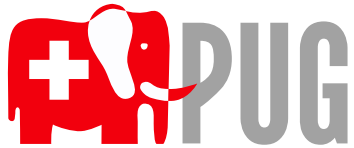
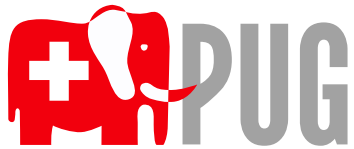


Lightning Talks

Vik Fearing
& Possibly You?



Boldly Migrate to PostgreSQL with credativ-pg-migrator

Josef Machytka
josef.machytka@gmail.com

credativ

Boldly Migrate to PostgreSQL with credativ-pg-migrator

Use our new Open Source Tool
Your Data Deserves the Best

Josef Machytka <josef.machytka@credativ.de>

2025-06-27 Swiss PostgreSQL Day 2025 lightning talk

credativ GmbH

- Founded 1999 in Jülich, Germany
- Close ties to Open-Source Community
- More than 40 Open-Source experts
- Consulting, development, training, support (3rd-level / 24x7)
- Open-Source infrastructure with Linux, Kubernetes and Proxmox
- Open-Source databases with PostgreSQL
- DevSecOps with Ansible, Puppet, Terraform and others
- Since 2025 independent owner-managed company again

Main Reasons for Migration

- Growing licensing payments for proprietary databases
- Lack of support and new features for legacy DBs
- No plans for future development of some legacy DBs
- In some cases lack of administrators, shrinking community
- Legacy DBs often run on old hardware / outdated OS
- Knowledge is often lost
- “History became legend. Legend became myth.”

AI images without credits
created by the author
using DeepDreamGenerator

Different Migration Tools Exist

credativ

- Different tools are available for migrations
- Open source, and commercial - with specific use cases
- Some use CDC (Change Data Capture) & minimal downtime
- Others require downtime and offline migration

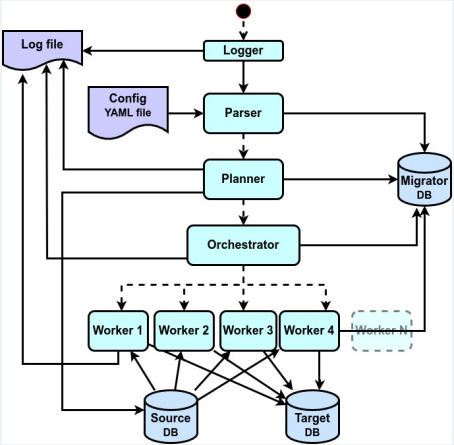
- Changes in open source projects can be slow
- Some of them are not maintained anymore
- Our idea is to unify multiple migration use cases
- To dynamically address issues we created credativ-pg-migrator
- We can quickly add new features for specific use cases

Meet credativ-pg-migrator

- Inspired by pgloader created by Dimitri Fontaine
- Intended for offline migrations, speed depends on hardware
- Written in Python - JDBC, ODBC or python native DB access
- Other languages have limited support for older DBs
- Written in classes, dynamicly pluggable connectors
- Uses well documented and stable libraries
- Pyodbc, JayDeBeApi, cx_Oracle, psycopg2, mysql, ...
- YAML configuration file, text log file

credativ-pg-migrator - Architecture

- Modular structure: Parser, Planner, Orchestrator, Workers
- Runs parallel workers, one reader and writer per table
- Speed of migration depends mainly on the hardware
- Creates and fills migration protocol tables
- Protocol tables contain all details about migrated objects
- Outputs detailed INFO and DEBUG messages



credativ-pg-migrator - Features

- Target database is always PostgreSQL
- Supports 8 different source databases
- Informix, Oracle, Sybase ASE, SQL Anywhere,
- IBM DB2 LUW, MS SQL Server, MySQL/MariaDB
- PostgreSQL to PostgreSQL for special use cases
- Migrates complete data model from all of them
- Tables, data, indexes, constraints, views
- Allows multiple custom adjustments

credativ-pg-migrator - Customizations

- Configurable scope - only schema, only data, both
- Yes/No, Include/exclude - tables, views, funcs, triggers

- Custom defined adjustments:
- Replace Data types (numeric PKs to BIGINT, UDF to standard type)
- Replace Default values (different names of SQL funcs)
- Subtitutions of Remote objects references (cross-db refs like db1@tab1)
- Limitatins for Data migrations (time limits: created_at >=, ID in SELECT)

credativ-pg-migrator - Other Features

- From Informix can convert Functions, Procedures, Triggers
- Funcs, procs, triggers - yes/no, include/exclude
- Success rate of code conversion 80 to 90%
- Errors mostly due to missing tables, renamed objects/columns
- Some statements may require small manual adjustments
- Conversion of code can be easily added for other DBs too

Future plans for credativ-pg-migrator

- Currently we are adding:
- Pre-migration analysis of source database
- Analysis of source tables for partitioning/data
- Configurable partitioning for target tables
- Migration of materialized views for relevant DBs
- Conversion of procs, funcs, triggers for supported DBs on demand
- Other source databases on demand

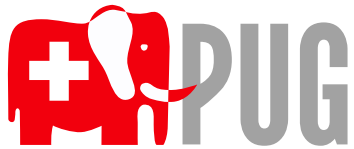
Open Source Version 0.9.0

GitHub repository - github.com/credativ/credativ-pg-migrator

Released under the GNU General Public License, version 3 (or any later version)

Available also on PyPi - pypi.org/project/credativ-pg-migrator/

I created this tool and many thanks to my colleague Michael Banck
for all the hard work with open sourcing & publishing it!

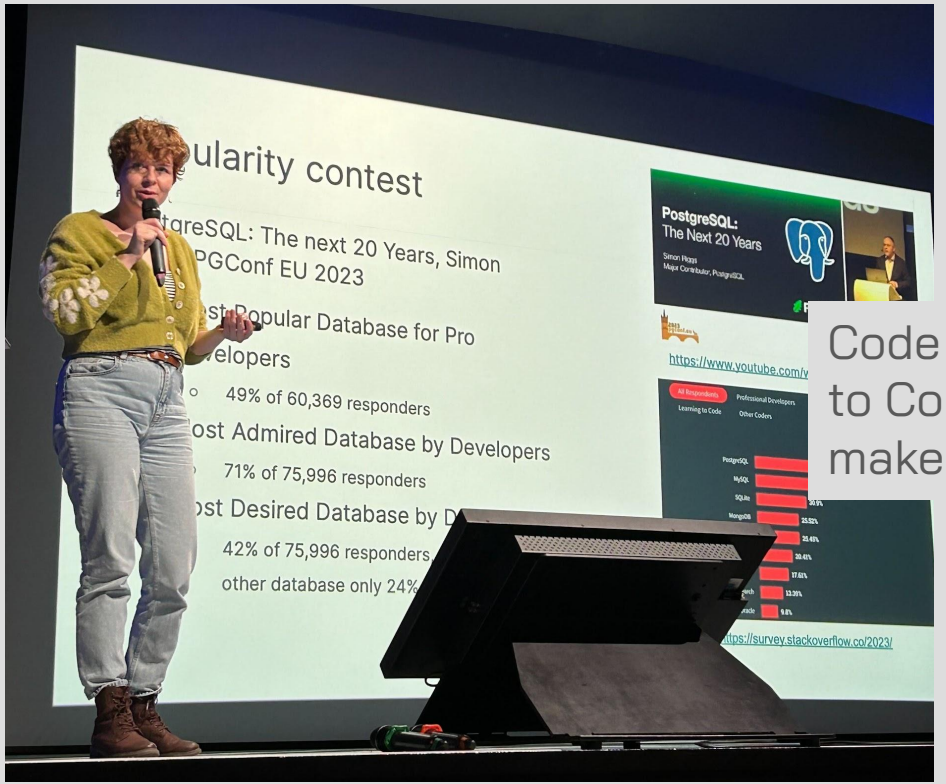


Kudos for the rest of us

Pavlo Golub
pavlo.golub@gmail.com

Kudos for the rest of us

Celebrating all contributions to the PostgreSQL project.



Code contributions and patches to Core are very visible, but don't make a community.

Floor Dress

Contributions of w/c 2024-10-07 (week 41)

Jimmy Angelakos | Oct. 12, 2024 | Category: contributions

- Claire Giordano hosted Tom Lane on the Talking Postgres podcast on Wed Oct 10th.
- Robert Haas hosted the October Hacking Workshop.
- Teresa Giacomini, Isaac Alves, My Nguyen produced the Activity Book for Postgres v4 with advice from Ariana Padilla Acosta, Adam Wolk, Derk van Veen, Boriss Mejias, and several others.
- Katharine Saar organized and hosted the San Francisco Bay Area PUG meeting on October 8, 2024, "Leveraging a PL/RUST Extension to Protect Sensitive Data" with Rajan Palanivel.
- Kim Jan...

Contributions of w/c 2024-09-30 (week 40)

Floor Drees | Oct. 7, 2024 | Category: contributions

- Dirk Krautschick and Christian Gohmann organized the 3. PostgreSQL User Group NRW MeetUp. Christoph Mönch-Teggeder and Christoph Berg presented at the Meetup.
- Gülçin Yıldırım Jelinek organized the Prague PostgreSQL Meetup: September Edition. Adam Wolk and Mayuresh Bagayatkar (Slides) presented at the Meetup.
- Andreas Scherbaum, Jonathan Katz, Mark Wong, Michael Alan Brewer, Mila Zhou and Pat Wright organized PGConf NYC 2024. Chelsea Dole, Daniel Gustafsson and Jonathan Katz selected the talks f...

Types of Contributions

Contributions to the PostgreSQL Project that may be considered by the committee include, but are not limited to:

- Code: Author, review, test, and/or commit patches that are pushed to the primary PostgreSQL git repository, and/or to closely related external projects such as PostGIS, pgjdbc, PGAdmin
- Translation: user facing messages in source code, documentation
- Community mailing list participation: report bugs, suggest features, contribute to ongoing discussions, answer questions, list moderation
- Governance of the PostgreSQL project or community recognized NPOs: Core Team, NPO Board of Directors, NPO Officers, NPO Committees
- Maintenance and operation of community controlled infrastructure: Sysadmin Team, Web Team
- Other Community recognized committee participation: Security, Code of Conduct
- Management of the development process and software lifecycle: Commitfest managers, Release Management Team, Release Team, Packagers, Buildfarm animal maintenance
- Organization and execution of community recognized conferences: Organizing committee, Selection Committee, Speakers, Volunteers
- Open Education: PostgreSQL related Blogs, articles, uncompensated training/tutorials
- Open Support (in addition to community mailing lists): #postgresql IRC channel, Postgres slack channel, Stack Overflow

If there is a type of contribution that you feel should be listed here or expanded upon, email **contributors@postgresql.org**.

https://www.postgresql.org/about/policies/contributors/

Types of Contributions

Contributions to the PostgreSQL Project that may be considered by the committee include, but are not limited to:

- Code: Author, review, test, and/or commit patches that are pushed to the primary PostgreSQL git repository, and/or to closely related external projects such as PostGIS, pgjdbc, PGAdmin
- Translation: user facing messages in source code, documentation
- Community mailing list participation: report bugs, suggest features, contribute to ongoing discussions, answer questions, list moderation
- Governance of the PostgreSQL project or community recognized NPOs: Core Team, NPO Board of Directors, NPO Officers, NPO Committees
- Maintenance and operation of community controlled infrastructure: Sysadmin Team, Web Team
- Other Community recognized committee participation: Security, Code of Conduct
- Management of the development process and software lifecycle: Commitfest managers, Release Management Team, Release Team, Packagers, Buildfarm animal maintenance
- Organization and execution of community recognized conferences: Organizing committee, Selection Committee, Speakers, Volunteers
- Open Education: PostgreSQL related Blogs, articles, uncompensated training/tutorials
- Open Support (in addition to community mailing lists): #postgresql IRC channel, Postgres slack channel, Stack Overflow

If there is a type of contribution that you feel should be listed here or expanded upon, email **contributors@postgresql.org**.

https://www.postgresql.org/about/policies/contributors/

About postgres-contrib.org

postgres-contrib.org is a new website, started in July 2024 by members of the PostgreSQL community, highlighting contributions to the project by the amazing people standing behind it.

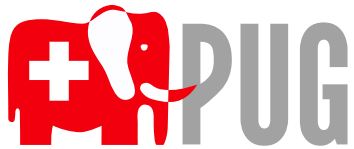
Many contributions to and for the PostgreSQL Project happen outside of writing code. This was the topic of the Increase Community Participation session at PGConf.dev 2024.

The following people contributed to this list, and the general idea: Andreas Scherbaum, Boriss Mejías, Chris Ellis, Floor Drees, Jimmy Angelakos and Pavlo Golub.

If you would like to add something to the list of contributions please contact us by email.

Recognize!

- https://postgres-contrib.org/
- Send submissions to:
  - contact@postgres-contrib.org



Efficient Web Development with SeaORM and PostgreSQL.

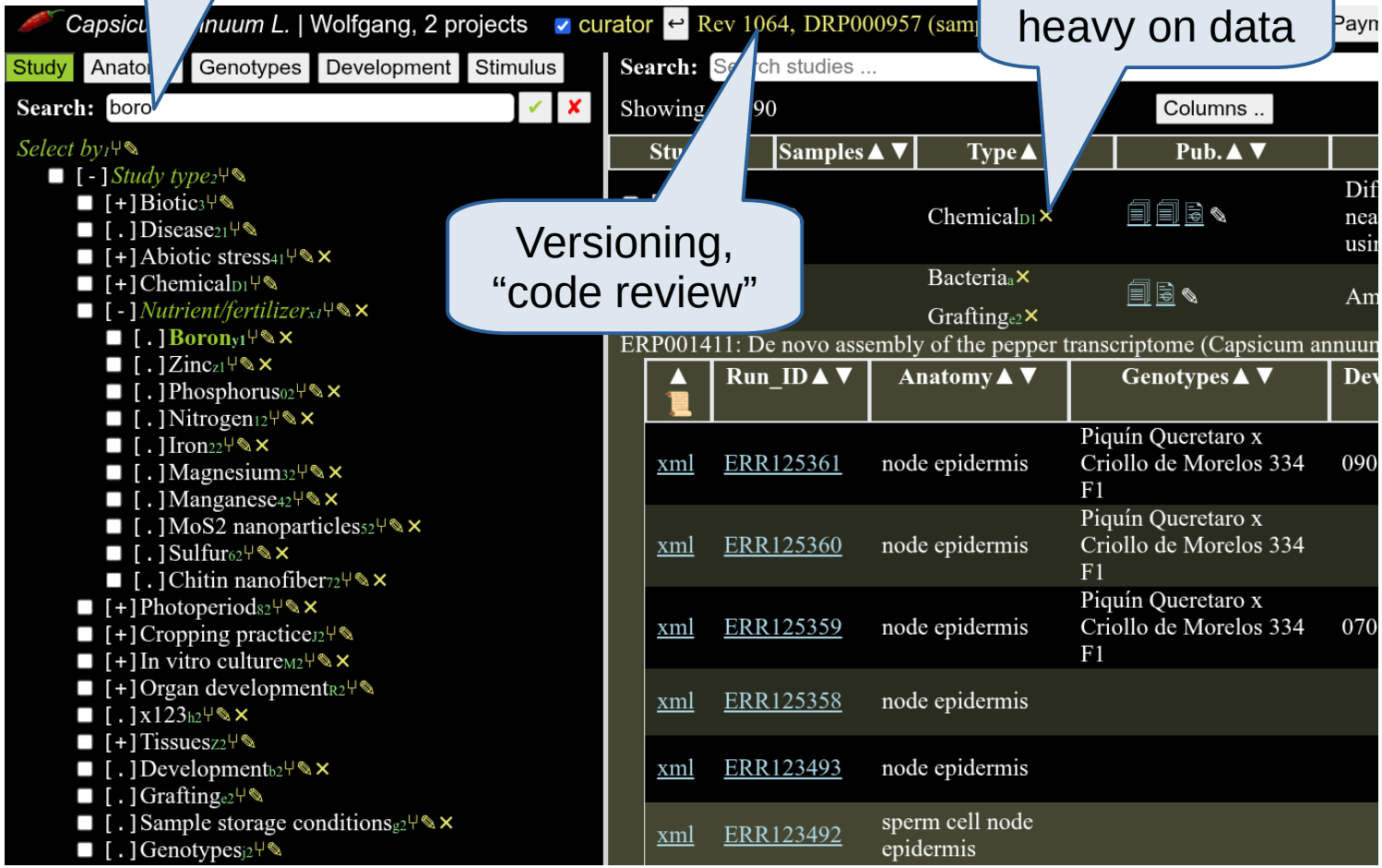
Audrius Meškauskas
audrius.meskauskas@gmail.com

Our application

Trees with search

Nested tables heavy on data

Versioning, “code review”



SeaORM

ORM mapping with Rust

```rust
#[sea_orm(table_name = "experiment_study")]
pub struct Model {
    #[sea_orm(primary_key)]
    pub id: i32,

    // Does not change with revision.
    pub canonical_id: i32,

    #[sea_orm(nullable)]
    pub revision: Option<i32>,

    #[sea_orm(column_type = "Json")]
    pub metadata: Value,
```

SeaORM

PostgreSQL NULL talks very nicely to Rust Option

```
revision        INTEGER DEFAULT NULL,
```

```
if let Some(rev :i32 ) = study.revision {
    // Field is not null and rev is integer
} else {
    // field is null
```

SeaORM

Error handling is both enforced and easy

```
async fn find_or_create_revision_row(
    is_curator: bool,
    db: &DatabaseTransaction,
) -> Result<current_revision::Model> {
    let current :Model = match current_revision::Entity::find()
        .filter(current_revision::Column::ForCurator.eq(is_curator))
        .one(db)
        .await?
    {
        Some(revision :Model ) => revision,
        None => {
            return Err(anyhow!("Missing row in revision table"));
        }
    };
    Ok(current)
}
```

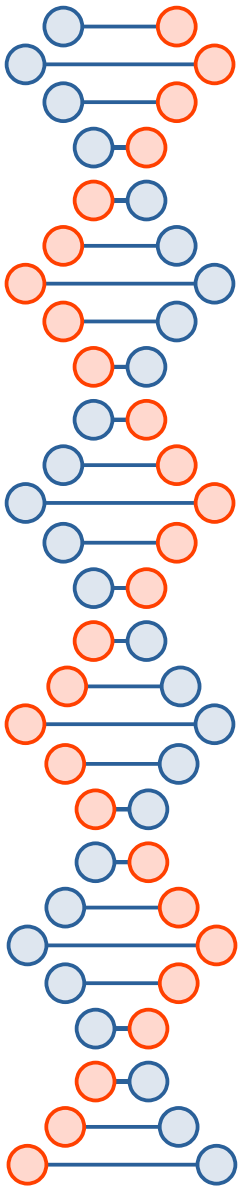
? =“or error”

Sea ORM is anynchronous

```rust
pub async fn read_trees(kinds: &[TreeKind], curator: bool) -> Result<Trees> {
    let trees :Vec<Tree> = try_join_all(
        kinds.iter().map(|kind :&TreeKind |
            let root :TreeNode = read_tree(
            info!("Loaded tree {}", kind);
            Ok::<Tree, anyhow::Error>(Tree {
                kind: *kind,
                root,
            })
        })
    ).await?;

    Ok(Trees { trees })
}
```

Try independently all trees in enum array passed, join to one result when all done



JSONB is very powerful

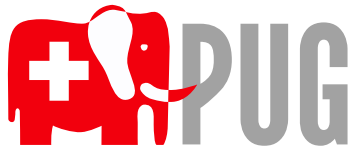
```
CREATE TABLE experiment_sample (
    id                  SERIAL PRIMARY KEY,
    annotations         JSONB,
```

```
#[sea_orm(table_name = "experiment_sample")]
pub struct Model {
    #[sea_orm(primary_key)]
    pub id: i32,

    // BTreeMap<TreeKind, Column>,
    #[sea_orm(column_type = "Json", nullable)]
    pub annotations: Option<Value>,
```

```
        annotations: Set(Some(json!(sample.annotations))),
```

This can take huge structure of any complexity that is defined in Rust – so kind of schema



NULL is unknown (yet)

Franck Pachot
franck@pachot.net

In SQL a NULL column is unknown yet a NULL primary key is known not existing

= (comparison)

NULL is unknown, comparing to NULL has an unknown result.
WHERE NOT IN (...,NULL)

'' (empty string)

NULL is not an empty string, except in some databases (Oracle)
NULL is not a value

Indexing

Some databases may index null entries, some may not (Oracle)

UNIQUE constraint

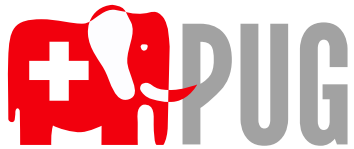
Two unknown values not guaranteed to be unique (same for distinct), except NULLS NOT DISTINCT

Foreign Key raised only on known violation (beware of nullable compound FK)

Outer Join returns NULL keys for no rows (non existing is the absence of a row)

MIN, MAX with null values (or empty sets)
GREATEST, LEAST should return null (not PG)

Storage size of tables with many nullable columns (takes no space at the end of a row)



pgstat_snap – a (very) poor man's ASH for PostgreSQL

Raphael Debinski
raphael.debinski@postfinance.ch

PostFinance

pgstat_snap – a (very) poor man's ASH for PostgreSQL

Swiss PGDay 2025

Use case - why we created this

- Every day at 8:31 an application suffered from a severe performance impact
- Instead of 5ms their calls took over one second
- At 8:32 everything was okay again
- Database was one of 20 in a PG cluster
- Pg_stat_statements and pg_stat_activity are cumulative and lack any timestamps
- Grafana is based on the cumulative values, some DBs had a line of trillions of rows inserted
- Without knowing what we are looking for, it was very hard to find the culprit
- Eventually we figured it out, one database was loading DWH data (6-200mio rows) every day at 8:31

Pgstat_snap – what it does

- Simple sql script with some pgPl/sql functions
- When CALLed, collects timestamped snapshots of pg_stat_statements and pg_stat_activity
- Two views provide the difference between each snapshot for every combination of queryid and datid

Workflow

1. Install it when needed:

```
psql
\i /path/to/pgstat_snap.sql
```

2. Collect Snapshots, for example every second for 60 seconds

```
CALL pgstat_snap.create_snapshot(1, 60);
```

3. Analyze

```
select * from pgstat_snap_diff order by 1;
```

4. Uninstall

```
SELECT pgstat_snap.uninstall();
DROP SCHEMA pgstat_snap CASCADE;
```

Sample Output

```
select * from pgstat_snap_diff order by 1;
```

snapshot_time	queryid	query	datname	username	wait_event_type	wait_event	rows_d	calls_d	exec_ms_d	sb_hit_d	sb_read_d	sb_dirt_d	sb_write_d
2025-03-25 11:00:19	4380144606300689468	UPDATE pgbench_tell	postgres	postgres	Lock	transactionid	4485	4485	986.262098	22827	0	0	0
2025-03-25 11:00:20	4380144606300689468	UPDATE pgbench_tell	postgres	postgres	Lock	transactionid	1204	1204	228.822413	6115	0	0	0
2025-03-25 11:00:20	7073332947325598809	UPDATE pgbench_bran	postgres	postgres	Lock	transactionid	1204	1204	1758.190499	5655	0	0	0
2025-03-25 11:00:21	7073332947325598809	UPDATE pgbench_bran	postgres	postgres	Lock	transactionid	1273	1273	2009.227575	6024	0	0	0
2025-03-25 11:00:22	2931033680287349001	UPDATE pgbench_acco	postgres	postgres	Client	ClientRead	9377	9377	1818.464415	66121	3699		

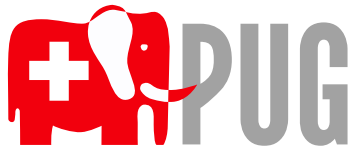
Other queries

- What was every query doing?
  ```
  select * from pgstat_snap_diff order by 2,1;
  ```
- Which database touched the most rows?
  ```
  select sum(rows_d),datname from pgstat_snap_diff group by datname;
  ```
- Which query DML touched the most rows?
  ```
  select sum(rows_d),queryid,query from pgstat_snap_diff where
  upper(query) not like 'SELECT%' group by queryid,query;
  ```
- What wait events happened which weren’t of type client?
  ```
  select * from pgstat_snap_diff where wait_event_type is not null and
  wait_event_type <> 'Client' order by 2,1;
  ```

Get it on github

Script and full documentation:

https://github.com/raphideb/pgstat_snap



ML for Systems and Systems for ML

Luigi Nardi
luigi@dbtune.com

ML for Systems and Systems for ML

Luigi Nardi, Ph.D.

Founder & CEO, DBtune

dbtune

MLSys: ML for Systems and Systems for ML

"ML for Systems" uses ML to make computer systems better, while
"Systems for ML" makes better systems so that ML can be better

Luigi Nardi

Systems for ML

Focus: Building and optimizing the computer systems that are necessary to support the training and deployment of ML models

Systems for ML

Focus: Building and optimizing the computer systems that are necessary to support the training and deployment of ML models

Examples:

- Developing efficient hardware accelerators for ML workloads — like TPUs
- Creating software frameworks and tools for managing ML pipelines — like TF
- Creating systems for data management for ML — like pg_vector

Systems for ML

Focus: Building and optimizing the computer systems that are necessary to support the training and deployment of ML models

Examples:

- Developing efficient hardware accelerators for ML workloads — like TPUs
- Creating software frameworks and tools for managing ML pipelines — like TF
- Creating systems for data management for ML — like pg_vector

Goal: To enable faster, more efficient, and more scalable ML by providing the necessary system-level support

ML for Systems

Focus: Using ML techniques to improve the design, operation, and optimization of computer systems

ML for Systems

Focus: Using ML techniques to improve the design, operation, and optimization of computer systems

Examples:

- Using ML to optimize network traffic routing — like Homunculus
- Using ML to optimize compilers — like BaCO
- Using ML to optimize database management — like DBtune

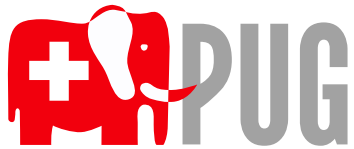
ML for Systems

Focus: Using ML techniques to improve the design, operation, and optimization of computer systems

Examples:

- Using ML to optimize network traffic routing — like Homunculus
- Using ML to optimize compilers — like BaCO
- Using ML to optimize database management — like DBtune

Goal: To replace or enhance traditional, often manual or heuristic-based, system designs with more adaptive and efficient ML-driven solutions



Travel Romania efficiently
Graph search with CTEs

Johannes Graën
johannes@selfnet.de

Travel Romania efficiently

Graph search with CTEs

Johannes Graën

Friday 27th June, 2025

Traveling in Romania

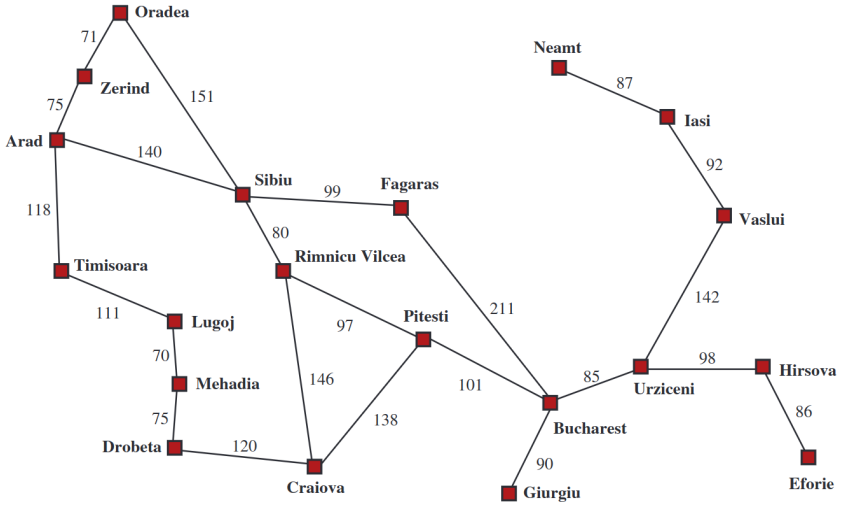


Figure 3.1 A simplified road map of part of Romania, with road distances in miles.

Figure by: Russell, S. J., & Norvig, P. (2010). Artificial intelligence a modern approach. London.

Map representation in DB – cities

```
CREATE TABLE city (id int, name text);

INSERT INTO city (id, name) VALUES (1,'Oradea'), (2,'Arad'),
  (3,'Timisoara'), (4,'Lugoj'), (5,'Mehadia'),
  (6,'Drobeta'), (7,'Sibiu'), (8,'Zerind'),
  (9,'Rimnicu Vilcea'), (10,'Craiova'), (11,'Pitesti'),
  (12,'Fagaras'), (13,'Bucharest'), (14,'Giurgiu'),
  (15,'Urziceni'), (16,'Vaslui'), (17,'Iasi'),
  (18,'Neamt'), (19,'Hirsova'), (20,'Eforie');
```

Map representation in DB – connections

```
CREATE TABLE conn (c1 int, c2 int, cost int);

INSERT INTO conn (c1, c2, cost) VALUES (1,7,151), (1,8,71),
  (8,2,75), (2,7,140), (2,3,118), (3,4,111), (4,5,70),
  (5,6,75), (6,10,120), (7,12,99), (12,13,211), (7,9,80),
  (9,10,146), (9,11,97), (10,11,138), (11,13,101),
  (13,14,90), (13,15,85), (15,19,98), (19,20,86),
  (15,16,142), (16,17,92), (17,18,87);
```

Map representation in DB – bidirectional cost

```
CREATE VIEW biconn (source, target, cost) AS
  SELECT c1, c2, cost
  FROM conn
  UNION ALL
  SELECT c2, c1, cost
  FROM conn;
```

Questions to answer

- Which paths lead from Neamt to Arad?
- Which cities can be reached from Rimniu Vilcea with costs lower than 300?
- Which is the most expensive route through Romania (visiting every place only once)?

CTE – Initial query (start position)

```
SELECT
  0           AS step,
  id          AS curr_loc,
  ARRAY[id]   AS path_ids,
  ARRAY[name] AS path_names,
  0           AS total_cost
FROM city
WHERE name = 'Neamt';
```

```
step       | 0
curr_loc   | 18
path_ids   | {18}
path_names | {Neamt}
total_cost | 0
```

CTE – Next step (identify connected cities)

```
SELECT
  initial.step+1                    AS step,
  biconn.target                     AS curr_loc,
  initial.path_ids||biconn.target   AS path_ids,
  initial.path_names||city.name     AS path_names,
  initial.total_cost+biconn.cost    AS total_cost
FROM (
  SELECT
    0            AS step,
    id           AS curr_loc,
    ARRAY[id]    AS path_ids,
    ARRAY[name]  AS path_names,
    0            AS total_cost
  FROM city
  WHERE name = 'Neamt'
) AS initial, biconn
LEFT JOIN city ON city.id = biconn.target
WHERE biconn.source = initial.curr_loc;
```

CTE – Next steps (continue searching)

```
WITH RECURSIVE search_graph(step, curr_loc, path_ids, path_names, total_cost) AS
(
    SELECT 0 AS step, id AS curr_loc, ARRAY[id] AS path_ids,
        ARRAY[name] AS path_names, 0 AS total_cost
    FROM city
    WHERE name = 'Neamt'
    UNION ALL
    SELECT step+1, target, path_ids||target, path_names||name, total_cost+cost
    FROM search_graph, biconn
    LEFT JOIN city ON city.id = biconn.target
    WHERE biconn.source = curr_loc
    AND biconn.target NOT IN (SELECT unnest(search_graph.path_ids))
)
SELECT *
FROM search_graph;
```

Routes from Neamt to Arad (with costs less than 1000)

```
WITH RECURSIVE search_graph(step, curr_loc, path_ids, path_names, total_cost) AS
(...)
SELECT *
FROM search_graph
JOIN city ON curr_loc = city.id
WHERE city.name = 'Arad'
AND total_cost < 1000;
```

Results

```
step        | 7
curr_loc    | 2
path_ids    | {18,17,16,15,13,12,7,2}
path_names  | {Neamt,Iasi,Vaslui,Urziceni,Bucharest,Fagaras,Sibiu,Arad}
total_cost  | 856
id          | 2
name        | Arad

step        | 8
curr_loc    | 2
path_ids    | {18,17,16,15,13,11,9,7,2}
path_names  | {Neamt,Iasi,Vaslui,Urziceni,Bucharest,Pitesti,Rimnicu Vilcea,Sibiu,Arad}
total_cost  | 824
id          | 2
name        | Arad

step        | 10
curr_loc    | 2
path_ids    | {18,17,16,15,13,11,9,7,1,8,2}
path_names  | {Neamt,Iasi,Vaslui,Urziceni,Bucharest,Pitesti,Rimnicu Vilcea,Sibiu,Oradea,Zerind,Arad}
total_cost  | 981
id          | 2
name        | Arad
```

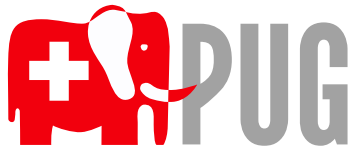
Cities reachable from Rimnicu Vilcea with cost < 300

```
WITH RECURSIVE search_graph(step, curr_loc, path_ids, path_names, total_cost) AS
(
    SELECT 0 AS step, id AS curr_loc, ARRAY[id] AS path_ids,
        ARRAY[name] AS path_names, 0 AS total_cost
    FROM city
    WHERE name = 'Rimnicu Vilcea'
    UNION ALL
    (...)
)
SELECT *
FROM search_graph
WHERE total_cost < 300
ORDER BY total_cost;
```

Results

step	c~loc	path_ids	path_names	t~cost
0	9	{9}	{Rimnicu Vilcea}	0
1	7	{9,7}	{Rimnicu Vilcea,Sibiu}	80
1	11	{9,11}	{Rimnicu Vilcea,Pitesti}	97
1	10	{9,10}	{Rimnicu Vilcea,Craiova}	146
2	12	{9,7,12}	{Rimnicu Vilcea,Sibiu,Fagaras}	179
2	13	{9,11,13}	{Rimnicu Vilcea,Pitesti,Bucharest}	198
2	2	{9,7,2}	{Rimnicu Vilcea,Sibiu,Arad}	220
2	1	{9,7,1}	{Rimnicu Vilcea,Sibiu,Oradea}	231
2	10	{9,11,10}	{Rimnicu Vilcea,Pitesti,Craiova}	235
2	6	{9,10,6}	{Rimnicu Vilcea,Craiova,Drobeta}	266
3	15	{9,11,13,15}	{Rimnicu Vilcea,Pitesti,Bucharest,Urziceni}	283
2	11	{9,10,11}	{Rimnicu Vilcea,Craiova,Pitesti}	284
3	14	{9,11,13,14}	{Rimnicu Vilcea,Pitesti,Bucharest,Giurgiu}	288
3	8	{9,7,2,8}	{Rimnicu Vilcea,Sibiu,Arad,Zerind}	295

(14 rows)



PG patching with GitLab pipelines

Michael Hegyi
lionel.rieder@postfinance.ch

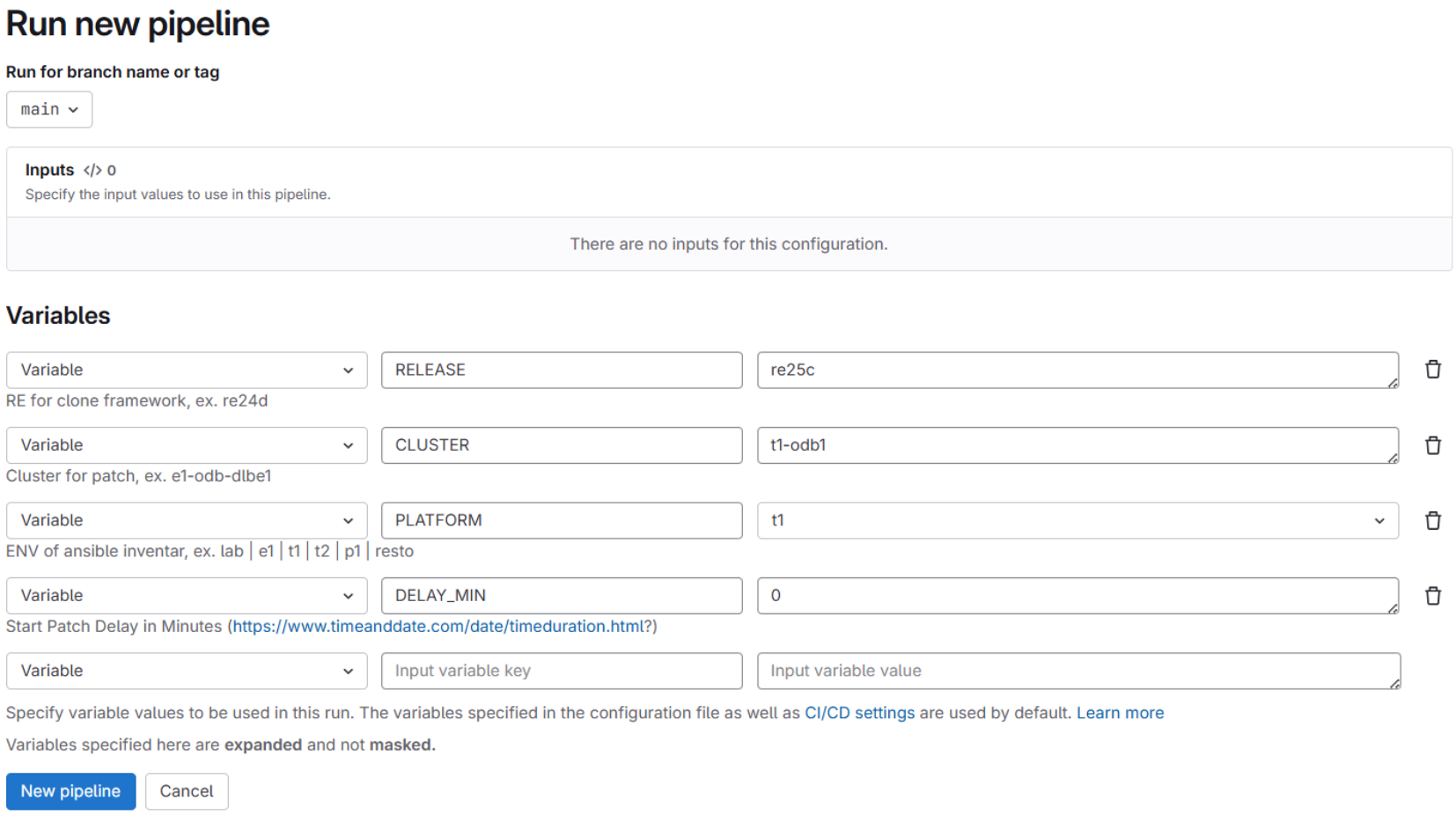
PostFinance

PG patching with GitLab pipelines

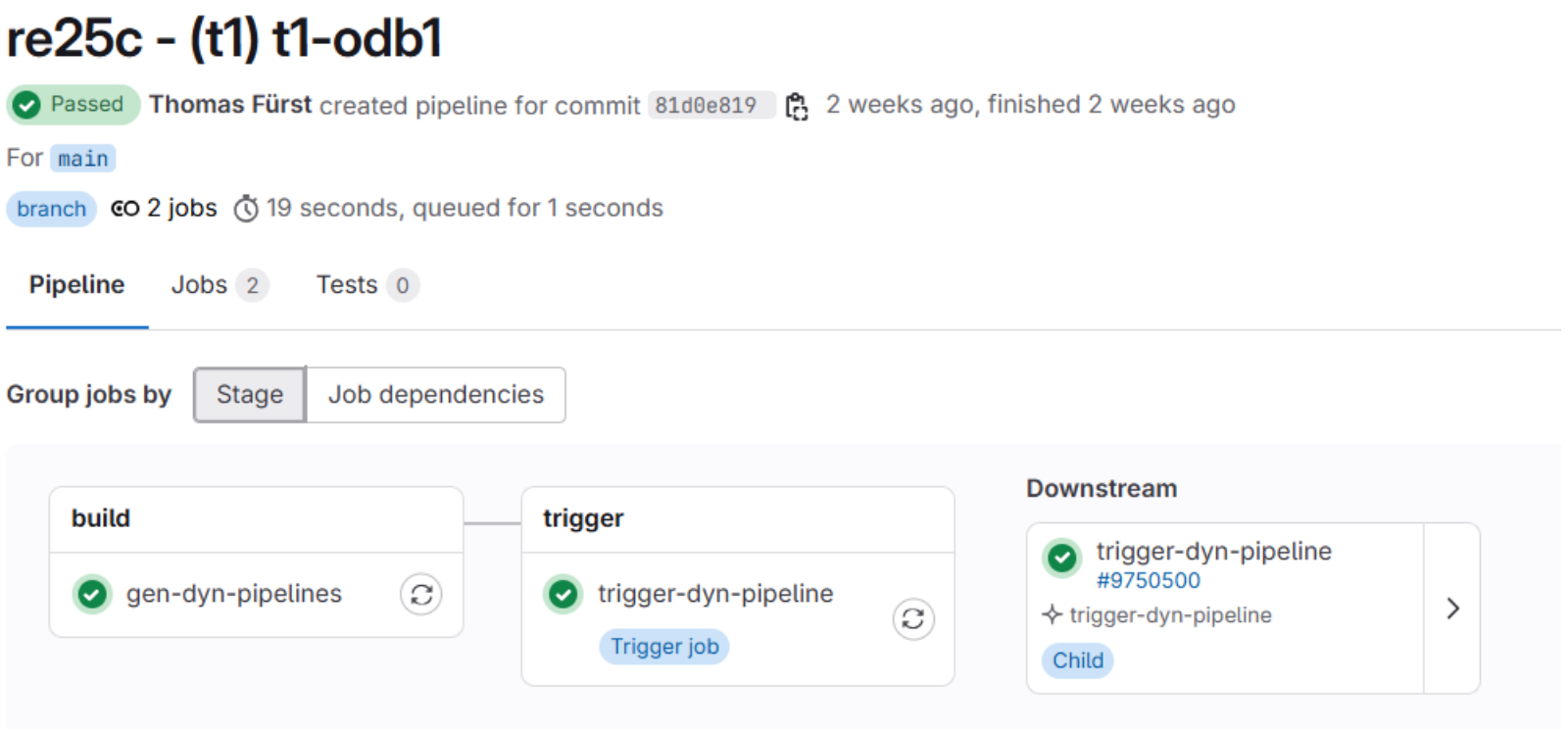
Swiss PGDay, Jun. 2025

Michael Hegyi

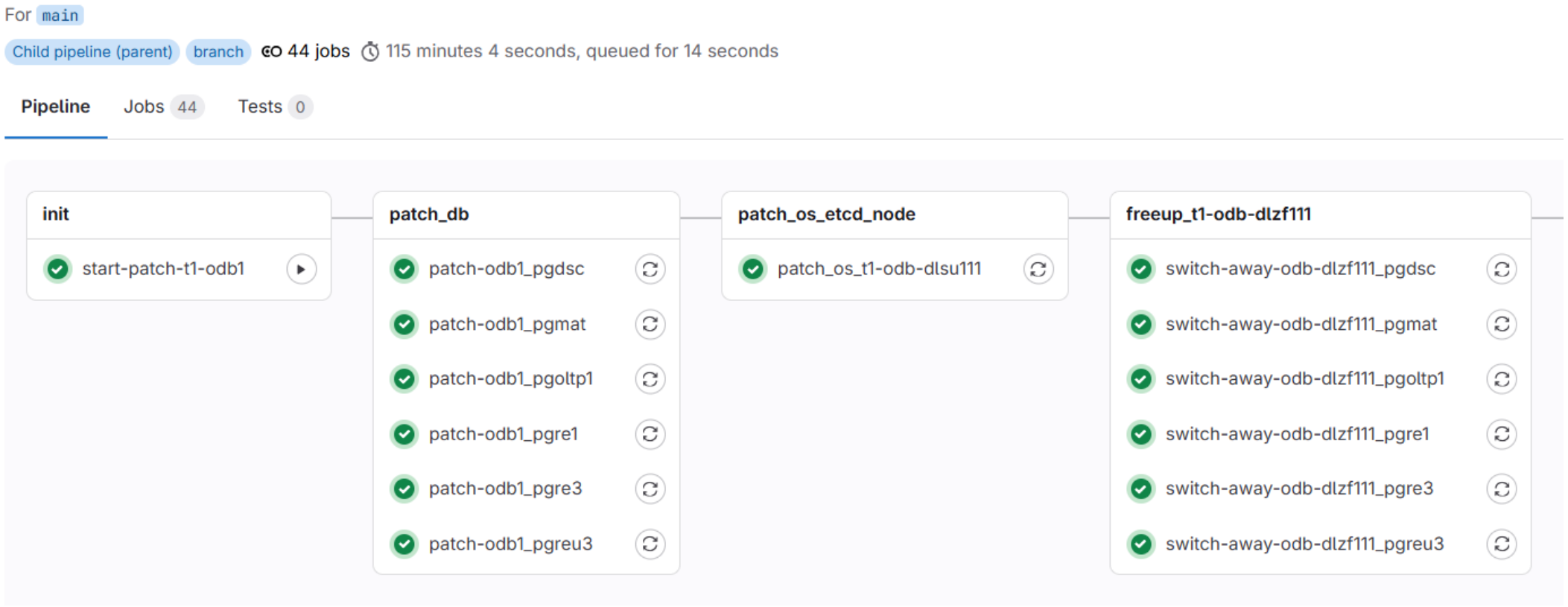
Define «Release Kit's» and Server (Ansible-Cluster)



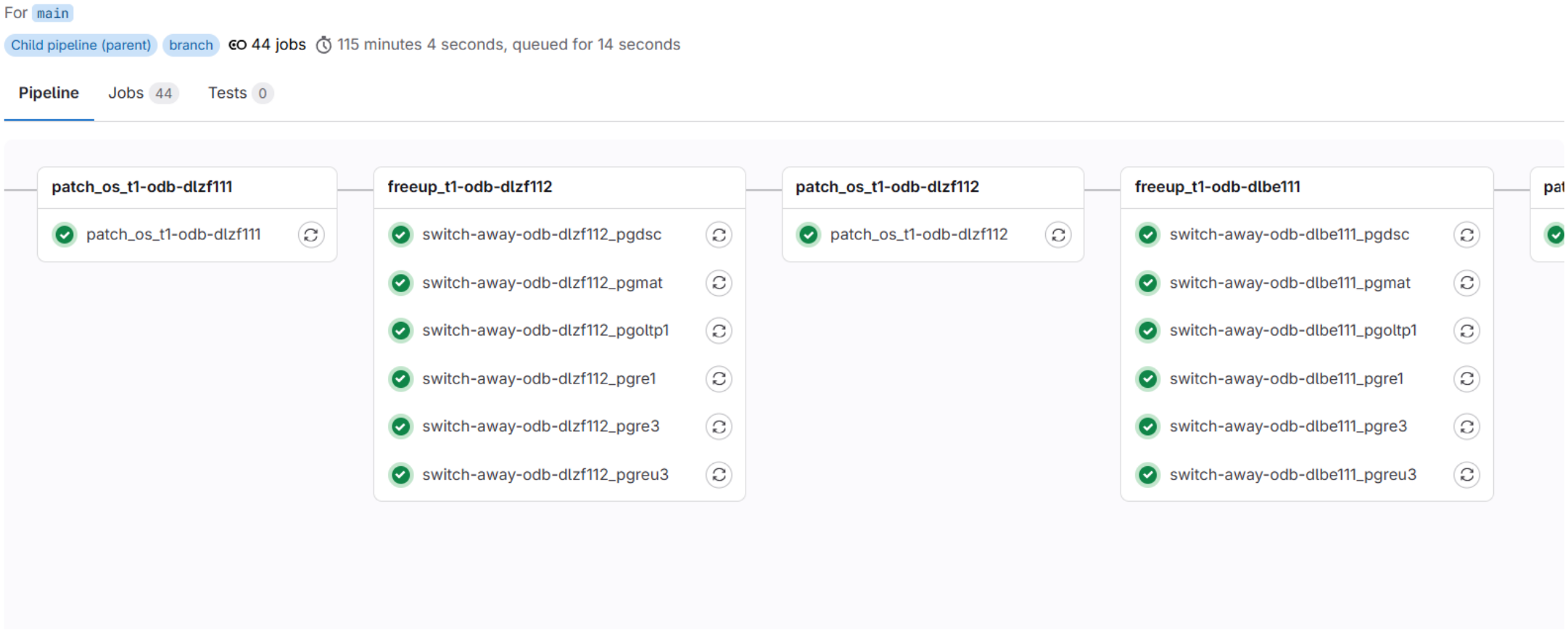
gen-dyn-pipelines → Collects realtime information of all PGClusters and servers and triggers a downstream pipeline



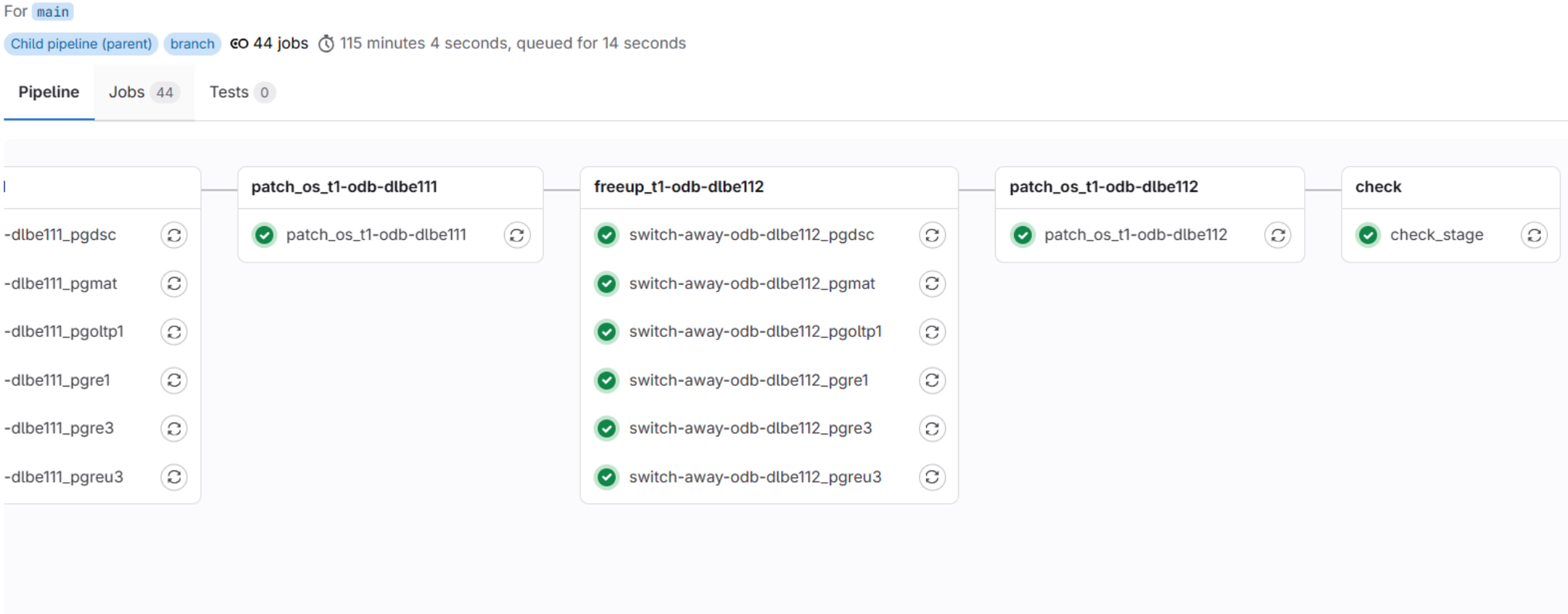
Downstream I: Starts with delay – patch all PG DB Cluster (correct minor-version) → patch etcd-node and switchover primaries to last node



Downstream II: Now patching OS (incl. patroni, pgexporter, pg BackRest …) and switchover primaries from 2. node to 1. node



Downstream III: and so on… until last node
→ Last step: quality-check – everything correctly patched?

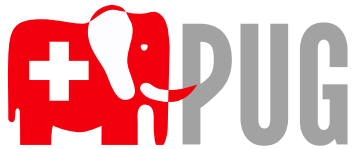


Everything is patched…

… lot of work done ;-)

Thanks

PostFinance



Postgres and Web3

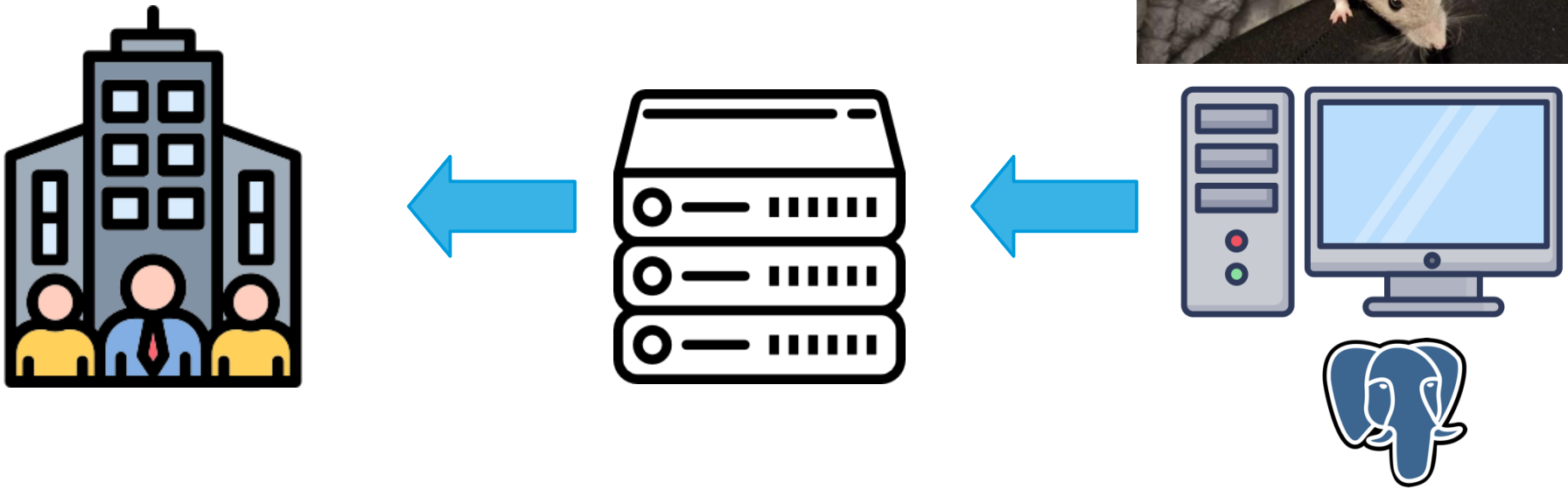
Marlene Retterer
marlene.reiterer@cybertec.at

Postgres with Web 3

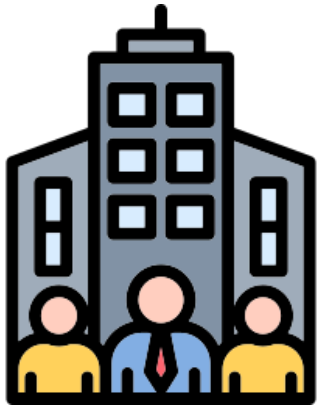
Marlene Reiterer

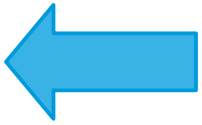
This is Karl

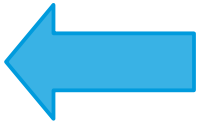
Web2 service with postgres backend

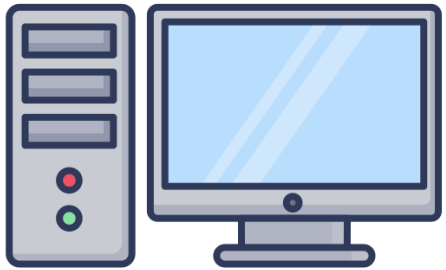


Web2 service with Postgres backend



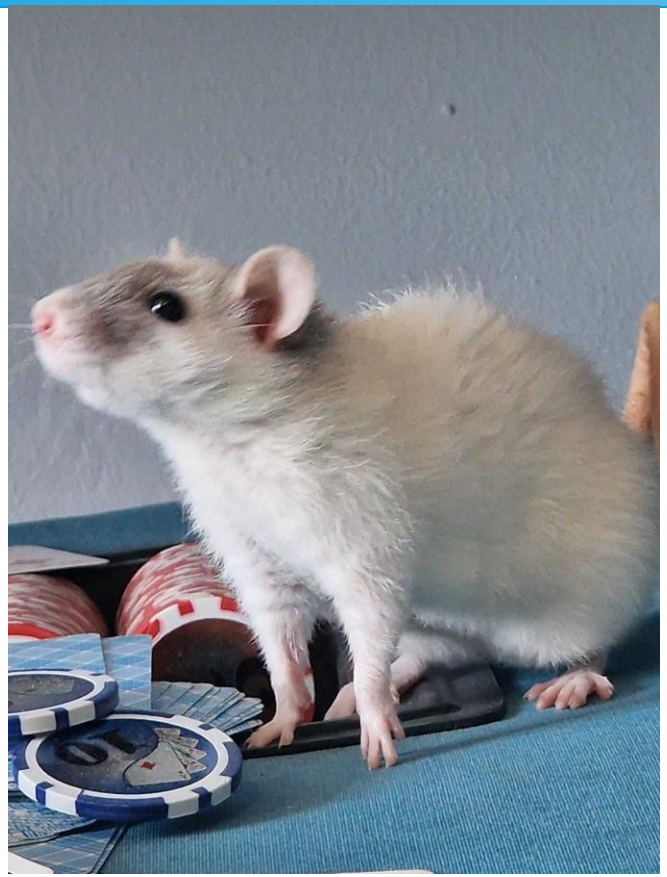




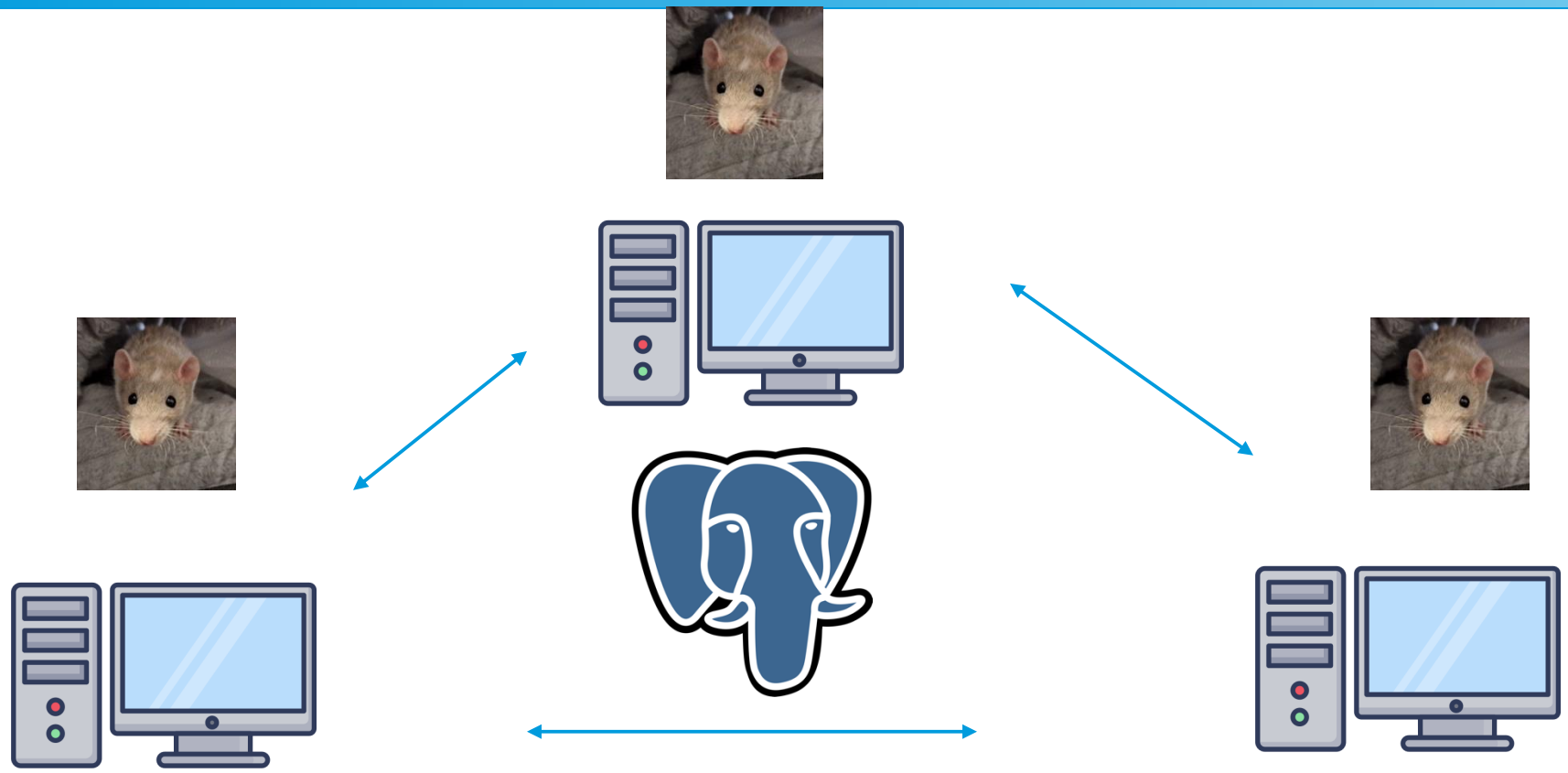


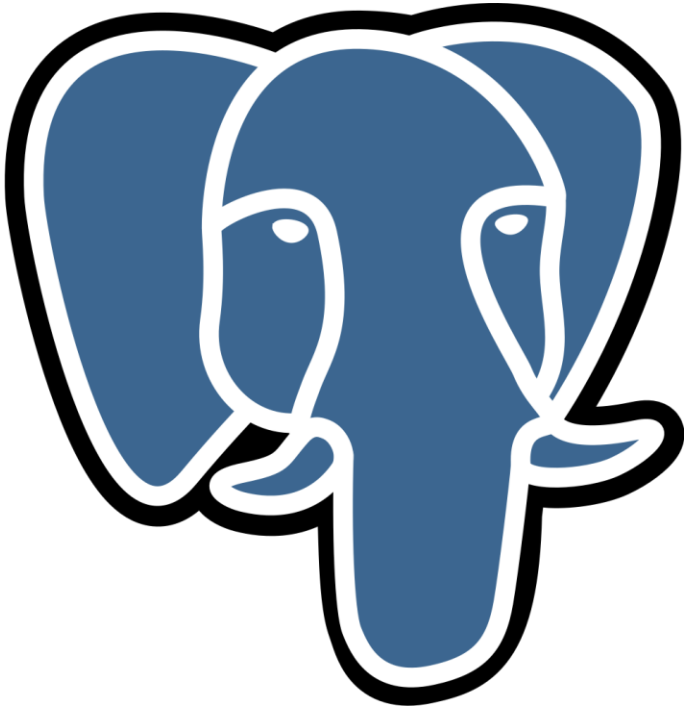
This is Icarus (evil)

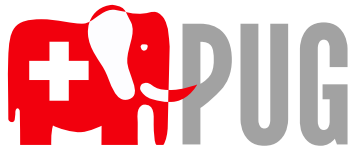
$$$



$$
$







Postgres on spot VM-s?

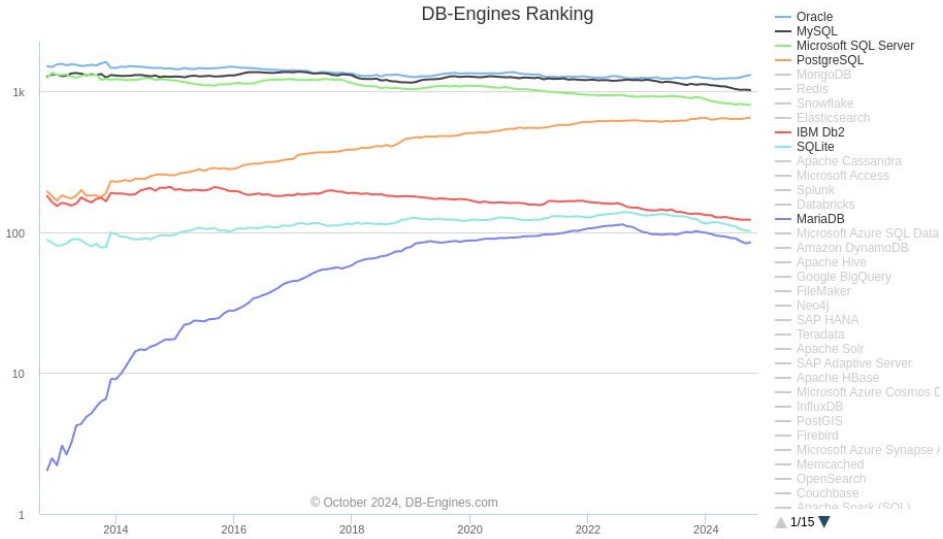
Kaarel Moppel
kaarel.moppel@gmail.com

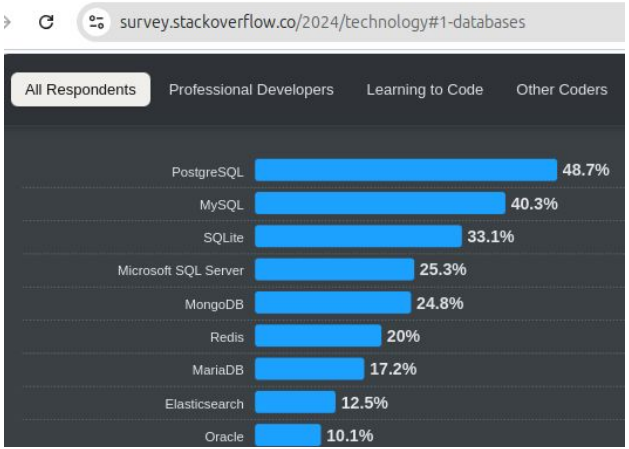
Postgres on Spot VMs?

Kaarel Moppel

Freelance PostgreSQL Consultant

SLIDES





The Register

Open source PostgreSQL named DBMS of the year by DB-Engines

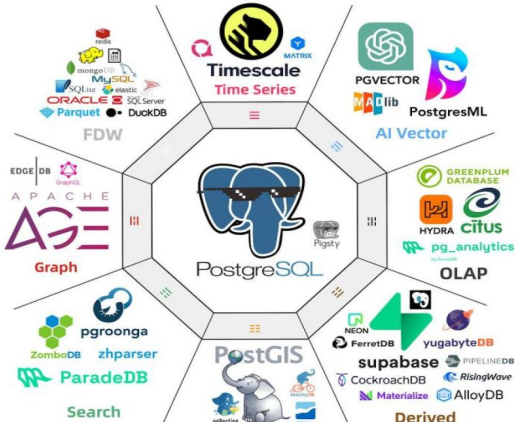
Already more than 37 years old, the relational system continues to gain popularity

Lindsay Clark

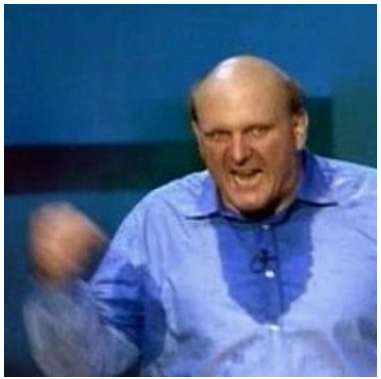
Wed 3 Jan 2024 // 17:00 UTC

Peter Zaitsev • 1st
Entrepreneur | Driving Success with MySQL, MariaDB, MongoDB & Postgr…
3d •

"Postgres is eating the database world" - Do you agree ?
https://lnkd.in/ecUyy5Rj #postgres



Postgres, postgres, postgres



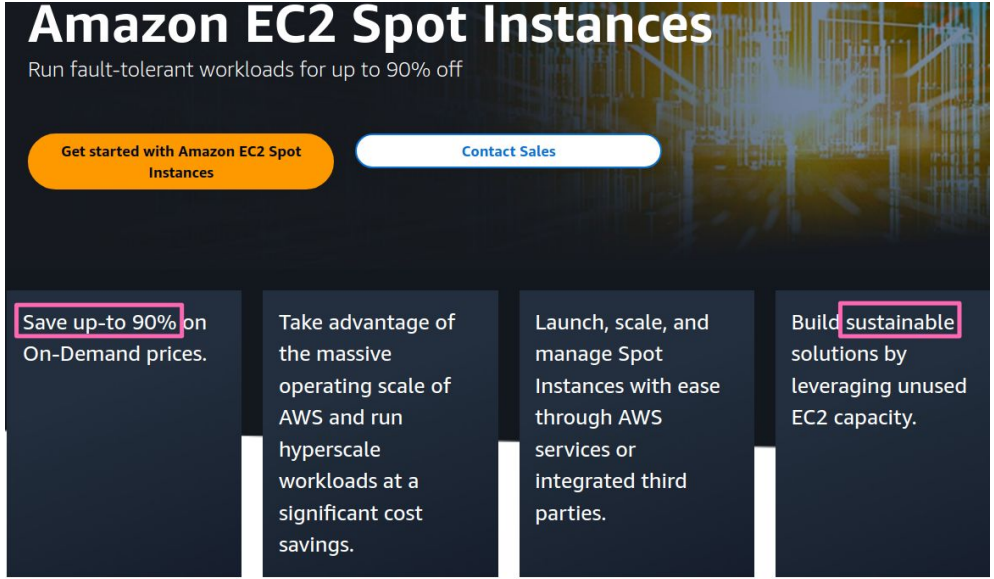
We know it - the tech world in general knows it - it’s already friggin awesome!

But wait... what if one could make it even better? Like 5x better - in about 3 minutes?

How on earth???

- Am I talking about async IO?
- Or perhaps vectorized instructions?
- Or a new columnar engine?
- ...
- The Elephant in the room - managed database costs...

Spot instances!



Amazon EC2 Spot Instances let you take advantage of unused EC2 capacity in the AWS cloud and are available at up to a 90% discount compared to On-Demand prices. You can use Spot Instances for various stateless, fault-tolerant, or flexible applications such as big data, containerized workloads, CI/CD, web servers, high-performance computing (HPC), and test & development workloads.

ChatGPT

What is the estimate percentage of really business critical databases?

The percentage of **business-critical databases** can vary significantly depending on the size, industry, and operational needs of a company. However, based on industry surveys and studies, it is estimated that **15% to 25%** of an organization's databases are considered **business-critical**.

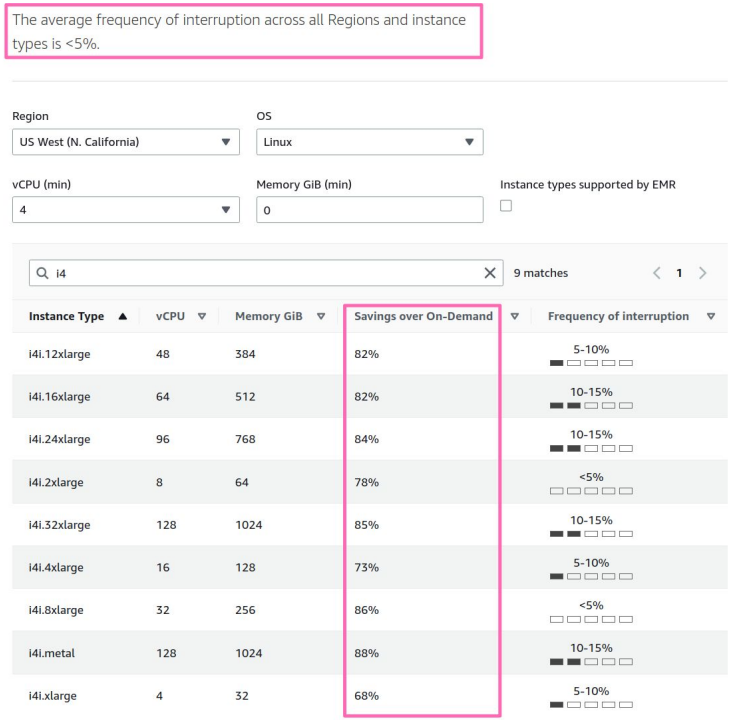
A selection of non-critical workloads

- Dev / staging environments
- Feature testing “forks”
- Backup / DR restore testing
- Production snapshots for ad-hoc reporting / mutations
- Testing effects / feasibility of HW upgrades
- Testing critical maintenance operations like VACUUM FULL-s
- ...

Spot is actually not “that” scary

If to use the Spot Instance advisor [tool](#) by AWS

On average, one can expect to run a few months uninterrupted!



The average frequency of interruption across all Regions and instance types is <5%.

Region: US West (N. California)

OS: Linux

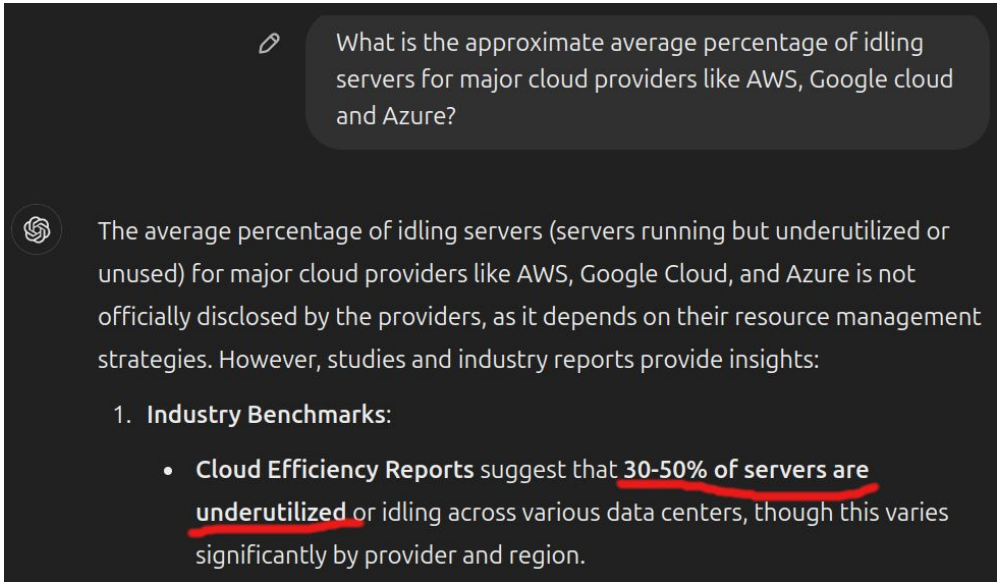
vCPU (min): 4

Memory GiB (min): 0

Instance types supported by EMR

i4 — 9 matches — 1

Instance Type	vCPU	Memory GiB	Savings over On-Demand	Frequency of interruption
i4i.12xlarge	48	384	82%	5-10%
i4i.16xlarge	64	512	82%	10-15%
i4i.24xlarge	96	768	84%	10-15%
i4i.2xlarge	8	64	78%	<5%
i4i.32xlarge	128	1024	85%	10-15%
i4i.4xlarge	16	128	73%	5-10%
i4i.8xlarge	32	256	86%	<5%
i4i.metal	128	1024	88%	10-15%
i4i.xlarge	4	32	68%	5-10%



What is the approximate average percentage of idling servers for major cloud providers like AWS, Google cloud and Azure?

The average percentage of idling servers (servers running but underutilized or unused) for major cloud providers like AWS, Google Cloud, and Azure is not officially disclosed by the providers, as it depends on their resource management strategies. However, studies and industry reports provide insights:

1. **Industry Benchmarks**:
   - **Cloud Efficiency Reports** suggest that **30-50% of servers are underutilized** or idling across various data centers, though this varies significantly by provider and region.

Stateful databases + stateless Spot ?

- Only for the crazy? Not really...if to pick the right use case and use the Spot statistics
  - Expected uptimes still in 99.9 - 99.95% range!
- Can get messy of course in practice without automation ...
- Wouldn’t it be nice if someone else deals with the annoying details and gives us *one-liner Postgres at unbeatable* price*?

PG Spot Operator

psql "$(**pg_spot_operator** --region=eu-north-1 --ram-min=64 --storage-min=500 \

--storage-type=local --tuning-profile=analytics --instance-name=mypg1 \

--admin-user=pgspotops --admin-user-password=topsecret123 **--connstr-only**)"

...

INFO Current Spot discount rate in AZ eu-north-1a: -75.5% (spot $126.6 vs on-demand $516.2)

...

psql (16.4 (Ubuntu 16.4-1.pgdg24.04+2))

SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression: off)

Type "help" for help.

pgspotops@postgres=>

~6x savings compared to RDS!

** Assumes local AWS CLI setup*

“UI” - CLI / Docker or a YAML manifest

```
pipx install pg-spot-operator

pg_spot_operator \
  --region ^eu \
  --ram-min 256 \
  --check-price


docker run --rm -e REGION=^eu \
  -e RAM_MIN=256 \
  -e CHECK_PRICE=y \
  pgspotops/pg-spot-operator:latest
```

```yaml
api_version: v1
kind: pg_spot_operator_instance
cloud: aws
region: eu-south-2
instance_name: hello-aws
expiration_date: "2024-12-22 00:00+03"
vm:
  cpu_min: 4
  ram_min: 16
  storage_min: 500
  volume_iops: 10000
os:
  extra_packages: [ pgbadger, postgresql-16-cron ]
  ssh_pub_key_paths: [ ~/.ssh/my_key.pub ]
user_tags:
  app: backend
postgresql:
  admin_is_superuser: false
  app_db_name: app
  admin_user: admin
  tuning_profile: oltp  # none | default | oltp | analytics | web
  admin_user_password: !vault |
    $ANSIBLE_VAULT;1.1;AES256
    306433643563343037396265346239376137333865353466613631663231386
```

Integrating with applications

Options:

- A "setup finished" callback script to propagate Postgres / VM connect data somewhere
- Running in pipe-friendly *“--connstr-only”* mode
- Specifying an S3 (or compatible) bucket to push the connect string into
- Writing the connect string to a file on the engine node
- Following output formats available:
  - *--connstr-format = auto** | *ssh* | *ansible* | *postgres*

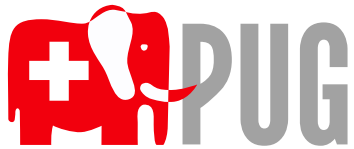
Tips for practical usage

- AWS advertised ~5% avg. eviction rate is nowhere to be seen ...
- Local storage instances have much less evictions
  - And the best $$, as EBS has no Spot discounts
- Larger, especially metal, instances have less evictions
- Need to increase the Spot CPU quotas for heavier usage
- Helpful to choose the best region near you:
  - *pg_spot_operator --list-avg-spot-savings*

github.com/pg-spot-ops/pg-spot-operator

Licence: ~~Functional Source License, Version 1.1~~, Apache 2.0

All kinds of feedback, feature requests or just a ⭐ would be very much appreciated!



The tower of Babel

Laurenz Albe
laurenz.albe@cybertec.at