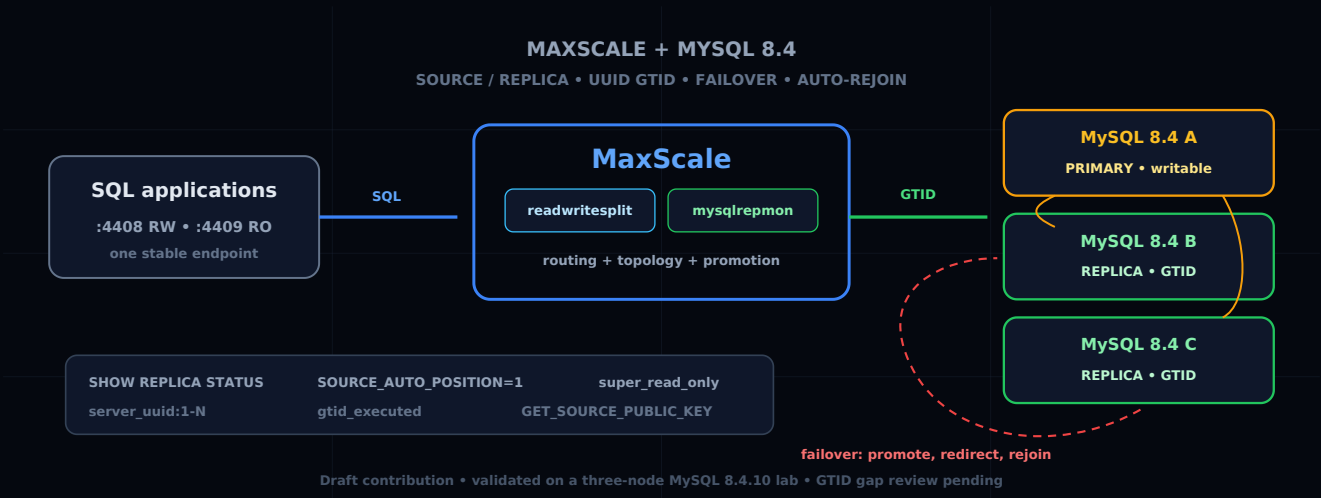
MaxScale i MySQL 8.4: failover GTID zgodny z terminologią Source/Replica

Aurélien LEQUOY · July 27, 2026

MAXSCALE MYSQL MYSQL-8.4 REPLICATION GTID FAILOVER HIGH-AVAILABILITY PROXY



> ***Stan projektu — 27 lipca 2026 r.*** *Upstream PR #421 proponuje dla MaxScale moduł* `mysqlrepmon` *opisany w tym artykule. PR celowo pozostaje wersją draft: kod został skompilowany i przetestowany w laboratorium MySQL 8.4.10, ale nie znajduje się jeszcze w oficjalnym wydaniu MaxScale. Nie należy przedstawiać go jako funkcji produkcyjnej wspieranej przez producenta.*

Problem w jednym zdaniu

MaxScale historycznie monitoruje topologie replikacji MariaDB / MySQL za pomocą `mariadbmon`. MySQL 8.4 usunął jednak dawne polecenia `SLAVE` / `MASTER`, używa GTID opartych na UUID i wymaga innej składni do rekonfiguracji replikacji.

Skutek może być mylący: proxy nadal przyjmuje połączenia SQL, podczas gdy monitor nie potrafi już prawidłowo odczytać topologii, promować repliki ani dołączyć dawnego primary.

Trzeba rozdzielić trzy funkcje:

1. **obserwowanie** topologii i stanu wątków replikacji;

2. **modyfikowanie** topologii podczas switchover, failover lub rejoin;
3. **routowanie** sesji aplikacyjnych do bieżącego primary i dostępnych replik.

Zgodność z MySQL 8.4 musi działać poprawnie na wszystkich trzech poziomach. Sama zamiana `SHOW SLAVE STATUS` na `SHOW REPLICA STATUS` nie wystarcza.

Dlaczego MySQL 8.4 łamie historyczne założenia

MySQL 8.0.22 wprowadził terminologię Source/Replica, zachowując jednocześnie część historycznych aliasów. MySQL 8.4 kończy tę migrację: przestarzałe polecenia zostały usunięte.

Operacja	MariaDB / stary dialekt	MySQL 8.4
Odczyt stanu repliki	`SHOW SLAVE STATUS`	`SHOW REPLICA STATUS`
Konfiguracja źródła	`CHANGE MASTER TO`	`CHANGE REPLICATION SOURCE TO`
Uruchomienie replikacji	`START SLAVE`	`START REPLICA`
Zatrzymanie replikacji	`STOP SLAVE`	`STOP REPLICA`
Wyczyszczenie konfiguracji	`RESET SLAVE`	`RESET REPLICA`
Odczyt stanu binloga	`SHOW MASTER STATUS`	`SHOW BINARY LOG STATUS`
Reset GTID	`RESET MASTER`	`RESET BINARY LOGS AND GTIDS`

Zmieniają się również nazwy zwracanych kolumn:

Stara nazwa	Nazwa w MySQL 8.4
`Master_Host`	`Source_Host`
`Master_Port`	`Source_Port`
`Master_Server_Id`	`Source_Server_Id`
`Slave_IO_Running`	`Replica_IO_Running`
`Slave_SQL_Running`	`Replica_SQL_Running`
`Seconds_Behind_Master`	`Seconds_Behind_Source`
`Master_Log_File`	`Source_Log_File`

Stara nazwa	Nazwa w MySQL 8.4
`Read_Master_Log_Pos`	`Read_Source_Log_Pos`
`Exec_Master_Log_Pos`	`Exec_Source_Log_Pos`
`Channel_Name`	`Channel_Name`

Monitor, który nadal wysyła `SHOW SLAVE STATUS`, natychmiast kończy się błędem składni. Monitor wysyłający właściwe zapytanie, lecz nadal szukający `Slave_IO_Running`, błędnie uzna, że replikacja nie działa.

Prawdziwa różnica: model GTID

Najważniejszą różnicą nie jest słownictwo SQL, lecz sposób reprezentacji transakcji.

GTID MariaDB

MariaDB zapisuje GTID w postaci:

```
domain_id-server_id-sequence
```

Przykład:

```
0-101-7842
```

Domena jest częścią modelu. Monitor MariaDB potrafi porównywać takie pozycje i korzysta między innymi z `MASTER_USE_GTID`.

GTID MySQL

MySQL reprezentuje zbiór GTID za pomocą UUID serwerów i przedziałów:

```
155fa720-7623-11f1-9a78-bc241174cab8:1-76
```

Historia może zawierać wiele UUID oraz nieciągłe przedziały:

```
155fa720-7623-11f1-9a78-bc241174cab8:1-10:20-30,
7fd8c42a-7630-11f1-99af-bc241174cab8:1-8
```

MySQL udostępnia między innymi:

- `@@global.server_uuid` do identyfikacji źródła;
- `@@global.gtid_executed` dla zastosowanych transakcji;
- `@@global.gtid_purged` dla GTID, których binlogi zostały usunięte;
- `Retrieved_Gtid_Set` w `SHOW REPLICA STATUS` dla odebranych transakcji.

Zmiana źródła wykorzystuje automatyczne pozycjonowanie:

```
CHANGE REPLICATION SOURCE TO
    SOURCE_HOST = '10.0.0.12',
    SOURCE_PORT = 3306,
    SOURCE_USER = 'repl',
    SOURCE_PASSWORD = 'secret',
    SOURCE_AUTO_POSITION = 1,
    GET_SOURCE_PUBLIC_KEY = 1
FOR CHANNEL '';
```

Serwer sam wybiera właściwy punkt wznowienia na podstawie zbiorów GTID, bez podawania pliku i pozycji binloga.

Dlaczego moduł `mysqlrepmon`

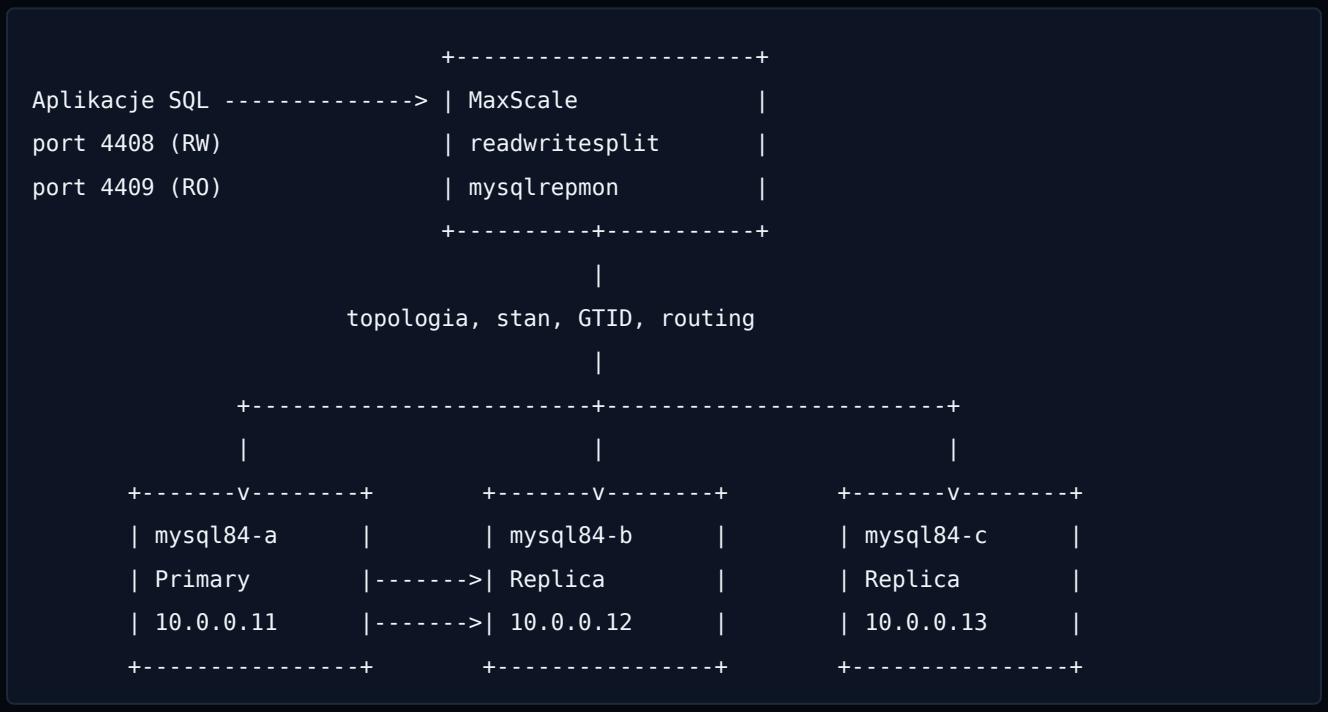
Obok `mariadbmon` propozycja dodaje monitor przeznaczony dla MySQL. Wykorzystuje sprawdzony silnik podejmowania decyzji i manipulowania klastrem, ale zastępuje elementy zależne od serwera:

- odczyt `SHOW REPLICA STATUS` oraz kolumn `Source_*` / `Replica_*`;
- odczyt `server_uuid`, `gtid_executed` i `gtid_purged`;
- użycie `CHANGE REPLICATION SOURCE TO ... SOURCE_AUTO_POSITION=1`;
- polecenia `START`, `STOP` i `RESET REPLICA ... FOR CHANNEL`;
- włączenie `super_read_only=1` podczas degradacji serwera;
- użycie `RESET BINARY LOGS AND GTIDS` dla ręcznego resetu;
- sprawdzanie `enforce_gtid_consistency` i `log_replica_updates`.

Moduł zachowuje parametry operacyjne znane z `mariadbmon`: `auto_failover`, `auto_rejoin`, `enforce_read_only_slaves` oraz polecenia switchover, failover i rejoin.

Architektura referencyjna

Przetestowana architektura obejmuje trzy serwery MySQL 8.4 i jeden węzeł MaxScale:



Aplikacje nigdy nie znają adresu primary. Łączą się z listenerem MaxScale. Monitor decyduje, który serwer ma wewnętrzną rolę `Master` w MaxScale; `readwritesplit` kieruje zapisy do niego i rozdziela odczyty.

> *Sam MaxScale również musi być redundantny. Idealny failover baz danych nie pomoże, jeśli jedyne proxy SQL jest pojedynczym punktem awarii. W produkcji należy przewidzieć co najmniej dwa węzły MaxScale w trybie aktywny/pasywny, VIP lub load balancer oraz odpowiedni mechanizm blokad kooperacyjnych.*

1. Przygotowanie każdego serwera MySQL 8.4

Każdy serwer, który może zostać promowany, musi potrafić tworzyć własne binlogi i przekazywać odebrane transakcje.

Konfiguracja bazowa:

```
[mysqld]
server_id=101
```

```
log_bin=mysql-bin
binlog_format=ROW

gtid_mode=ON
enforce_gtid_consistency=ON
log_replica_updates=ON

relay_log_recovery=ON
```

Na każdym węźle użyj innego `server_id`, na przykład `101`, `102`, `103`.

`server_uuid` również musi być unikalny. Jest przechowywany w pliku `auto.cnf` katalogu danych. Po sklonowaniu maszyny lub datadiru nie wolno uruchamiać wielu serwerów z tym samym `server_uuid`.

Na replikach:

```
read_only=ON
super_read_only=ON
```

Na primary:

```
read_only=OFF
super_read_only=OFF
```

Po restarcie sprawdź niezmienniki:

```
SELECT
    @@hostname,
    @@server_id,
    @@server_uuid,
    @@gtid_mode,
    @@enforce_gtid_consistency,
    @@log_bin,
    @@log_replica_updates,
    @@read_only,
    @@super_read_only\G

SELECT @@global.gtid_executed\G
SHOW BINARY LOG STATUS\G
SHOW REPLICA STATUS\G
```

Replika nadająca się do promocji musi mieć `log_bin=1`, `log_replica_updates=1`, oba aktywne wątki replikacji i kontrolowane opóźnienie.

2. Utworzenie oddzielnych kont

Należy rozdzielić co najmniej:

- konto monitorowania i modyfikowania topologii;
- konto używane przez repliki do odczytu binlogów;
- konta aplikacyjne przechodzące przez proxy.

Konto monitora

Tylko do obserwacji:

```
CREATE USER 'maxscale_mon'@'10.0.0.10'
    IDENTIFIED BY 'a-long-random-password';

GRANT REPLICATION CLIENT
    ON *.* TO 'maxscale_mon'@'10.0.0.10';
```

Do failover, switchover i rejoin monitor musi również móc zatrzymywać i rekonfigurować replikację, zmieniać zmienne read-only oraz kontrolować połączenia:

```
GRANT REPLICATION SLAVE,
      REPLICATION CLIENT,
      PROCESS,
      SUPER,
      SYSTEM_VARIABLES_ADMIN,
      REPLICATION_SLAVE_ADMIN,
      CONNECTION_ADMIN
    ON *.* TO 'maxscale_mon'@'10.0.0.10';
```

Historyczny przywilej `SUPER` nadal jest wymagany przez niektóre ścieżki zgodności. W finalnej wersji modułu należy ponownie ocenić dokładną listę i usunąć wszystkie zbędne uprawnienia.

Konto replikacji

```
CREATE USER 'repl'@'10.0.0.%'
    IDENTIFIED BY 'another-long-random-password'
    REQUIRE SSL;

GRANT REPLICATION SLAVE
    ON *.* TO 'repl'@'10.0.0.%';
```

Preferuj TLS. Przy domyślnym w MySQL 8.4 uwierzytelnianiu `caching_sha2_password` przez nieszyfrowane połączenie polecenie zmiany źródła musi podać `GET_SOURCE_PUBLIC_KEY=1` albo ścieżkę do publicznego klucza RSA. Moduł dodaje tę opcję, gdy TLS nie jest włączony.

Konto usługi do uwierzytelniania klientów

Parametr `user` usługi MaxScale nie oznacza konta, na którym wykonywane są wszystkie zapytania aplikacji. Umożliwia on modułowi User Account Manager wczytanie kont i uprawnień z backendów:

```
CREATE USER 'maxscale_route'@'10.0.0.10'
    IDENTIFIED BY 'route-password';

GRANT SELECT ON mysql.user
    TO 'maxscale_route'@'10.0.0.10';
GRANT SELECT ON mysql.db
    TO 'maxscale_route'@'10.0.0.10';
GRANT SELECT ON mysql.tables_priv
    TO 'maxscale_route'@'10.0.0.10';
GRANT SELECT ON mysql.columns_priv
    TO 'maxscale_route'@'10.0.0.10';
GRANT SELECT ON mysql.procs_priv
    TO 'maxscale_route'@'10.0.0.10';
GRANT SELECT ON mysql.proxies_priv
    TO 'maxscale_route'@'10.0.0.10';
GRANT SHOW DATABASES ON *.*
    TO 'maxscale_route'@'10.0.0.10';
```

Konta aplikacyjne nadal muszą istnieć na serwerach MySQL z tym samym hasłem i spójnymi grantami. Ponieważ backendy widzą połączenie przychodzące z MaxScale, część `@host` musi zezwalać na adres proxy.

3. Budowa wkładu z odtwarzalnego SHA

Ponieważ PR nadal jest wersją draft, standardowy pakiet MaxScale nie zawiera `mysqlrepmon`. Wyłącznie do laboratorium zbuduj dokładny publiczny SHA:

```
git clone https://github.com/mariadb-corporation/MaxScale.git
cd MaxScale

git fetch https://github.com/Esysteme/MaxScale.git \
  aurelien/mysql-8.4-replication-support
git checkout --detach f61776a5b19cf295636c94a8a3dea6d81fcd61fb

cmake -S . -B build \
  -DCMAKE_BUILD_TYPE=Release \
  -DCMAKE_INSTALL_PREFIX=/usr/local/maxscale-mysql84

cmake --build build --target mariadbmon mysqlrepmon test_cycle_find -j2
ctest --test-dir build --output-on-failure \
  -R '^test_mariadbmon_cycle_find$'

cmake --build build -j2
sudo cmake --install build
```

Budowa ukierunkowana musi utworzyć `libmysqlrepmon.so`. Poprawka CMake w tej propozycji jawnie wykorzystuje `LIBSSH_LIBRARY` i `LIBSSH_INCLUDE_DIR` wykryte przez MaxScale, zamiast zakładać istnienie wewnętrznego targetu `libssh`.

Uruchomienie odizolowanego buildu

Prefiks `/usr/local/maxscale-mysql84` zapobiega nadpisaniu zainstalowanego pakietu. W zamian usługa pakietu `maxscale.service` nadal uruchamia `/usr/bin/maxscale` i nie załaduje nowego modułu. W laboratorium utwórz `/etc/systemd/system/maxscale-mysql84.service`:

```
[Unit]
Description=MaxScale MySQL 8.4 compiled build
After=network-online.target
Wants=network-online.target

[Service]
Type=simple
```

```
User=maxscale
Group=maxscale
RuntimeDirectory=maxscale
RuntimeDirectoryMode=0755
ExecStart=/usr/local/maxscale-mysql84/bin/maxscale --nodaemon --config=/etc/maxscale.cnf --
logdir=/var/log/maxscale --datadir=/var/lib/maxscale --cachedir=/var/cache/maxscale --
libdir=/usr/local/maxscale-mysql84/lib/maxscale --piddir=/run/maxscale
Restart=on-failure
RestartSec=5s
LimitNOFILE=65535

[Install]
WantedBy=multi-user.target
```

Konto systemowe `maxscale` i katalogi danych muszą istnieć; zwykle tworzy je zainstalowany pakiet MaxScale zapewniający środowisko uruchomieniowe. Następnie przeładuj systemd:

```
sudo install -d -o maxscale -g maxscale -m 0755 \
  /var/lib/maxscale /var/cache/maxscale /var/log/maxscale
sudo systemctl daemon-reload
```

4. Konfiguracja MaxScale

Kompletny minimalny przykład:

```
[maxscale]
threads=auto
admin_host=127.0.0.1
admin_port=8989

[mysql84-a]
type=server
address=10.0.0.11
port=3306
protocol=MariaDBBackend

[mysql84-b]
type=server
address=10.0.0.12
port=3306
```

```
protocol=MariaDBBackend

[mysql84-c]
type=server
address=10.0.0.13
port=3306
protocol=MariaDBBackend

[MySQL84-Monitor]
type=monitor
module=mysqlrepmon
servers=mysql84-a,mysql84-b,mysql84-c
user=maxscale_mon
password=a-long-random-password
monitor_interval=2000ms

assume_unique_hostnames=true
auto_failover=safe
auto_rejoin=true
enforce_read_only_slaves=true

replication_user=repl
replication_password=another-long-random-password
replication_master_ssl=true

[MySQL84-RW-Service]
type=service
router=readwritesplit
servers=mysql84-a,mysql84-b,mysql84-c
user=maxscale_route
password=route-password

master_reconnection=true
master_failure_mode=fail_on_write

[MySQL84-RW-Listener]
type=listener
service=MySQL84-RW-Service
protocol=MariaDBClient
port=4408

[MySQL84-RO-Service]
```

```
type=service
router=readconnroute
servers=mysql84-a,mysql84-b,mysql84-c
user=maxscale_route
password=route-password
router_options=slave

[MySQL84-RO-Listener]
type=listener
service=MySQL84-RO-Service
protocol=MariaDBClient
port=4409
```

Hasła pozostają tutaj jawne wyłącznie dla czytelności przykładu. W eksploatacji użyj szyfrowania sekretów oferowanego przez MaxScale, restrykcyjnych uprawnień do pliku konfiguracyjnego oraz udokumentowanej rotacji.

`assume_unique_hostnames=true` jest wymagane, aby włączyć `auto_failover` i `auto_rejoin`. Każdy backend musi więc udostępniać unikatowy, stabilny adres lub hostname zgodny ze źródłem zapisanym przez replicas; użyj `private_address`, jeśli sieć replikacji korzysta z innego adresu.

Dlaczego zacząć od `auto_failover=safe`? Ten tryb odmawia operacji, gdy monitor wykryje, że promocja wyraźnie spowodowałaby utratę transakcji. Nie zmienia replikacji asynchronicznej w synchroniczną, ale dodaje użyteczną barierę podczas testów.

Dlaczego `master_reconnection=true` i `master_failure_mode=fail_on_write`? Sesja `readwritesplit` może zachować kontekst i połączyć się z nowym primary, o ile podczas okna bez primary nie nadejdzie zapis i nie ma otwartej transakcji. Bez tych ustawień failover bazy może się udać, a wszystkie sesje aplikacji i tak zostaną zerwane.

5. Weryfikacja przed pierwszym testem

Sprawdź konfigurację i uruchom MaxScale:

```
sudo /usr/local/maxscale-mysql84/bin/maxscale \
  --config=/etc/maxscale.cnf \
  --libdir=/usr/local/maxscale-mysql84/lib/maxscale \
  --config-check

sudo systemctl disable --now maxscale.service 2>/dev/null || true
```

```
sudo systemctl enable --now maxscale-mysql84.service
systemctl --no-pager --full status maxscale-mysql84.service
```

Sprawdź widoczność modułu i topologii:

```
maxctrl list modules | grep -E 'mariadbmon|mysqlrepmon'
maxctrl list servers
maxctrl list monitors
maxctrl show monitor MySQL84-Monitor
```

Oczekiwany stan:

- jeden serwer `Master, Running`;
- dwa serwery `Slave, Running`;
- brak serwerów `Down`;
- aktywny monitor;
- listenery `4408` i `4409` są otwarte.

Sprawdź także MySQL bezpośrednio:

```
mysql -h 10.0.0.12 -e "SHOW REPLICA STATUS\\G"
mysql -h 10.0.0.13 -e "SHOW REPLICA STATUS\\G"
```

Najważniejsze pola:

```
Replica_IO_Running: Yes
Replica_SQL_Running: Yes
Seconds_Behind_Source: 0
Auto_Position: 1
```

6. Przebieg failover

Gdy primary znika, monitor robi więcej niż tylko zmienia etykietę:

1. czeka przez liczbę cykli zdefiniowaną przez `failcount` i w miarę możliwości potwierdza rzeczywistą awarię;
2. porównuje stan replik i wybiera kandydatkę do promocji;
3. zatrzymuje replikację na kandydatce;

4. wyłącza `read_only` na nowym primary;
5. przekierowuje pozostałe repliki przez `CHANGE REPLICATION SOURCE TO ... SOURCE_AUTO_POSITION=1` ;
6. ponownie uruchamia ich wątki replikacji;
7. `readwritesplit` wykrywa nową rolę i kieruje kolejne zapisy do nowego primary.

Do degradacji primary `mysqlrepmon` używa:

```
SET GLOBAL super_read_only = 1;
```

To silniejsza ochrona niż `read_only=1` : nawet konto z uprawnieniami administracyjnymi nie powinno przypadkowo dalej zapisywać do dawnego primary.

7. Test kontrolowanego switchover

Przed symulacją twardej awarii sprawdź switchover:

```
maxctrl call command mysqlrepmon switchover \
  MySQL84-Monitor mysql84-b mysql84-a
```

Następnie sprawdź:

```
maxctrl list servers

mysql -h 10.0.0.11 -e \
  "SELECT @@hostname, @@read_only, @@super_read_only, @@global.gtid_executed\\G"
mysql -h 10.0.0.12 -e \
  "SELECT @@hostname, @@read_only, @@super_read_only, @@global.gtid_executed\\G"
mysql -h 10.0.0.13 -e \
  "SELECT @@hostname, @@read_only, @@super_read_only, @@global.gtid_executed\\G"
```

Nowy primary musi być zapisywalny. Dwa pozostałe węzły powinny mieć `super_read_only=1` i replikować z niego.

8. Test prawdziwej awarii bez ułatwień

Najpierw utwórz dane przez MaxScale:

```
CREATE DATABASE maxscale_ha_test;
CREATE TABLE maxscale_ha_test.events (
    id BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
    source_hostname VARCHAR(255) NOT NULL,
    created_at TIMESTAMP(6) NOT NULL DEFAULT CURRENT_TIMESTAMP(6),
    PRIMARY KEY (id)
) ENGINE=InnoDB;

INSERT INTO maxscale_ha_test.events(source_hostname)
VALUES (@@hostname);
```

Sprawdź ich obecność na replikach, a następnie zatrzymaj primary na poziomie usługi lub maszyny. Nie ustawiaj go jedynie w maintenance w MaxScale: to test decyzji administracyjnej, a nie wykrywania awarii.

Podczas testu:

```
watch -n 1 'maxctrl list servers'
```

W drugim terminalu:

```
journalctl -u maxscale -f
```

Kryteria sukcesu:

- wybrany zostaje dokładnie jeden nowy primary;
- pozostałe repliki wskazują na niego;
- zapis przez listener `4408` działa po promocji;
- odczyty przez `4409` pozostają spójne;
- dawny primary po powrocie dołącza jako replika;
- zbiory `gtid_executed` zbiegają się.

Co potwierdziło laboratorium MySQL 8.4.10

Scenariusz wykonano na trzech węzłach MySQL 8.4.10:

- początkowe wykrycie primary i dwóch replik;
- zatrzymanie primary;

- promocja jednej repliki;
- przekierowanie pozostałej repliki do nowego źródła;
- powrót dawnego primary i automatyczny rejoin jako replika;
- drugi failover w przeciwnym kierunku;
- końcowa zbieżność do tego samego zbioru `gtid_executed` na trzech serwerach;
- poprawny routing odczytów i zapisów przez MaxScale;
- odrzucanie zapisów na listenerze read-only.

Test potwierdza ścieżkę funkcjonalną dla ciągłych historii GTID. Nie dowodzi jeszcze poprawności wszystkich przypadków brzegowych wymaganych do integracji upstream.

Uwierzytelnianie: nie myl dwóch awarii

Build MaxScale użyty w laboratorium odrzucał konta aplikacyjne wykorzystujące hash `caching_sha2_password` z MySQL 8.4:

```
Stored password hash length is 70 when 40 was expected
```

Komunikat pochodzi z authenticatora protokołu MaxScale w tym buildzie, a nie z monitora replikacji. Replikacja i routing mogą działać prawidłowo, mimo że uwierzytelnianie klienta nie działa.

W laboratorium dedykowane konto probe z `mysql_native_password` pozwoliło odizolować problem. Nie jest to zalecenie ogólne: MySQL 8.4 domyślnie wyłącza ten historyczny plugin. W produkcji należy użyć buildu i authenticatora MaxScale zgodnego z `caching_sha2_password` albo oficjalnie wspieranej metody uwierzytelniania, zamiast globalnie osłabiać konfigurację MySQL.

Ta sama zasada dotyczy:

```
401 Unauthorized
```

na porcie REST: oznacza to, że `maxctrl` nie ma prawidłowych danych administracyjnych. Nie dowodzi awarii monitora ani routingu SQL.

Tabela diagnostyczna

Objaw	Prawdopodobna przyczyna	Weryfikacja
Błąd składni przy `SHOW SLAVE STATUS`	stary moduł lub dialekt MariaDB użyty z MySQL 8.4	log MaxScale, faktycznie wysłane zapytanie
Serwer bez roli repliki	kolumny `Replica_*` nie są mapowane	bezpośrednie `SHOW REPLICA STATUS\G`
Repliki nie można promować	wyłączone `log_bin`, `log_replica_updates` lub GTID	zmienne globalne i log monitora
Rejoin odrzucony	transakcje errant lub niezgodna historia	porównaj `gtid_executed` / `gtid_purged`
`Stored password hash length is 70...`	authenticator niezgodny z `caching_sha2_password`	log MaxScale, plugin konta
`401 Unauthorized` w `maxctrl`	błędne dane REST	konfiguracja administracyjna MaxScale
Sesja zerwana mimo promocji	router nie może się połączyć, otwarta transakcja lub wyczerpana historia poleceń sesji	`master_reconnection`, `master_failure_mode`, log routera
Dwa zapisywalne primary	niewystarczający fencing/read-only albo konkurencyjne monitory	`super_read_only`, blokady kooperacyjne, stan sieci

Znane ograniczenie: luki w zbiorach GTID

Aby ponownie wykorzystać wewnętrzny silnik `mariadbmon`, propozycja mapuje każdy UUID MySQL na syntetyczną domenę i konwertuje jego przedziały do istniejącej reprezentacji wewnętrznej.

W obecnym stanie zachowuje najwyższy numer transakcji dla każdego UUID. Dlatego:

```
uuid:1-10:20-30
```

jest podsumowane tak, jakby pozycja osiągnęła `30`. Informacja o braku transakcji od 11 do 19 nie jest wiernie reprezentowana.

Historie w laboratorium były ciągłe. W infrastrukturze z GTID wstrzykniętymi, filtrowanymi, usuniętymi lub errant takie uproszczenie może zniekształcić porównanie kandydatek.

To główny powód, dla którego PR pozostaje wersją draft. Zanim moduł będzie można uznać za gotowy produkcyjnie, potrzebne są:

- prawidłowa reprezentacja lub porównywanie nieciągłych przedziałów;
- testy jednostkowe z wieloma UUID i lukami;
- pokrycie GTID errant i wyczyszczonych historii;
- pełna CI upstream;
- przegląd maintainerów MaxScale.

Czego failover nie gwarantuje

Monitor nie usuwa właściwości replikacji asynchronicznej.

- **Zerowy RPO nie jest gwarantowany**: transakcja potwierdzona przez dawny primary mogła jeszcze nie dotrzeć do repliki.
- **Otwarte transakcje**: sesji w trakcie transakcji nie zawsze można przenieść bez błędu.
- **Split-brain**: jeśli dawny primary pozostaje dostępny dla części aplikacji, potrzebny jest prawdziwy fencing sieciowy lub systemowy.
- **Złożone topologie**: multi-source, replikacja kołowa i relaye wymagają osobnej strategii.
- **Rozbieżne dane**: `auto_rejoin` nie może po cichu nadpisywać transakcji errant.
- **Pojedyncze proxy**: MaxScale trzeba zredundować niezależnie.

Prawidłowy test wysokiej dostępności mierzy więc cztery różne rzeczy: wykrycie, promocję, zbieżność danych i ciągłość aplikacji.

Checklist przed produkcją

- [ ] przetestowana, odtwarzalna kopia zapasowa;
- [ ] trzy unikalne wartości `server_id` i `server_uuid`;
- [ ] GTID oraz `log_replica_updates` włączone wszędzie;
- [ ] TLS między MaxScale a backendami;
- [ ] oddzielne konta monitora, replikacji i aplikacji;
- [ ] `super_read_only=1` potwierdzone na każdej replice;

- [ ] kontrolowany switchover sprawdzony przed twardym failover;
- [ ] test zapisu przez MaxScale po promocji;
- [ ] test rejoin dawnego primary;
- [ ] końcowe porównanie zbiorów `gtid_executed`;
- [ ] udokumentowane zachowanie transakcji aplikacji;
- [ ] przetestowany fencing i redundancja MaxScale;
- [ ] alert przy każdej promocji i rozbieżności GTID;
- [ ] moduł upstream ukończony, przejrzany i wspierany przed produkcją.

Dlaczego publikujemy tę propozycję

MySQL 8.4 jest wydaniem LTS. Historyczne polecenia nie wrócą. Bezterminowe utrzymywanie aliasów w narzędziach monitorujących nie rozwiązuje różnicy modelu GTID ani składni rekonfiguracji.

Dedykowany moduł jasno wyznacza granicę:

- `mariadbmon` pozostaje natywny dla dialektu i GTID MariaDB;
- `mysqlrepmon` zawiera reguły specyficzne dla MySQL 8.x;
- wspólny silnik failover pozostaje współdzielony;
- testy mogą weryfikować każdą rodzinę bez mnożenia rozproszonych warunków.

PR MaxScale #421 zawiera kod, dokumentację, polecenia walidacyjne, wyniki laboratorium oraz znane ograniczenie GTID. Celem wersji draft jest właśnie uzyskanie przeglądu tej reprezentacji, zanim będzie można stwierdzić, że każda historia MySQL jest bezpieczna.

Więcej informacji

- MySQL 8.4 — nowości i usunięte polecenia replikacji
- MySQL 8.4 — konfiguracja replikacji
- MySQL 8.4 — zmiana źródła i `GET_SOURCE_PUBLIC_KEY`
- MaxScale — parametry MariaDB Monitor
- MaxScale — router readwritesplit

- Instalacja MySQL 8.4 na Debianie 13
- Przejście z Master/Slave na Source/Replica

Chcesz zobaczyć tę obsługę w MaxScale?

Jeśli zgodność z MySQL 8.4 byłaby dla Ciebie przydatna, przeczytaj propozycję, przetestuj gałąź i okaż wsparcie bezpośrednio w PR. Informacje z rzeczywistych wdrożeń, złożone przypadki GTID i przeglądy techniczne pomogą zmienić ten draft w wkład, który naprawdę można zintegrować:

Wesprzyj PR MaxScale #421