# Backup i replikacja

Logi binarne, mysqldump, backup fizyczny i replikacja.


- Logi binarne: archiwizacja i czyszczenie
- mysqldump i backup logiczny
- Backup fizyczny: Percona XtraBackup i mariabackup
- Replikacja w MySQL i MariaDB

# Logi binarne: archiwizacja i czyszczenie

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

*Binlog nie jest logiem błędów ani logiem zapytań — jest zapisem każdej zmiany danych, na którym stoi zarówno replikacja, jak i odtwarzanie do punktu w czasie, i który domyślnie serwer wcale nie musi prowadzić.*

**Skróty użyte na tej stronie (3)**

| Skrót | Co znaczy |
|---|---|
| **CLI** | Command Line Interface: praca przez wpisywanie komend w terminalu |
| **PITR** | Point-In-Time Recovery: odtworzenie bazy do dowolnej chwili, np. 14:31 |
| **SQL** | Structured Query Language: język zapytań do relacyjnych baz danych |

# Fiszka

**TYLKO ODCZYT**

## Wylistowanie plików binloga: SHOW BINARY LOGS

```
SHOW BINARY LOGS;
```

Przykład:

```
+------------------+-----------+
| Log_name         | File_size |
+------------------+-----------+
| my01-bin.000041 | 104857802 |
| my01-bin.000042 | 104857600 |
| my01-bin.000043 |  18402913 |
```

```
+------------------+-----------+
3 rows in set (0.00 sec)
```

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

**TYLKO ODCZYT**

## Odczytanie surowej zawartości pliku binloga: SHOW BINLOG EVENTS

```
SHOW BINLOG EVENTS IN 'my01-bin.000043';
```

Przykład:

```
+--------------------+-----+------------+-----------+-------------
+---------------------------------------+
| Log_name           | Pos | Event_type | Server_id | End_log_pos |
Info                                    |
+--------------------+-----+------------+-----------+-------------
+---------------------------------------+
| my01-bin.000043 |   4 | Format_desc|         1 |         126 | Server ver: 8.0.35,
Binlog ver: 4      |
| my01-bin.000043 | 126 | Query      |         1 |         234 |
BEGIN                                   |
| my01-bin.000043 | 234 | Table_map  |         1 |         298 | table_id: 112
(produkcja.zamowienia)   |
| my01-bin.000043 | 298 | Update_rows|         1 |         512 | table_id: 112 flags:
STMT_END_F        |
+--------------------+-----+------------+-----------+-------------
+---------------------------------------+
4 rows in set (0.00 sec)
```

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

**TYLKO ODCZYT**

## Usunięcie logów starszych niż wskazana data: PURGE BINARY LOGS BEFORE

```
PURGE BINARY LOGS BEFORE NOW() - INTERVAL 3 DAY;
```

Z poziomu CLI SQL.

□□ **CLI** = Command Line Interface: praca przez wpisywanie komend w terminalu · **SQL** = Structured Query Language: język zapytań do relacyjnych baz danych

□□ TYLKO ODCZYT

## Usunięcie logów starszych niż wskazany plik: PURGE BINARY LOGS TO

```
PURGE BINARY LOGS TO 'my01-bin.000040';
```

Wariant z nazwą pliku zamiast daty.

□□ ZMIENIA SYSTEM

## Konwersja binloga na czytelny SQL: mysqlbinlog

```
mysqlbinlog SciezkaDoPliku > SciezkaDoPliku.sql
```

□□ **SQL** = Structured Query Language: język zapytań do relacyjnych baz danych · **mysqlbinlog** = narzędzie konwertujące binarną zawartość pliku binloga MySQL/MariaDB na czytelny SQL

□□ ZMIENIA SYSTEM

## Odtworzenie do punktu w czasie po zakresie dat: mysqlbinlog --start-datetime

```
mysqlbinlog --start-datetime="2026-03-14 02:00:00" --stop-datetime="2026-03-14 09:41:00"
my01-bin.000043 | mysql -u NazwaUsera -p
```

Wariant z zakresem czasowym, typowy przy PITR po przywróceniu backupu pełnego z nocy — odtwarzamy wszystko, co wydarzyło się między backupem a chwilą tuż przed awarią.

□□ **PITR** = Point-In-Time Recovery: odtworzenie bazy do dowolnej chwili, np. 14:31 · **mysqlbinlog** = narzędzie konwertujące binarną zawartość pliku binloga MySQL/MariaDB na czytelny SQL

**ZMIENIA SYSTEM**

### Odtworzenie do punktu w czasie po pozycji bajtowej: mysqlbinlog --stop-position

```
mysqlbinlog --stop-position=298 my01-bin.000043 | mysql -u NazwaUsera -p
```

Wariant z pozycją bajtową zamiast czasu, precyzyjniejszy, gdy znamy dokładną pozycję z `SHOW BINLOG EVENTS` (na przykład tuż przed błędnym `DELETE` bez `WHERE`).

**mysqlbinlog** = narzędzie konwertujące binarną zawartość pliku binloga MySQL/MariaDB na czytelny SQL

# Z życia wzięte

**Z ŻYCIA WZIĘTE**

Ktoś z zespołu Oli uruchamia na my01 `UPDATE` bez `WHERE` na tabeli cen w `sklep_b2b` tuż przed piętnastą. Odtwarzamy backup pełny sprzed nocy, a potem `mysqlbinlog` z zakresem czasu aż do minuty tuż przed błędnym poleceniem, żeby nie stracić zamówień złożonych rano. Cennik wraca do stanu sprzed pomyłki, a zamówienia z całego dnia zostają nietknięte.

# Czym jest binlog

**Pojęcie: binary log (binlog).** Sekwencyjny zapis wszystkich poleceń zmieniających dane — `INSERT`, `UPDATE`, `DELETE`, `CREATE`, `DROP` — utrwalany w osobnych plikach, niezależnie od tego, czy transakcja została jeszcze zatwierdzona. Binlog jest podstawą dwóch mechanizmów: replikacji, opisanej na stronie Replikacja, gdzie replika po prostu odtwarza u siebie strumień zdarzeń przeczytany z binloga źródła, oraz odtwarzania do punktu w czasie (PITR), gdzie po przywróceniu backupu pełnego odtwarzamy zdarzenia z binloga aż do wybranego momentu tuż przed awarią.

W przeciwieństwie do redo logu InnoDB, binlog nie jest domyślnie włączony na wszystkich instalacjach — MySQL i MariaDB uruchamiają się bez niego, dopóki nie ustawimy `log_bin` w `my.cnf`. Bez binloga replikacja jest niemożliwa, a odtworzenie da się cofnąć wyłącznie do momentu ostatniego backupu pełnego, bez żadnej granulacji pomiędzy.

# Binlog kontra redo log InnoDB

Oba logi zapisują zmiany, oba są sekwencyjne, i to jedyne, co mają wspólnego — mylenie ich jest częstym błędem osoby przychodzącej z innego silnika, gdzie jeden log robi obie te role naraz.

| Cecha | Binlog | Redo log InnoDB |
|---|---|---|
| Poziom | logiczny — zapisane polecenie albo wiersze zmian | fizyczny — zmiana konkretnej strony na dysku |
| Zależność od silnika | niezależny od silnika składowania | wyłącznie InnoDB |
| Cel | replikacja, PITR, audyt zmian | crash recovery samego InnoDB |
| Domyślnie włączony | nie (do włączenia w `my.cnf`) | tak, zawsze |
| Cykliczność | rotacja przy rozmiarze/restarcie, kasowany dopiero po `PURGE` albo wygaśnięciu | nadpisywany cyklicznie po każdym checkpoincie |
| Możliwość odtworzenia na innej wersji silnika | tak, format logiczny | nie, format wewnętrzny InnoDB |

# Formaty ROW, STATEMENT, MIXED

> **Pojęcie: format binloga.** Sposób, w jaki zmiana trafia do logu — deklarujemy parametrem `binlog_format` w `my.cnf`.

**STATEMENT** zapisuje samo polecenie SQL tak, jak zostało wykonane — `UPDATE zamowienia SET status = 'wyslane' WHERE data < '2026-01-01'`. Log jest kompaktowy, ale niebezpieczny przy funkcjach niedeterministycznych (`NOW()`, `RAND()`, `UUID()`), bo replika wykonując to samo polecenie może dostać inny wynik niż źródło.

**ROW** zapisuje nie polecenie, tylko faktyczne zmienione wiersze — który wiersz, jakie wartości przed i po zmianie. Bezpieczny niezależnie od użytych funkcji, kosztem większego rozmiaru logu przy operacjach masowych (`UPDATE` na milionie wierszy zapisuje milion wpisów, nie jedno polecenie). Domyślny od MySQL 5.7.7.

**MIXED** zaczyna od `STATEMENT` i automatycznie przełącza się na `ROW` dla poleceń rozpoznanych jako niedeterministyczne — kompromis, dziś rzadziej wybierany niż samo `ROW`, bo różnica w rozmiarze logu przestała być istotnym problemem przy dzisiejszych pojemnościach dysków.

| Format | Rozmiar logu | Bezpieczeństwo replikacji | Domyślny od |
|---|---|---|---|

| | | | |
|---|---|---|---|
| STATEMENT | najmniejszy | ryzykowny przy funkcjach niedeterministycznych | — |
| ROW | największy | bezpieczny zawsze | MySQL 5.7.7 |
| MIXED | pośredni | bezpieczny, przełącza się automatycznie | — |

# Ustawienia archiwizacji logów transakcyjnych

Binlogi z automatu nie odkładają się na dysk, dopóki nie włączymy tego w pliku konfiguracyjnym `my.cnf` / `my.ini`. W pliku wyszukujemy sekcję:

```
# ***** Group Replication Related *****
# Specifies the base name to use for binary log files. With binary logging
# enabled, the server logs all statements that change data to the binary
# log, which is used for backup and replication.
```

Usuwamy znak komentarza `#` przed sekcją albo dopisujemy poniższe parametry:

```
binlog-format = ROW
expire-logs-days = 2
log-bin="SciezkaDoFolderuLogow"
```

`log-bin` włącza binlog i ustawia ścieżkę oraz prefiks nazwy plików, `binlog-format` ustawia format opisany wyżej, `expire-logs-days` ustawia automatyczne usuwanie starych plików po dwóch dniach — parametr przestarzały od MySQL 8.0 na rzecz `binlog_expire_logs_seconds` opisanego niżej. Zmiana `log-bin` wymaga restartu serwera.

# SHOW BINARY LOGS i SHOW BINLOG EVENTS

> **Pojęcie: SHOW BINARY LOGS.** Zwraca listę aktualnie istniejących plików binloga wraz z ich rozmiarem.

> **Pojęcie: SHOW BINLOG EVENTS.** Zwraca surową zawartość pojedynczego pliku binloga w formie czytelnej dla człowieka — pozycje bajtowe zdarzeń i ich typy, przydatne przy ręcznym znajdowaniu punktu, do którego chcemy odtwarzać.

# Purge logów starszych niż 3 dni

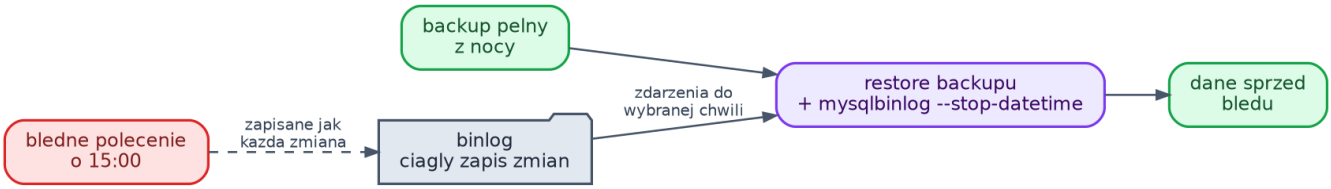
> **Pojęcie: PURGE BINARY LOGS.** Trwale usuwa pliki binloga starsze niż wskazana data ( `BEFORE` ) albo wszystkie starsze niż wskazany plik ( `TO` ), zarówno z dysku, jak i z pliku indeksu `.index` , w którym serwer trzyma listę aktywnych plików.

Automatyczne odpowiedniki tego ręcznego czyszczenia to parametry `expire-logs-days` (do 8.0) albo `binlog_expire_logs_seconds` (od 8.0, jednostka sekundowa, domyślnie 30 dni) — ustawione w `my.cnf` , zwalniają z ręcznego wykonywania `PURGE` po każdym backupie. Ręczny `PURGE` pozostaje przydatny wtedy, gdy trzeba natychmiast odzyskać miejsce na dysku, nie czekając na automatyczne wygaśnięcie.

Czyszczenie logu, który jest jeszcze potrzebny replice niepodłączonej od dłuższego czasu, kończy się tym, że replika po powrocie nie może dogonić źródła — brakującego pliku binloga nie da się już odzyskać, a jedynym wyjściem zostaje ponowne zbudowanie repliki od pełnego backupu.

# mysqlbinlog — odczyt i PITR

> **Pojęcie: mysqlbinlog.** Narzędzie CLI konwertujące binarną zawartość pliku binloga na czytelny SQL, gotowy albo do przeczytania okiem, albo do wykonania klientem `mysql` przy odtwarzaniu do punktu w czasie.




*Backup pełny z nocy plus binlog aż do wybranej chwili odtwarzają dane sprzed błędnego polecenia (kliknięcie otwiera pełny rozmiar)*

# Częste błędy

**Włączenie binloga bez ustawienia wygasania.** `log-bin` bez `expire-logs-days`/`binlog_expire_logs_seconds` i bez ręcznego `PURGE` w harmonogramie oznacza katalog logów rosnący bez końca, aż do zapełnienia dysku — to samo zdarzenie, które kończy się awarią całej instancji, bo `mysqld` nie potrafi zapisać kolejnej zmiany.

**PURGE logów wciąż potrzebnych replice.** Usunięcie pliku binloga, którego pozycja replikacji jeszcze nie minęła u którejkolwiek podłączonej repliki, zrywa replikację nieodwracalnie — przed czyszczeniem ręcznym warto sprawdzić pozycję najwolniejszej repliki przez `SHOW REPLICA STATUS` opisane na stronie Replikacja.

**Powiązane:** Replikacja · mysqldump i backup logiczny · Log transakcyjny: WAL, redo, transaction log

**Źródła:** dokumentacja MySQL 8.0, rozdział „The Binary Log" i „mysqlbinlog — Utility for Processing Binary Log Files" · procedura autora

# mysqldump i backup logiczny

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

*`mysqldump` bierze dane przez zwykłe połączenie SQL, nie przez kopiowanie plików — wynikiem jest zestaw poleceń `CREATE`/`INSERT`, który odtwarza się dowolnym klientem `mysql`, nawet na innej wersji silnika.*

### Skróty użyte na tej stronie (3)

| Skrót | Co znaczy |
|---|---|
| **B2B** | business to business: sprzedaż i systemy skierowane do innych firm, nie do klientów indywidualnych |
| **MVCC** | Multi-Version Concurrency Control: baza trzyma kilka wersji wiersza, żeby czytający nie blokował piszącego |
| **SQL** | Structured Query Language: język zapytań do relacyjnych baz danych |

# Fiszka

**OSTROŻNIE · NIEODWRACALNE**

## Utworzenie skryptu backupu wszystkich baz: NazwaSkryptu.sh

```
#!/bin/bash

USER="NazwaUseraBackupu"
PASSWORD="HasloBackup"
#OUTPUT="SciezkaZapisuPlikow"

#Ewentualne usuwanie starych backupów
#rm "$OUTPUTDIR/*gz" > /dev/null 2>&1
```

```
databases=`mysql -u $USER -p$PASSWORD -e "SHOW DATABASES;" | tr -d "| " | grep -v
Database`

for db in $databases; do
    if [[ "$db" != "information_schema" ]] && [[ "$db" != "performance_schema" ]] && [[
"$db" != "mysql" ]] && [[ "$db" != _* ]] ; then
        echo "Dumping database: $db"
        mysqldump -u $USER -p$PASSWORD --databases $db > `date +%Y%m%d`.$db.sql

       # Ewentualne pakowanie plików SQL
       # gzip $OUTPUT/`date +%Y%m%d`.$db.sql
    fi
done
```

Zamiast jednego pliku ze wszystkim naraz, na potrzeby backupu produkcyjnego wygodniejszy jest jeden plik na bazę — łatwiej odtworzyć pojedynczą bazę bez dotykania reszty, i łatwiej zorientować się w rozmiarze każdej z nich osobno. Poniższy skrypt robi dokładnie to: pyta serwer o listę baz i zrzuca każdą po kolei, pomijając bazy systemowe. Tworzymy plik `NazwaSkryptu.sh`. Konto `$USER` powinno być kontem dedykowanym pod backup, opisanym na stronie Użytkownicy i uprawnienia, nie kontem `root`.

**echo** = od angielskiego echo: wypisuje podany tekst na standardowe wyjście, często łączone z przekierowaniem do pliku · **mysqldump** = zrzut bazy MySQL do pliku

**ZMIENIA SYSTEM**

## Nadanie prawa wykonywania skryptowi: chmod +x

```
chmod +x NazwaSkryptu.sh
```

Nadajemy uprawnienia do uruchamiania (atrybut `x`) dla pliku.

**chmod** = change mode: zmień prawa dostępu

**TYLKO ODCZYT**

## Uruchomienie skryptu backupu: ./NazwaSkryptu.sh

```
./NazwaSkryptu.sh
```

Wykonujemy poprzez to polecenie.

Przykład:

```
Dumping database: produkcja
Dumping database: zamowienia_archiwum
```

Skrypt w obecnej formie pomija dodatkowo bazę `sys` tylko pośrednio — trafia w warunek `_*` tylko wtedy, gdy lokalna instalacja nazwała ją z podkreślnikiem; na standardowej instalacji `sys` warto dopisać jawnie do listy wykluczeń, jeśli backup ma pomijać także ją. Można też dodać skrypt do `crontab`, żeby wykonywał się automatycznie o stałej porze.

**□□ ZMIENIA SYSTEM**

## Zrzut jednej bazy: mysqldump

```
mysqldump -h NazwaSerwera -u NazwaUsera -p --single-transaction --routines --triggers
NazwaBazy > SciezkaDoPliku
```

Zrzucamy jedną wskazaną bazę ze strukturą, danymi, procedurami i triggerami, w spójnym stanie dzięki `--single-transaction`.

Przykład:

```
mysqldump -h my01 -u nazwa_usera -p --single-transaction --routines --triggers produkcja
> /home/nazwa_usera/produkcja.sql
```

Polecenie przy sukcesie nic nie wypisuje na ekran — cały wynik trafia do wskazanego pliku.

□□ **mysqldump** = zrzut bazy MySQL do pliku

**□□ ZMIENIA SYSTEM**

## Zrzut jednej tabeli: mysqldump

```
mysqldump -h NazwaSerwera -u NazwaUsera -p NazwaBazy NazwaTabeli > SciezkaDoPliku
```

Zrzucamy strukturę i dane dokładnie jednej tabeli, bez reszty bazy — wygodne przed ryzykowną operacją na tej jednej tabeli.

Przykład:

```
mysqldump -h my01 -u nazwa_usera -p produkcja zamowienia >
/home/nazwa_usera/zamowienia.sql
```

**mysqldump** = zrzut bazy MySQL do pliku

**ZMIENIA SYSTEM**

### Zrzut samej struktury, bez danych: mysqldump --no-data

```
mysqldump --no-data -h NazwaSerwera -u NazwaUsera -p NazwaBazy > SciezkaDoPliku
```

Zrzucamy wyłącznie definicje tabel, bez ani jednego wiersza danych — przydatne przy przygotowaniu pustej kopii bazy pod środowisko testowe.

Przykład:

```
-- MySQL dump 10.13  Distrib 8.0.35
-- Host: my01    Database: produkcja
```

**mysqldump** = zrzut bazy MySQL do pliku

**TYLKO ODCZYT**

### Odtworzenie zrzutu: mysql < plik

```
mysql -h NazwaSerwera -u NazwaUsera -p NazwaBazy < SciezkaDoPliku
```

Odtwarzamy zrzut tekstowy zwykłym klientem `mysql`, tak jakby jego treść wpisano ręcznie w sesji — ten sam mechanizm co metakomenda `SOURCE` opisana na stronie Klient mysql i połączenie. Baza docelowa musi już istnieć, `mysql` sam jej nie tworzy, chyba że zrzut wykonano z `--databases`, który zawiera własne `CREATE DATABASE`.

Przykład:

```
mysql -h my01 -u nazwa_usera -p produkcja < /home/nazwa_usera/produkcja.sql
```

# Z życia wzięte

**Z ŻYCIA WZIĘTE**

Skrypt backupu `sklep_b2b` na my01 działa co noc od lat, ale nikt nie sprawdzał, czy obejmuje też procedury składowane. Po awarii dysku i odtworzeniu zrzutu aplikacja sklepu B2B zgłasza błąd o brakującej procedurze liczącej rabaty hurtowe. Poprawiamy skrypt, dopisując flagę `--routines`, i robimy backup ponownie, zanim ktokolwiek zdąży tego potrzebować na produkcji.

# Backup logiczny a backup fizyczny

**Pojęcie: backup logiczny.** Zrzut danych w postaci poleceń SQL, wykonany przez odpytanie działającej instancji, a nie przez skopiowanie plików z dysku — w odróżnieniu od backupu fizycznego, jakim zajmuje się Backup fizyczny: Percona XtraBackup i mariabackup, backup logiczny nie wymaga zatrzymania serwera ani identycznej wersji silnika przy odtwarzaniu, ale jest wolniejszy przy dużych bazach, bo dane trzeba faktycznie odczytać i przepisać na SQL, zamiast skopiować bajt w bajt. Ogólny podział backupów rozwija strona Backup pełny, różnicowy, przyrostowy, logi — `mysqldump` w tym podziale jest zawsze backupem pełnym, nigdy przyrostowym.

# Kluczowe flagi mysqldump

**--single-transaction** — otwiera jedną transakcję z poziomem izolacji `REPEATABLE READ` na cały czas trwania zrzutu, korzystając z MVCC InnoDB, żeby uzyskać spójny obraz danych bez blokowania zapisów na tabelach w trakcie backupu. Działa wyłącznie dla tabel InnoDB — na MyISAM nie ma efektu, bo ten silnik nie zna transakcji.

**--routines**, --triggers, --events — dołączają do zrzutu odpowiednio procedury składowane i funkcje, triggery oraz zdarzenia zaplanowane (`EVENT`), które domyślnie `mysqldump` pomija poza triggerami (te są domyślnie włączone) — łatwo o sytuację, w której zrzut „wszystkich baz" nie zawiera ani jednej procedury składowanej, bo `--routines` trzeba podać jawnie.

**--master-data** — dopisuje do zrzutu komentarz z pozycją binloga source'a w chwili wykonania zrzutu, gotowy do użycia przy stawianiu nowej repliki opisanej na stronie Replikacja. Wartość `2` zapisuje pozycję jako komentarz SQL (bezpieczniejsze, bo nieaktywne przy zwykłym imporcie), wartość `1` jako aktywne polecenie `CHANGE REPLICATION SOURCE TO`.

**--all-databases** — zrzuca wszystkie bazy instancji naraz do jednego pliku, łącznie z bazą `mysql` zawierającą konta — podejście wygodne, ale mniej elastyczne niż zrzut każdej bazy osobno, bo

odtworzenie pojedynczej bazy ze wspólnego pliku wymaga ręcznego wycięcia jej fragmentu.

# mysqlpump i mydumper — wspomniane

> **Pojęcie: mysqlpump.** Narzędzie dostarczane przez Oracle od MySQL 5.7.8 jako potencjalny następca `mysqldump` — potrafi zrzucać obiekty równolegle wieloma wątkami, co skraca czas backupu dużych instancji, ale nie jest rozwijane tak aktywnie jak `mysqldump` i nie obsługuje `--single-transaction` w pełni tak samo bezproblemowo. W praktyce rzadko wybierany jako domyślne narzędzie backupu.

> **Pojęcie: mydumper.** Zewnętrzne narzędzie open source (nie wchodzi w skład dystrybucji MySQL/MariaDB), zrzucające dane wieloma wątkami równolegle do osobnych plików na tabelę, z odpowiadającym mu `myloader` do odtwarzania — realny wybór przy bazach na tyle dużych, że jednowątkowy `mysqldump` przestaje mieścić się w oknie backupowym.

# Dlaczego --lock-all-tables boli na produkcji

> **Pojęcie: --lock-all-tables.** Flaga wymuszająca globalną blokadę odczytu (`FLUSH TABLES WITH READ LOCK`) na wszystkich tabelach wszystkich baz na czas trwania całego zrzutu — jedyny sposób na spójność między tabelami dla silników bez MVCC (MyISAM), ale na instancji z ruchem produkcyjnym oznacza zamrożenie każdego zapisu do dowolnej bazy na cały czas trwania dumpa, nie tylko tej jednej zrzucanej. Backup bazy ważącej kilkadziesiąt gigabajtów potrafi trwać kilkanaście minut — tyle samo czasu aplikacja produkcyjna stoi z zawieszonymi zapisami. Dla tabel InnoDB `--single-transaction` daje tę samą spójność bez blokowania niczego, dlatego na instalacjach czysto InnoDB `--lock-all-tables` nie ma uzasadnienia.

# Częste błędy

**Zrzut bez --single-transaction na obciążonej bazie InnoDB.** Bez tej flagi `mysqldump` domyślnie i tak stara się o spójność przez blokady tabelowe, co na dużej, aktywnej bazie InnoDB oznacza niepotrzebne blokowanie zapisów przez cały czas trwania zrzutu, mimo że MVCC pozwoliłoby tego uniknąć.

**Pominięcie --routines przy migracji na nowy serwer.** Zrzut bez tej flagi wygląda na kompletny — wszystkie tabele i dane są na miejscu — dopóki aplikacja nie wywoła procedury składowanej, której po prostu nie ma, bo backup jej nie objął.

**Powiązane:** Użytkownicy i uprawnienia · Backup fizyczny: Percona XtraBackup i mariabackup · Klient mysql i połączenie · Backup pełny, różnicowy, przyrostowy, logi

**Źródła:** dokumentacja MySQL 8.0, rozdział „mysqldump — A Database Backup Program" · procedura autora (skrypt backupu wszystkich baz)

# Backup fizyczny: Percona XtraBackup i mariabackup

**Poziom: średni** **Dotyczy: MySQL 5.7–8.0, MariaDB 10.x–11.x** **Czas: ~20 min**

*Tam, gdzie `mysqldump` przestaje mieścić się w oknie backupowym, wchodzi backup fizyczny — kopiowanie plików InnoDB bajt w bajt, bez odczytywania i przepisywania danych na SQL.*

**Skróty użyte na tej stronie (3)**

| Skrót | Co znaczy |
|---|---|
| **LSN** | Log Sequence Number: rosnący numer pozycji w dzienniku transakcyjnym, po którym silnik rozpoznaje kolejność zmian |
| **RTO** | Recovery Time Objective: jak długo wolno nie działać przy awarii |
| **SQL** | Structured Query Language: język zapytań do relacyjnych baz danych |

# Fiszka

**TYLKO ODCZYT**

## Wykonanie backupu gorącego: xtrabackup --backup

```
xtrabackup --backup --target-dir=SciezkaDoKatalogu --user=NazwaUsera --
password=HasloUzytkownika
```

Faza kopiowania — narzędzie kopiuje pliki InnoDB, śledząc jednocześnie redo log, żeby wychwycić zmiany zachodzące w trakcie samego kopiowania.

**TYLKO ODCZYT**

## Przygotowanie kopii do uruchomienia: xtrabackup --prepare

```
xtrabackup --prepare --target-dir=SciezkaDoKatalogu
```

Faza przygotowania — stosuje do skopiowanych plików zmiany zebrane z redo logu w trakcie kopiowania, doprowadzając kopię do stanu spójnego, gotowego do uruchomienia, tak jakby to był normalnie zamknięty katalog danych.

**□□ TYLKO ODCZYT**

## Odtworzenie kopii do katalogu danych: xtrabackup --copy-back

```
xtrabackup --copy-back --target-dir=SciezkaDoKatalogu --datadir=SciezkaDoDatadir
```

Faza odtworzenia — kopiuje przygotowane pliki z powrotem do docelowego `datadir`, który musi być pusty.

**□□ ZMIENIA SYSTEM**

## Przywrócenie właściciela plików po odtworzeniu: chown

```
chown -R mysql:mysql SciezkaDoDatadir
```

Po `--copy-back` katalog danych ma właściciela `root` (bo skopiowany przez proces uruchomiony z odpowiednimi uprawnieniami) — przed startem `mysqld` konieczna jest zmiana właściciela z powrotem na konto systemowe `mysql`.

□□ **chown** = change owner: zmień właściciela

**□□ TYLKO ODCZYT**

## Wykonanie backupu przyrostowego: xtrabackup --backup --incremental-basedir

```
xtrabackup --backup --target-dir=SciezkaDoKatalogu --incremental-
basedir=SciezkaDoBackupuBazowego
```

**□□ TYLKO ODCZYT**

### Przygotowanie łańcucha backupu przyrostowego: xtrabackup --prepare --apply-log-only

```
xtrabackup --prepare --apply-log-only --target-dir=SciezkaDoBackupuBazowego
xtrabackup --prepare --apply-log-only --target-dir=SciezkaDoBackupuBazowego --
incremental-dir=SciezkaDoPrzyrostu1
xtrabackup --prepare --target-dir=SciezkaDoBackupuBazowego --incremental-
dir=SciezkaDoPrzyrostu2
```

Odtworzenie backupu przyrostowego wymaga przygotowania każdego backupu w kolejności chronologicznej, zaczynając od bazowego, z flagą `--incremental-dir` wskazującą kolejny przyrost nakładany na już przygotowany backup bazowy. Flaga `--apply-log-only` na wszystkich krokach poza ostatnim zatrzymuje przygotowanie tuż przed cofnięciem niezatwierdzonych transakcji (`rollback`) — cofnięcie wykonane przedwcześnie uniemożliwiłoby poprawne nałożenie kolejnego przyrostu.

**□□ TYLKO ODCZYT**

### Wykonanie pełnego cyklu backupu w MariaDB: mariabackup

```
mariabackup --backup --target-dir=SciezkaDoKatalogu --user=NazwaUsera --
password=HasloUzytkownika
mariabackup --prepare --target-dir=SciezkaDoKatalogu
mariabackup --copy-back --target-dir=SciezkaDoKatalogu
```

`xtrabackup` oficjalnie wspiera tylko MySQL i Percona Server, nie MariaDB — próba użycia `xtrabackup` na instalacji MariaDB kończy się niekompatybilnością formatu metadanych na nowszych wersjach obu silników, dlatego wybór narzędzia idzie zawsze zgodnie z silnikiem, nie na przemian.

□□ **mariabackup** = narzędzie kopii zapasowej MariaDB, które robi spójną kopię bazy bez zatrzymywania serwera; używa go skrypt kopii MailCow

# Z życia wzięte

**□□ Z ŻYCIA WZIĘTE**

Zrzut `mysqldump` całej bazy `sklep_b2b` w listopadzie trwa już ponad dwie godziny i zaczyna kolidować z oknem nocnym magazynu. Przechodzimy na `mariabackup`, które kopiuje pliki InnoDB bez blokowania zapisów aplikacji sklepowej. Pierwsze odtworzenie testowe kończy się błędem, bo ktoś pominął fazę `--prepare` przed `--copy-back`, więc poprawiamy procedurę i testujemy jeszcze raz.

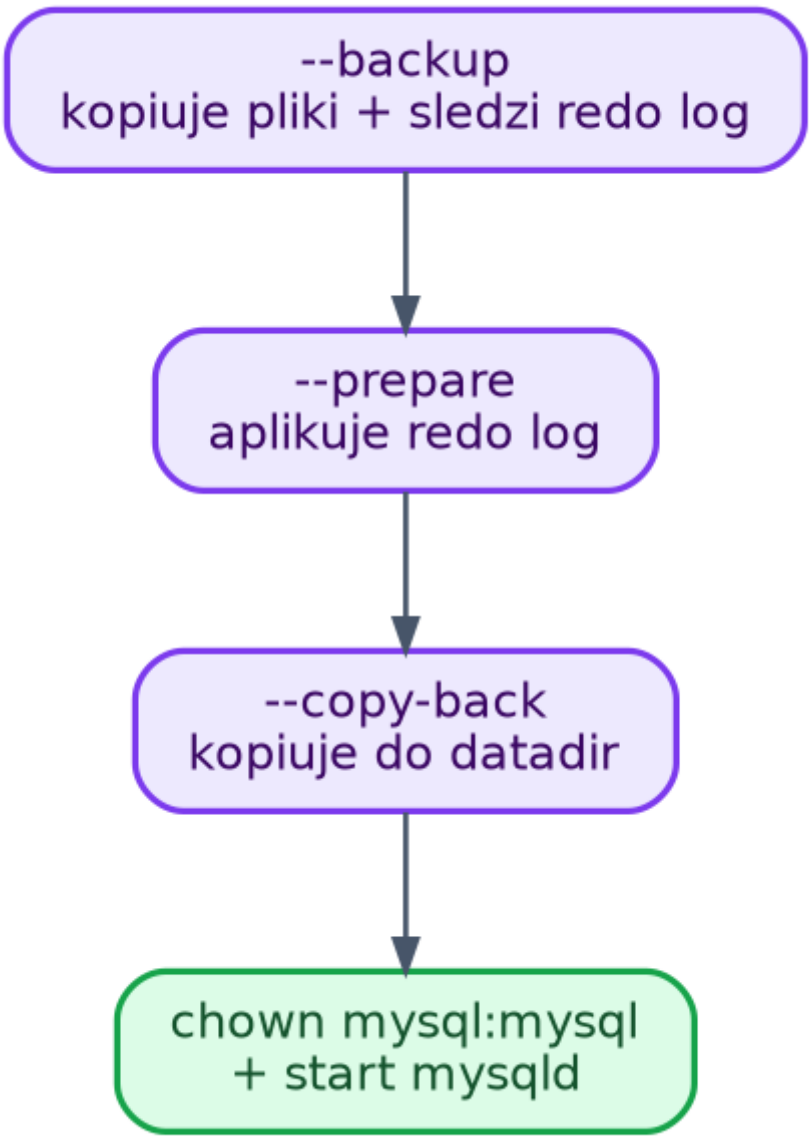
# Czym jest backup fizyczny gorący

**Pojęcie: backup fizyczny.** Kopia plików tworzących bazę danych na poziomie systemu plików — pliki `.ibd`, `ibdata1`, redo log — zamiast zestawu poleceń SQL jak przy backupie logicznym opisanym na stronie mysqldump i backup logiczny. Odtworzenie polega na skopiowaniu tych plików z powrotem na miejsce, bez ponownego budowania indeksów i bez wykonywania pojedynczo milionów poleceń `INSERT` — dlatego backup fizyczny dużej bazy jest zwykle wielokrotnie szybszy do odtworzenia niż logiczny.

**Pojęcie: backup gorący (hot backup).** Backup fizyczny wykonany na działającej instancji, bez zatrzymywania serwera i bez blokowania zapisów na czas kopiowania — w odróżnieniu od prostego `cp -r` katalogu danych, które wymagałoby zatrzymania `mysqld`, żeby uniknąć skopiowania niespójnego stanu plików w trakcie zapisu. Zarówno `xtrabackup`, jak i `mariabackup` są narzędziami do backupu gorącego.

# xtrabackup --backup, --prepare, --copy-back

**Pojęcie: Percona XtraBackup.** Darmowe narzędzie firmy Percona do backupu gorącego instancji MySQL (oraz Percona Server), działające w trzech fazach rozdzielonych w czasie.




*Faza backup kopiuje pliki i sledzi redo log, prepare aplikuje redo log, copy-back kopiuje przygotowane pliki do datadir (kliknięcie otwiera pełny rozmiar)*

# Backup przyrostowy

> **Pojęcie: backup przyrostowy w XtraBackup.** Backup zawierający wyłącznie strony danych zmienione od poprzedniego backupu (pełnego albo innego przyrostowego) — mechanizm szerzej opisany na stronie Backup pełny, różnicowy, przyrostowy, logi. XtraBackup rozpoznaje zmienione strony po numerze LSN (Log Sequence Number)

zapisanym w każdej stronie InnoDB, porównując go z LSN zapamiętanym przy backupie bazowym.

# mariabackup — odpowiednik dla MariaDB

**Pojęcie: mariabackup.** Fork `xtrabackup` utrzymywany przez MariaDB Foundation i dołączony domyślnie do pakietów serwera MariaDB (od 10.1) — składnia poleceń jest praktycznie identyczna z `xtrabackup` (`--backup`, `--prepare`, `--copy-back`), co jest zamierzone, żeby migracja procedur backupowych między silnikami wymagała minimalnych zmian.

# Porównanie z mysqldump

| Cecha | mysqldump (logiczny) | xtrabackup / mariabackup (fizyczny) |
|---|---|---|
| Czas backupu dużej bazy | rośnie proporcjonalnie do liczby wierszy | rośnie proporcjonalnie do rozmiaru plików na dysku, zwykle szybszy |
| Czas odtworzenia | wolny — ponowne wykonanie każdego INSERT i budowa indeksów | szybki — kopiowanie plików, indeksy już gotowe |
| Rozmiar backupu | mniejszy, tekst SQL kompresuje się dobrze | większy, surowe pliki binarne |
| Wymagana wersja silnika przy odtworzeniu | dowolna nowsza | zgodna wersja i architektura silnika |
| Backup przyrostowy | nie istnieje | tak, oparty na LSN |
| Backup pojedynczej tabeli | tak, natywnie | ograniczony (`--tables-file`, wymaga `innodb_file_per_table`) |
| Obciążenie serwera w trakcie backupu | odczyt przez SQL, obciąża optymalizator i bufor | odczyt plików na poziomie systemu plików, mniejszy narzut na sam silnik |

Backup logiczny wygrywa tam, gdzie liczy się przenośność (inna wersja silnika, inny system operacyjny, potrzeba wglądu w treść zrzutu) i mały rozmiar produkcji dziennej. Backup fizyczny wygrywa tam, gdzie liczy się RTO — czas potrzebny na przywrócenie działania po awarii — bo

odtworzenie kilkusetgigabajtowej bazy z plików trwa minuty, a z zrzutu logicznego potrafi trwać godziny.

# Częste błędy

**Pominięcie fazy --prepare przed --copy-back.** Skopiowane, ale nieprzygotowane pliki są w stanie niespójnym — próba uruchomienia `mysqld` na takim katalogu kończy się błędem podczas odzyskiwania po awarii albo, gorzej, cichym uszkodzeniem danych.

**Mieszanie xtrabackup i mariabackup między silnikami.** Format wewnętrznych metadanych backupu obu narzędzi rozjechał się na tyle, że backup wykonany jednym narzędziem nie zawsze da się poprawnie przygotować drugim — wybór narzędzia idzie zawsze zgodnie z silnikiem docelowym.

**Powiązane:** mysqldump i backup logiczny · Architektura MySQL i MariaDB: silniki składowania, pliki, procesy · Backup pełny, różnicowy, przyrostowy, logi

**Źródła:** dokumentacja Percona XtraBackup, rozdział „The Backup Cycle" · dokumentacja MariaDB, rozdział „Mariabackup Overview"

# 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

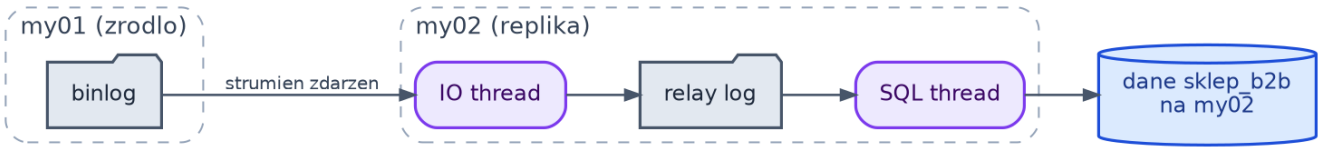
**Z ŻYCIA WZIĘTE**

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"