ASBD TP3 : Restauration de données

Auteurs:

- CONTRERAS Baptiste p1809436
- CLEMENT Florent p1601511

Quelle est la différence entre une réplication logique et une réplication physique ?

→ La réplication logique consiste qu'à chaque événement (update / delete …) on l'a joue sur l'ensemble des réplicats. Quant à la réplication physique, elle consiste à "copier" des blocs au niveau disque. Cela transmet beaucoup plus d'informations comme les index, les données etc.

Qu'est-ce qu'une réplication en WAL shipping ?

C'est une méthode d'archivage continue, le nœud queen sauvegarde ses WALs à interval régulier dans un emplacement dédié. Les réplicats iront eux même périodiquement rapatrier les WALs et les appliquer. Cette méthode ne nécessite pas de connexion réseau persistante.

Qu'est-ce qu'une réplication en mode "streaming" ?

Dans le mode streaming, dès qu'une WAL est généré par la queen, elle est directement envoyée et appliquée sur les réplicats au travers d'une communication persistante.

Une réplication synchrone peut-elle fonctionner en WAL shipping ?

Oui, la réplication synchrone est possible en WAL shipping mais cela dégrade les performances. En effet, qu'on envoie un ou plusieurs blocs de données (selon où se trouve les données que l'on modifie durant la transaction) qui contiennent beaucoup plus d'information que nécessaire.

Quels sont les inconvénients d'une réplication synchrone ?

Si jamais une requête “destructrice” est jouée sur la Queen, elle affecte immédiatement le réplicat. De plus le mode synchrone peut ajouter de la latence que l’ensemble des réplicats aient fini de jouer la modification avant de renvoyer une confirmation.

Prérequis

Nous utilisons **pgbackrest** comme utilitaire de sauvegarde pour nos WAL, si il n’est pas installé :

Pour l’installer pgbackrest :

```
sudo apt-get update && sudo apt-get install -y pgbackrest
```

La configuration minimum :

```
sudo vim /etc/pgbackrest.conf
[global]
repo-path=/var/lib/pgbackrest
retention-full=2

[main]
db-path=/var/lib/postgresql/13/main
```

1. Création du cluster sur le réplicat

Pour commencer, nous allons préparer la machine réplica.

Sur la machine réplica on crée un nouveau cluster

```
sudo pg_ctlcluster create 13 main
```

Nous avons décidé arbitrairement de garder “**main**” comme nom de cluster sur le réplica. Si vous choisissez un nom différent, il faudra veiller à remplacer “main” par votre nom dans la suite.

Une fois créée vous obtiendrez ce résultat:

```
Ver Cluster  Port Status Owner    Data directory
Log file
13  main 5432 down   postgres /var/lib/postgresql/13/replica1
```

```
/var/log/postgresql/postgresql-13-replica1.log
```

2. Préparation de la Queen

2.1. Configuration

On va maintenant se connecter sur la Queen et modifier un peu de configuration pour permettre la réplication.

```
sudo vim /etc/postgresql/13/main/postgresql.conf
```

```
wal_level = replica
wal_log_hints = on
archive_mode = on # (change requires restart)
archive_command = 'pgbackrest --stanza=main archive-push %p'
max_wal_senders = 10
wal_keep_size = 64
hot_standby = on
```

2.2. Autorisation

Il faut maintenant créer à notre queen pour la réplica.

La première étape consiste à créer un nouvel user postgresql :

```
sudo su postgres &&  psql -c "CREATE USER rep_user WITH
replication;"
```

Et autoriser la connexion dans pg_hba.conf

```
sudo vim /etc/postgresql/13/main/pg_hba.conf
```

```
host  replication       rep_user       192.168.73.234/24
```

```
trust
```

En ajoutant cette ligne suivante à notre réplica, défini comme étant le serveur **192.168.73.234** va pouvoir se connecter sur la base réplication, avec le user **rep_user** sans mot de passe. Il serait judicieux de renforcer la sécurité, en ajoutant, par exemple, un mot de passe mais nous ne traiterons pas de cela ici.

C'est fini pour la configuration de la Queen, on peut la démarrer et retourner sur le réplica pour finir la mise en place de la réplication.

```
sudo service postgresql restart
```

3. Configuration du réplica

3.1. Modification de la configuration du cluster

Assurez vous postgres est bien arrêté :

```
sudo service postgresql status
```

Si ce n'est pas le cas, stoppez le cluster :

```
sudo service postgresql stop
```

Nous allons éditer le fichier de configuration de postgres :

```
sudo vim /etc/postgresql/13/main/postgresql.conf
```

```
wal_level = replica
wal_log_hints = on
archive_mode = on # (change requires restart)
max_wal_senders = 10
wal_keep_size = 64
hot_standby = on
restore_command = 'pgbackrest --log-level-console=info
--stanza=main archive-get %f "%p"'
recovery_target_timeline = 'latest'
primary_conninfo = 'user=rep_user
passfile=''/var/lib/postgresql/.pgpass'' host''192.168.73.216''
port=5432 sslmode=prefer sslcompression=1 krbsrvname=postgres
target_session_attrs=any'
archive_cleanup_command = 'pg_archivecleanup
/var/lib/postgresql/13/pg_archive/main %r'
```

3.2 Ajout d'un utilisateur local

Ici aussi nous allons ajouter le user postgresql rep_user.

```
psql -c "CREATE USER rep_user WITH replication;"
```

Une fois ceci fait nous pouvons éditer le fichier le fichier pga_hba :

```
sudo vim /etc/postgresql/13/main/pg_hba.conf
```

```
local replication rep_user trust
```

3.3 Synchronisation de l'état initial de la Queen

Pour partir sur de bonnes bases, nous devons faire en sorte que la réplica soit similaire à notre Queen, nous allons supprimer le répertoire /var/lib/postgresql/13/main/ et le synchroniser avec celui de la Queen.

```
sudo rm -rf  /var/lib/postgresql/13/main
```

```
pg_basebackup -h 192.168.73.216 -D /var/lib/postgresql/13/main -U
rep_user -w -P -R
```

Pour démystifier un peu cette commande :

- -h 192.168.73.216 pour préciser l'ip de la queen
- -U pour spécifier l'utilisateur créé précédemment sur la Queen
- -w pour skip le mot de passe
- -D pour la destination

Pour plus d'information sur les possibilités de cette commande :
https://www.postgresql.org/docs/13/app-pgbasebackup.html

```
sudo service postgresql start
```

Si le cluster main n'est pas lancé vous pouvez lancer cette commande :

```
sudo pg_ctlcluster 13 main start
```

Ainsi via la commande :

```
sudo pg_lsclusters
```

Vous obtiendrez ce résultat :

```
Ver Cluster Port Status          Owner    Data directory
Log file
13  main    5432 online,recovery postgres /var/lib/postgresql/13/main
/var/log/postgresql/postgresql-13-main.log
```

4. Tests

Pour tester notre réplication, nous allons créer une table sur la Queen et vérifier qu'elle est présente dans le cluster réplica.

Queen :

```
psql -c "CREATE DATABSE testReplication;"
CREATE DATABASE
```

```
psql -c "\l"
```

```
 postgres        | postgres | UTF8     | C.UTF-8 | C.UTF-8 |
 template0       | postgres | UTF8     | C.UTF-8 | C.UTF-8 | =c/postgres
+
                 |          |          |         |         |
postgres=CTc/postgres
 template1       | postgres | UTF8     | C.UTF-8 | C.UTF-8 | =c/postgres
+
                 |          |          |         |         |
postgres=CTc/postgres
 testreplication | postgres | UTF8     | C.UTF-8 | C.UTF-8 |
```

Sur le réplica :

```
postgres@tiw3-asbd-postgres-26:/home/ubuntu$ psql -c "\l"
                                  List of databases
      Name       |  Owner   | Encoding | Collate |  Ctype  |
Access privileges
-----------------+----------+----------+---------+---------+------
-----------------
 postgres        | postgres | UTF8     | C.UTF-8 | C.UTF-8 |
 template0       | postgres | UTF8     | C.UTF-8 | C.UTF-8 |
=c/postgres          +
                 |          |          |         |         |
postgres=CTc/postgres
 template1       | postgres | UTF8     | C.UTF-8 | C.UTF-8 |
=c/postgres          +
                 |          |          |         |         |
postgres=CTc/postgres
 testreplication | postgres | UTF8     | C.UTF-8 | C.UTF-8 |
```

On remarque que la database **testreplication** est présente sur le réplica sans aucune intervention de notre part !

4.1. Vérification vue système

```
SELECT * FROM pg_stat_replication ;
```

```
   pid   | usesysid | usename  | application_name |  client_addr   |
client_hostname | client_port |         backend_start         |
backend_xmin |   state   | sent_lsn  | write_lsn | flush_lsn |
replay_lsn | write_lag | flush_lag | replay_lag | sync_priority |
sync_state |          reply_time
---------+----------+----------+------------------+----------------+----
-------------+-------------+-------------------------------+------------
--+-----------+-----------+-----------+-----------+-----------+--------
---+-----------+------------+--------------+-----------+--------------
-----------------
 1346799 |    16384 | rep_user | 13/main          | 192.168.73.234 |
|       32820 | 2021-05-23 10:56:13.257079+02 |              | streaming
| 0/40011A0 | 0/40011A0 | 0/40011A0 | 0/40011A0  |           |
|            |             0 | async      | 2021-05-23
11:42:26.845379+02
(1 row)
```

4.2. Vérification en direct sur le réplica

Nous créons une table sur **testreplication**:

```
psql -d testreplication
```

```
CREATE TABLE mytable( id integer );
```

Nous avons maintenant ce résultat sur la Queen :

```
testreplication=# \dt+
                            List of relations
 Schema |  Name   | Type  |  Owner   | Persistence |  Size   |
Description
--------+---------+-------+----------+-------------+---------+----------
---
 public | mytable | table | postgres | permanent   | 0 bytes |
```

Et sur la réplication :

```
testreplication=# \dt+
                             List of relations
 Schema |  Name   | Type  |  Owner   | Persistence |  Size   |
Description
--------+---------+-------+----------+-------------+---------+----------
---
 public | mytable | table | postgres | permanent   | 0 bytes |
(1 row)
```

4.3. Changement du WAL courant

```
postgres=# SELECT pg_walfile_name(pg_current_wal_lsn());
     pg_walfile_name
--------------------------
 000000010000000000000013
(1 row)
```

```
postgres=# SELECT pg_switch_wal();
 pg_switch_wal
---------------
 0/13025B68
(1 row)
```

```
postgres=# SELECT pg_walfile_name(pg_current_wal_lsn());
     pg_walfile_name
--------------------------
 000000010000000000000014
```

```
select * from pg_stat_replication()


   pid   | usesysid | usename  | application_name |  client_addr   |
```

```
client_hostname | client_port |         backend_start         |
backend_xmin |   state    |  sent_lsn  | write_lsn  | flush_lsn  |
replay_lsn | write_lag | flush_lag | replay_lag | sync_priority |
sync_state |          reply_time
---------+----------+----------+-----------------+----------------+----
-------------+-------------+--------------------------------+-----------
--+-----------+-----------+-----------+-----------+-----------+-----
------+-----------+-----------+---------------+-----------+-----------
--------------------
 1346799 |    16384 | rep_user | 13/main          | 192.168.73.234 |
|       32820 | 2021-05-23 10:56:13.257079+02 |               | streaming
| 0/14000148 | 0/14000148 | 0/14000148 | 0/14000148 |           |
|            |            0 | async      | 2021-05-23
15:00:58.142679+02
(1 row)
```

```
select * from pg_stat_wal_receiver;

  pid   |  status   | receive_start_lsn | receive_start_tli |
written_lsn | flushed_lsn | received_tli |      last_msg_send_time
|     last_msg_receipt_time     | latest_end_lsn |
latest_end_time        | slot_name |  sender_host   | sender_port |
conninfo
--------+-----------+-------------------+-------------------+-----------
--+-------------+--------------+--------------------------------+--------
-----------------------+----------------+--------------------------------
+-----------+----------------+-------------+----------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
----------------------------------------------------------------------------
---------------------------
 521910 | streaming | 0/4000000         |                 1 | 0/14000148
| 0/14000148  |           1 | 2021-05-23 15:06:38.039899+02 |
2021-05-23 15:06:38.186993+02 | 0/14000148     | 2021-05-23
14:17:36.596121+02 |           | 192.168.73.216 |        5432 |
user=rep_user passfile=/root/.pgpass channel_binding=prefer
dbname=replication host=192.168.73.216 port=5432
fallback_application_name=13/main sslmode=prefer sslcompression=0
ssl_min_protocol_version=TLSv1.2 gssencmode=prefer krbsrvname=postgres
```

```
target_session_attrs=any
```

4.4 Passage de la réplication en synchrone

Pour passer la réplication en synchrone il faut juste ajouter la ligne

synchronous_standby_names = '"192.168.73.234"'

dans la configuration postgres de la queen, redémarrer le cluster et voilà !