asynchroniczna oparta o binlog

Pojęcie: replikacja w MySQL/MariaDB. Mechanizm, w którym jedna instancja (źródło, w starszym nazewnictwie „master") wysyła strumień zdarzeń ze swojego binloga, opisanego na stronie Logi binarne: archiwizacja i czyszczenie, do jednej lub wielu instancji-replik (w starszym nazewnictwie „slave"), a każda replika odtwarza te zdarzenia u siebie w tej samej kolejności. Domyślny tryb jest asynchroniczny — źródło potwierdza `COMMIT` klientowi natychmiast, nie czekając, aż którakolwiek replika faktycznie odtworzy tę zmianę u siebie. Ogólny podział trybów replikacji rozwija strona fundamentów Replikacja: synchroniczna, asynchroniczna, logiczna, fizyczna — replikacja MySQL/MariaDB oparta o binlog jest w tym podziale zawsze logiczna, nigdy fizyczna.

Wewnątrz repliki działają dwa wątki: IO thread, który łączy się ze źródłem i ściąga strumień zdarzeń binloga do lokalnego relay logu, oraz SQL thread (albo więcej wątków przy replikacji równoległej), który czyta relay log i faktycznie wykonuje zdarzenia lokalnie. Rozdzielenie na dwa wątki oznacza, że ściąganie danych ze źródła i ich aplikowanie lokalnie mogą iść w różnym tempie — dokładnie to rozjechanie mierzy `Seconds_Behind_Source` opisane niżej.



Binlog źródła płynie do IO thread repliki, ten zapisuje relay log, a SQL thread aplikuje zdarzenia jako dane repliki (kliknięcie otwiera pełny rozmiar)

Pozycja kontra GTID

Pojęcie: pozycja binloga. Klasyczny, starszy sposób identyfikacji miejsca w strumieniu replikacji — nazwa pliku binloga plus pozycja bajtowa wewnątrz niego, na przykład `my01-bin.000043:512`. Replika pamięta, gdzie skończyła, i przy ponownym połączeniu prosi źródło o kontynuację od tej dokładnej pozycji. Problem pojawia się przy zmianie źródła (failover) — pozycja z poprzedniego źródła nic nie znaczy dla nowego, bo każde źródło prowadzi własną, niezależną numerację plików.

Pojęcie: GTID (Global Transaction Identifier). Unikalny identyfikator nadawany każdej transakcji w momencie jej zatwierdzenia na źródle, w formacie `NazwaSID:numer`, gdzie `NazwaSID` to identyfikator instancji, a numer rośnie sekwencyjnie. Replika śledzi zbiór już wykonanych GTID-ów zamiast pozycji bajtowej — przy zmianie źródła nowa replika po prostu prosi o „wszystko, czego jeszcze nie mam", niezależnie od tego, z którego pliku binloga to pochodzi, co znacząco upraszcza failover. GTID w MySQL i GTID w MariaDB to dwie niekompatybilne implementacje tego samego pomysłu — replikacji między silnikami z użyciem GTID się nie łączy.

Cecha	Pozycja binloga	GTID
Identyfikacja transakcji	plik + offset bajtowy	globalnie unikalny identyfikator
Zachowanie przy failover	wymaga ręcznego ustalenia nowej pozycji	automatyczne dogranie brakujących transakcji
Kompatybilność MySQL/MariaDB	format wspólny	implementacje niekompatybilne
Konfiguracja	`CHANGE REPLICATION SOURCE TO ... SOURCE_LOG_FILE=..., SOURCE_LOG_POS=...`	`CHANGE REPLICATION SOURCE TO ... SOURCE_AUTO_POSITION=1`

Konfiguracja źródła i repliki

Źródło potrzebuje włączonego binloga (`log_bin` w `my.cnf`, opisane na stronie [my.cnf: konfiguracja serwera]) oraz unikalnego `server_id`, różnego dla każdej instancji w topologii:

```
server-id = 1
log_bin = my01-bin
```

Replika potrzebuje własnego, innego `server_id` oraz — jeśli korzystamy z GTID — włączonego `gtid_mode`:

```
server-id = 2
log_bin = my02-bin
gtid_mode = ON
enforce_gtid_consistency = ON
```

CHANGE REPLICATION SOURCE TO

> **Pojęcie: CHANGE REPLICATION SOURCE TO.** Polecenie wykonywane na replice, wskazujące, z którym źródłem ma się połączyć i jakim kontem — nazwa obowiązująca od MySQL 8.0.23 i MariaDB 10.5.1, wcześniej `CHANGE MASTER TO` (wciąż działa jako alias w nowszych wersjach, ze względu na kompatybilność wsteczną).

SHOW REPLICA STATUS i Seconds_Behind_Source

> **Pojęcie: SHOW REPLICA STATUS.** Polecenie wykonywane na replice, zwracające pełny stan obu wątków replikacji — najważniejsze pole diagnostyczne to `Seconds_Behind_Source` (do niedawna `Seconds_Behind_Master`), mówiące, o ile sekund opóźniona jest replika względem źródła.

Typowe błędy i pominięcie transakcji

Najczęstszą przyczyną zatrzymania `SQL thread` jest próba wykonania na replice polecenia, które kończy się błędem różnym od tego na źródle — na przykład `UNIQUE` naruszone przez dane wstawione ręcznie bezpośrednio na replice, czego reguła „replika jest tylko do odczytu" (`read_only = ON` w `my.cnf`) ma zapobiegać. `Replica_SQL_Running` przechodzi wtedy na `No`, a `Last_SQL_Error` opisuje dokładny błąd.

Replikacja półsynchroniczna — wspomniana

> **Pojęcie: replikacja półsynchroniczna (semi-sync).** Wariant pośredni między pełną asynchronicznością a pełną synchronicznością — źródło czeka z potwierdzeniem `COMMIT` klientowi, aż co najmniej jedna replika potwierdzi odebranie zdarzenia do swojego relay logu (niekoniecznie jego wykonanie), po czym wraca do trybu asynchronicznego, jeśli żadna replika nie potwierdzi w ustalonym czasie. Włączana wtyczkami `rpl_semi_sync_source` i `rpl_semi_sync_replica` po obu stronach — kompromis zmniejszający ryzyko utraty danych przy awarii źródła, kosztem nieco większego opóźnienia zapisów, bez pełnych gwarancji replikacji w pełni synchronicznej.

Częste błędy

Brak read_only na replice. Bez tego ustawienia nic nie stoi na przeszkodzie, żeby ktoś (albo jakaś aplikacja podłączona przez pomyłkę) zapisał dane bezpośrednio na replice — takie dane nigdy nie trafią z powrotem na źródło i przy pierwszym konflikcie z przychodzącym strumieniem zatrzymują `SQL thread`.

Ignorowanie rosnącego Seconds_Behind_Source. Stale rosnące opóźnienie oznacza, że replika nie nadąża z aplikowaniem zmian — najczęściej dlatego, że pojedynczy `SQL thread` (bez replikacji równoległej) nie jest w stanie przetworzyć strumienia równie szybko, jak wiele równoległych wątków zapisu robi to na źródle.

Powiązane: Logi binarne: archiwizacja i czyszczenie · Użytkownicy i uprawnienia · Replikacja: synchroniczna, asynchroniczna, logiczna, fizyczna

Źródła: dokumentacja MySQL 8.0, rozdział „Replication" · dokumentacja MariaDB, rozdział „Standard Replication"

Revision #41
Created 2026-09-25 00:00:40 UTC by Tomasz Konieczny
Updated 2026-10-26 14:08:28 UTC by Tomasz Konieczny