Open in app ↗

Medium Search Write

PostgreSQL Recipes: Log Files Analysis with Bash Command Line

Josef Machytka

4 min read · 18 hours ago

1

In my previous article, I described how to log all statements in PostgreSQL into a log file. This is very important when analyzing problems such as locks, performance issues, etc. However, a log file with extended logging can quickly grow very large. So, how can we analyze it effectively?

In this article, I will describe ad-hoc analysis from the Linux command line. These commands are particularly useful when I have only remote terminal access and cannot download log files or install additional software on the remote machine. In such situations, I must rely on the standard Linux tools that are typically available.

Of course, there are many problems we might need to analyze in PostgreSQL logs, and I cannot cover all of them. Instead, I'll share some techniques that I find most useful. You can easily combine them to get other results tailored to your needs.

Log Analysis from the Linux Command Line

Linux provides simple yet powerful tools that we can combine to quickly get an overview and some basic statistics from PostgreSQL log files.

Select Log Entries for a Specific Time Range

We can use the `awk` command to extract log entries for a specific time range and then process them further in the pipeline:

```
awk '/2025-01-06 16:44:00/,/2025-01-06 16:46:59/' postgresql.log | ....
```

Statistics of Errors

As I described in my previous article about logging statements, we can customize prefix of the logging line to print multiple valuable information. But no matter how we customize it, the logging message itself will be always the same. PostgreSQL uses multiple prefixes in messages to distinguish content of the message: ERROR, WARNING, DETAIL, STATEMENT, LOG etc. We can very simply get basic overview of errors from log using:

```
grep "ERROR" postgresql.log|awk -F 'ERROR:' '{print $2}'|sort|uniq -c
```

Example output:

```
      1   canceling statement due to lock timeout at character 29
     30   duplicate key value violates unique constraint "groups_unique_key"
   1020   duplicate key value violates unique constraint "users_unique_key"
```

```
31   duplicate key value violates unique constraint "user_info_pkey"
 8   duplicate key value violates unique constraint "locations_unique_key"
```

Select Top 10 Slowest Queries

If statements are logged (log_statement=all), we can select the 10 slowest queries. The command truncates each query to the first 150 characters for better readability (some queries can be extremely long). This simplified version assumes that the statement is logged on the same line:

```
grep "duration:" postgresql.log |awk -F 'duration: | ms  ' '{split($3, a, "state
```

Example output:

```
2288.651 ms - SELECT order_data.id, m.field_id, m.creation_date, order_data.type
819.534 ms - SELECT template.id FROM template, order_data, contact_status s, con
445.420 ms - SELECT lt_order.order_id, lt_order.type, lt_order.ltr_id FROM lt_or
328.381 ms - SELECT patient.id, patient.type, patient.creation_date, patient.upd
280.101 ms - SELECT billing_state.workflow_state, billing_state.check_state, bil
216.010 ms - SELECT order_data.id, diag.type FROM  diagnosis diag, order_data, t
211.970 ms - SELECT order_data.id, m.field_id, m.creation_date, order_data.type,
210.548 ms - SELECT order_data.id, m.field_id, m.creation_date, order_data.type,
173.405 ms - SELECT template.id FROM template, order_data, contact_status s, con
146.788 ms - SELECT order_data_history.id, m.field_id, m.creation_date, order_da
```

If a statement is missing or incomplete, search for the query runtime in the original log. These runtimes are usually unique enough to narrow down the results.

Top IPs Accessing the Database

If connections are being logged (log_connections=on), we can find the IP addresses from which the database is most frequently accessed, use:

```
grep "connection received" postgresql.log|awk -F 'host=| port=' '{print $2}'|sor
```

Example output:

```
   1264 10.112.199.140
    915 [local]
    655 10.112.233.8
    650 10.112.233.9
    636 10.112.233.5
    633 10.112.233.6
    632 10.112.233.10
    629 10.112.233.11
    622 10.112.233.4
    610 10.112.233.7
```

Count Disconnections Per Minute

If disconnections are being logged (log_disconnections=on), we can analyze for example amount of disconnections per minute. Here's a simple way to do this:

```
grep disconnection postgresql.log |cut -d':' -f1-2|sort| uniq -c
```

Example output:

```
371 2024-10-07 20:00
348 2024-10-07 20:01
401 2024-10-07 20:02
255 2024-10-07 20:03
242 2024-10-07 20:04
365 2024-10-07 20:05
353 2024-10-07 20:06
418 2024-10-07 20:07
```

Top 5 Slowest Checkpoints

If checkpoints are being logged (log_checkpoints=on), we can list slowest checkpoints using:

```
grep "checkpoint complete" log_incident_iptv.txt|awk -F 'total=' '{print $2}'|aw
```

Example output:

```
270.073 s
96.715 s
58.153 s
56.039 s
48.809 s
```

Summary

These are just a few examples of ad-hoc analyses that can be performed on PostgreSQL log files using the Bash command line. In the next article, I'll discuss how to use `pgBadger` for more detailed log analysis.

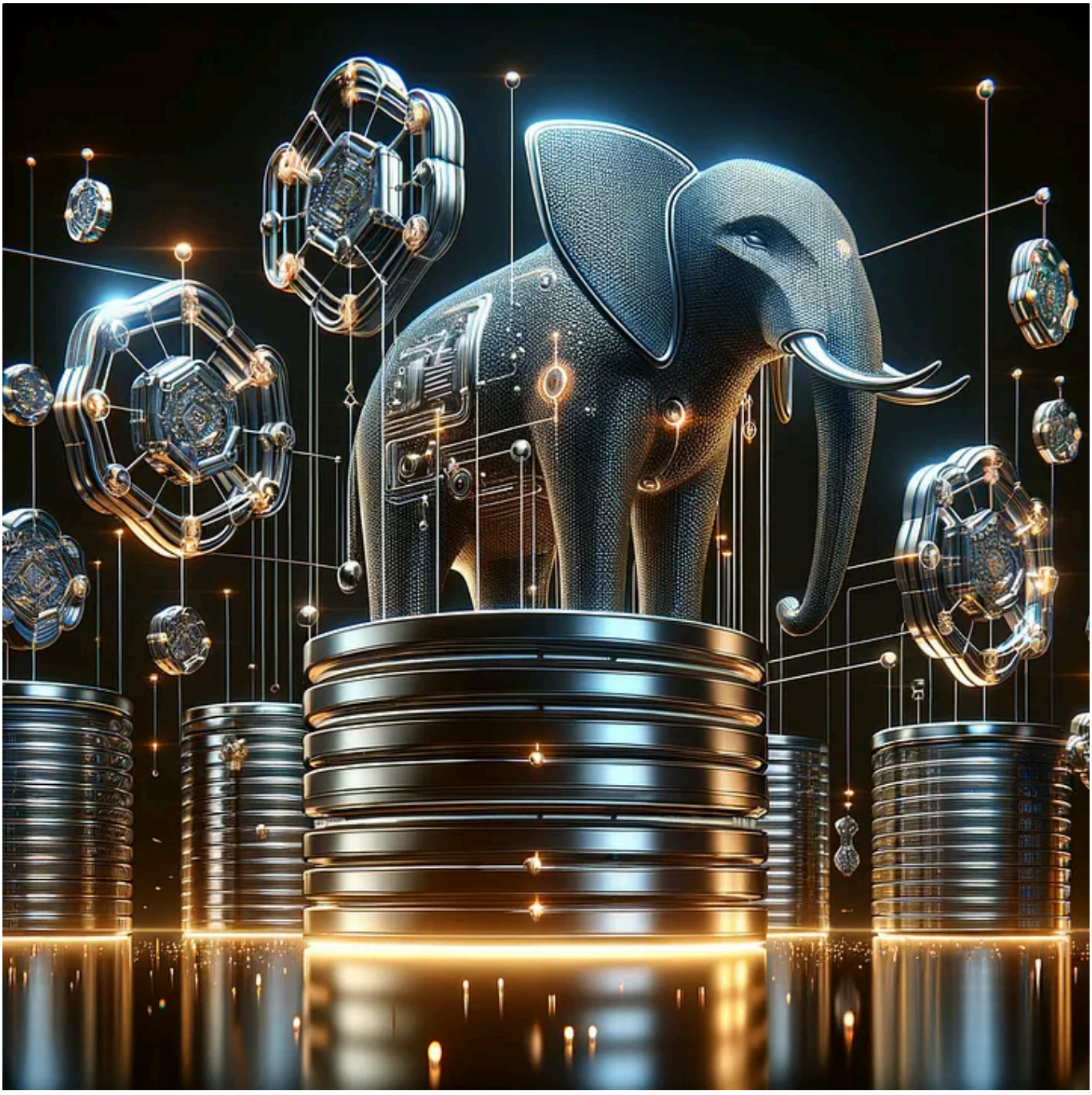


Image created by the author using DeepDreamGenerator

Postgresql Bash Log File Analysis

Written by Josef Machytka

51 Followers · 10 Following

Edit profile

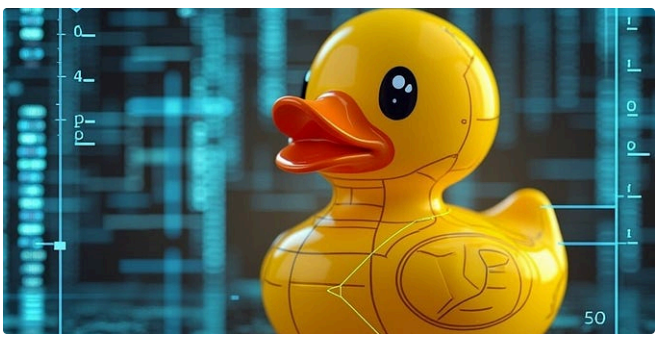
I work as PostgreSQL specialist & database reliability engineer at NetApp Deutschland, Open Source Services division.

No responses yet

What are your thoughts?

Respond

More from Josef Machytka



 Josef Machytka

 Josef Machytka

DuckDB Database File as a New Standard for Sharing Data?

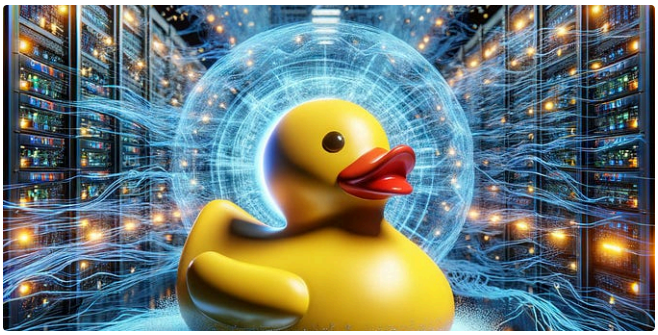
This is not my original idea; I came across it in an excellent article titled “DuckDB Beyond th...

Dec 30, 2024 17 2

DuckDB as a Rudimentary Data Migration Tool

After exploring how to use DuckDB as an intelligent ETL tool for PostgreSQL, and how...

Nov 30, 2024 26 2

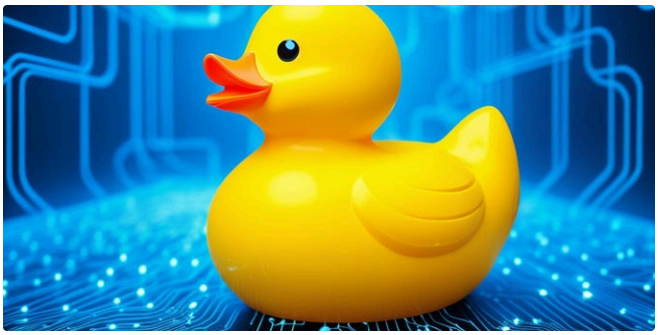


Josef Machytka

Quick and Easy Data Exports to Parquet Format Using DuckDB

The Parquet format has become almost an industry standard for Data Lakes and Data...

Dec 5, 2024 11



Josef Machytka

Extending DuckDB ETL Capabilities with Python

DuckDB has recently become my go-to solution for small ETL tasks. It is an...

Nov 24, 2024 26

See all from Josef Machytka

Recommended from Medium

In Dev Genius by Aleksei Aleinikov

10 Ways to Work with Large Files in Python: Effortlessly Handle...

Handling large text files in Python can feel overwhelming. When files grow into...

Dec 1, 2024 225 3

Romanchechyotkin

Make MinIO and Postgresql work together

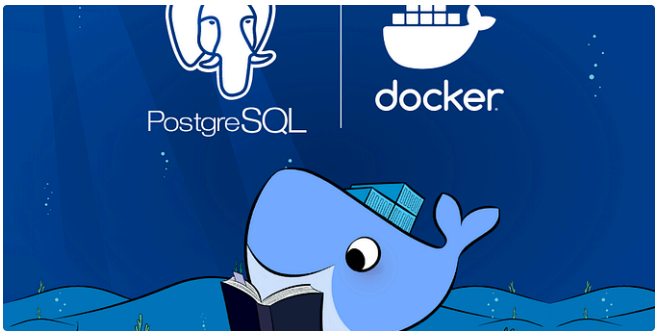
Hello eveeryone, in this article i want to tell you how i did integration between MinIO and...

Jul 31, 2024 65

Lists

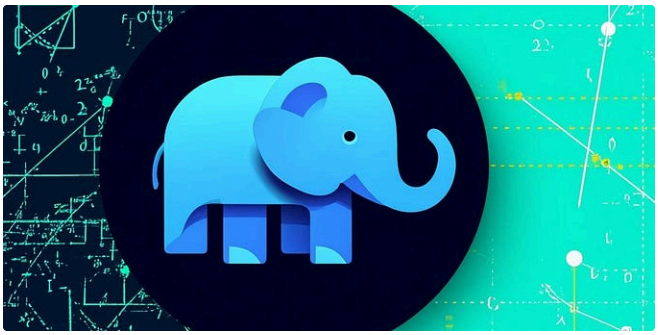
General Coding Knowledge

20 stories · 1850 saves



Andy Pickup

Developing in Python with Dev Containers—Part 5: Working with...



Phantompete

Vector Search with pgvector and OCI Database with PostgreSQL—...

Setting up PostgreSQL in a local container and ensuring network connectivity with our...

Dec 30, 2024 7

In this post we will go through the basics of leveraging pgvector combined with OCI...

Dec 5, 2024 4

Nishaanth S P

PostgreSQL—See how data is stored!

Both tables and materialized views are stored in files, with each file having a minimum size...

Dec 20, 2024 60

In Towards Data Engineering by Sandun Lakshan

10 Best Python Text User Interface (TUI) Libraries for 2025

Text-based user interfaces (TUIs) are a simple way to create interactive applications that ru...

Dec 11, 2024 34

See more recommendations