PostgreSQL et les connexions en cours

PostgreSQL est un outil génial, si génial qu'il permet d'avoir des connexions entre un service et lui (oui je sais, c'est logique ^^)

Prérequis

Il est recommandé d'avoir un accès superuser pour effectuer ces actions. Généralement on les executent lorsqu'il y a des problèmes d'accès à la base de donnée (des connexions ouverte mais morte).

Connaitre le nombre de connexion en simultanée sur une base de donnée

Pour connaitre le nombre de connexion à une base de donnée, tapez la commande suivante avec `<DBNAME>` votre base de donnée

```
select datname, usename, pid, application_name, client_addr, client_hostname, client_port, backend_start,
query_start, query
from pg_stat_activity
where datname = '<DBNAME>';
```

Vous devez avoir un retour similaire :

```
mastodon=# select datname, usename, pid, application_name, client_addr, client_hostname, client_port,
backend_start, query_start, query
from pg_stat_activity
```

```
where datname = 'mastodon';
 datname  | usename  |  pid  |          application_name          | client_addr | client_hostname | client_port |
backend_start         |          query_start
|

query

----------+----------+-------+-----------------------------------+-------------+-----------------+-------------+------------------------------
+------------------------------
+-----------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------
 mastodon | mastodon | 31057 | puma: cluster worker 1: 8526 [www] | 10.10.10.3  |                 |       49250 |
2020-01-28 17:25:39.587535+01 | 2020-01-28 17:35:04.662738+01 | SELECT  "accounts".* FROM "accounts"
WHERE "accounts"."uri" = $1 LIMIT $2
 mastodon | mastodon | 29827 | puma: cluster worker 3: 8526 [www] | 10.10.10.3  |                 |       49216 |
2020-01-28 17:24:23.619165+01 | 2020-01-28 17:35:16.13175+01  | SELECT  "accounts".* FROM "accounts"
WHERE "accounts"."uri" = $1 LIMIT $2
 mastodon | mastodon |  3180 | puma: cluster worker 3: 8526 [www] | 10.10.10.3  |                 |       50088 | 2020-
01-28 17:35:12.39345+01  | 2020-01-28 17:35:12.426536+01 | SELECT "status_pins"."status_id" FROM
"status_pins" WHERE 1=0 AND "status_pins"."account_id" = $1
 mastodon | mastodon | 31061 | sidekiq 5.2.7 www [0 of 50 busy]   | 10.10.10.3  |                 |       49260 | 2020-
01-28 17:25:39.714381+01 | 2020-01-28 17:35:16.176494+01 | COMMIT
```

Couper des connexions

TIMBER !

Pour couper une connexion, rien de plus simple :

```
SELECT
    pg_terminate_backend(pid)
FROM
    pg_stat_activity
WHERE
    pid <> pg_backend_pid()
    AND pid = '<PID>'
    AND datname = '<DBNAME>';
```

Avec `<PID>` le PID que vous souhaitez couper (et que vous avez récupéré plus haut) et `<DBNAME>` le nom de votre base de donnée. La ligne `pid <> pg_backend_pid()` permet de ne pas se couper sa propre connexion (mesure de sécurité).

Pour couper toutes les connexions d'une base de donnée :

```
SELECT
    pg_terminate_backend(pid)
FROM
    pg_stat_activity
WHERE
    pid <> pg_backend_pid()
    AND datname = '<DBNAME>';
```

Révision #2
Créé 28 janvier 2020 11:17:22 par Dryusdan
Mis à jour 28 janvier 2020 17:37:50 par Dryusdan