PgConf.eu 2017 – pgBadger@Dev meeting

pgBadger interoperability

Developer : Gilles Darold
Target: Analyze PostgreSQL Logs
Ancestor: pgFouine
History: v1.0 on 2012

Why using pgBadger from your app?

Most information can be read dynamically from statistics tables.

Some others not.

“ I know everything about your logs! ”

Information only from logs

- Sessions duration
- Queries that wait the most
- Most frequents events (error, warning, ...)
- Stores examples of queries with params
- Show auto_explain plan

pgBadger has an incremental mode to aggregate statistics over the week.

How to get statistics from pgBadger

Default: HTML report

- great to have a standalone report
- but not very useful for your app

Binary, very efficient to load and manipulate.

- but Perl centric

JSON output for every one!

Use pgBadger JSON reports

Build your own reports from pgBadger:

```
$ pgbadger -w /var/log/postgresql/* -o out.json
```

then parse “out.json” and build your custom report for your application.

(-w : only reports events/errors statistics)

PgConf.eu 2017 – pgBadger@Dev meeting

Need some more work to:

- Describe the JSON document structure.
- Add an example like for binary output with tools/pgbadger_tools script.

But need time overall!

PgConf.eu 2017 – pgBadger@Dev meeting

“ But wait I have a companion… “

PgFormatter interoperability

Can be called from command line or as a web app to format:

- SQL queries
- PLPGSQL code

```
$ echo "SELECT * FROM table;" | ./pg_format
SELECT
    *
FROM
    TABLE;
```

Demo: http://sqlformat.darold.net/

PgConf.eu 2017 - pgFormatter@Dev meeting

Two output format:

- Text or HTML at command line (-F)
- HTML in CGI mode

```
$ perl pg_format -F html -
select * from tb;

<span class="kw1_u">SELECT</span>
    <span class="sy0">*</span>
<span class="kw1_u">FROM</span>
    tb;
```

PgConf.eu 2017 – pgFormatter@Dev meeting

Ask for features!
Contribute!