IV. Logiczna replikacja strumieniowa w SZBD Postgres

Replikacja strumieniowa zaprezentowana na poprzednich zajęciach jest replikacją fizyczną. Oznacza to, że:

- Zmiany we wszystkich obiektach podstawowej bazy danych są replikowane do czuwających baz danych.
- W danym momencie tylko jedna baza danych jest bazą podstawową, do której użytkownicy mogą wprowadzać zmiany.
- Czuwające bazy danych są uruchomione w trybie tylko do odczytu, w związku z tym nie mogą posiadać własnych obiektów oprócz tych, które są replikowane z podstawowej bazy danych.

Wady te łagodzi zastosowanie logicznej replikacji strumieniowej. Mechanizm ten, podobnie jak wcześniej poznana fizyczna replikacja strumieniowa, korzysta z dziennika bazy danych do transferu zmian w obiektach między bazami danych. W nazewnictwie systemu Postgres pliki dziennika bazy danych nazywane są WAL (ang. Write Ahead Log). Mechanizm logicznej replikacji strumieniowej wykorzystuje dwa rodzaje baz danych: *dostawcy* (ang. providers) i *subskrybenci* (ang. subscribers). *Dostawcy* dostarczają dane, natomiast *subskrybenci* replikują dane *dostawców*. Dane można modyfikować zarówno w bazach typu *dostawca* jak i *subskrybent*. Jednakże tylko zmiany wprowadzane w dostawcach są replikowane do subskrybentów. Zmiany wprowadzone w subskrybentach pozostają lokalne i mogą być przyczyną konfliktów. Logiczna replikacji strumieniowa może być kaskadowa, to znaczy, że subskrybent może być dostawcą dla innych subskrybentów.

Celem zajęć jest zapoznanie się z logiczną replikacją strumieniową dostarczaną przez system Postgres od wersji 10.

1. Przygotowanie środowiska

1. Zaloguj się do maszyny wirtualnej jako użytkownik `rbd` używając hasła `RBD#7102`.
2. Otwórz okno terminala, który nazwiemy terminalem pomocniczym.
3. Usuń z klastra Kubernetes obiekty, które zostały utworzone w poprzednim tutorialu, wykonaj w terminalu pomocniczym następujące polecenia:
   ```
   kubectl delete -f rbd-citus.yaml
   kubectl delete pvc pgsql-rbd-citus-disk-pgsql-citus-sts-0
   kubectl delete pvc pgsql-rbd-citus-disk-pgsql-citus-sts-1
   kubectl delete pvc pgsql-rbd-citus-disk-pgsql-citus-sts-2
   kubectl delete -f rbd-citus-coord.yaml
   kubectl delete pvc postgres-db-rbd-citus-coord-1-0
   kubectl delete pvc postgres-db-rbd-citus-coord-2-0
   kubectl delete pvc postgres-db-rbd-citus-coord-3-0
   kubectl delete pvc postgres-db-rbd-citus-coord-4-0
   ```
4. Skorzystamy ze StatefulSet, które zostały zdefiniowane w pierwszym tutorialu, w tym celu wykonaj w terminalu pomocniczym następujące polecenia:
   ```
   k3d image import rbd/postgres13:1.0 --cluster RBDcluster
   kubectl apply -f  rbd1.yaml
   kubectl apply -f  rbd2.yaml
   ```

5. Otwórz 2 nowe zakładki w oknie terminala, pierwszą nazwij provider-1, a drugą subscriber-1.
6. W terminalu provider-1 uruchom poniższe polecenie w celu przyłączenia się do bazy danych, którą nazwiemy provider-1.
   `psql -U postgres -h localhost -p 5432`
   Użyj hasła rbd1 w celu uwierzytelnienia użytkownika postgres.
   Uwaga: jeżeli narzędzie psql nieoczekiwanie traci połączenie ze serwerem zamiast adresu localhost użyj jednego z adresów węzła loadbalancer. W celu pozyskania tych adresów użyj polecenia: `kubectl get svc`. Wykorzystaj jeden z adresów z kolumny EXTERNAL-IP.
7. W bazie danych provider-1 zwiększ ilość informacji generowanych do plików dziennika bazy danych, które umożliwią logiczną replikację. W tym celu wykonaj polecenie:
   `alter system set wal_level=logical;`
8. W terminalu pomocniczym zrestartuj Pod obsługujący bazę danych provider-1. Użyj polecenia:
   `kubectl rollout restart sts pgsql-rbd1`
9. W terminalu pomocniczym zrestartuj połączenie do bazy danych provider-1. Możesz w tym celu wykonać dwukrotnie to samo polecenie, np.:
   `show wal_level;`

2. Strumieniowa replikacja logiczna z jednym dostawcą.

1. W bazie danych provider-1 utwórz użytkownika *repl*, który będzie służyć do uwierzytelnienia połączenia z bazy danych subskrybenta do bazy danych dostawcy. Uruchom polecenie:
   `CREATE ROLE repl WITH REPLICATION LOGIN PASSWORD 'rbd1repl';`
2. W bazie danych provider-1 utwórz obiekty, których zawartość będzie replikowana, skorzystaj z poniższego polecenia:
   `\i ~/loggers/loggers.sql`
3. W bazie danych provider-1 nadaj użytkownikowi *repl* prawa odczytu do obiektów, które zostały utworzone w poprzednim kroku, użyj poniższe polecenie:
   `GRANT select ON ALL TABLES IN SCHEMA public TO repl;`
   Polecenie to nadaje uprawnienia do wszystkich obecnie istniejących tabel w schemacie public, czyli tylko tych, które zostały utworzone w poprzednim kroku
4. W celu udostępnienia wybranych tabel mechanizmowi logicznej replikacji utwórz w bazie danych provider-1 publikację o nazwie meas_publication.
   `CREATE PUBLICATION meas_publication;`
5. W bazie danych provider-1 dodaj do publikacji meas_publication tabelę organizations. Skorzystaj z poniższego polecenia:
   `ALTER PUBLICATION meas_publication ADD TABLE organizations;`
6. W bazie danych provider-1 sprawdź za pomocą poniższego polecenia jakie publikacje znajdują się w bazie danych:
   `select pubname from pg_publication;`
7. W bazie danych provider-1 zobacz jakie tabele zostały opublikowane, w tym celu uruchom poniższe polecenie:
   `select * from pg_publication_tables;`
8. W terminalu subscriber-1 uruchom poniższe polecenie w celu przyłączenia się do bazy danych, którą nazwiemy subscriber-1.

```
psql -U postgres -h localhost -p 5433
```

9. W bazie danych subscriber-1 utwórz obiekty, do których będą replikowana dane z obiektów dostawcy, skorzystaj z poniższego polecenia:
   ```
   \i ~/loggers/loggers.sql
   ```
   Mechanizm logicznej replikacji nie propaguje operacji DDL. Dzięki temu podejściu schematy obiektów w bazie danych dostawcy i subskrybenta mogą być różne.
10. W bazie danych subscriber-1 utwórz subskrypcję meas_subscription, która umożliwi replikację danych z bazy danych dostawcy do bazy danych subskrybenta, uruchom poniższe polecenie:
    ```
    CREATE SUBSCRIPTION meas_subscription CONNECTION
    'host=pgsql-rbd1-lb port=5432 user=repl password=rbd1repl dbname=postgres'
    PUBLICATION meas_publication;
    ```
11. W bazie danych subscriber-1 wyświetl utworzone subskrypcje, skorzystaj z poniższego polecenia:
    ```
    select * from pg_subscription;
    ```
12. W bazie danych provider-1 wstaw dane do tabeli organizations:
    ```
    \i ~/loggers/organizations.dmp
    ```
13. W bazie danych subscriber-1 wyświetl status replikacji:
    ```
    select subname as substription_name, relname as table_name,
            case srsubstate
            when 'i' then 'initialize'
            when 'd' then 'data is being copied'
            when 's' then 'synchronized'
            when 'r' then 'ready (normal replication)'
            end
    from pg_subscription_rel r join pg_subscription s on r.srsubid=s.oid
       join pg_class c on r.srrelid=c.oid;
    ```
14. W bazie danych subscriber-1 sprawdź liczbę wierszy w tabeli organizations. Czy replikacja zmian się powiodła? [Raport]
15. W bazie danych provider-1 dodaj tabelę loggers do publikacji meas_publication. [Raport]
16. W bazie danych provider-1 wstaw dane do tabeli loggers:
    ```
    \i ~/loggers/loggers.dmp
    ```
17. W bazie danych subscriber-1 sprawdź liczbę wierszy w tabeli loggers. Czy replikacja zmian się powiodła? [Raport]
18. Powtórz krok 13, czy w wyniku znajdują się informacje o statusie replikacji tabeli loggers? [Raport]
19. W bazie danych subscriber-1 odśwież subskrypcję meas_subscription:
    ```
    ALTER SUBSCRIPTION meas_subscription REFRESH PUBLICATION;
    ```
20. Powtórz krok 17 oraz 18.
21. Sprawdź replikację wyników operacji aktualizacji i usuwania danych z tabeli loggers w bazie danych provider-1. [Raport]
22. Sprawdź co się stanie jeżeli w trakcie wykonywania zmian w bazie danych provider-1 baza danych subscriber-1 będzie niedostępna, wykonaj następujący eksperyment [Raport]:
    a. Zmniejsz liczbę replik Pod bazy danych subscriber-1 do zera
    b. W bazie danych provider-1 wprowadź zmiany do replikowanego obiektu.

c. Zwiększ liczbę replik Pod bazy danych subscriber-1 do jednej
d. W bazie danych subscriber-1 sprawdź czy zmiany zostały zreplikowane.

23. Sprawdź definicję parametru wal_keep_size. Jak jego wartość może mieć wpływ na wynik eksperymentu z kroku 22? [Raport]

3. Strumieniowa replikacja logiczna z wieloma dostawcami.

1. Utwórz nowy StatefulSet, który będzie zawierał bazę danych provider-2. W tym celu skopiuj plik manifestów rbd2.yaml do pliku rbd3.yaml. Za pomocą ulubionego edytora zamień w pliku rbd3.yaml wszystkie napisy rbd2 na rbd3, zamień również port wykorzystywany przez loadbalancer z 5433 na 5434. Wykonaj wdrożenie manifestów znajdujących się w pliku rbd3.yaml.
2. W oknie terminalu otwórz nową zakładkę w oknie terminala, nazwij ją provider-2.
3. W terminalu provider-2 uruchom poniższe polecenie w celu przyłączenia się do bazy danych, którą nazwiemy provider-2.
   ```
   psql -U postgres -h localhost -p 5434
   ```
4. W bazie danych provider-2:
   4.1. Zmień odpowiednio wartość parametru wal_level i zrestartuj bazę danych provider-2
   4.2. Utwórz tabele korzystając ze skryptu ~/loggers/loggers.sql
   4.3. Utwórz użytkownika repl
   4.4. Nadaj użytkownikowi repl uprawnienia do obiektów utworzonych za pomocą skryptu ~/loggers/loggers.sql
   4.5. Utwórz publikację meas2_publication i dodaj do niej tabelę organizations
5. W bazie danych subscriber-1 utwórz subskrypcję meas2_subscription, która umożliwi replikację danych z bazy danych provider-2.
6. W bazie danych provider-2 dodaj organizację o identyfikatorze -40. Czy zmiany przepropagowały się do baz danych subscriber-1? [Raport]
7. Przeprowadź eksperyment wprowadzenia konfliktu przez wstawienie do bazy danych provider-2 organizacji, która już istnieje w bazie danych provider-1 i subscriber-1. Wykonaj bazie danych provider-2 sekwencję następujących operacji:
   ```
   insert into organizations values(50, 'Konflitowa', 'CLIENT');
   select pg_current_wal_lsn ();
   insert into organizations values(-50, 'Nie Konflitowa', 'CLIENT');
   ```
   Funkcja *pg_current_wal_lsn*() zwraca bieżącą pozycję zapisu w pliku WAL.
8. W bazie danych subscriber-1 sprawdź wartości dla organizacji o identyfikatorach 50 i -50. Czy propagacja zmian się powiodła? [Raport]
9. Sprawdź w logu bazy danych subscriber-1 co się wydarzyło. W tym celu w terminalu pomocniczym uruchom poniższe polecenie:
   ```
   kubectl logs pgsql-rbd2-0 -f
   ```
10. W celu rozwiązania konfliktu pominiemy w bazie danych subscriber-1 wstawienie "konfliktowej" organizacji.
    10.1. Wykonaj poniższe polecenie:
    ```
    select s.subname, r.*
    from pg_subscription s join pg_replication_origin_status r
     on concat('pg_',s.oid)=r.external_id;
    ```
    Odczytaj wartość kolumny external_id dla subskrypcji meas2_subscription.

10.2. Wykonaj poniższe polecenie, użyj odpowiednio external_id uzyskanego w punkcie 10.1 oraz WAL LSN uzyskanego w punkcie 7:

```
select pg_replication_origin_advance('external_id','WAL LSN');
```

11. Sprawdź w terminalu pomocniczym nowe wpisy w logu bazy danych subscriber-1. Czy replikacja została wznowiona? [Raport]
12. W bazie danych subscriber-1 sprawdź wartości dla organizacji o identyfikatorach 50 i -50. Wyjaśnij uzyskany wynik [Raport]