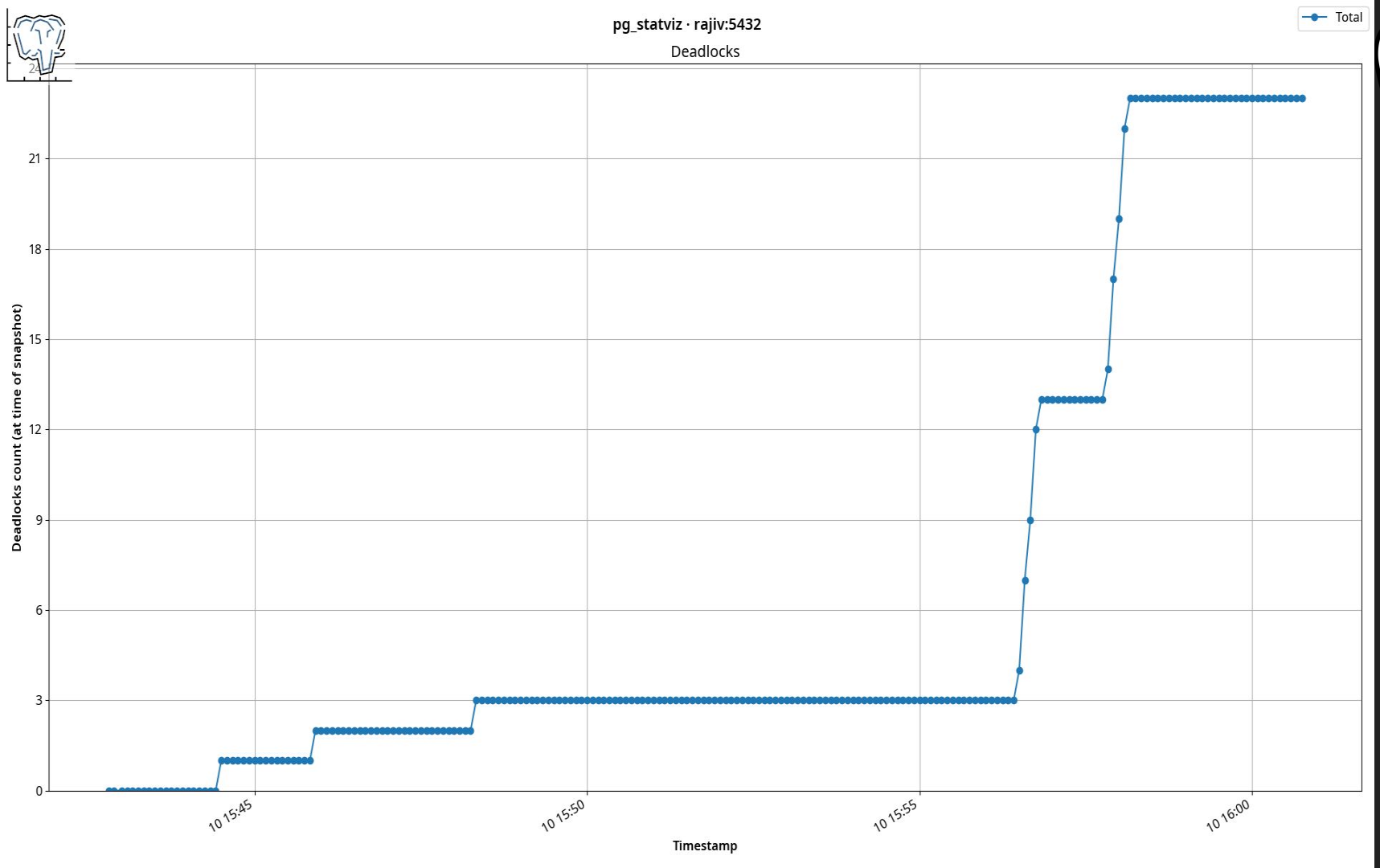
Charting My Path: Journey of a GSoC 2023 Contributor to PostgreSQL with pg_statviz

Rajiv Harlalka

One Fine Morning.....


pg_statviz · rajiv:5432
Deadlocks
Total
Deadlocks count (at time of snapshot)
0
3
6
9
12
15
18
21
10 15:45
10 15:50
10 15:55
10 16:00
Timestamp

$ whoami

Rajiv Harlalka

- A student from India
- Member of Kharagpur Open Source Society (kossiitkgp.org), a student led club at IIT Kharagpur, India
- (New) Member to the PostgreSQL community
- Terminal Guy, Loves Nature and plays Squash.

$ whatis

pg_statviz

- A minimalist extension and utility pair

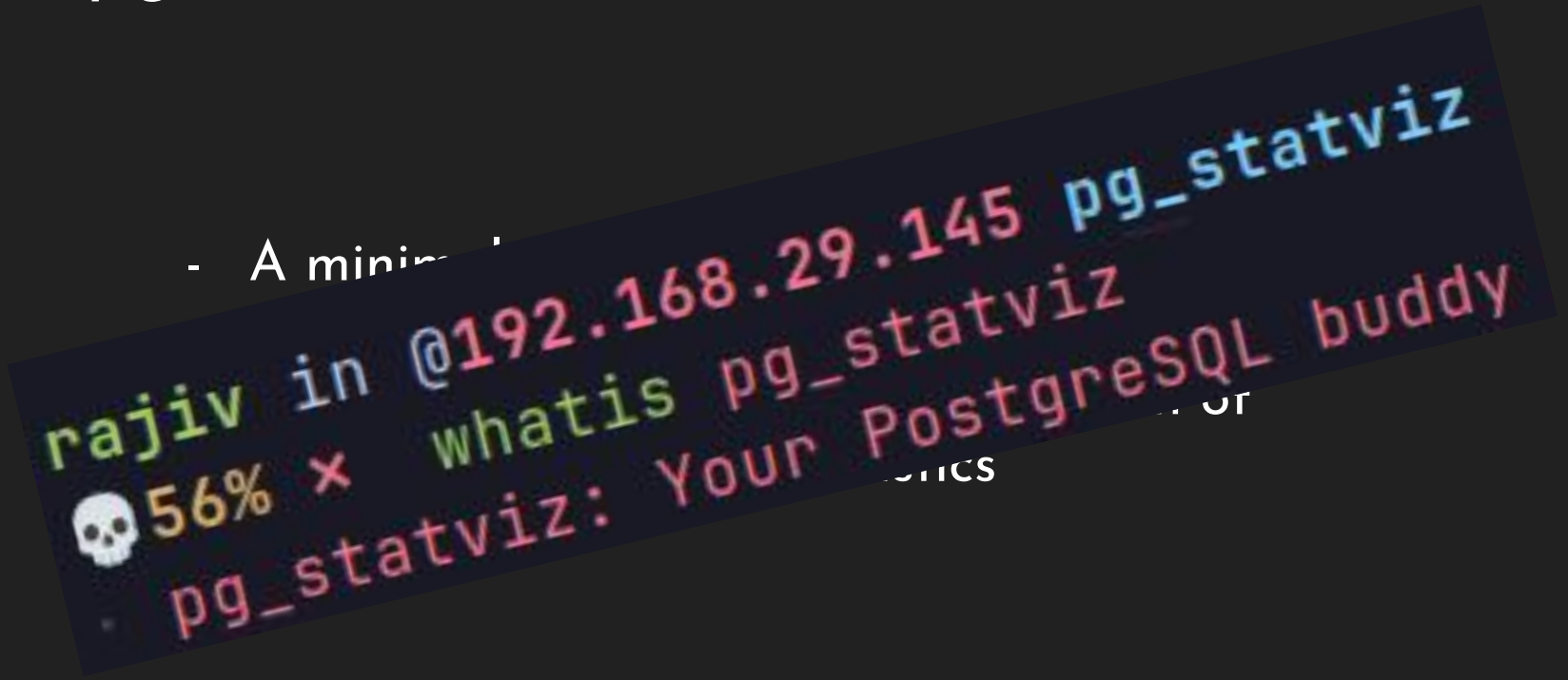
$ whatis

pg_statviz

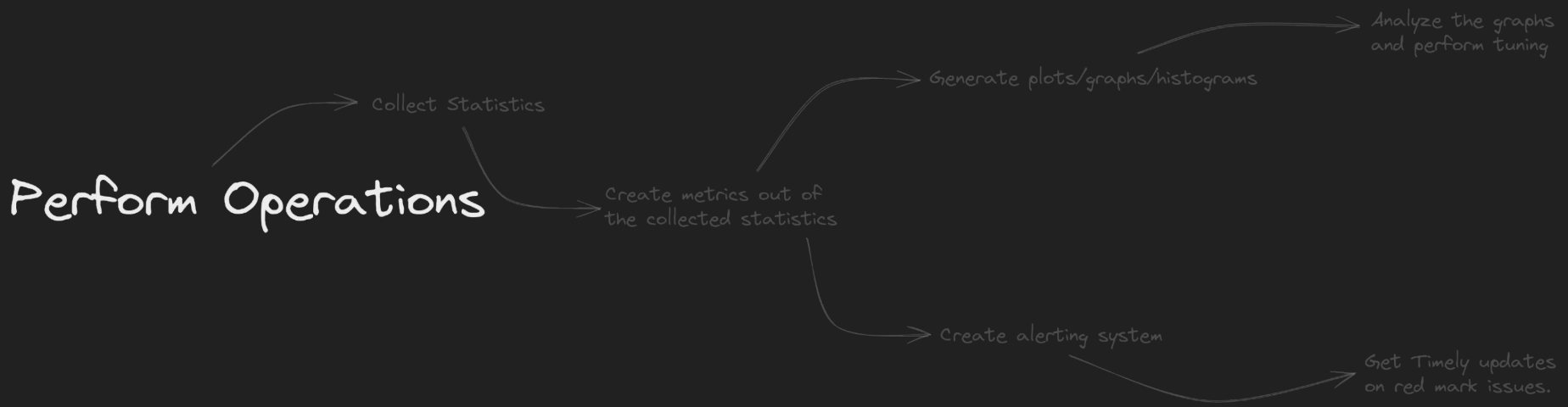
- A minimalist extension and utility pair
- Time series analysis and visualization of PostgreSQL internal statistics

$ whatis

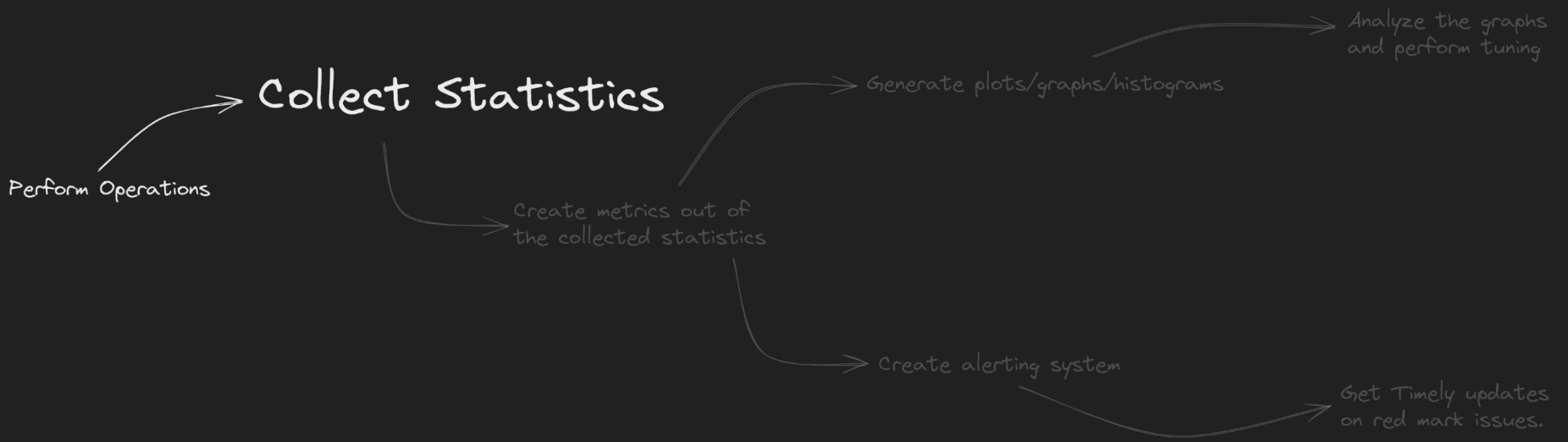
pg_statviz



A Day in Life of:

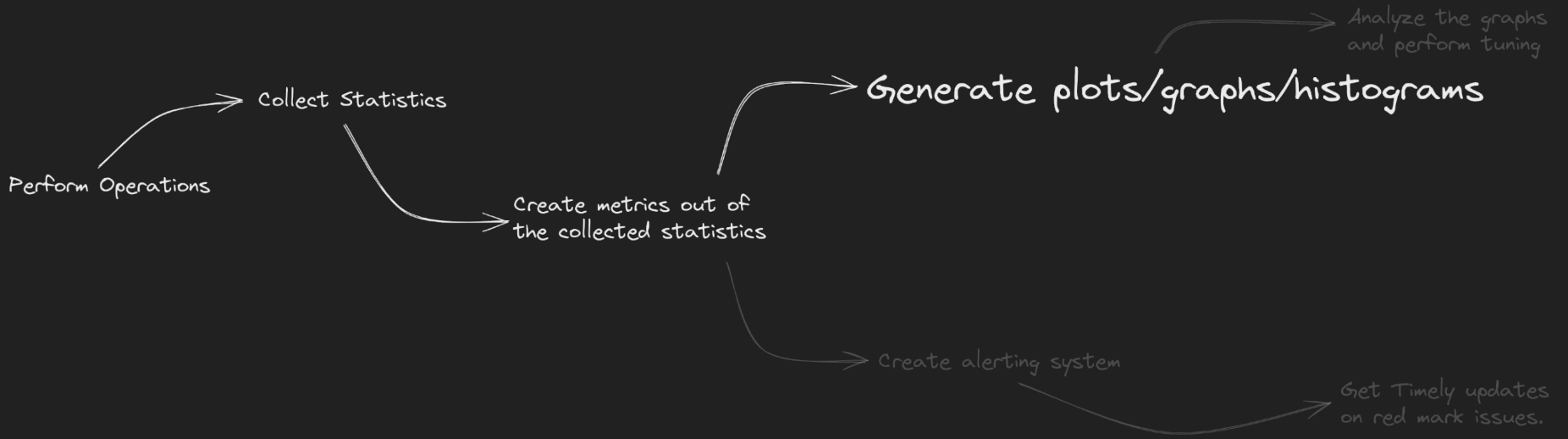


A Day in Life of:

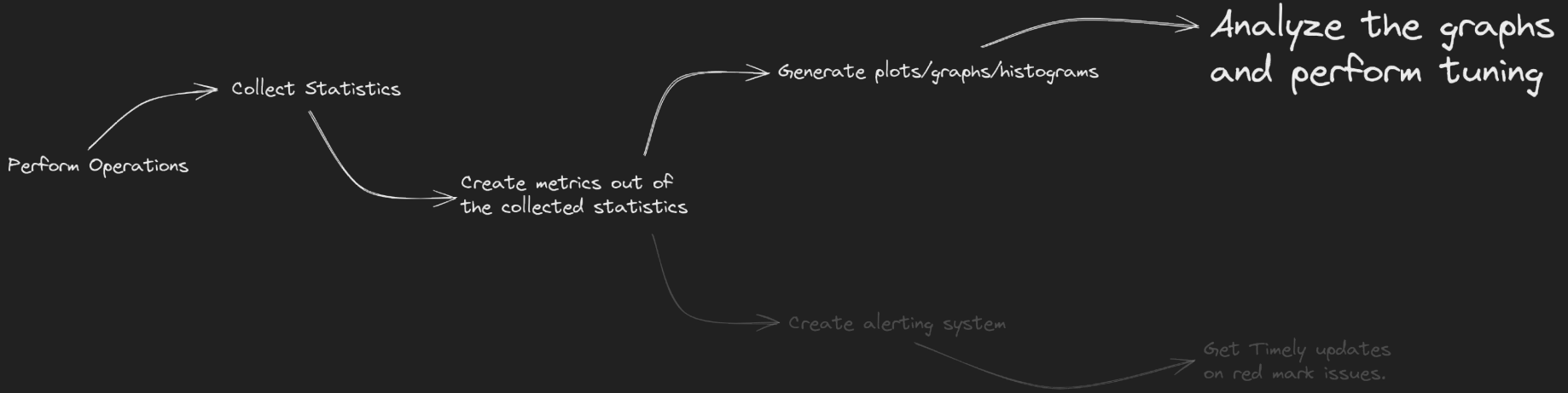


A Day in Life of:

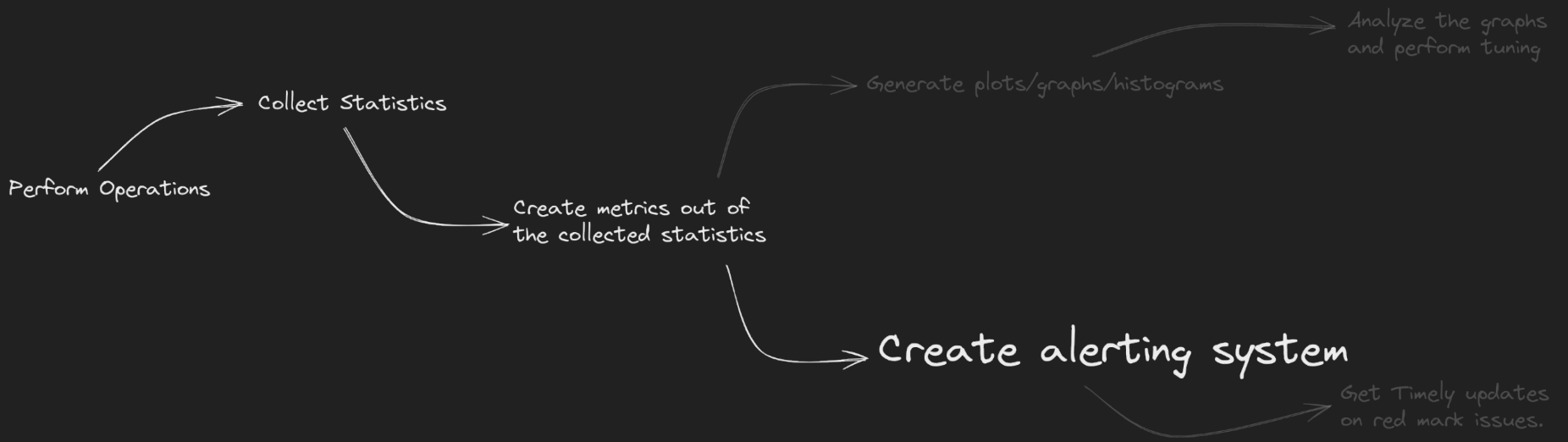
A Day in Life of:



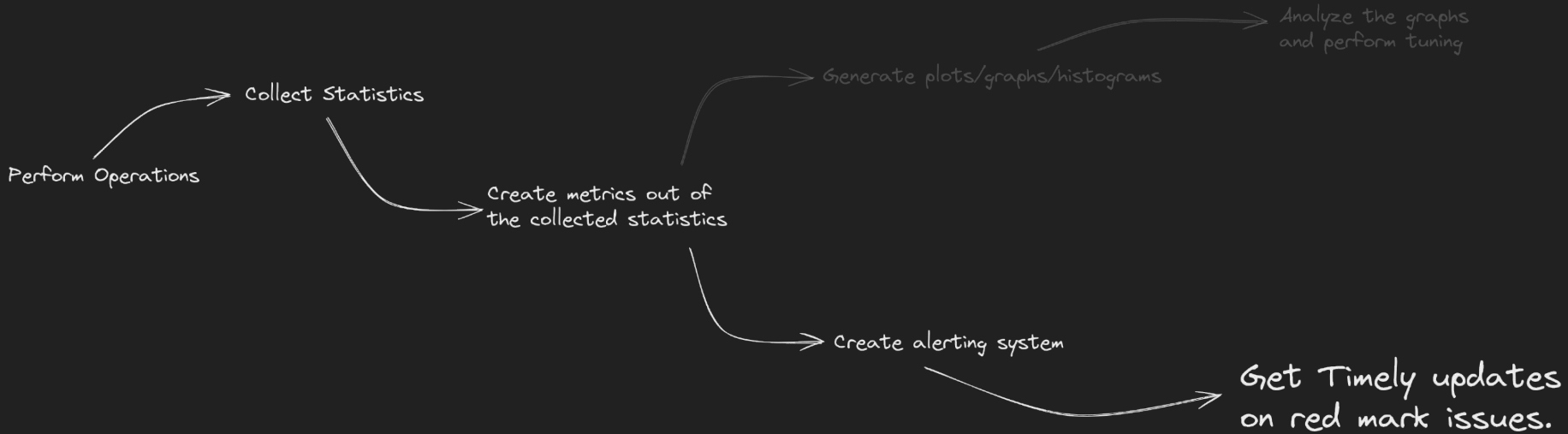
A Day in Life of:



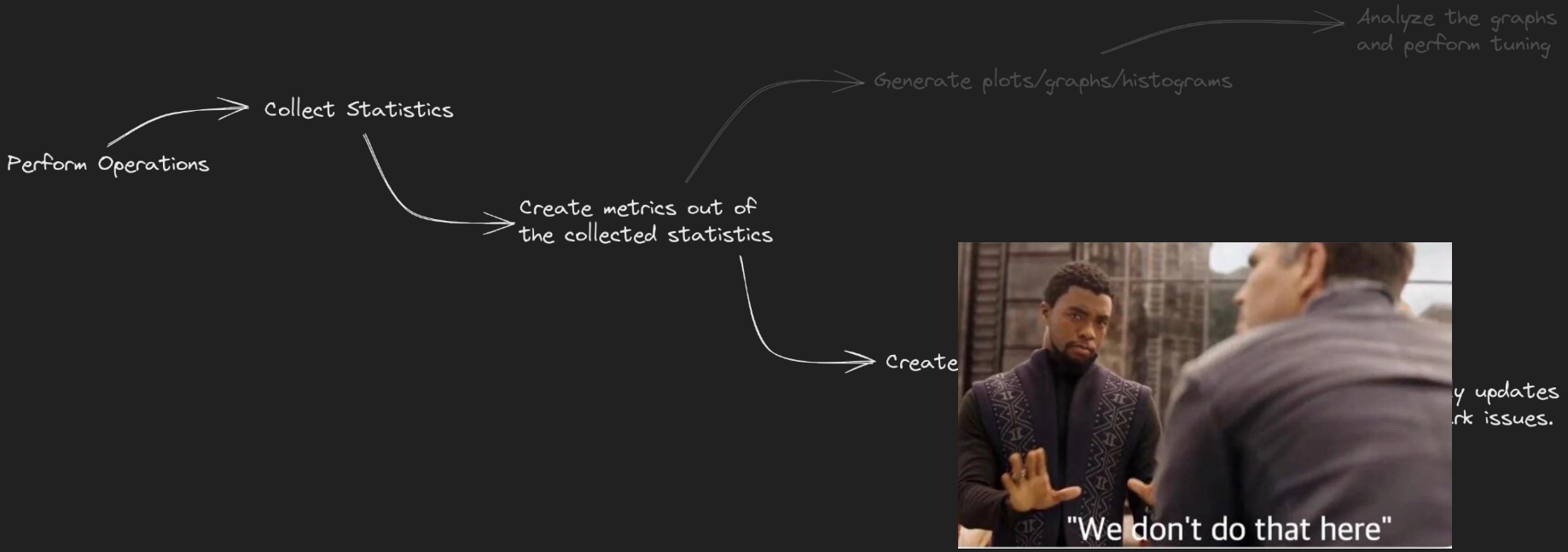
A Day in Life of:



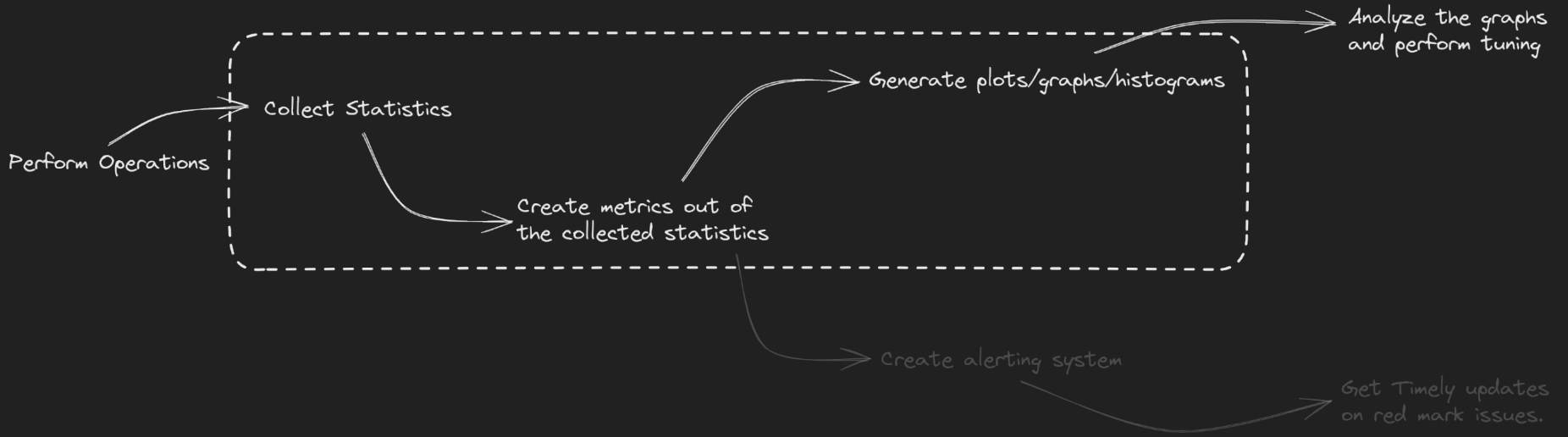
A Day in Life of:



A Day in Life of:



A Day in Life of:



What pg_statviz focuses at

Why

pg_statviz

- Track PostgreSQL performance over time and potentially perform tuning or troubleshooting

Why

pg_statviz

- Track PostgreSQL performance over time and potentially perform tuning or troubleshooting

Why

pg_statviz

- Track PostgreSQL performance over time and potentially perform tuning or troubleshooting

- Understand about your system at a look.

Why

pg_statviz

- Track PostgreSQL performance over time and potentially perform tuning or troubleshooting
- Understand about your system at a look.
- Does not need knowledge about the internal tables of PostgreSQL.

Design Philosophy

- Minimal

Design Philosophy

- Minimal
- Modular

Design Philosophy

- Minimal
- Modular
- Designed with K.I.S.S and UNIX philosophy in mind

Design Philosophy

- Minimal
- Modular
- Designed with K.I.S.S and UNIX philosophy in mind

Does what it is meant to do.

Design Philosophy

- Minimal
- Modular
- Designed with K.I.S.S and UNIX philosophy in mind

Does what it is meant to do.

Provide Insights

Design Philosophy

Components:

- PostgreSQL Extension
- Python Utility Pair

Design Philosophy

Components:

- PostgreSQL Extension
- Python Utility Pair

Design Philosophy

Components:

- PostgreSQL Extension
- Python Utility Pair

No need to restart Database

Design Philosophy

Components:

- PostgreSQL Extension
- Python Utility Pair

No need to restart Database

No addition to shared_preload_libraries

Installation Guide

Getting hold of the extension:

- **Get the packaged one**

```
sudo dnf install pg_statviz_extension-<pg_version>
```

```
pgxn install pg_statviz
```

Installation Guide

Getting hold of the extension:

- Get the packaged one

```
sudo dnf install pg_statviz_extension-<pg_version>
```

```
pgxn install pg_statviz
```

- **Be Brave and build it:**
  $ make install

Installation Guide

Extension:

- **CREATE EXTENSION pg_statviz;**

Python Utility:

- **pip install pg_statviz;**

Installati

Extension:

- **CREATE**

Python Utilit

- **pip inst**

Usage: Extension

- To create snapshots the user role should have either superuser or pg_monitor priviledge

Usage: Extension

- Role should have either superuser or pg_monitor privilege

To take a snapshot:

SELECT pgstatviz.snapshot();

Usage: Extension

- Role should have either superuser or pg_monitor privilege

To take a snapshot:

SELECT pgstatviz.snapshot();

```
(postgres@localhost:5432) 12:12:12 [postgres]
# select pgstatviz.snapshot();
NOTICE:  created pg_statviz snapshot
             snapshot
---------------------------------
 2023-12-05 12:12:20.228703+05:30
(1 row)
```

Usage: Utility Pair

```
💀68% →  pg_statviz --help
usage: pg_statviz [--help] [--version] [-d DBNAME] [-h HOSTNAME] [-p PORT] [-U USERNAME] [-W] [-D FROM TO] [-O OUTPUTDIR]
                  {analyze,buf,cache,checkp,conn,lock,tuple,wait,wal,xact} ...

run all analysis modules

positional arguments:
  {analyze,buf,cache,checkp,conn,lock,tuple,wait,wal,xact}
    analyze             run all analysis modules
    buf                 run buffers written analysis module
    cache               run cache hit ratio analysis module
    checkp              run checkpoint analysis module
    conn                run connection count analysis module
    lock                run locks analysis module
    tuple               run tuple count analysis module
    wait                run wait events analysis module
    wal                 run WAL generation analysis module
    xact                run transaction count analysis module
```

Usage: Utility Pair

```
rajiv in @192.168.29.145 pg_statviz on 🌱 update-readme [📦🤷] is 📦 v0.5
💀67% →  ./src/run_pg_statviz -d postgres -h localhost -U postgres
INFO:pg_statviz.modules.buf:Running buffers written analysis
INFO:pg_statviz.modules.buf:Saving pg_statviz_rajiv_5432_buf.png
INFO:pg_statviz.modules.buf:Saving pg_statviz_rajiv_5432_buf_rate.png
INFO:pg_statviz.modules.checkp:Running checkpoint analysis
INFO:pg_statviz.modules.checkp:Saving pg_statviz_rajiv_5432_checkp.png
INFO:pg_statviz.modules.checkp:Saving pg_statviz_rajiv_5432_checkp_rate.png
INFO:pg_statviz.modules.cache:Running cache hit ratio analysis
INFO:pg_statviz.modules.cache:Saving pg_statviz_rajiv_5432_cache.png
INFO:pg_statviz.modules.conn:Running connection count analysis
INFO:pg_statviz.modules.conn:Saving pg_statviz_rajiv_5432_conn_status.png
INFO:pg_statviz.modules.conn:Saving pg_statviz_rajiv_5432_conn_user.png
INFO:pg_statviz.modules.lock:Running locks analysis
INFO:pg_statviz.modules.lock:Saving pg_statviz_rajiv_5432_lock.png
INFO:pg_statviz.modules.tuple:Running tuple count analysis
INFO:pg_statviz.modules.tuple:Saving pg_statviz_rajiv_5432_tuple.png
INFO:pg_statviz.modules.wait:Running wait events analysis
INFO:pg_statviz.modules.wait:Saving pg_statviz_rajiv_5432_wait.png
INFO:pg_statviz.modules.wal:Running WAL generation analysis
INFO:pg_statviz.modules.wal:Saving pg_statviz_rajiv_5432_wal.png
INFO:pg_statviz.modules.wal:Saving pg_statviz_rajiv_5432_wal_rate.png
```

PostgreSQL Internal Statistics

- Statistics on Internal activity collected via Collector reported through views.
  - Most on table/index information on row & disk block levels

- 24 Cumulative Statistics Views
  - Collected through Statistics Collector
  - collection and reporting of information about server activity

PostgreSQL Internal Statistics

- Statistics on Internal activity collected via Collector reported through views.
  - Most on table/index information on row & disk block levels

- 24 Cumulative Statistics Views
  - Collected through Statistics Collector
  - collection and reporting of information about server activity

PostgreSQL Internal Statistics

- 10 Dynamic Statistics Views
  - Deals with statistics of things currently happening.
  - Eg: replication, vacuum

- Though cumulative, gone once reset.

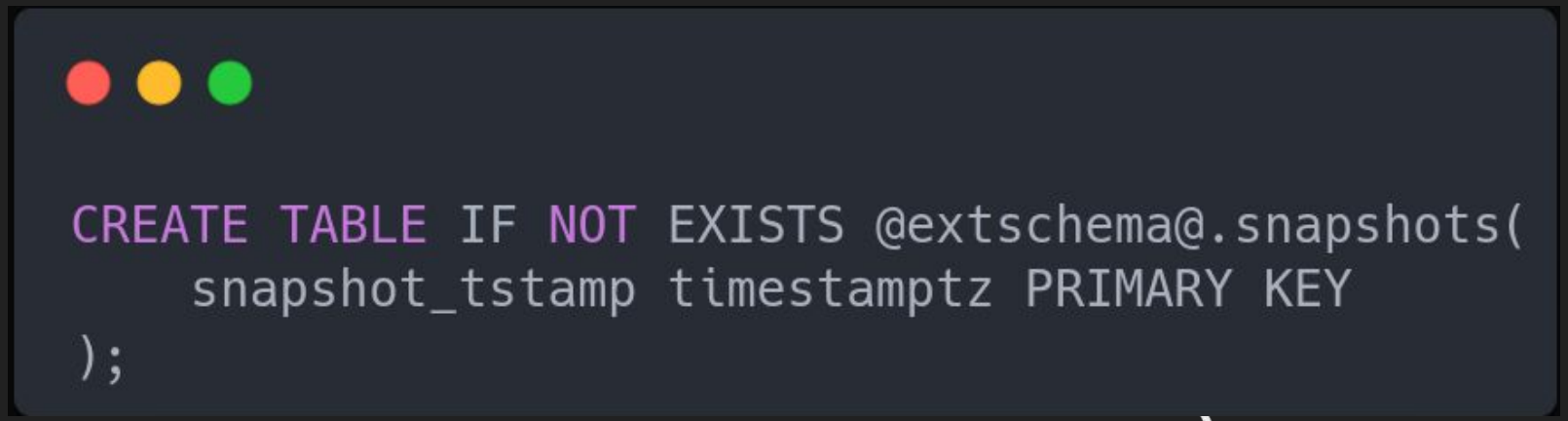
PostgreSQL Internal Statistics

- 10 Dynamic Statistics Views
  - Deals with statistics of things currently happening.
  - Eg: replication, vacuum

- Though cumulative, gone once reset.

A look behind the curtain

A look behind the curtain

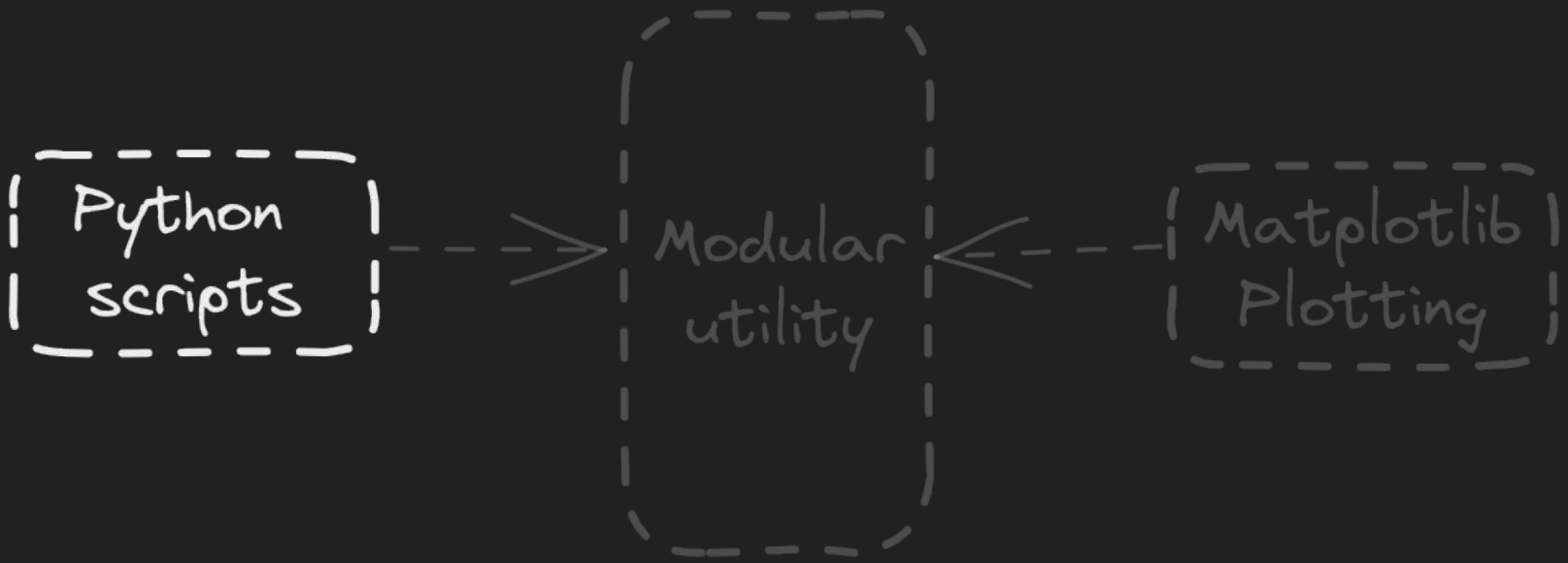


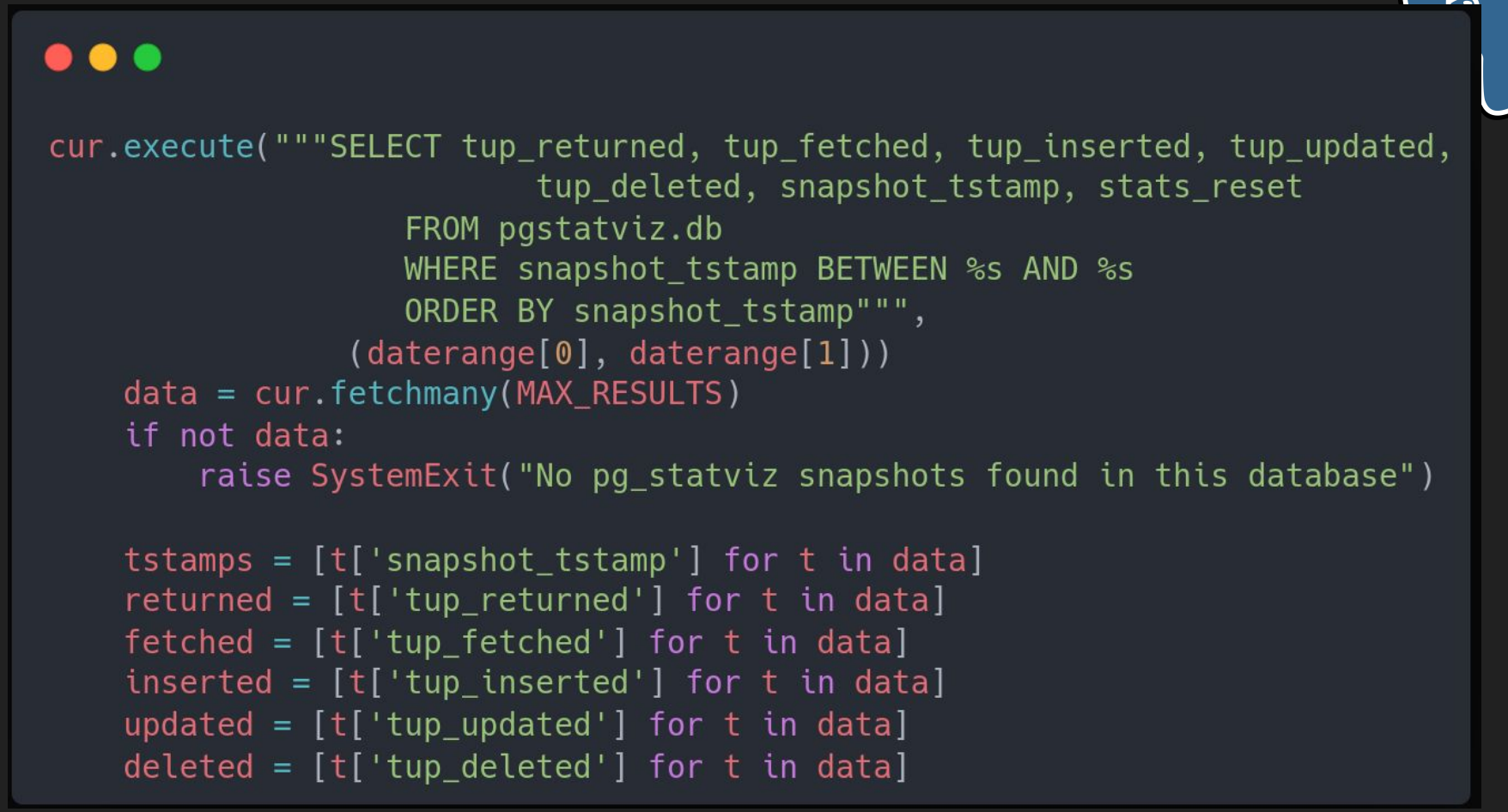
A look behind the curtain

A look behind the curtain

```
(postgres@localhost:5432) 19:18:24 [postgres]
# \df pgstatviz.*
                                        List of functions
  Schema   |       Name       |     Result data type     |          Argument data types           | Type
-----------+------------------+--------------------------+----------------------------------------+------
 pgstatviz | delete_snapshots | void                     |                                        | func
 pgstatviz | snapshot         | timestamp with time zone |                                        | func
 pgstatviz | snapshot_buf     | void                     | snapshot_tstamp timestamp with time zone | func
 pgstatviz | snapshot_conf    | void                     | snapshot_tstamp timestamp with time zone | func
 pgstatviz | snapshot_conn    | void                     | snapshot_tstamp timestamp with time zone | func
 pgstatviz | snapshot_db      | void                     | snapshot_tstamp timestamp with time zone | func
 pgstatviz | snapshot_lock    | void                     | snapshot_tstamp timestamp with time zone | func
 pgstatviz | snapshot_wait    | void                     | snapshot_tstamp timestamp with time zone | func
 pgstatviz | snapshot_wal     | void                     | snapshot_tstamp timestamp with time zone | func
(9 rows)
```

A look behind the curtain

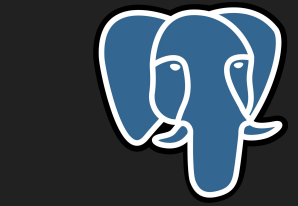



cur.execute("""SELECT tup_returned, tup_fetched, tup_inserted, tup_updated,
tup_deleted, snapshot_tstamp, stats_reset
FROM pgstatviz.db
WHERE snapshot_tstamp BETWEEN %s AND %s
ORDER BY snapshot_tstamp""",
(daterange[0], daterange[1]))
data = cur.fetchmany(MAX_RESULTS)
if not data:
raise SystemExit("No pg_statviz snapshots found in this database")
tstamps = [t['snapshot_tstamp'] for t in data]
returned = [t['tup_returned'] for t in data]
fetched = [t['tup_fetched'] for t in data]
inserted = [t['tup_inserted'] for t in data]
updated = [t['tup_updated'] for t in data]
deleted = [t['tup_deleted'] for t in data]

A look

Pytho
script

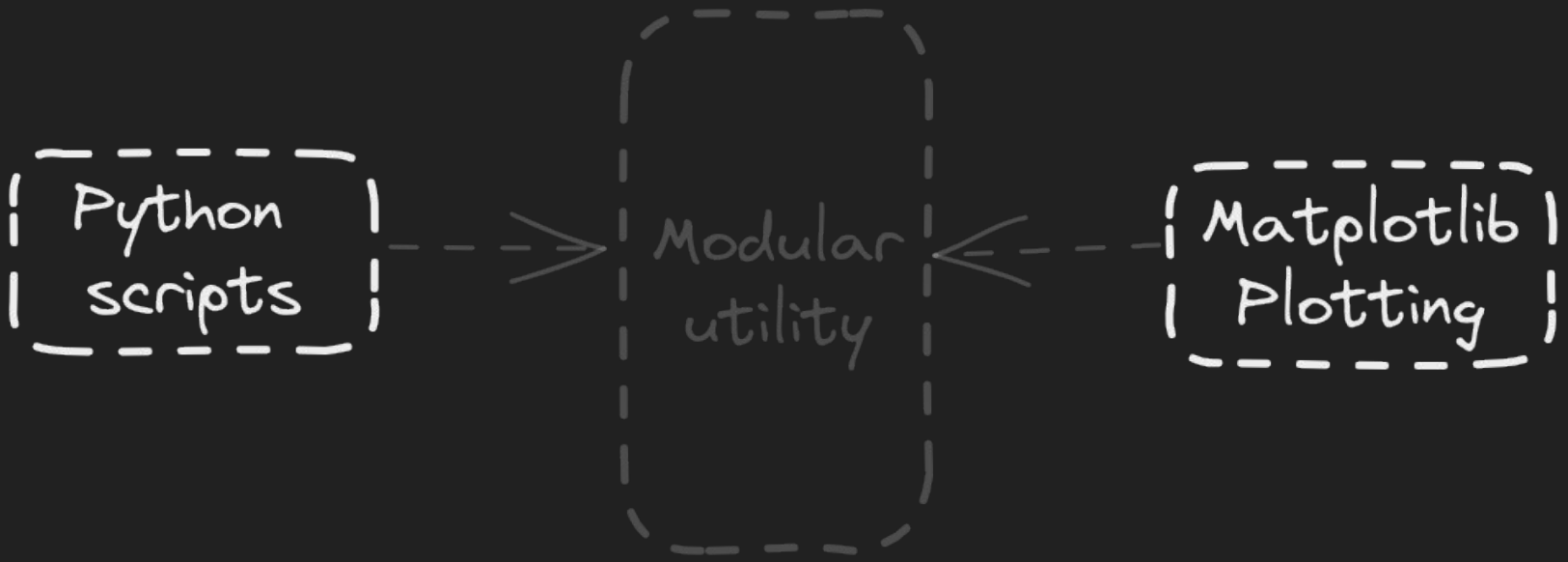
matplotlib
otting



```
waitkinds = []
for w in wevents:
    for e in w:
        if 'wait_event' in e:
            wk = {'wait_event_type': e['wait_event_type'],
                  'wait_event': e['wait_event']}
            if wk not in waitkinds:
                waitkinds += wk,

# Plot as many of each wait event kind we have per snapshot
plt, fig = plot.setup()
plt.suptitle(f"pg_statviz · {info['hostname']}:{port}",
             fontweight='semibold')
plt.title("Wait events")
for wk in waitkinds:
    wc = []
    for w in wevents:
        if not w:
            wc += 0,
        else:
            found = False
            for e in w:
                if wk.items() <= e.items():
                    wc += e['wait_event_count'],
                    found = True
            if not found:
                wc += 0
```

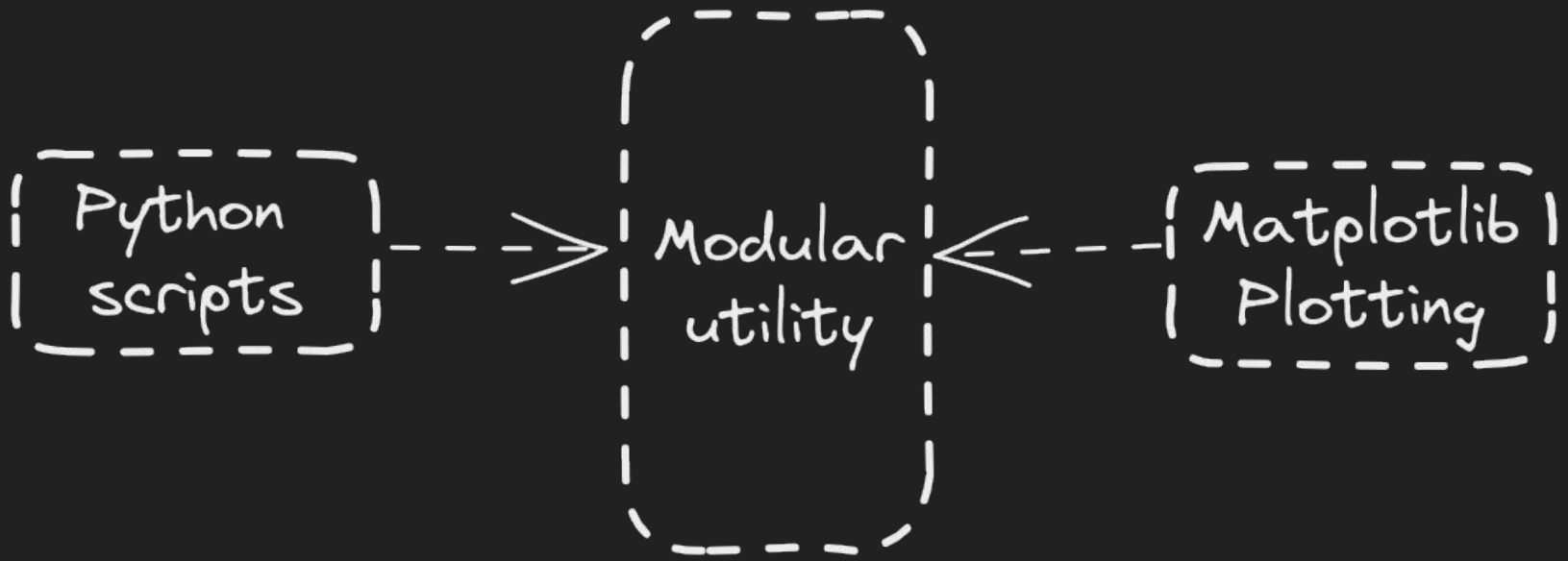
A look behind the curtain



A look behind the curtain

```python
def setup():
    for f in ["NotoSans-Regular.ttf", "NotoSans-SemiBold.ttf"]:
        f = importlib.resources.files("pg_statviz.libs").joinpath(f)
        fnt.fontManager.addfont(f)
    plt.rcParams['font.family'] = 'Noto Sans'
    plt.rcParams['font.size'] = 12
    base_image_path = importlib.resources.files("pg_statviz.libs")\
        .joinpath("pg_statviz.png")
    im = plt.imread(str(base_image_path))
    height = im.shape[0]
    fig = plt.figure(figsize=(19.2, 10.8))
    fig.figimage(im, 5, (fig.bbox.ymax - height - 6), zorder=3)
    plt.grid(visible=True)
    plt.ticklabel_format(axis='y', style='plain')
    plt.gcf().autofmt_xdate()
    return plt, fig
```

A look behind the curtain

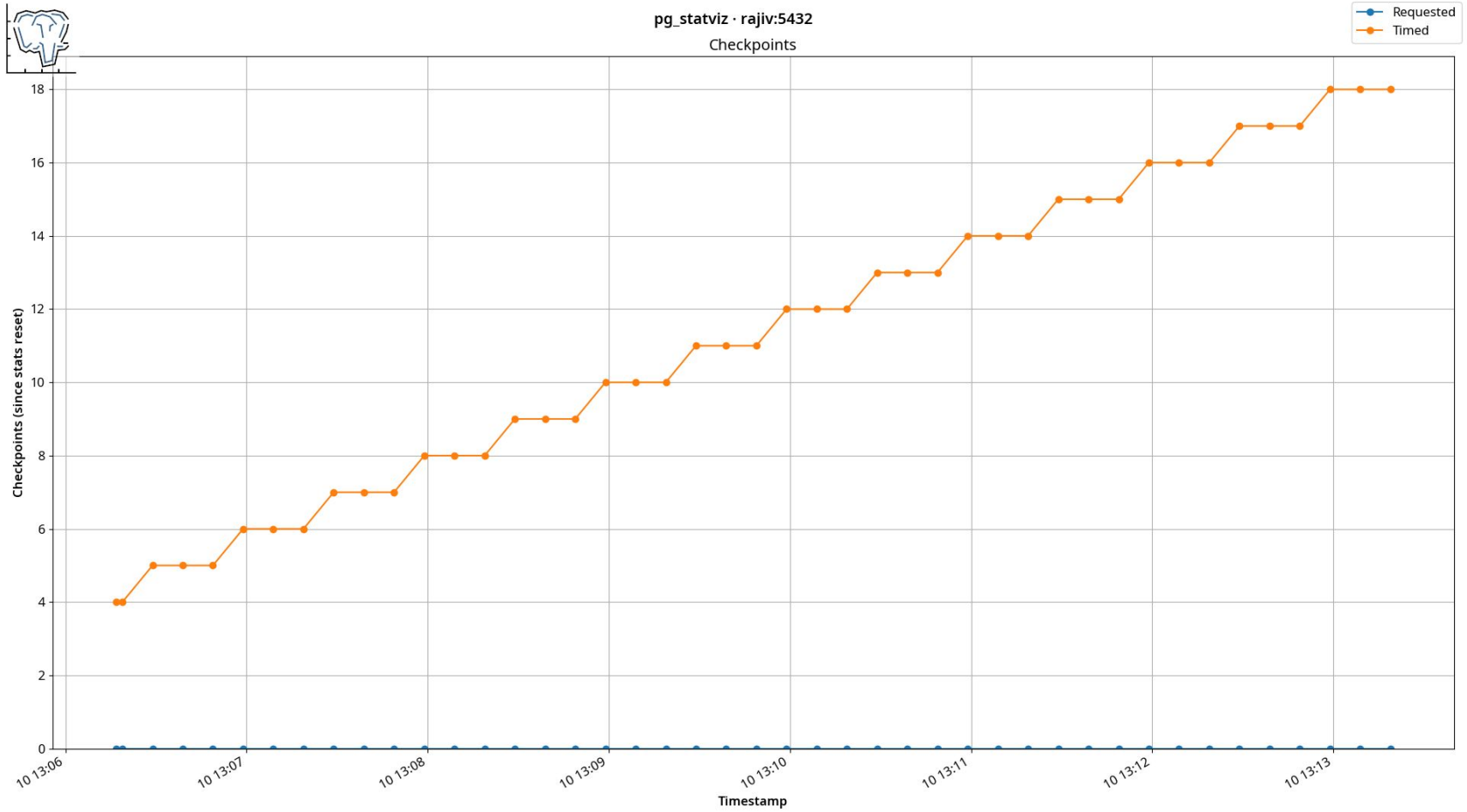


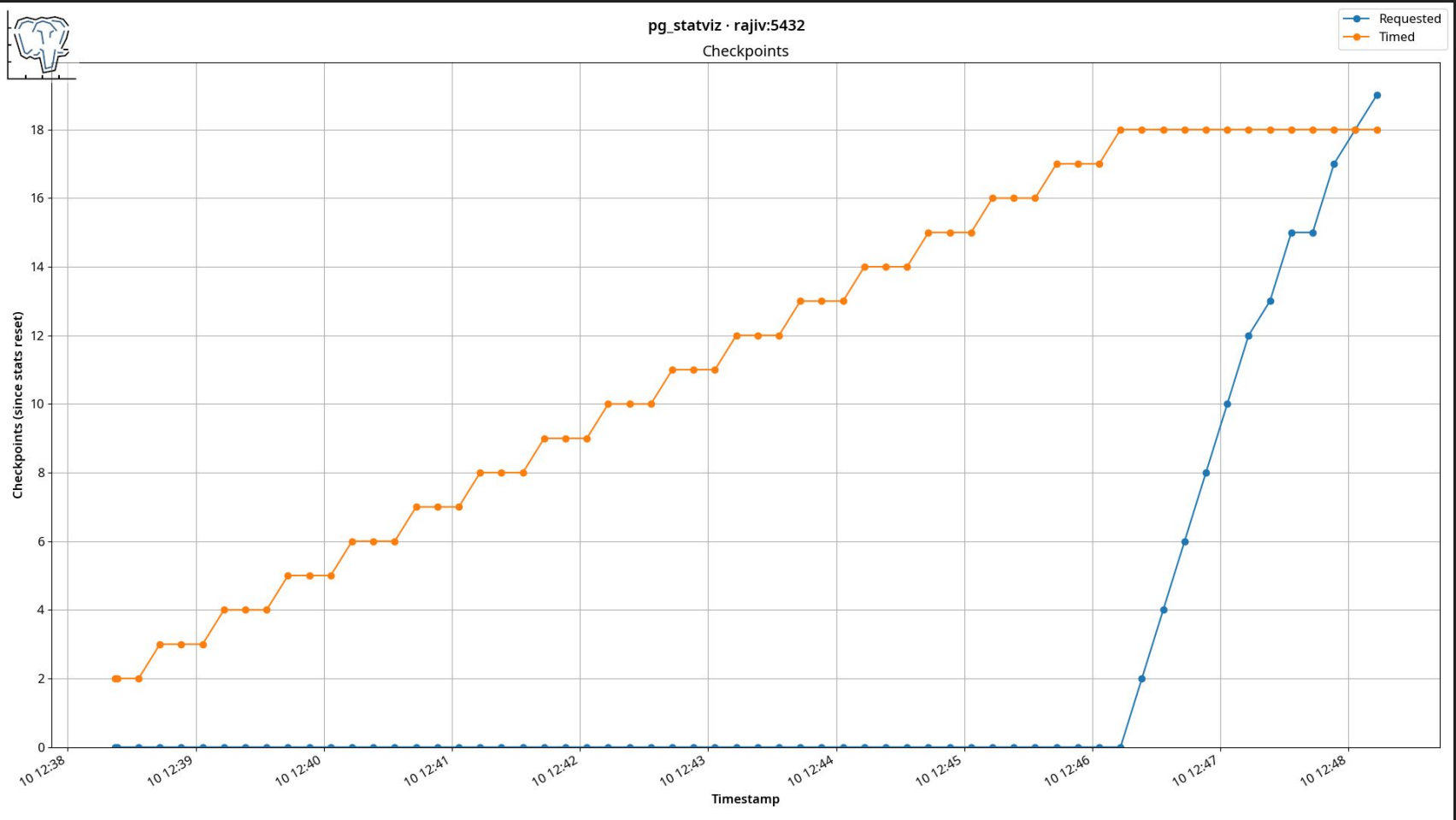
A look behind the curtain

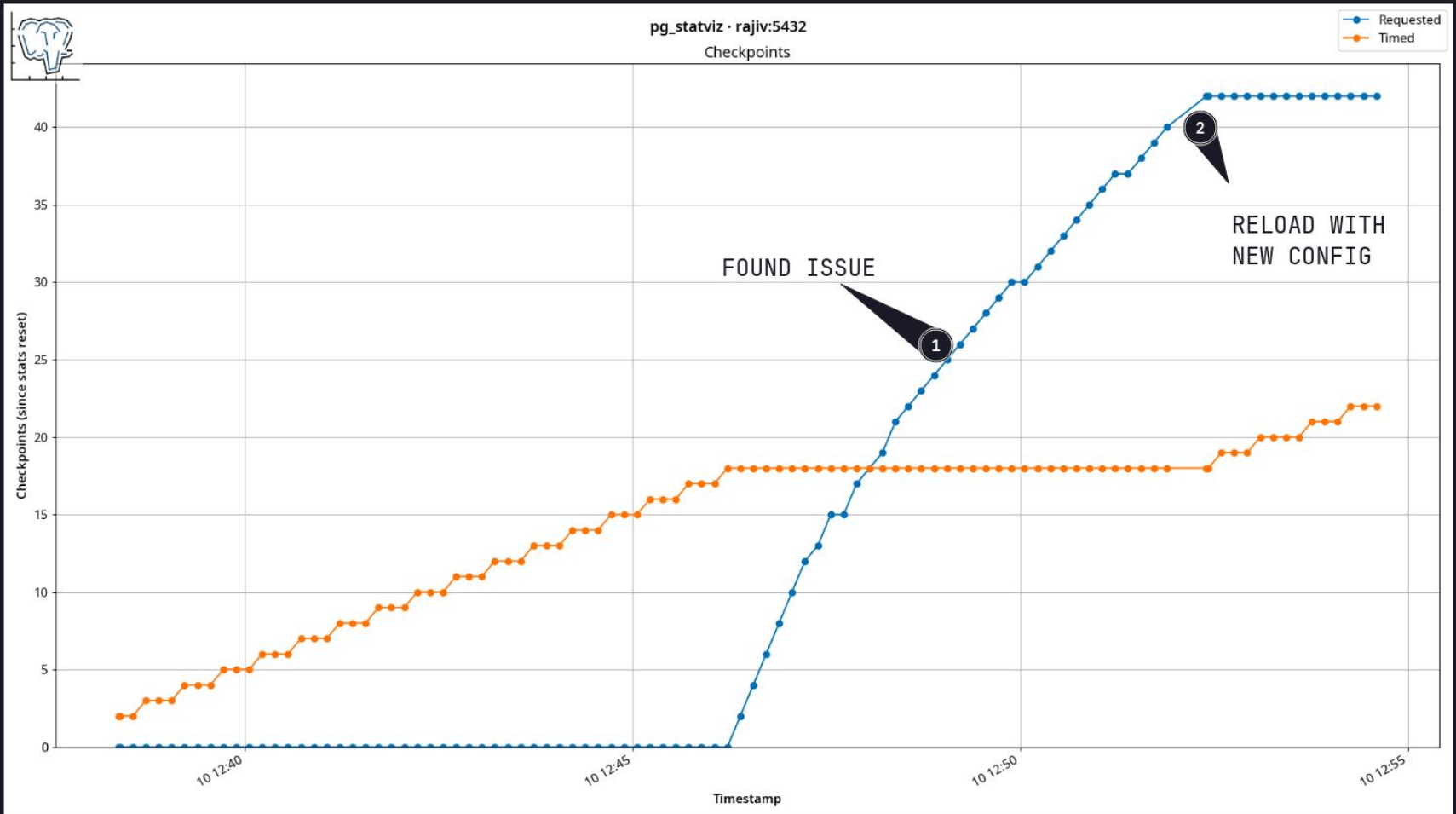
```
import sys
from argh import ArghParser
from pg_statviz.modules.analyze import analyze
from pg_statviz.modules.buf import buf
from pg_statviz.modules.cache import cache
from pg_statviz.modules.checkp import checkp
from pg_statviz.modules.conn import conn
from pg_statviz.modules.io import io
from pg_statviz.modules.lock import lock
from pg_statviz.modules.tuple import tuple
from pg_statviz.modules.wait import wait
from pg_statviz.modules.wal import wal
from pg_statviz.modules.xact import xact
```

Let's play detective:

Checkpointer


pg_statviz · rajiv:5432
Checkpoints
Requested
Timed
Checkpoints (since stats reset)
0
2
4
6
8
10
12
14
16
18
10 13:06
10 13:07
10 13:08
10 13:09
10 13:10
10 13:11
10 13:12
10 13:13
Timestamp


pg_statviz · rajiv:5432
Checkpoints
Requested
Timed
Checkpoints (since stats reset)
0
2
4
6
8
10
12
14
16
18
10 12:38
10 12:39
10 12:40
10 12:41
10 12:42
10 12:43
10 12:44
10 12:45
10 12:46
10 12:47
10 12:48
Timestamp


pg_statviz · rajiv:5432
Checkpoints
Requested
Timed
FOUND ISSUE
1
2
RELOAD WITH
NEW CONFIG
Checkpoints (since stats reset)
0
5
10
15
20
25
30
35
40
10 12:40
10 12:45
10 12:50
10 12:55
Timestamp

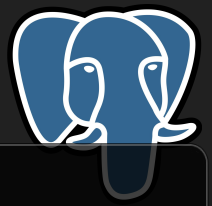
But what if you need a
Custom visualization

Create statistics collector

```
CREATE TABLE IF NOT EXISTS @extschema@.all_tab(
    snapshot_tstamp timestamptz REFERENCES @extschema@.snapshots(snapshot_tstamp) ON DELETE
CASCADE PRIMARY KEY,
    seq_scan int,
    idx_scan int);

CREATE OR REPLACE FUNCTION @extschema@.snapshot_all_tab(snapshot_tstamp timestamptz)
RETURNS void
AS $$
    INSERT INTO @extschema@.all_tab (
            snapshot_tstamp,
            seq_scan,
            idx_scan)
        SELECT
            snapshot_tstamp,
            seq_scan,
            idx_scan
        FROM pg_stat_all_tables
        WHERE relname = 'pgbench_accounts';
$$ LANGUAGE SQL;
```

Create visualization generator

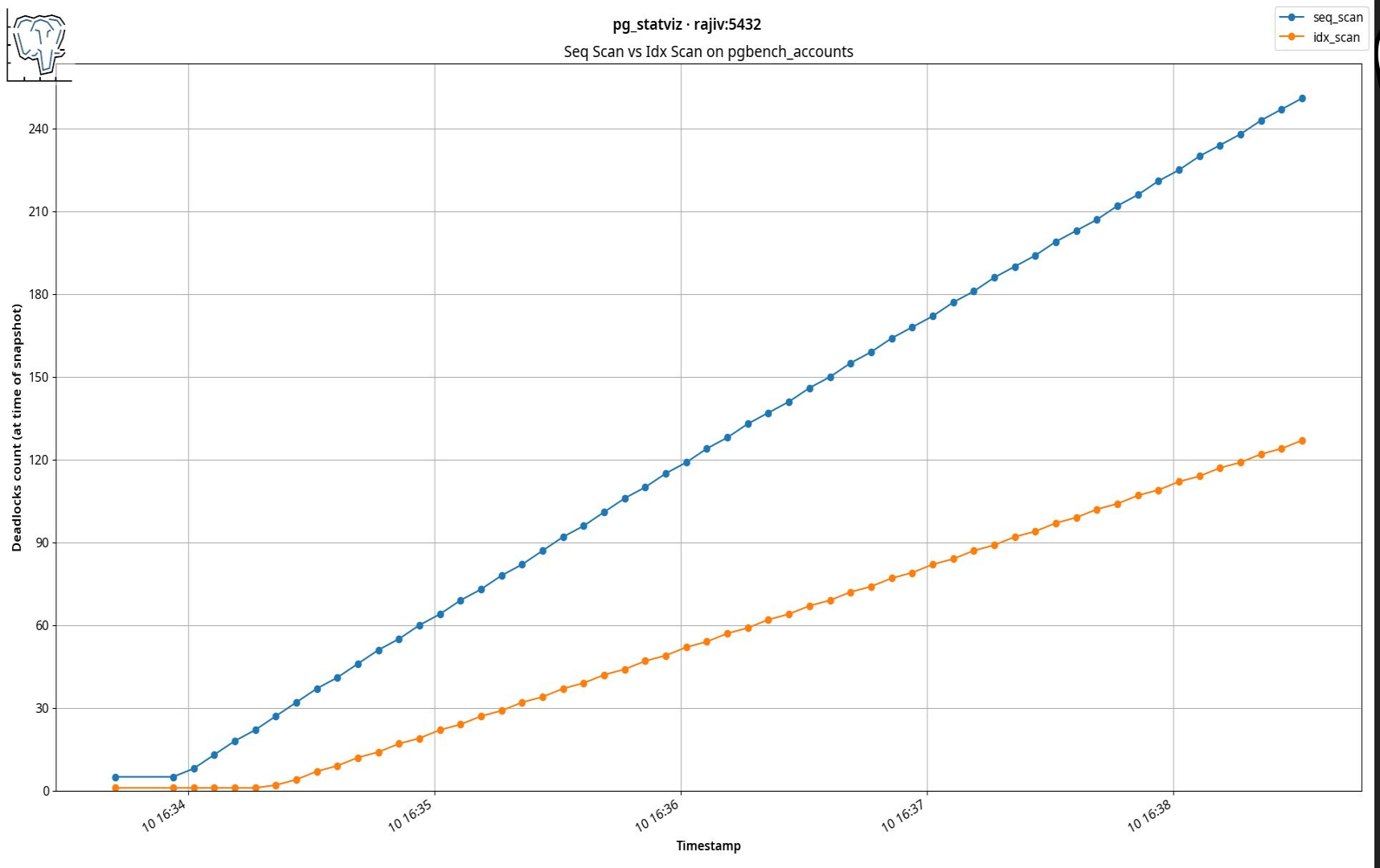


```
 # Retrieve the snapshots from DB
    cur = conn.cursor()
    cur.execute("""SELECT snapshot_tstamp,seq_scan,
idx_scan
                   FROM pgstatviz.all_tab
                   WHERE snapshot_tstamp BETWEEN %s
AND %s
                   ORDER BY snapshot_tstamp DESC""",
                (daterange[0], daterange[1]))
    data = cur.fetchmany(MAX_RESULTS)
    if not data:
        raise SystemExit("No pg_statviz snapshots
found in this database")

    tstamps = [ts['snapshot_tstamp'] for ts in data]
    seq_scan = [t['seq_scan'] for t in data]
    idx_scan = [t['idx_scan'] for t in data]
```

```
    # Plot as many of each scan type kind we have per
snapshot
    plt, fig = plot.setup()
    plt.suptitle(f"pg_statviz · {info['hostname']}:{port}",
                 fontweight='semibold')
    plt.title("Seq Scan vs Idx Scan on pgbench_accounts")
    # Plot total wait events
    plt.plot_date(tstamps, seq_scan,
                  label='seq_scan', aa=True,
linestyle='solid')
    plt.plot_date(tstamps, idx_scan,
                  label='idx_scan', aa=True,
linestyle='solid')
    fig.axes[0].set_ylim(bottom=0)

fig.gca().yaxis.set_major_locator(MaxNLocator(integer=True))
    plt.xlabel("Timestamp", fontweight='semibold')
```


pg_statviz · rajiv:5432
Seq Scan vs Idx Scan on pgbench_accounts
seq_scan
idx_scan
Deadlocks count (at time of snapshot)
0
30
60
90
120
150
180
210
240
10 16:34
10 16:35
10 16:36
10 16:37
10 16:38
Timestamp

Before we move on, a bit how did I manage to do all this while pursuing my college...

Google Summer of Code (GSoC)

- Global, online program focused on bringing new contributors into open source software development

Google Summer of Code (GSoC)

- Global, online program focused on bringing new contributors into open source software development
- Started in 2005 by Google and PostgreSQL has been a part since 2006

Google Summer of Code (GSoC)

- Global, online program focused on bringing new contributors into open source software development
- Started in 2005 by Google and PostgreSQL has been a part since 2006
- Contributors are paired with mentors from open source organizations, gaining exposure to real-world software development techniques

Opportunity for Students

- Chance to work on real world open source projects of interest
- Receive mentoring from experienced people in the industry
- Connect to a larger community of the org and fellow GSoCer's.
- Monetary and other Incentives

Opportunity for Students

- Chance to work on real world open source projects of interest

Opportunity for Students

- Chance to work on real world open source projects of interest
- Receive mentoring from experienced people in the industry

Opportunity for Students

- Chance to work on real world open source projects of interest
- Receive mentoring from experienced people in the industry
- Connect to a larger community of the org and fellow GSoCer's.

Opportunity for Students

- Chance to work on real world open source projects of interest
- Receive mentoring from experienced people in the industry
- Connect to a larger community of the org and fellow GSoCer's.
- Monetary and other Incentives

Opportunity for Orgs/Mentors

- Get contributions to your project

Opportunity for Orgs/Mentors

- Get contributions to your project
- Provide opportunity to students to begin their open source journey

Extending the tool

Extending the tool

- BYOM(Bring your own metric)

Extending the tool

- BYOM(Bring your own metric)
  - Snapshot the necessary statistics required.

Extending the tool

- BYOM(Bring your own metric)
  - Snapshot the necessary statistics required.
  - Update extension on the database

Extending the tool

- BYOM(Bring your own metric)
  - Snapshot the necessary statistics required.
  - Update extension on the database
  - Create module for your metric.

Extending the tool

- BYOM(Bring your own metric)
  - Snapshot the necessary statistics required.
  - Update extension on the database
  - Create module for your metric.
  - Export metrics in form of graphs.

Extending the tool

- BYOM(Bring your own metric)
  - Snapshot the necessary statistics required.
  - Update extension on the database
  - Create module for your metric.
  - Export metrics in form of graphs.
- Create live monitoring dashboards.

Extending the tool

- BYOM(Bring your own metric)
  - Snapshot the necessary statistics required.
  - Update extension on the database
  - Create module for your metric.
  - Export metrics in form of graphs.
- Create live monitoring dashboards.
- An alerting solution

Wrapping up..

- A simple tool to gain insights on your database
- Doesn’t involve dependencies
- Not required to restart the db
- Modular
- Open to contributions

Thanks for tuning in...

linkedin.com/in/rajivharlalka

github.com/rajivharlalka

twitter.com/rajivharlalka09

Project Link:

pg_statviz: github.com/vyruss/pg_statviz