ASBD TP2 : Restauration de données

Auteurs:

- CONTRERAS Baptiste p1809436
- CLEMENT Florent p1601511

Quelques informations avant de débuter :

Qu'est-ce qu'une sauvegarde full ?

→ C'est une sauvegarde totale des données. (très coûteux en temps et en ressources)

Qu'est-ce qu'une sauvegarde incrémentale ?

→ Un sauvegarde incrémentale permet de sauvegarder les modifications apportées à un fichier depuis la dernière sauvegarde. C'est l'équivalent de faire un git diff du fichier et de le sauvegarder

Quelle est la différence entre une sauvegarde incrémentale et une sauvegarde différentielle ?

→ La sauvegarde incrémentale à l'avantage d'utiliser moins de ressource mais elle est plus longue à recharger, car il faut rejouer tous les changements des fichiers depuis le début, tandis que la sauvegarde différentielle nécessite qu'un simple copier / coller des fichiers à restaurer.

Que signifie PITR ?

→ Point-in-time Recovery

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

→ Une sauvegarde logique permet de sauvegarder les schémas et leurs données tandis que les sauvegarde physique vont se focaliser sur les fichiers "brut" du SGBD, comme les fichier block, de configuration, les logs.

Que signifie WAL ?

→ Write Ahead log

Pourquoi a-t-on besoin des WALs lors d'une sauvegarde physique ?

→ Les WALs font partie intégrante des fichiers constituant une base de données postgres, il est donc normal de les sauvegarder lors d'une sauvegarde physique. De plus, le fait d'avoir les WALs sous la main offre la possibilité de faire du PITR, et de pouvoir restaurer les données à un moment précis.

Restauration

Après s'être connecté à la base de données nous pouvons créer la base de données pour le bench:

```
postgres=# CREATE DATABASE benchdb;
CREATE DATABASE
```

Nous pouvons maintenant initialisé le bench via la commande suivante:

```
pgbench -s 10 -i benchdb
```

Une fois le bench initialisé, nous vérifions si les tables **pgbench_tellers** et **pgbench_accounts** sont bien présentent:

```
benchdb=# select count(*) from pgbench_tellers;
 count
-------
   100
(1 row)
----------------------
```

```
benchdb=# select count(*) from pgbench_accounts;
  count
---------
 1000000
(1 row)
```

Installation de pgBackRest

Commençons par installer l'outil. Sur ubuntu c'est relativement simple :

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

Vérifions ensuite que la commande **pgbackrest** est fonctionnelle:

```
ubuntu@tiw3-asbd-postgres-25:~$ pgbackrest
```

```
pgBackRest 2.33 - General help

Usage:
    pgbackrest [options] [command]

Commands:
    archive-get     Get a WAL segment from the archive.
    archive-push    Push a WAL segment to the archive.
    backup          Backup a database cluster.
    check           Check the configuration.
    expire          Expire backups that exceed retention.
    help            Get help.
    info            Retrieve information about backups.
    repo-get        Get a file from a repository.
    repo-ls         List files in a repository.
    restore         Restore a database cluster.
    stanza-create   Create the required stanza data.
    stanza-delete   Delete a stanza.
    stanza-upgrade  Upgrade a stanza.
    start           Allow pgBackRest processes to run.
    stop            Stop pgBackRest processes from running.
    version         Get version.

Use 'pgbackrest help [command]' for more information.
```

L'utilitaire installé, passons à la création de notre première sauvegarde:

```
sudo vim /etc/pgbackrest.conf
```

Nous allons éditer le fichier de configuration avec les valeurs suivantes :

```
[global]
repo-path=/var/lib/pgbackrest
retention-full=2

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

Nous lançons maintenant un archivage avec **pgbackrest** :

```
sudo -u postgres pgbackrest --stanza=main --log-level-console=info
stanza-create
```

Nous pouvons ensuite vérifier que la backup a bien eu lieu:

```
sudo ls /var/lib/pgbackrest
archive  backup
```

Sauvegarde full

Pour commencer la sauvegarde incrémentale nous allons commencer par modifier la configuration:

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

```
archive_command = 'pgbackrest --stanza=main archive-push %p'
archive_mode = on
listen_addresses = '*'
log_line_prefix = ''
max_wal_senders = 3
wal_level = logical
```

Une fois la configuration prête nous pouvons lancer la backup:

```
sudo -u postgres pgbackrest --stanza=main --log-level-console=info
backup
```

Une fois la backup est terminée nous pouvons vérifier s'il elle est correct :

```
sudo -u postgres pgbackrest --stanza=main --log-level-console=info check
```

Sauvegarde incrémentale

Pour commencer, nous allons vider la table **pgbench_tellers** :

```
benchdb=# DELETE FROM pgbench_tellers;
DELETE 100
```

```
benchdb=# SELECT COUNT(*) FROM pgbench_tellers;
 count
-------
     0
(1 row)
```

Effectuons ensuite une sauvegarde incrémentale

```
sudo -u postgres pgbackrest --type=incr --stanza=main
--log-level-console=info backup
```

```
2021-05-05 14:42:29.725 P00   INFO: backup command begin 2.33:
--exec-id=917673-64354456 --log-level-console=info
--pg1-path=/var/lib/postgresql/13/main --pg1-port=5432
--repo1-path=/var/lib/pgbackrest --repo1-retention-full=2 --stanza=main
--type=incr
2021-05-05 14:42:30.462 P00   INFO: last backup label =
20210505-142426F, version = 2.33
2021-05-05 14:42:30.463 P00   INFO: execute non-exclusive
pg_start_backup(): backup begins after the next regular checkpoint
completes
2021-05-05 14:42:31.370 P00   INFO: backup start archive =
000000010000000000000000E, lsn = 0/E000028
2021-05-05 14:42:32.149 P01   INFO: backup file
/var/lib/postgresql/13/main/base/16418/2619 (152KB, 49%) checksum
3a8896acb764a83889ded2b953e72966d3b54f9c
2021-05-05 14:42:32.167 P01   INFO: backup file
/var/lib/postgresql/13/main/base/16418/1259 (104KB, 84%) checksum
ac93b2d4a202aab67014caaa7b90a3c9544ae64b
2021-05-05 14:42:32.181 P01   INFO: backup file
/var/lib/postgresql/13/main/base/16418/16437 (16KB, 89%) checksum
4ab740dad5198354d878abce94e9aebac8219e2a
2021-05-05 14:42:32.248 P01   INFO: backup file
/var/lib/postgresql/13/main/base/16418/16434_fsm (16KB, 94%) checksum
1c20282d25c2f6a16e04b74a9b93717f61e8d66f
2021-05-05 14:42:32.263 P01   INFO: backup file
/var/lib/postgresql/13/main/pg_xact/0000 (8KB, 97%) checksum
18a8a2b2f6256d46a835a89205504522f67049cd
2021-05-05 14:42:32.278 P01   INFO: backup file
/var/lib/postgresql/13/main/global/pg_control (8KB, 99%) checksum
efc28d36ffd0544653cd2d77f2e79a51da3d50ff
2021-05-05 14:42:32.296 P01   INFO: backup file
/var/lib/postgresql/13/main/pg_logical/replorigin_checkpoint (8B, 100%)
checksum 347fc8f2df71bd4436e38bd1516ccd7ea0d46532
2021-05-05 14:42:32.310 P01   INFO: backup file
/var/lib/postgresql/13/main/base/16418/16434_vm (0B, 100%)
2021-05-05 14:42:32.425 P01   INFO: backup file
/var/lib/postgresql/13/main/base/16418/16434 (0B, 100%)
2021-05-05 14:42:32.432 P00   INFO: incr backup size = 304KB
2021-05-05 14:42:32.432 P00   INFO: execute non-exclusive
pg_stop_backup() and wait for all WAL segments to archive
2021-05-05 14:42:33.238 P00   INFO: backup stop archive =
000000010000000000000000E, lsn = 0/E000100
```

```
2021-05-05 14:42:33.490 P00   INFO: check archive for segment(s)
000000010000000000000000E:000000010000000000000000E
2021-05-05 14:42:33.657 P00   INFO: new backup label =
20210505-142426F_20210505-144230I
2021-05-05 14:42:33.881 P00   INFO: backup command end: completed
successfully (4157ms)
2021-05-05 14:42:33.882 P00   INFO: expire command begin 2.33:
--exec-id=917673-64354456 --log-level-console=info
--repo1-path=/var/lib/pgbackrest --repo1-retention-full=2 --stanza=main
2021-05-05 14:42:33.915 P00   INFO: expire command end: completed
successfully (33ms)
```

Comme le montre

La sauvegarde incrémentale terminée avec succès, nous allons supprimer les lignes de avec bid = 2

```
benchdb=# DELETE FROM pgbench_accounts WHERE bid=2;
DELETE 100000
```

```
benchdb=# SELECT COUNT(*) FROM pgbench_accounts;
 count
--------
 900000
(1 row)
```

On vérifie que nos sauvegarde sont bien présente dans le dossier dédié :

```
ubuntu@tiw3-asbd-postgres-25:~$ sudo ls /var/lib/pgbackrest/backup/main
20210505-142426F  20210505-142426F_20210505-144230I  backup.history
backup.info  backup.info.copy  latest
```

Restauration de nos sauvegarde :

Maintenant que nous avons modifié nos données, nous souhaiterions tout annuler pour revenir à l'état initial. Nous allons appliquer les sauvegardes une à une et remonter le temps.

Restauration de la sauvegarde incrémentale :

Avant toute chose, il faut couper postgres avant de lancer la restauration :

```
sudo service postgresql stop
```

Pour plus de détails sur les commandes et les options ci-dessous, voici le lien du guide officiel https://pgbackrest.org/user-guide.html (partie 10)

```
sudo -u postgres pgbackrest --stanza=main --log-level-console=info
--delta --set=20210505-142426F_20210505-144230I restore
```

On ajoute l'option **--delta** pour laisser le soin à pgbacktest de supprimer les fichiers nécessaires pour faire une restauration. (c'est faisable à la main pour pourquoi s'embêter ?)

L'autre option, cette fois très importante est **--set**. Elle permet de préciser une sauvegarde à appliquer (si on veut appliquer une différente de la dernière)

```
sudo -u postgres pgbackrest --stanza=main --log-level-console=info
--type=time --target="2021-05-05 14:24:26" --delta
--set=20210505-142426F_20210505-144230I restore
```

Une fois la backup terminée on retrouve bien les valeurs que nous avions avant la suppression.

```
benchdb=# SELECT COUNT(*) FROM pgbench_accounts where bid=2;
 count
--------
 100000
(1 row)
```

```
benchdb=# SELECT COUNT(*) FROM pgbench_accounts;
  count
---------
 1000000
(1 row)
```

Toutefois un problème survient après le backup, il nous est plus possible de faire d'opération car la base de données se met en standby.
Pour éviter que ce problème survienne, nous devons rajouter l'option
--target-action=promote pour pas que la base de données se mette en **standby**.

```
sudo -u postgres pgbackrest --stanza=main --log-level-console=info
--set=20210505-142426F --type=time --target-action=promote
```

```
--target="2021-05-05 14:24:26" --delta  restore
```

```
SELECT COUNT(*) FROM pgbench_tellers;
 count
-------
   100
(1 row)
```

Utilisation des WAL pour identifier une erreur

Génération de transaction

```
while :; do pgbench -c 4 -j 1 -T 60 benchdb; sleep 1; done
```

Récupération d'information sur notre base de données

```
select
coalesce(tbs.oid, db.dattablespace) as tablespace,
db.oid as database,
t.relfilenode as table
from pg_class t left outer join pg_tablespace tbs
on t.reltablespace=tbs.oid
cross join pg_database db
where t.relname='pgbench_accounts'
and db.datname=current_database();
```

```
 tablespace | database | table
------------+----------+-------
       1663 |    16418 | 16431
```

On commet une erreur -> requête destructrice

```
benchdb=# DELETE FROM pgbench_tellers;
DELETE 100
```

Nooon, notre table a été vidé…

```
SELECT COUNT(*) FROM pgbench_tellers;
 count
```

```
-------
     0
```

Nous pouvons ensuite exécuter la commande suivante pour lire le wal :

```
pg_waldump -p /var/lib/postgresql/13/main/pg_wal WAL_NAME
```

Toutefois, malgré le fait d'avoir essayé sur tous nos fichiers wal, il nous a été impossible d'en lire un. Nous tombions sur cette erreur :

```
pg_waldump: fatal: could not find a valid record after
```

Nous supposons que les Wal sont en avance par rapport à l'état où la sauvegarde a été appliquée. Il se peut qu'au moment où on a restauré la base de données les WAL n'ont pas été restauré ce qui les a désynchronisés.