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
 5397  \_ /opt/postgresql/10/bin/postgres
 5398      \_ postgres: logger process
 5400      \_ postgres: checkpointer process
 5401      \_ postgres: writer process
 5402      \_ postgres: wal writer process
 5403      \_ postgres: autovacuum launcher process
 5404      \_ postgres: archiver process   last was
000000010000000000000010
 5405      \_ postgres: stats collector process
 5406      \_ postgres: bgworker: logical replication launcher
```

Note *› 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 :*

```
 1653  \_ /opt/postgresql/10/bin/postgres
 5666  \_ /opt/postgresql/10/bin/postgres
 5667      \_ postgres: prod: logger process
```

```
5669      \_ postgres: prod: checkpointer process
5670      \_ postgres: prod: writer process
5671      \_ postgres: prod: wal writer process
5672      \_ postgres: prod: autovacuum launcher process
5673      \_ postgres: prod: archiver process
5674      \_ postgres: prod: stats collector process
5675      \_ postgres: prod: bgworker: 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).

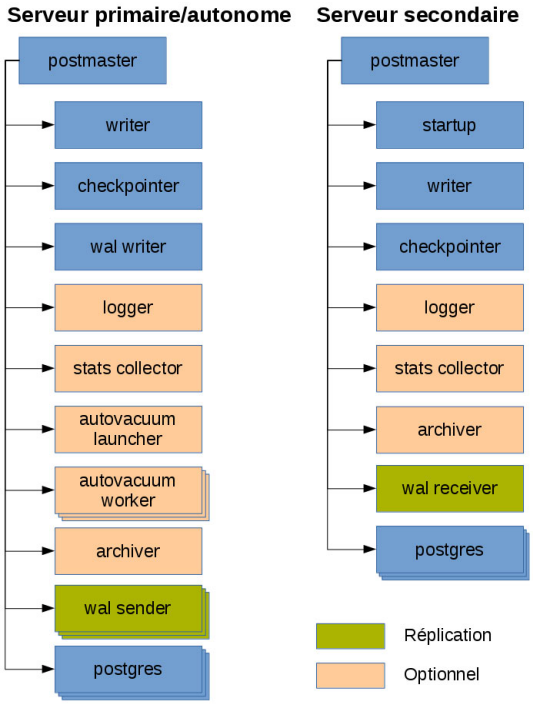
Les processus de gestion sont peu nombreux et disponibles en un seul exemplaire sur le système pour une même instance (à l'exception notable de l'`autovacuum worker`) :

- 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`.

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 d'arrêt brutal du serveur. Il avertit le processus `postmaster` quand il entame le rejeu, il l'avertit de nouveau quand il est arrivé au point de cohérence, ce qui permet au processus `postmaster` de changer l'état de l'instance à chaque avancée. La Figure 4.2 montre les différentes étapes de l'initialisation du point de vue de la machine à états.