[PgBadger]
Utilisation avancée

Auteur: Gilles Darold

PgDays France Juin 2019

PgBadger est simple

Un objectif : transportable, pas de prérequis par défaut !

pgbadger /var/log/postgresql/*postgresql-11.log*

PgBadger est puissant !

Auto-détection des formats de log.

Traitement parallèle des fichiers de log.

Rapport professionnel et complet.

http://pgbadger.darold.net/samplev7.html

S’adapter aux formats de log

log_destination = stderr, syslog, csvlog, jsonlog

--prefix ‘%t [%p] : usr=%u,db=%d,client=%h ‘

La simplicité ne fait pas tout !

Satisfaire tous les cas possibles.

Plus de 80 options en ligne de commande !

pgbadger --help

Options de filtrage des logs

Uniquement les erreurs : --watch-mode

Filtres temporels : --begin, --end, --include-time, --exclude-time

Filtres sur Ip / base de données / utilisateur / application :

--dbclient, --dbname, --dbuser, --appname
--exclude-client, --exclude-db, --exclude-user, --exclude-appname

Filtres sur les lignes du log :

--exclude-line,
--include-query, --exclude-query
--include-file, --exclude-file

```
pgbadger --begin='07:00' --end='22:00' --exclude-appname='psql'   \
         --exclude-user='postgres' --dbname='dbprod01' postgresql-11.log
```

Options de contrôle des rapports

Possibilité de supprimer des sections du rapport :

--disable-* : error, type, query, session, connection, lock, temporary, checkpoint, autovacuum, hourly

- Supprimer tous les graphes du rapport : --nograph
- Désactiver le formatage des requêtes : --nohighlight et --no-prettify
- Changer le titre (par défaut « pgBadger ») : --title
- Anonymisation des requêtes : --anonymize

Détails plus fins :

- --average, 5 minutes par défaut
- --histo-average, 60 minutes par défaut

Augmenter le nombre de requêtes du rapport : --top (20), --sample (3)

Modifier les rapports

Le graphisme du rapport ne vous convient pas ?

Modifiez la feuille de style dans le répertoire des sources :

resources/pgbadger.css

et l'intégrez les modifications au script pgbadger :

perl tools/updt_embedded_rsc.pl

Nécessite la présence de *yui-compressor* pour la minimisation du code.

Générer vos propres rapports 1/2

Vous voulez générer votre propre rapport ou utiliser uniquement certains métriques remontés par pgBadger ?

Aucun problème, utilisez la sortie binaire pouvoir manipuler les données extraites par pgBadger.

```
pgbadger -o pgbadger-data.bin -j 4 postgresql-11.log
```

Inspirez vous du script Perl « *tools/pgbadger_tools* » pour savoir comment manipuler les données.

Générer vos propres rapports 2/2

Vous ne maîtrisez pas bien le langage Perl ?

Aucun problème, utilisez la sortie JSON pour pouvoir manipuler les données dans votre langage de programmation favori.

Utilisez *--prettify-json* pour une lecture humaine du fichier de sortie.

```
pgbadger --prettify-json -o pgbadger-data.json -j 4 postgresql-11.log
```

Nécessite l’installation du module Perl JSON::XS.

Exemple de sortie JSON


```
"top_slowest" : {
   "postgres" : [
      [
         "112.775",
         "2019-01-26 08:35:48",
         "UPDATE pgbench_branches SET bbalance = bbalance + 4967 WHERE bid = 5;",
         "pgbench",
         "gilles",
         "[local]",
         "pgbench",
         null,
         null
      ]
   ]
},
"host_info" : {
   "postgres" : {
      "127.0.0.1" : {
         "UPDATE" : 100986,
         "SELECT|duration" : 1798.56000000015,
         "INSERT|duration" : 1650.54599999992,
         "SELECT" : 33663,
         "DDL|duration" : 6.472,
         "DDL" : 1,
         "OTHERS" : 2,
         "count" : 235631,
         "UPDATE|duration" : 11544.1209999998,
         "TCL" : 67320,
         "OTHERS|duration" : 12.549,
         "duration" : 39780.7109999942,
         "INSERT" : 33659,
         "TCL|duration" : 24768.4630000004
```

Problème des log intensifs

Avec une instance à très fort volume d'activité tracer toutes les requêtes peut générer une perte de performances.

log_min_duration_statement > 0ms

permet de ne tracer que les requêtes les plus lentes.

Pas satisfaisant si l'on veut avoir une vision de l'activité réelle sur l'instance.

PostgreSQL log sample

PostgreSQL v12 supporte l'échantillonnage des logs

log_transaction_sample_rate = 0.01

permet ici de tracer toutes les requêtes pour 1 % des transactions.

log_statement_sample_rate = 0.01

permet de ne tracer qu'un certain pourcentage des requêtes.

Merci à Adrien Nayrat pour le patch !

Extension pg_sampletolog

Pour les versions précédant PostgreSQL v12, installez l'extension *pg_sampletolog* :

https://github.com/anayrat/pg_sampletolog/

Même comportement que la fonctionnalité de PostgreSQL v12.

Compatible pgBadger à partir de la version 2.0

Mode incrémental (1/2)

Mettre à disposition de manière permanente les rapports.

pgbadger -I -O /var/www/pgbadger/ /var/log/postgresql/*.log

Exécution périodique, tous les jours ou toutes les heures.

Peu importe si les logs ont déjà été traités, pgbadger ne les reprendra pas.

Rapport accessible via : http://server.ip/pgbadger/

http://pgbadger.darold.net/demov6/index.html

Mode incrémental (2/2)

Limiter la rétention des statistiques en semaines : --retention

Séparer les données collectées des rapports :

pgbadger -I -O /*var*/lib/pgbadger -H /*var*/html/pgbadger/

Chaque rapport embarque les feuilles CSS et le code Javascript

-X, --extra-files : externalise les fichiers

Reconstruction des rapports depuis les statistiques : --rebuild

L'index global commence la semaine le dimanche : --start-monday

Traitement distant des logs

pgBadger peut être exécuté sur un serveur central et traiter des logs d'autres machines.

Mise à disposition des fichiers logs par accès SSH :

pgbadger --remote-host 192.168.1.100 /*home*/pglogs/*.log (obsolète)

pgbadger ssh://user@192.168.1.100//*home*/pglogs/*.log

Mise à disposition des logs via protocole HTTP :

pgbadger https://192.168.1.100/pglogs/user1/postgresql-11.log

Cloud ready ?

AWS RDB a un préfixe imposé [1] :

```
pgbadger -f stderr -p '%t:%r:%u@%d:[%p]:' postgresql.log
```

Azure a aussi un préfixe imposé moins complet [2]:

```
pgbadger -v -f stderr -p ‘%t-%c-‘ logs/postgresql.log
```

Heroku utilise un format spécial « logplex », il est supporté :

```
heroku logs -p postgres | pgbadger -o outfile.html  -
```

[1]https://codeinthehole.com/tips/using-pgbadger-with-aws-rds/
[2]https://www.danvy.tv/azure-postgresql-pgbadger/

Hébergement mutualisé

Les logs et les rapports pgBadger sont liés à une instance.

Obtenir un rapport (*nomdb-out.html)* par base de donnée :

pgbadger --explode /*var/log/postgresql/postgresql-11.log*

Pour le mode incrémental :

pgbadger --explode -I -O /*var/*lib/pgbadger -H /*var*/html/pgbadger/ ...

Un sous répertoire par base de données avec son rapport.

Les informations liées à l'instance (ex : checkpoints) sont reportées dans le rapport de la base postgres.

Merci ! Des questions ?

http://pgbadger.darold.net/

gilles@darold.net