CYBERTEC

DATA SCIENCE & POSTGRESQL

Monitoring PostgreSQL made simple

pgconf.eu 2023

Senior Consultant/Developer

Pavlo Golub

MAIL

pavlo.golub@cybertec.at

Social

pashagolub.github.io

WEB

www.cybertec-postgresql.com

CYBERTEC
DATA SCIENCE & POSTGRESQL
www.cybertec-postgresql.com

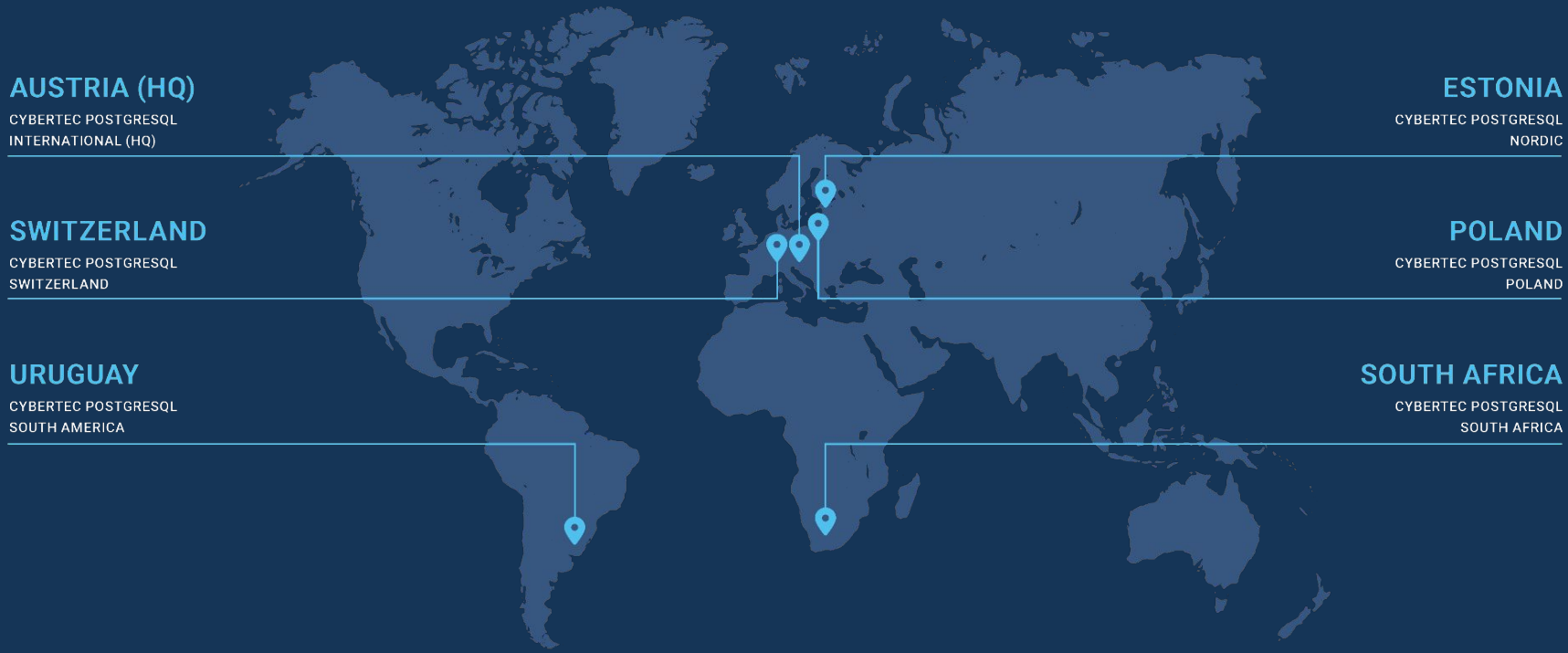
ABOUT **CYBERTEC**

Highly specialized,
fast growing
IT company

International Team
(10 countries),
six locations worldwide

PostgreSQL Services &
Support

Owner managed
since 2000



AUSTRIA (HQ)

CYBERTEC POSTGRESQL
INTERNATIONAL (HQ)

SWITZERLAND

CYBERTEC POSTGRESQL
SWITZERLAND

URUGUAY

CYBERTEC POSTGRESQL
SOUTH AMERICA

ESTONIA

CYBERTEC POSTGRESQL
NORDIC

POLAND

CYBERTEC POSTGRESQL
POLAND

SOUTH AFRICA

CYBERTEC POSTGRESQL
SOUTH AFRICA

CYBERTEC
DATA SCIENCE & POSTGRESQL
www.cybertec-postgresql.com

CUSTOMER SECTORS

- ICT
- Finance
- Regulatory
- Automotive
- Production
- Trade
- Universities
- and many more.

DATABASE - PRODUCTS

scalefield

CYPEX

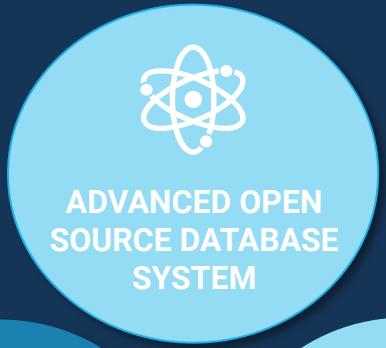
CYBERTEC MIGRATOR

DATA MASKING

CYBERTEC PGEE

POSTGRESQL TDE

WHY PostgreSQL?



www.cybertec-postgresql.com

CYBERTEC
DATA SCIENCE & POSTGRESQL
www.cybertec-postgresql.com

AGENDA

- Different levels of database monitoring
- PostgreSQL monitoring approaches
- PostgreSQL monitoring tools available
- pgwatch2
- pgwatch3? 😱

Monitoring PostgreSQL made simple

Different levels of database monitoring

Why to monitor?

- Failure / Downtime detection
- Slowness / Performance analysis
- Proactive predictions
- Maybe wasting money?

Different levels of database monitoring

- High level service availability
- System monitoring
- PostgreSQL land

High level service availability

- Try to periodically connect/query from an outside system

- DIY - e.g. a simple Cron script
- SaaS - lots of service providers

Who will guard the guards?

- You'll probably want two services for more critical stuff...

System monitoring

Too many tools, no real standards. Just make sure to understand what you're measuring!

- Do you know what does the CPU load number actually mean?
  - Is it a good metric?
- What's the difference between VIRT, RES, SHR memory values for a process?

PostgreSQL land

- Log analysis
- Statistics Collector
- Extensions

Log analysis

- “Just in case” storing of logs for possible ad hoc needs
  - Moving logs to a central place makes sense
  - rsync + Cron
- Active parsing
  - grep + Cron
  - DIY (file_fdw, Graylog, ELK, …)
  - pgBadger (JSON format)
  - Grafana Loki
  - Some cloud service (Loggly, Splunk, …)

Logging configuration

- Some settings to note
  - log_destination (CSV format recommended)
  - log_statement = ‘none’ (default)
  - log_min_duration_statement / log_duration
  - log_min_messages / log_min_error_statement

```
postgres=# SELECT count(*) FROM pg_settings
WHERE category LIKE 'Reporting and Logging%';
 count
-------
    43
```

Statistics Collector

- Not all `track_*` parameters enabled by default
- Dynamic views
  - `pg_stat_activity`, `pg_stat_(replication|wal_receiver)`
  - `pg_locks`, `pg_stat_ssl`, `pg_stat_progress_*`
- Cumulative views
  - Most `pg_stat_*` views
  - Long uptimes cause “lag” for problem detection
- Selective stats reset possible

Extensions

- Most notably **pg_stat_statements** (“top statements”)
- **pg_stat_monitor** (pg_stat_statements on steroids)
- **pgstattuple** (bloat)
- **pg_buffercache** (what’s in the shared buffers)
- **auto_explain** (e.g. to analyze “jumping runtimes”)
- **pg_qualstats** (WHERE predicate stats)
- **pg_stat_kcache** (to sample O/S metrics)
- …

Real life

Typically a mixed approach for bigger “shops”

- DYI
  - Log collection / parsing
  - Continuous storing of `pg_stat*` snapshots via some tool
  - Alerting and trends predictions (it’s hard!)

- Application performance monitoring (APM)
  - A more high level concept, requires some trust / lock-in
  - AppDynamics, New Relic, DataDog, …

Monitoring PostgreSQL made simple

PostgreSQL Monitoring Tools

PostgreSQL Monitoring Tools

FIND MORE

https://wiki.postgresql.org/wiki/Monitoring

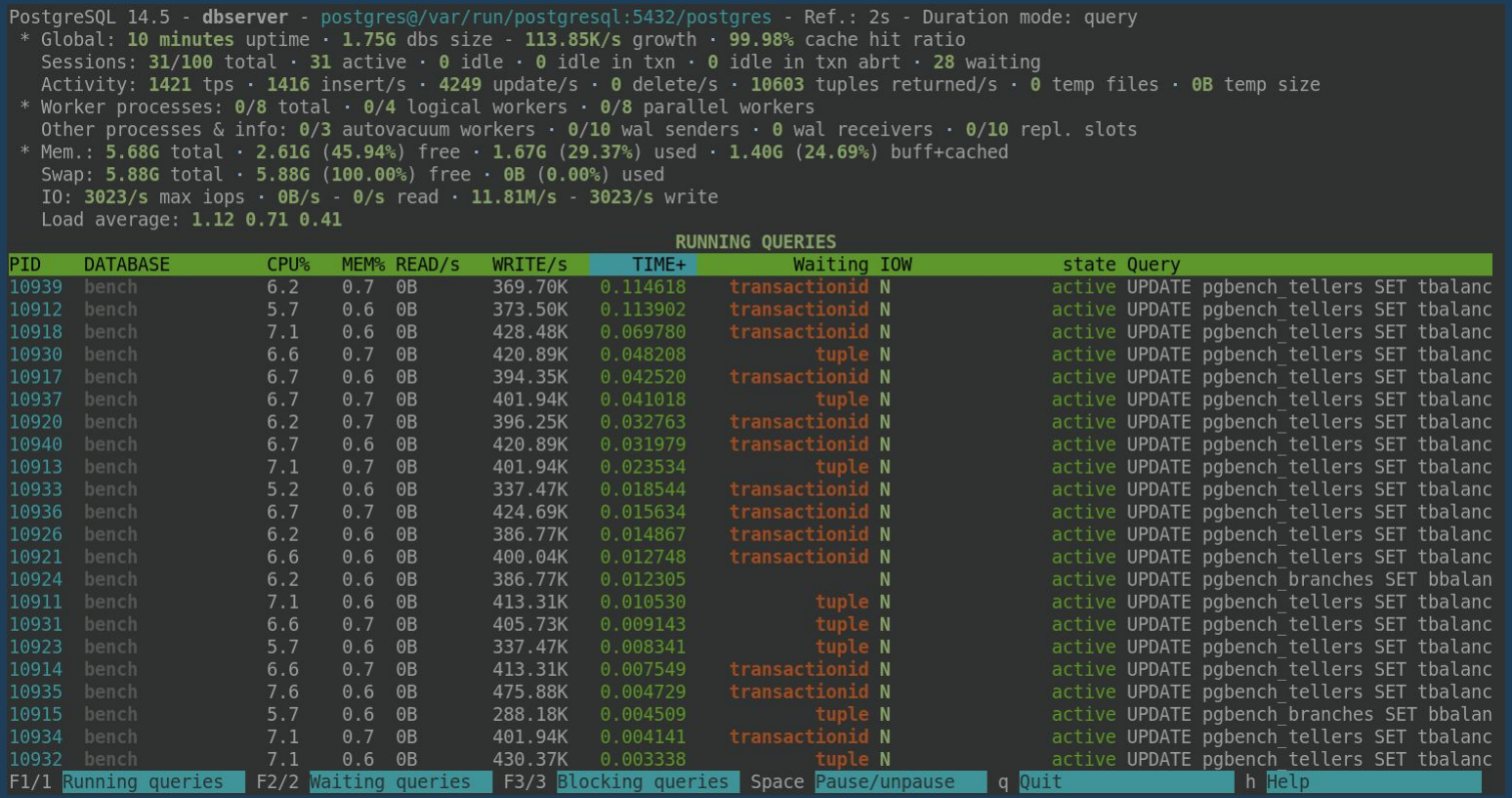
No shortage of tools!

Ad hoc troubleshooting

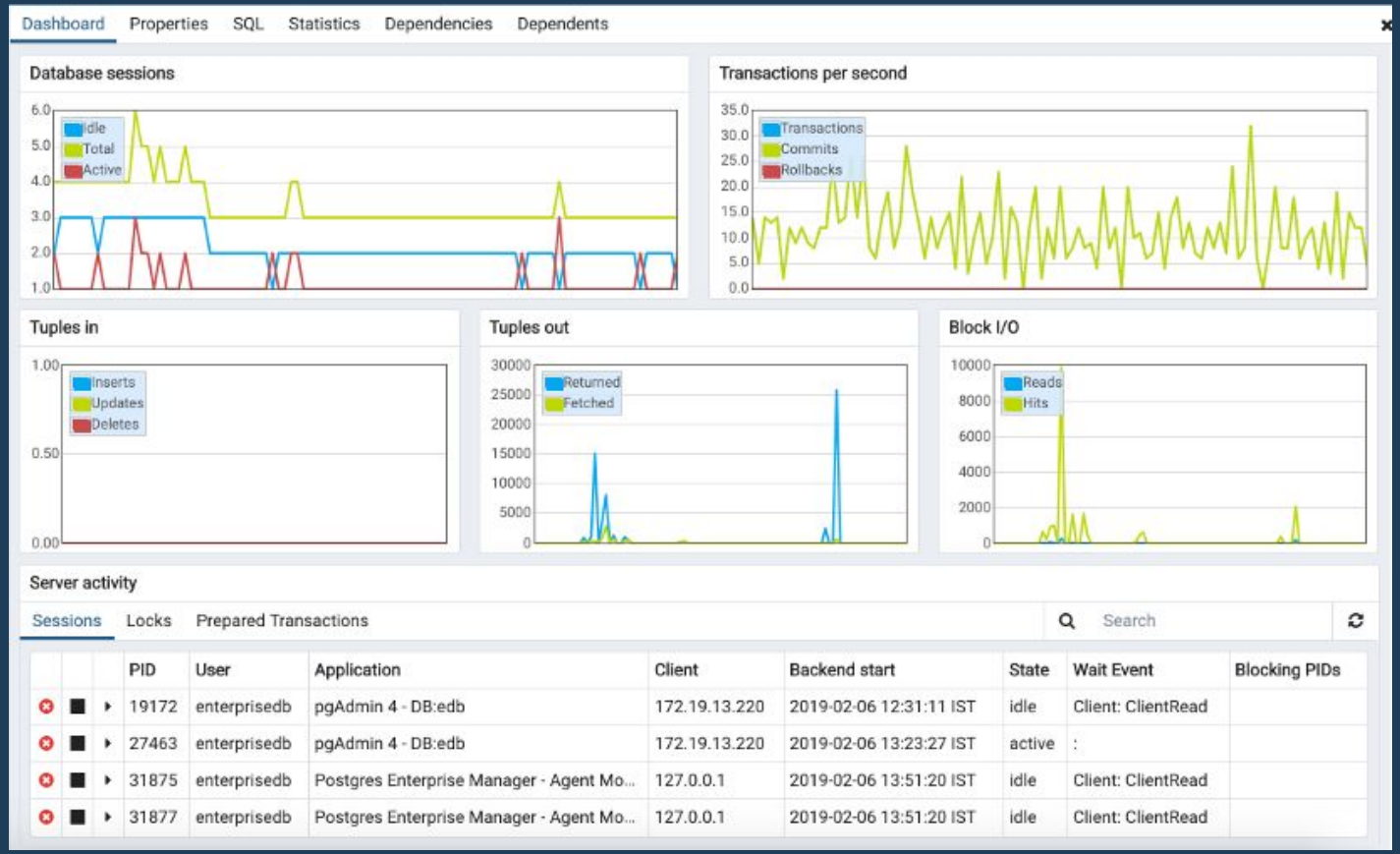
Open Source “ad-hoc” tools

- pg_activity
- pgcenter
- pg_top
- pghero
- pgAdmin4
- DBeaver
- ....

pg_activity


PostgreSQL 14.5 - dbserver - postgres@/var/run/postgresql:5432/postgres - Ref.: 2s - Duration mode: query
* Global: 10 minutes uptime · 1.75G dbs size - 113.85K/s growth · 99.98% cache hit ratio
Sessions: 31/100 total · 31 active · 0 idle · 0 idle in txn · 0 idle in txn abrt · 28 waiting
Activity: 1421 tps · 1416 insert/s · 4249 update/s · 0 delete/s · 10603 tuples returned/s · 0 temp files · 0B temp size
* Worker processes: 0/8 total · 0/4 logical workers · 0/8 parallel workers
Other processes & info: 0/3 autovacuum workers · 0/10 wal senders · 0 wal receivers · 0/10 repl. slots
* Mem.: 5.68G total · 2.61G (45.94%) free · 1.67G (29.37%) used · 1.40G (24.69%) buff+cached
Swap: 5.88G total · 5.88G (100.00%) free · 0B (0.00%) used
IO: 3023/s max iops · 0B/s - 0/s read · 11.81M/s - 3023/s write
Load average: 1.12 0.71 0.41
RUNNING QUERIES
PID DATABASE CPU% MEM% READ/s WRITE/s TIME+ Waiting IOW state Query
10939 bench 6.2 0.7 0B 369.70K 0.114618 transactionid N active UPDATE pgbench_tellers SET tbalanc
10912 bench 5.7 0.6 0B 373.50K 0.113902 transactionid N active UPDATE pgbench_tellers SET tbalanc
10918 bench 7.1 0.6 0B 428.48K 0.069780 transactionid N active UPDATE pgbench_tellers SET tbalanc
10930 bench 6.6 0.7 0B 420.89K 0.048208 tuple N active UPDATE pgbench_tellers SET tbalanc
10917 bench 6.7 0.6 0B 394.35K 0.042520 transactionid N active UPDATE pgbench_tellers SET tbalanc
10937 bench 6.7 0.7 0B 401.94K 0.041018 tuple N active UPDATE pgbench_tellers SET tbalanc
10920 bench 6.2 0.7 0B 396.25K 0.032763 transactionid N active UPDATE pgbench_tellers SET tbalanc
10940 bench 6.7 0.6 0B 420.89K 0.031979 transactionid N active UPDATE pgbench_tellers SET tbalanc
10913 bench 7.1 0.7 0B 401.94K 0.023534 tuple N active UPDATE pgbench_tellers SET tbalanc
10933 bench 5.2 0.6 0B 337.47K 0.018544 transactionid N active UPDATE pgbench_tellers SET tbalanc
10936 bench 6.7 0.7 0B 424.69K 0.015634 transactionid N active UPDATE pgbench_tellers SET tbalanc
10926 bench 6.2 0.6 0B 386.77K 0.014867 transactionid N active UPDATE pgbench_tellers SET tbalanc
10921 bench 6.6 0.6 0B 400.04K 0.012748 transactionid N active UPDATE pgbench_tellers SET tbalanc
10924 bench 6.2 0.6 0B 386.77K 0.012305 N active UPDATE pgbench_branches SET bbalan
10911 bench 7.1 0.6 0B 413.31K 0.010530 tuple N active UPDATE pgbench_tellers SET tbalanc
10931 bench 6.6 0.7 0B 405.73K 0.009143 tuple N active UPDATE pgbench_tellers SET tbalanc
10923 bench 5.7 0.6 0B 337.47K 0.008341 tuple N active UPDATE pgbench_tellers SET tbalanc
10914 bench 6.6 0.7 0B 413.31K 0.007549 transactionid N active UPDATE pgbench_tellers SET tbalanc
10935 bench 7.6 0.6 0B 475.88K 0.004729 transactionid N active UPDATE pgbench_tellers SET tbalanc
10915 bench 5.7 0.6 0B 288.18K 0.004509 tuple N active UPDATE pgbench_branches SET bbalan
10934 bench 7.1 0.7 0B 401.94K 0.004141 transactionid N active UPDATE pgbench_tellers SET tbalanc
10932 bench 7.1 0.6 0B 430.37K 0.003338 tuple N active UPDATE pgbench_tellers SET tbalanc
F1/1 Running queries F2/2 Waiting queries F3/3 Blocking queries Space Pause/unpause q Quit h Help

pgAdmin 4


Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
Database sessions
idle
Total
Active
Transactions per second
Transactions
Commits
Rollbacks
Tuples in
Inserts
Updates
Deletes
Tuples out
Returned
Fetched
Block I/O
Reads
Hits
Server activity
Sessions
Locks
Prepared Transactions
Search
PID
User
Application
Client
Backend start
State
Wait Event
Blocking PIDs
19172 enterprisedb pgAdmin 4 - DB:edb 172.19.13.220 2019-02-06 12:31:11 IST idle Client: ClientRead
27463 enterprisedb pgAdmin 4 - DB:edb 172.19.13.220 2019-02-06 13:23:27 IST active :
31875 enterprisedb Postgres Enterprise Manager - Agent Mo... 127.0.0.1 2019-02-06 13:51:20 IST idle Client: ClientRead
31877 enterprisedb Postgres Enterprise Manager - Agent Mo... 127.0.0.1 2019-02-06 13:51:20 IST idle Client: ClientRead

Continuous monitoring frameworks

Commercial (mostly “agent” based)

- AppDynamics
- New Relic
- Datadog
- Vividcortex
- EDB Enterprise Manager
- pganalyze
- …

Continuous monitoring frameworks

Generic Open Source

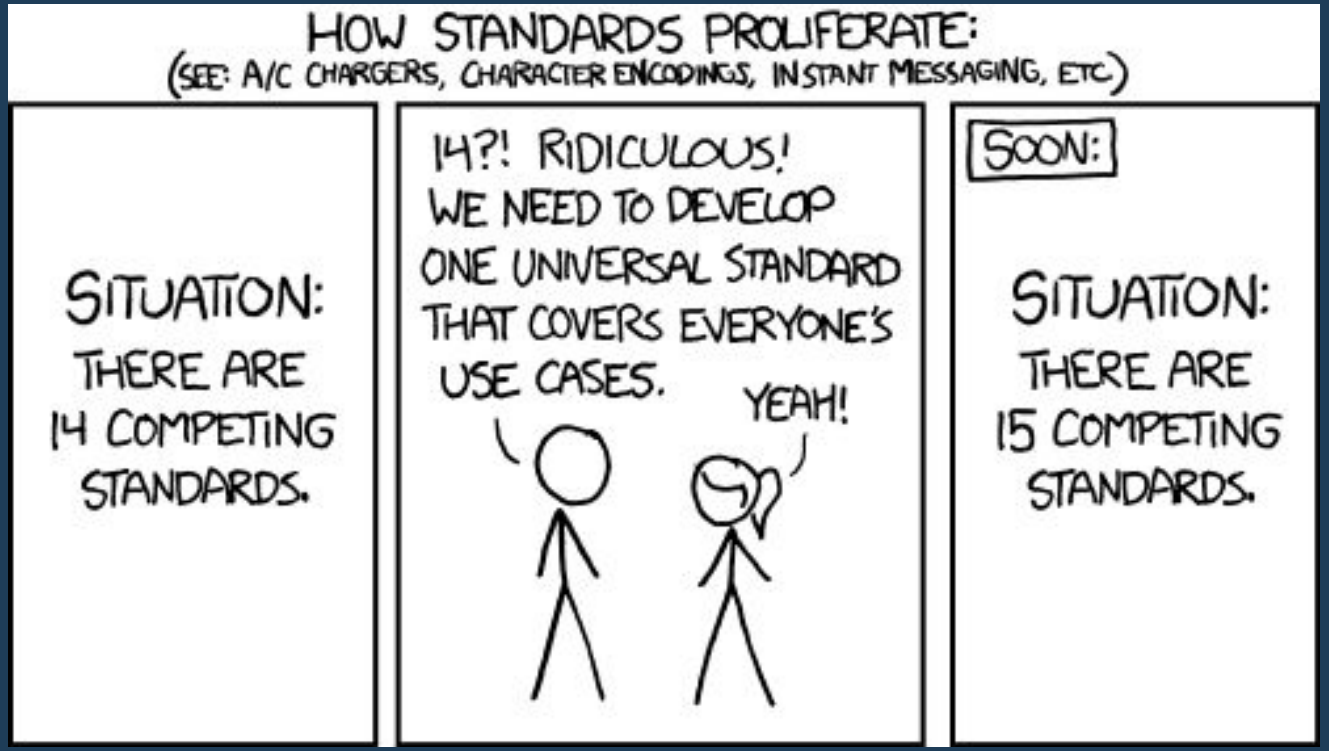
- Nagios
- Icinga
- Munin
- Zabbix

Base mostly on “check_postgres” script or derivatives

Postgres specific

- pghero
- PoWA (server side, quite advanced - pg_qualstats, pg_stat_kcache, pg_wait_sampling)
- PgCluu (server side, with “sar” system info)
- pg_statviz
- pgwatch2 (client or server side)
- …

pgwatch2?


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

Monitoring PostgreSQL made simple

pgwatch2

Thanks!

Kaarel Moppel

The author or pgwatch2

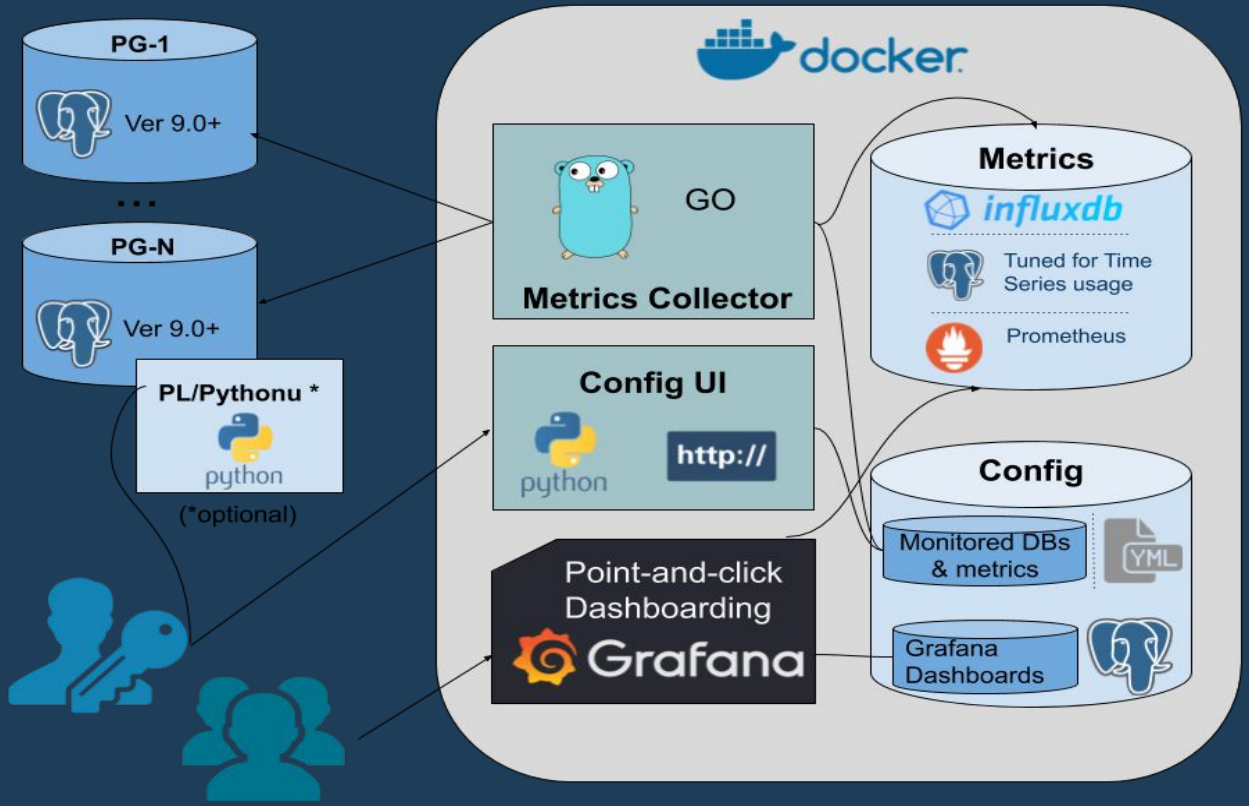
Main principles - why another tool?

- 1-minute setup
  - Docker (custom setup also possible)
- User changeable visuals / dashboarding
- Non-invasive
  - No extensions or superuser needed for base functionality
- Easy extensibility, do minimal work needed
  - SQL metrics
- Simple alerting via Grafana possible

Architecture components

- Metrics gathering daemon (Golang)
- Configuration database / YAML files / ENV vars
- Metrics storage layer
  - PostgreSQL (with TimescaleDB optionally)
  - InfluxDB (deprecated)
  - Graphite
  - Prometheus
  - Local file
- Optional simple Web UI for administration
- Grafana for intuitive point-and-click dashboarding

Architecture components



Features

- “Ready to go”
  - Default metrics cover almost all `pg_stat*` views
  - Pre-configured dashboards for almost all metrics
- Supports Postgres 9.0+ out of the box
- Configurable security - passwords, SSL
- Reuse of existing Postgres, Grafana, InfluxDB installations possible
- Kubernetes/OpenStack ready

Features

- Per DB setup with optional auto-discovery of all DBs of a cluster
- Change detection for table/index/sproc/configuration events
- AWS RDS CloudWatch metrics support
- PgBouncer & Pgpool2 metrics support
- Patroni support
- Very low resource requirements
- A Ping mode to test connectivity to monitored databases
- Extensible
  - Custom metrics via SQL, i.e. usable also for business layer!
  - Built-in log parsing and OS level metrics
  - Grafana has lot of plugins as well

Alerting

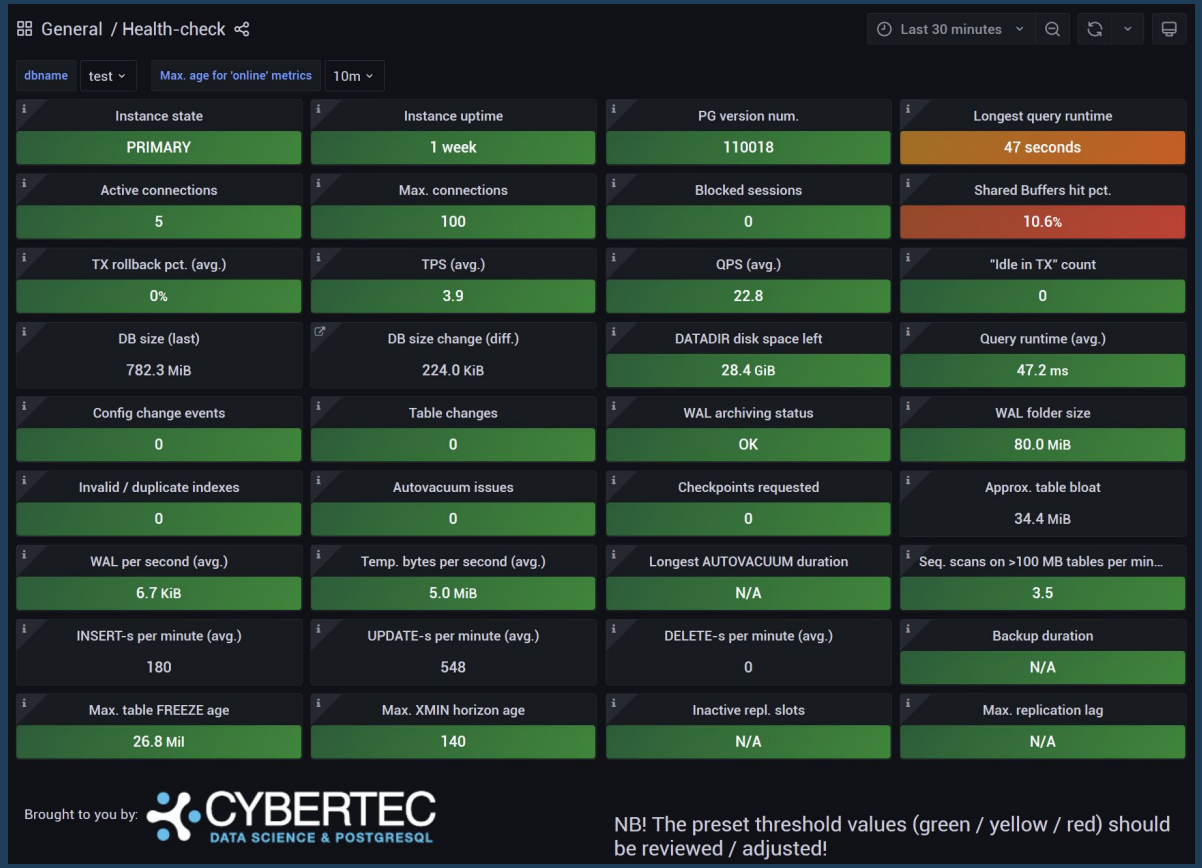
- Quite easy with Grafana, “point-and-click”
  - Most important alerting services covered
  - Email
  - Slack
  - PagerDuty
  - Web hooks
  - …
- Based on graph panels only currently
- Use **pg_timetable** scheduler to analyze and send alerts

Getting started

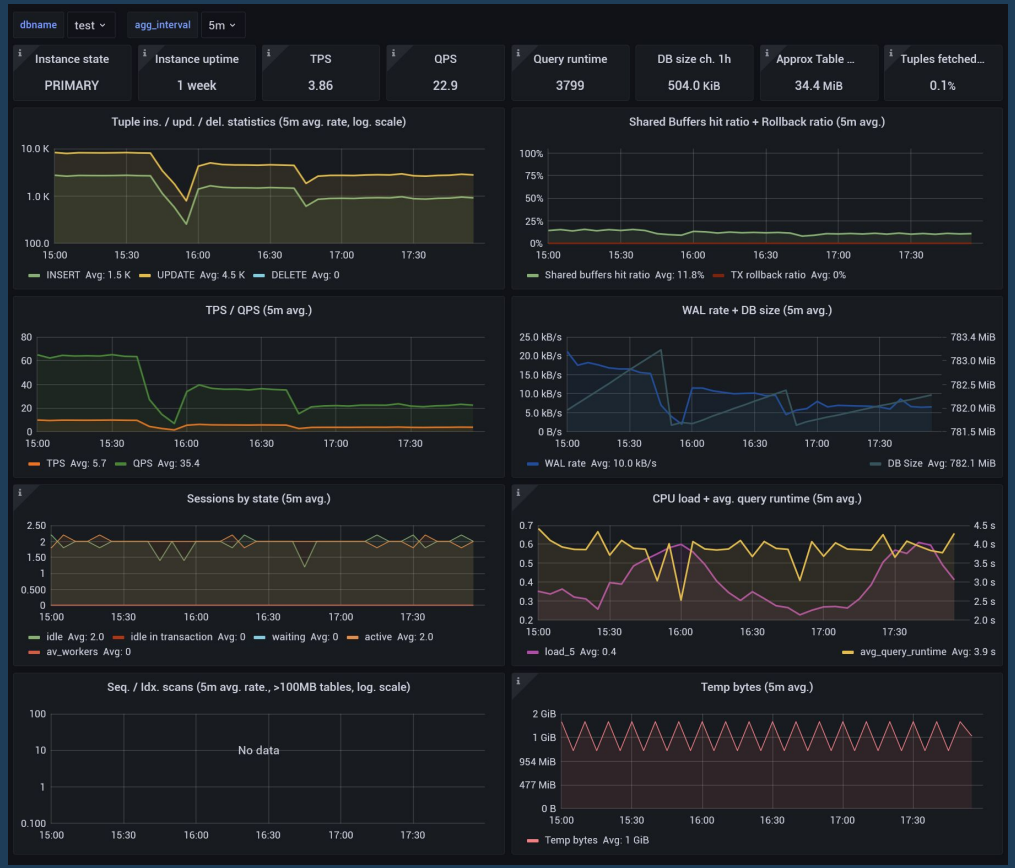
1.
```
docker run -d --restart=unless-stopped \
    -p 3000:3000 -p 8080:8080 \
    --name pw2 cybertec/pgwatch2-postgres
```
2. Wait some seconds and open browser at `localhost:8080`
3. Insert your DB connection strings
4. Start viewing/editing dashboards in 5 min...

or check online demo at **https://demo.pgwatch.com**

Health Check Dashboard



Database Overview Dashboard



Stat Statements Dashboard

Monitoring PostgreSQL made simple

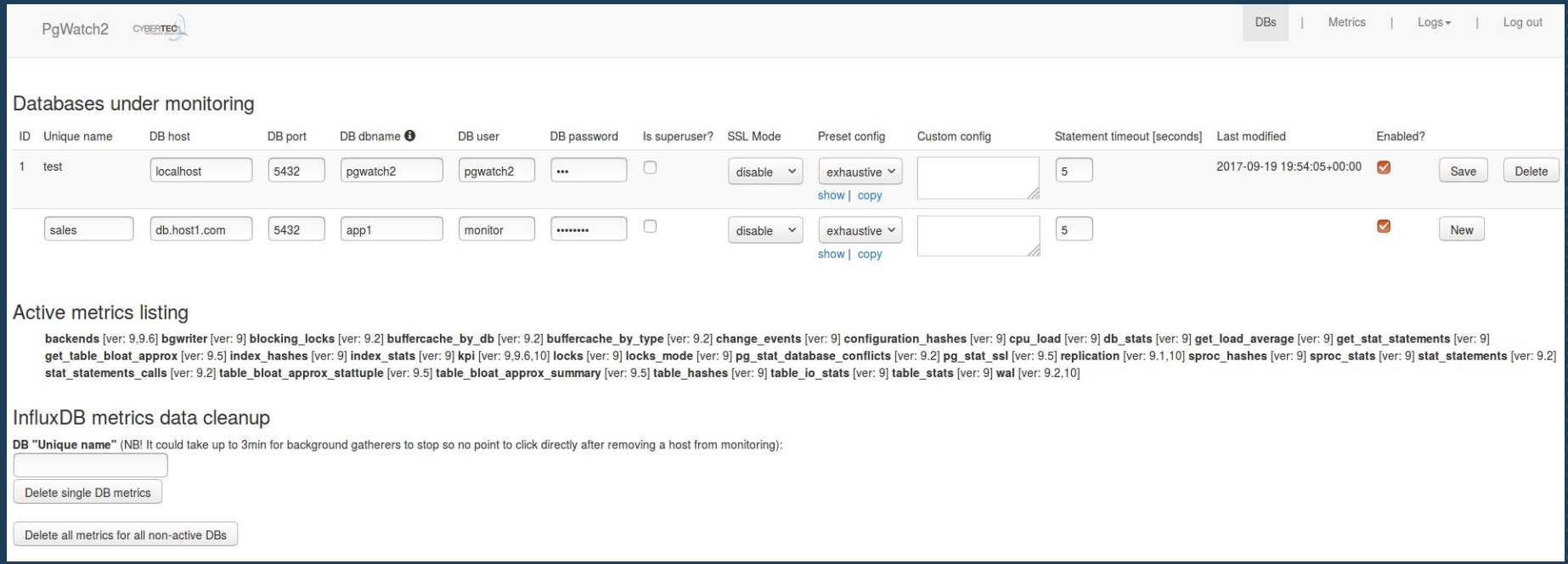
pgwatch3?

CLASSIFIED

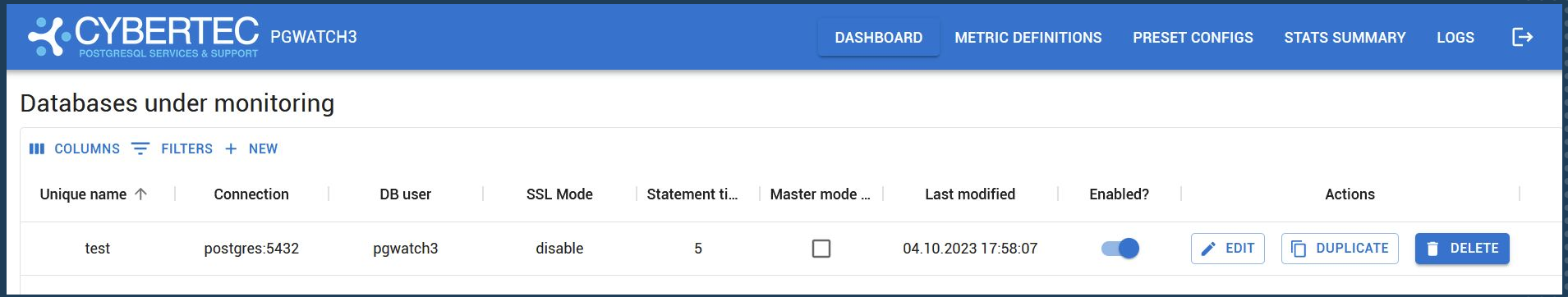

YO DAWG! HEARD YOU LIKE MONITORING
SO I'VE MADE YOU PGWATCH MAINTAINER.
NOW YOU CAN MONITOR WHILE YOU'RE PROGRAMMING
imgflip.com


YO DAWG! HEARD YOU LIKE MONITORING
SO I'VE MADE YOU PGWATCH MAINTAINER.
NOW YOU CAN MONITOR WHILE YOU'RE PROGRAMMING
imgflip.com

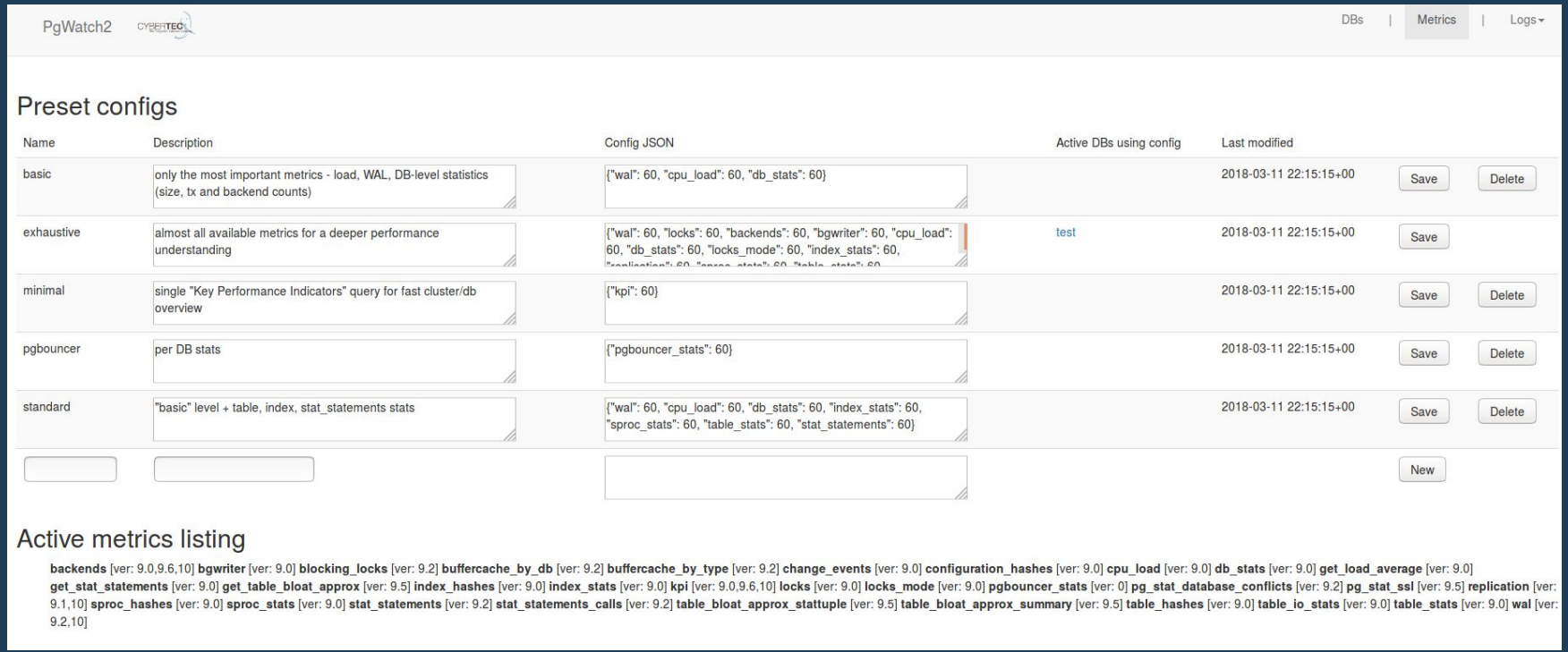
Web UI: Before



Web UI: After


CYBERTEC
POSTGRESQL SERVICES & SUPPORT
PGWATCH3
DASHBOARD
METRIC DEFINITIONS
PRESET CONFIGS
STATS SUMMARY
LOGS
Databases under monitoring
COLUMNS
FILTERS
NEW
Unique name
Connection
DB user
SSL Mode
Statement ti...
Master mode ...
Last modified
Enabled?
Actions
test
postgres:5432
pgwatch3
disable
5
04.10.2023 17:58:07
EDIT
DUPLICATE
DELETE

Web UI: Before



PgWatch2 CYBERTEC

DBs | Metrics | Logs

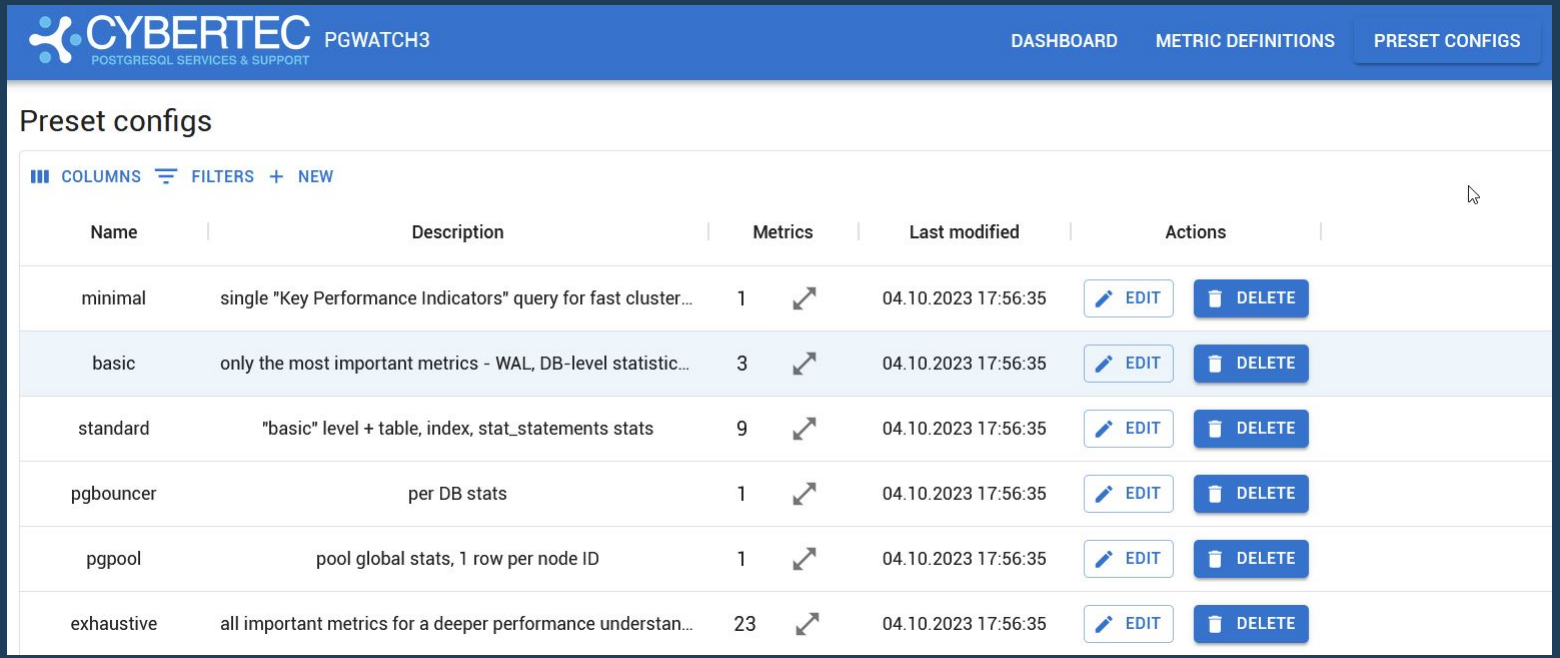
Preset configs

Name	Description	Config JSON	Active DBs using config	Last modified		
basic	only the most important metrics - load, WAL, DB-level statistics (size, tx and backend counts)	{"wal": 60, "cpu_load": 60, "db_stats": 60}		2018-03-11 22:15:15+00	Save	Delete
exhaustive	almost all available metrics for a deeper performance understanding	{"wal": 60, "locks": 60, "backends": 60, "bgwriter": 60, "cpu_load": 60, "db_stats": 60, "locks_mode": 60, "index_stats": 60,	test	2018-03-11 22:15:15+00	Save	
minimal	single "Key Performance Indicators" query for fast cluster/db overview	{"kpi": 60}		2018-03-11 22:15:15+00	Save	Delete
pgbouncer	per DB stats	{"pgbouncer_stats": 60}		2018-03-11 22:15:15+00	Save	Delete
standard	"basic" level + table, index, stat_statements stats	{"wal": 60, "cpu_load": 60, "db_stats": 60, "index_stats": 60, "sproc_stats": 60, "table_stats": 60, "stat_statements": 60}		2018-03-11 22:15:15+00	Save	Delete
					New	

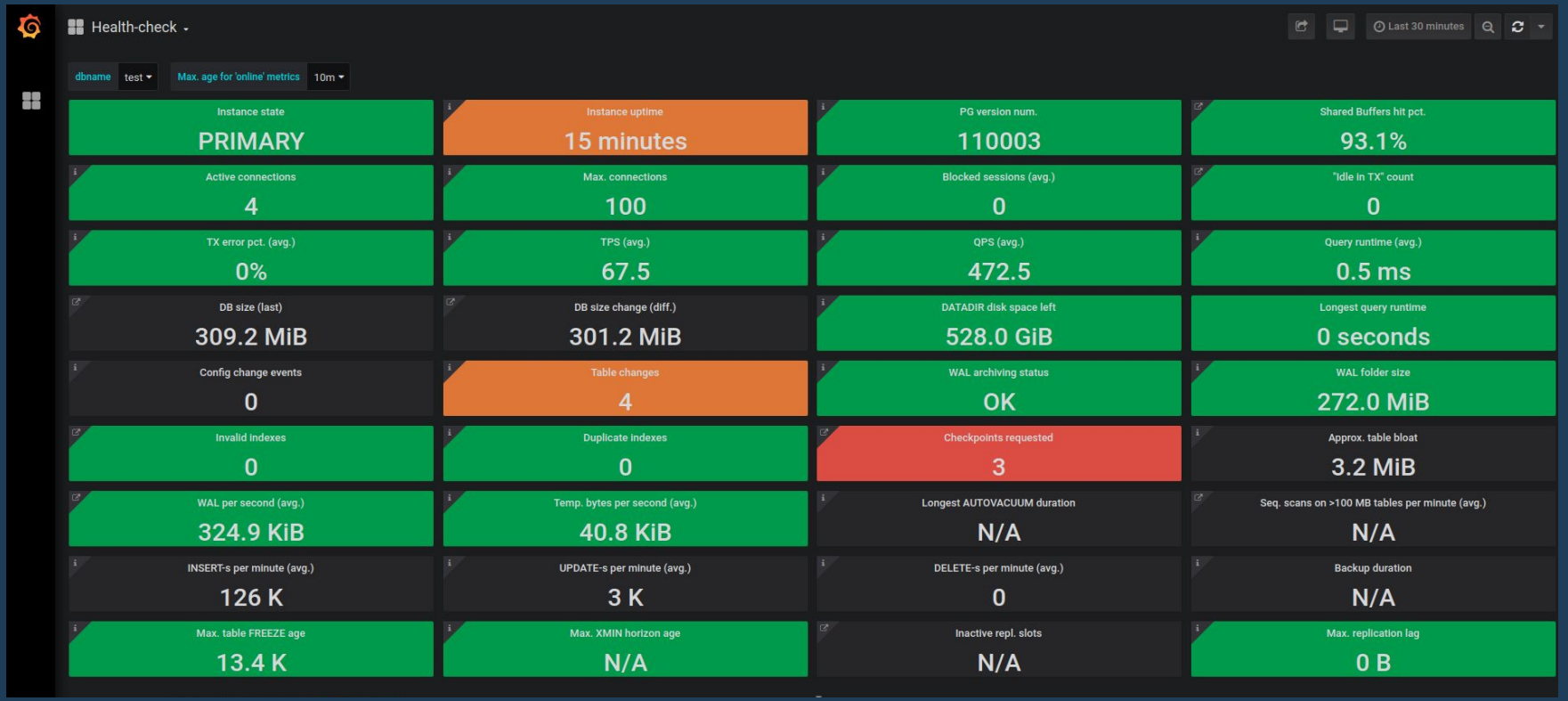
Active metrics listing

backends [ver: 9.0,9.6,10] **bgwriter** [ver: 9.0] **blocking_locks** [ver: 9.2] **buffercache_by_db** [ver: 9.2] **buffercache_by_type** [ver: 9.2] **change_events** [ver: 9.0] **configuration_hashes** [ver: 9.0] **cpu_load** [ver: 9.0] **db_stats** [ver: 9.0] **get_load_average** [ver: 9.0] **get_stat_statements** [ver: 9.0] **get_table_bloat_approx** [ver: 9.5] **index_hashes** [ver: 9.0] **index_stats** [ver: 9.0] **kpi** [ver: 9.0,9.6,10] **locks** [ver: 9.0] **locks_mode** [ver: 9.0] **pgbouncer_stats** [ver: 0] **pg_stat_database_conflicts** [ver: 9.2] **pg_stat_ssl** [ver: 9.5] **replication** [ver: 9.1,10] **sproc_hashes** [ver: 9.0] **sproc_stats** [ver: 9.0] **stat_statements** [ver: 9.2] **stat_statements_calls** [ver: 9.2] **table_bloat_approx_stattuple** [ver: 9.5] **table_bloat_approx_summary** [ver: 9.5] **table_hashes** [ver: 9.0] **table_io_stats** [ver: 9.0] **table_stats** [ver: 9.0] **wal** [ver: 9.2,10]

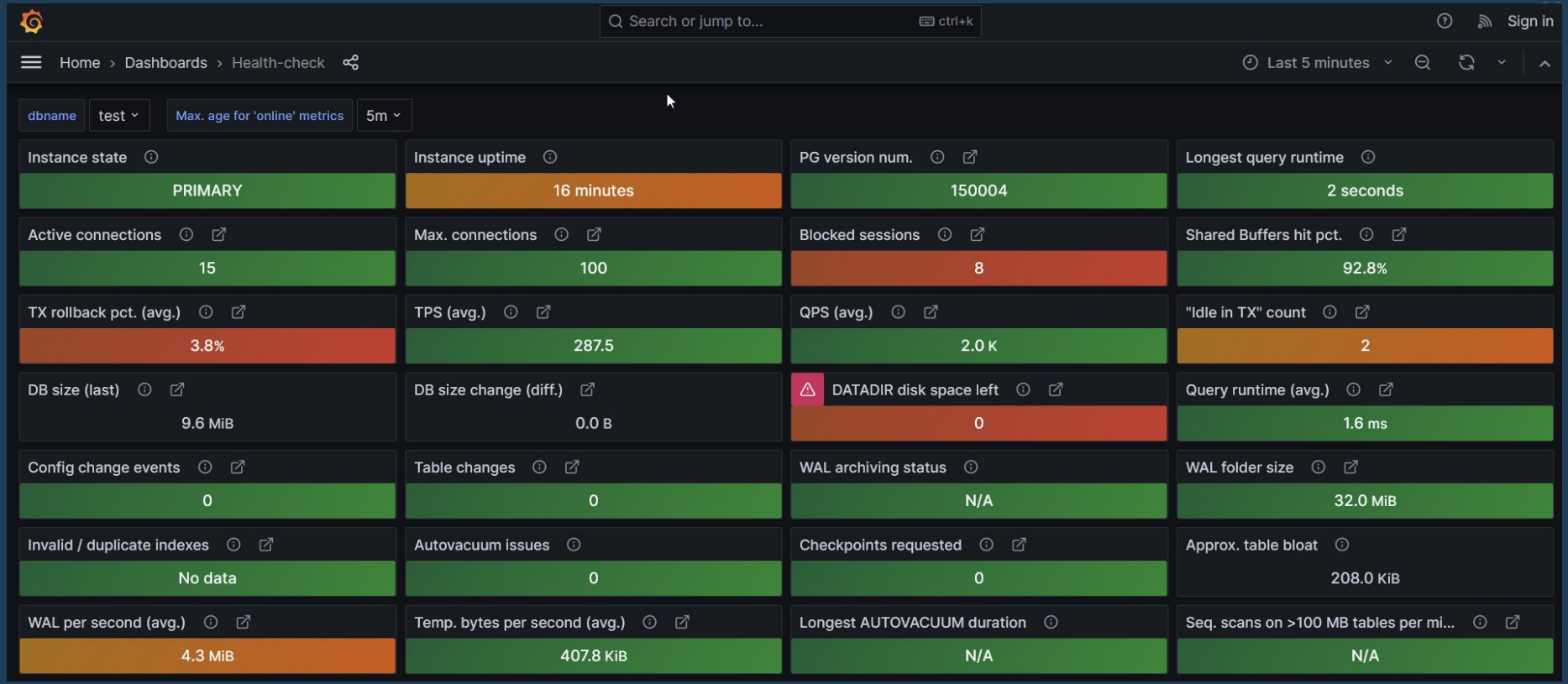
Web UI: After


CYBERTEC
POSTGRESQL SERVICES & SUPPORT
PGWATCH3
DASHBOARD
METRIC DEFINITIONS
PRESET CONFIGS
Preset configs
COLUMNS
FILTERS
NEW
Name
Description
Metrics
Last modified
Actions
minimal
single "Key Performance Indicators" query for fast cluster...
1
04.10.2023 17:56:35
EDIT
DELETE
basic
only the most important metrics - WAL, DB-level statistic...
3
04.10.2023 17:56:35
EDIT
DELETE
standard
"basic" level + table, index, stat_statements stats
9
04.10.2023 17:56:35
EDIT
DELETE
pgbouncer
per DB stats
1
04.10.2023 17:56:35
EDIT
DELETE
pgpool
pool global stats, 1 row per node ID
1
04.10.2023 17:56:35
EDIT
DELETE
exhaustive
all important metrics for a deeper performance understan...
23
04.10.2023 17:56:35
EDIT
DELETE

Dashboards: Before



Dashboards: After



Among other things

- TimescaleDB as a metric storage is on by default
  - compression is on
  - 45M rows of real world stat_statements data (6db * 3mo)
    - current storage: 70 GB → 8 GB compressed (ratio 8.9x)
    - upcoming feat: 11 GB → 763 MB (ratio 14.5x)

Among other things

- Parallel metric storages

```
src  309-allow-several-data-sinks ≡ ?2 ~8  1.21.4  go run . --host=localhost --user=pgwatch3 --password=pgwatch3 --dbname=pgwatch3 --pg-metric-store-conn-str=postgresql://pgwatch3:pgwatch3@localhost:5432/pgwatch3_metrics --json-storage-file=metrics.json --prometheus-listen-addr=127.0.0.1
2023-11-10 17:27:17.273 [INFO] Executing configuration schema scripts: 2
2023-11-10 17:27:17.276 [INFO] starting MetricsBatcher...
2023-11-10 17:27:17.278 [INFO] [file:metrics.json] JSON output enabled
2023-11-10 17:27:17.351 [INFO] Executing metric storage schema scripts: 6
2023-11-10 17:27:17.372 [INFO] [connstr:postgresql://pgwatch3:pgwatch3@localhost:5432/pgwatch3_metrics] PostgreSQL output enabled
2023-11-10 17:27:17.373 [INFO] [listen:127.0.0.1] Prometheus output enabled
2023-11-10 17:27:17.373 [INFO] updating metrics definitons from ConfigDB...
2023-11-10 17:27:17.392 [INFO] [databases:1] [metrics:89] host info refreshed
2023-11-10 17:27:17.657 [INFO] Writing 2 metric sets to JSON file at "metrics.json"...
2023-11-10 17:27:17.682 [INFO] wrote 1/1 rows from 2 metric sets to Postgres in 8.9 ms
2023-11-10 17:27:18.049 [INFO] [database:test] [metric:backends] [interval:60] starting gatherer
2023-11-10 17:27:18.052 [INFO] [database:test] [metric:table_stats] [interval:300] starting gatherer
```

Among other things

- Docker images
  - are multiplatform and are slimmer
  - no more monstrous all-in-one image
    - docker compose is shipped instead

```
docker images -a
REPOSITORY                      TAG     IMAGE ID         SIZE
cybertec/pgwatch2-postgres    latest    7a889edafe50    1.17GB
cybertec/pgwatch3-demo        latest    5e3b24fb5a6a    753MB
cybertec/pgwatch3             latest   2cbd8fc36e28    44.1MB
```

Be Inspired

“I would rather have questions that can't be answered than answers that can't be questioned.”

Richard Feynman

DON'T BE A STRANGER

PERSONAL GITHUB

www.github.com/pashagolub

CYBERTEC BLOG

www.cybertec-postgresql.com/en/blog/

CYBERTEC GITHUB

www.github.com/cybertec-postgresql

#StandWithUkraine