Tour d'horizon des moyens de sauvegarde et duplication de PostgreSQL

Du PITR à la réplication

2017.12.14

Dalibo - http://dalibo.com

http://www.dalibo.org/conferences

Du PITR à la réplication

Tour d'horizon des moyens de sauvegarde et duplication de PostgreSQL

TITRE : Du PITR à la réplication
SOUS-TITRE : Tour d'horizon des moyens de sauvegarde et duplication de PostgreSQL

REVISION: 2017.12.14

Contents

DIFFÉRENTES FAÇONS DE SAUVEGARDER

AVANT DE SE LANCER

- Que sauvegarder ?
- À quelle fréquence sauvegarder les données ?
- Quels supports ?
- Quels outils ?
- Quels moyens pour vérifier la restauration des sauvegardes ?

DIFFÉRENTES APPROCHES

- Sauvegarde à froid des fichiers
- Sauvegarde à chaud en SQL
- Sauvegarde à chaud des fichiers (PITR)

RTO/RPO

- `RPO` (Recovery Point Objective)
  - Perte de Données Maximale Admissible
- `RTO` (Recovery Time Objective)
  - Durée Maximale d'Interruption Admissible
- Définissent la politique de sauvegarde/restauration

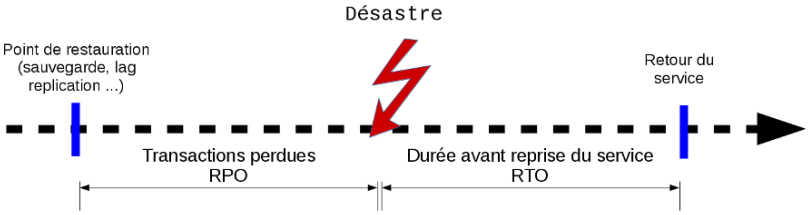


FIGURE 1: RTO ET RPO

POINTS D'ATTENTION

- Externaliser les sauvegardes
- Stocker plusieurs copies
  - de préférence dans des endroits différents
- Sauvegarder les fichiers de configuration
- Tester la restauration

SAUVEGARDES LOGIQUES

PRINCIPE

- Extraction à chaud des données dans un fichier
- Photo des données au début de l'opération
  - quelle que soit la durée de la sauvegarde
  - restauration à cet état
- Large choix d'options pour sélectionner les données

OUTILS

- Dump (Export)
  - `pg_dump` extrait le contenu d'une base en texte (SQL) ou binaire
  - `pg_dumpall` extrait une instance en totalité au format texte
- Restore (Import)
  - `psql` exécute le SQL des dumps au format texte
  - `pg_restore` restaure un dump binaire dans une base

FORMATS

Format	Dump	Restore
plain (ou SQL)	`pg_dump -Fp` ou `pg_dumpall`	`psql`
tar	`pg_dump -Ft`	`pg_restore`
custom	`pg_dump -Fc`	`pg_restore`
directory	`pg_dump -Fd`	`pg_restore`

AVANTAGES

- Simple et rapide à mettre en œuvre
- Sans interruption de service

- Indépendante de la version de PostgreSQL
- Granularité de sélection à l'objet
- Ne conserve pas la fragmentation des tables et des index

INCONVÉNIENTS

- Durée d'exécution dépendante des données et de l'activité
- Efficace pour des volumétries inférieures à 200 Go
- Restauration à l'instant du démarrage de l'export uniquement
- Impose d'utiliser plusieurs outils pour sauvegarder une instance complète
- Nécessite un `VACUUM ANALYZE` des tables à la restauration

SAUVEGARDES PHYSIQUES À FROID

PRINCIPES

- Une sauvegarde à froid impose l'arrêt de l'instance
- Outils système à utiliser pour sauvegarder et restaurer
- L'instance complète doit être sauvegardée :
  - PGDATA
  - tous les tablespaces
  - journaux de transactions (fichiers WAL)
  - fichiers de configuration

AVANTAGES

- Simple et rapide
- De nombreux outils disponibles
- Efficace pour les fortes volumétries

INCONVÉNIENTS

- Interruption de la production
- Sauvegarde de l'instance complète à un instant t
- Conservation de la fragmentation
- Impossible de changer d'architecture ou de version

SAUVEGARDES PHYSIQUES À CHAUD ET PITR

PRINCIPES

- Sauvegarde en deux parties :
  - les fichiers de données
  - les journaux de transactions archivés
- PostgreSQL archive les journaux de transactions
- L'administrateur réalise une copie des fichiers de données
- *Point In Time Recovery* : on applique les transactions archivées jusqu'à un point donné

LE MODE RECOVERY

- PostgreSQL écrit les données deux fois :
  - d'abord dans les journaux de transactions (fichiers WAL)
  - puis dans les fichiers de données de manière asynchrone
- En cas d'arrêt brutal, les journaux de transactions sont rejoués sur les fichiers de données
  - il s'agit du mode `recovery`
  - le rejeu des transactions permet de retourner à l'état cohérent
- Le contrôle du mode `recovery` permet le PITR

ARCHIVAGE DES JOURNAUX DE TRANSACTION

- Choisir le répertoire d'archivage
- Configurer PostgreSQL, dans `postgresql.conf` :
  - `wal_level` : `replica` ou `logical`
  - `archive_mode` : `on` ou `always`
  - `archive_command` : texte de la commande
  - `archive_timeout` : temps en secondes
- Redémarrer l'instance si `wal_level` et `archive_mode` ont changé, recharger sinon

---

SAUVEGARDE À CHAUD / BASEBACKUP

- L'archivage doit fonctionner sans erreur
- Se connecter et exécuter `SELECT pg_start_backup('label', true);`
- Copier l'ensemble des fichiers de l'instance :
  - fichiers de données (`$PGDATA`) et de configuration
  - tablespaces
- Mais ignorer :
  - fichier `postmaster.pid` et `postmaster.opts`
  - répertoires `pg_log`, `pg_xlog`/`pg_wal`, `pg_replslot`
- Se connecter et exécuter `SELECT pg_stop_backup();`

RESTAURATION PITR - FICHIERS DE DONNÉES

- Restaurer les fichiers et répertoires suivants :
  - $PGDATA
  - les tablespaces (mettre à jour `$PGDATA/pg_tblspc` si nécessaire)
  - fichiers de configuration
- Ne **pas** restaurer les fichiers suivants :
  - fichiers WAL de l'instance, i.e. `$PGDATA/pg_xlog` / `$PGDATA/pg_wal` (on restaurera les WAL archivés)
  - fichiers `postmaster.pid` et `postmaster.opts`
  - traces si elles sont incluses

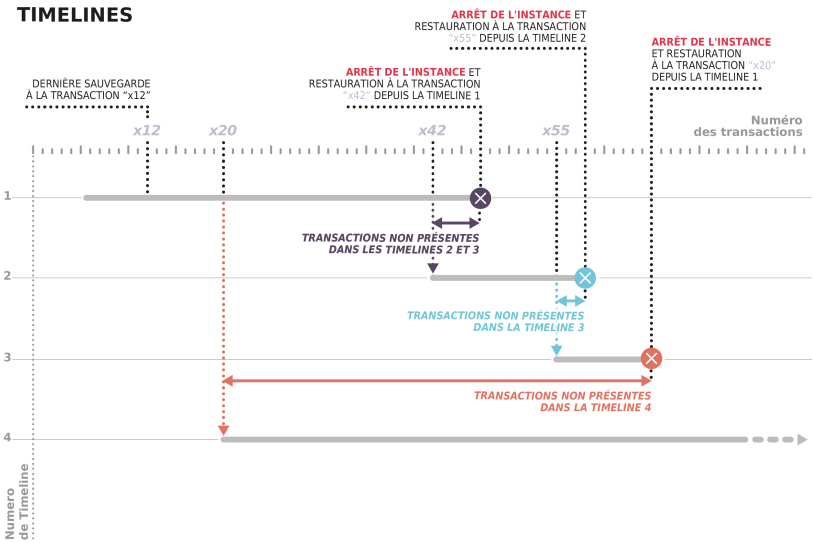
RESTAURATION PITR - RECOVERY.CONF

- PostgreSQL restaure lui-même les WAL archivés au démarrage
- L'instance doit pouvoir accéder aux fichiers WAL archivés
- Créer le fichier `$PGDATA/recovery.conf` :
  - `restore_command`
  - Cible : paramètres `recovery_target_*`

RESTAURATION PITR - DIFFÉRENTES TIMELINES

- En fin de `recovery`, la *timeline* change
  - l'historique des données prend une autre voie
  - le nom des WAL change pour éviter d'écraser des archives suivant le point d'arrêt
  - l'aiguillage est inscrit dans un fichier `.history`, archivé
- Permet de faire plusieurs restaurations PITR à partir du même *basebackup*
- `recovery_target_timeline` permet de choisir la *timeline* à suivre

RESTAURATION PITR - ILLUSTRATION DES TIMELINES



ADAPTER LA CONFIGURATION

- `postgresql.conf`
  - `port`
  - `listen_addresses`

    - `data_directory`
    -
- `pg_hba.conf`

AVANTAGES

- Sauvegarde sans interruption de service
- Restauration à la transaction près
- Efficace pour les fortes volumétries

INCONVÉNIENTS

- Restauration de l'instance complète
- Conservation de la fragmentation
- Impossible de changer d'architecture ou de version
- Point dans le temps souvent difficile à trouver

OUTILS

- Plusieurs outils implémentent ses mécanismes :
  - **pg_basebackup** : outil natif, utilisant une connection de réplication
  - **barman** : intéressant pour centraliser les backups de plusieurs instances
  - **pitrery** : orienté simplicité, peut sauvegarder un hot-standby
  - **pgbackrest** : gère les incrémentales
  - **wal-e** : stockage vers les cloud AWS, google, Azure et Swift.

RÉPLICATION PHYSIQUE

PRINCIPES

- La réplication se base sur les mécanismes du PITR
- Avec des optimisations :
  - Réplication en flux
  - Slots de réplication
- Un replicat est en mode recovery permanent et rejoue le WAL au fil de l'eau
- La replication logique s'affranchit du mode recovery

CONFIGURATION

- Sur le serveur principal :
  - Créer un role de connexion avec l'attribut `replication`
  - Configurer `pg_hba.conf`
  - Faire un basebackup (avec pg_basebackup)
  - Optionnel : créer un slot de replication `SELECT pg_create_physical_replication_slot('`
- Sur le serveur secondaire :
  - Restaurer le basebackup
  - Configurer `$PGDATA/recovery.conf` :
  - `standby_mode = on`
  - `primary_conninfo = 'host=ip_principal port=5432 user=repli'`
  - `recovery_target_timeline = 'latest'`
  - Optionnel : `primary_slot_name = 'nom'`

AVANTAGES

- Très simple à mettre en oeuvre grâce à pg_basebackup
- De nombreuses fonctionnalités :
  - Synchronisme
  - Cascade
  - Accès en lecture seule sur les réplicats

INCONVÉNIENTS

- Switch-over coûteux : il faut recréer un standby de (quasi) zéro
- **Ce n'est pas une sauvegarde**, les ordres suivants seraient répliqués :
  - Un `UPDATE` sans clause `WHERE`
  - `DROP TABLE` sur la "mauvaise" table
- Pas d'accès en écriture sur les réplicats

MERCI

- Nicolas Thauvin - nicolas.thauvin@dalibo.com
- http://dalibo.com - http://github.com/dalibo
- twitter: @orgrim
- blog : http://orgrim.net