4

Architecture des processus

PostgreSQL est un serveur reposant sur le concept du multiprocessus. À son démarrage, il exécute différents processus, suivant la configuration. Certains sont activés par défaut, d'autres non. Certains ne sont pas désactivables. La plupart sont configurables via différents paramètres. Ils ont tous leur utilité et assurent une cohérence dans le fonctionnement d'une instance. De ce fait, si un processus meurt sans raison, le serveur PostgreSQL peut se retrouver dans un état instable et forcera un redémarrage automatique.

Voici ce que donne la commande `ps` tout de suite après avoir démarré un serveur PostgreSQL :

```
$ ps fxo pid,cmd | grep [p]ostgres
626688  \_ /opt/postgresql/12/bin/postgres
626689      \_ postgres: logger
626691      \_ postgres: checkpointer
626692      \_ postgres: background writer
626693      \_ postgres: walwriter
626694      \_ postgres: autovacuum launcher
626695      \_ postgres: archiver   last was 000000010000000000000000B
626696      \_ postgres: stats collector
626697      \_ postgres: logical replication launcher
```

Noms des processus

Sous certains systèmes d'exploitation, postgres est le nom de tous les processus du serveur PostgreSQL. Autrement dit, il est impossible de les différencier. Ceci est gênant pour l'administrateur, pas pour le serveur en lui-même.

Néanmoins, il peut être intéressant de vérifier la valeur du paramètre `update_process_title`. Ce dernier permet d'activer et de désactiver la mise à jour du nom du processus. Il est activé par défaut, sauf sous Windows à partir de la version 10 de PostgreSQL en raison de la surcharge généralement importante que cause l'activation de ce paramètre sous ce système d'exploitation.

D'autre part, depuis la version 9.5, il est possible de donner un nom à l'instance. Pour cela, il faut configurer le paramètre `cluster_name` en utilisant uniquement des

caractères ASCII (tout autre caractère est remplacé par un point d'interrogation). Ce nom est affiché après le texte `postgres: ` , comme ici avec `prod` comme nom d'instance :

```
626907  \_ /opt/postgresql/12/bin/postgres
626908      \_ postgres: prod: logger
626910      \_ postgres: prod: checkpointer
626911      \_ postgres: prod: background writer
626912      \_ postgres: prod: walwriter
626913      \_ postgres: prod: autovacuum launcher
626914      \_ postgres: prod: archiver
626915      \_ postgres: prod: stats collector
626916      \_ postgres: prod: logical replication launcher
```

Il existe deux types de processus :

- les processus de gestion interne (par exemple, ceux chargés des écritures dans les fichiers de données, celui chargé des statistiques d'activité, etc.) ;
- les processus de communication avec les clients (plus simplement, ceux chargés de l'exécution des requêtes SQL et des flux de réplication).

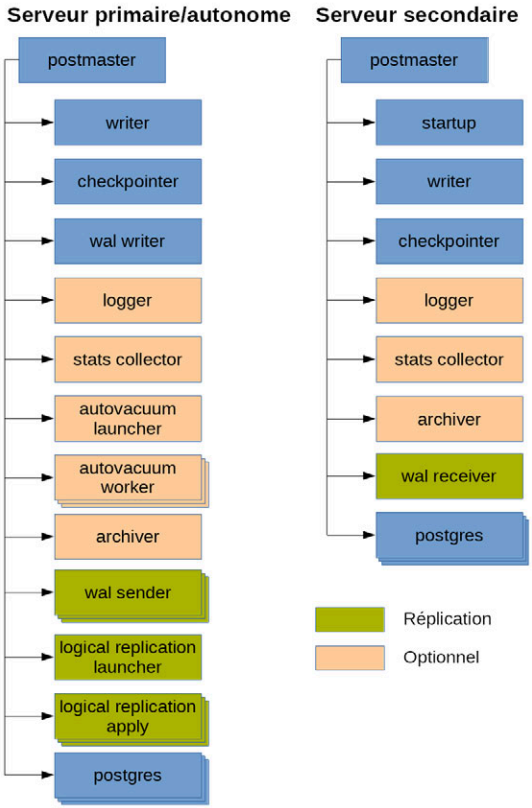
Les processus de gestion sont peu nombreux et disponibles en un seul exemplaire sur le système pour une même instance (à l'exception notable de l'`autovacuum worker`) :

- démarrage et gestion des sous-processus : `postmaster` et `startup` ;
- écriture dans les fichiers de données : `writer` et `checkpointer` ;
- écriture dans les journaux de transactions : `wal writer` ;
- gestion des statistiques d'activité : `stats collector` ;
- gestion des fichiers de traces : `logger` ;
- gestion automatique des `VACUUM` et `ANALYZE` : `autovacuum launcher` et `autovacuum worker` ;
- gestion de l'archivage des journaux de transactions : `archiver` ;
- gestion de la réplication par streaming : `wal receiver`, `logical replication apply`.

Les processus de communication client/serveur sont de deux types. Le premier gère les connexions standards (`postgres`). Le deuxième gère les connexions de réplication (`wal sender`).

Certains processus sont disponibles sur un serveur primaire ou sur un serveur autonome, alors que d'autres sont spécifiques aux serveurs secondaires. Certains sont optionnels, d'autres non. La Figure 4.1 en fournit une vue d'ensemble.

Figure 4.1 : Processus serveurs



4.1. Démarrage et gestion des sous-processus

Toute la phase d'initialisation est réalisée par deux processus : `postmaster` lui-même et `startup`. Le premier commence en réalisant quelques initialisations, puis lance le processus `startup`. Ces deux processus vont s'exécuter en parallèle. Le processus `startup` s'occupe de rejouer les changements contenus dans les journaux de transactions qui n'auraient pas eu le temps d'atteindre les fichiers de données. Cela peut survenir en cas