Monitoring PostgreSQL made simple

postgresql.istanbul 2026

Pavlo Golub

Senior Consultant/Developer

Email

pavlo.golub@cybertec.at

Social

pashagolub.github.io

www.cybertec-postgresql.com

@cybertec-postgresql

www.youtube.com/@cybertecpostgresql

PROUD CONTRIBUTOR TO PostgreSQL

Database Products & Tools

AUSTRIA (HQ)

CYBERTEC POSTGRESQL
INTERNATIONAL (HQ)

SWITZERLAND

CYBERTEC POSTGRESQL
SWITZERLAND

URUGUAY

CYBERTEC POSTGRESQL
SOUTH AMERICA

ESTONIA

CYBERTEC POSTGRESQL
NORDIC

POLAND

CYBERTEC POSTGRESQL
POLAND

INDIA

CYBERTEC POSTGRESQL
INDIA PRIVATE LIMITED

SOUTH AFRICA

CYBERTEC POSTGRESQL
SOUTH AFRICA

Agenda

1. Different levels of database monitoring
2. PostgreSQL monitoring approaches
3. PostgreSQL monitoring tools available
4. pgwatch
5. Live demo??? 😱

Monitoring PostgreSQL made simple

Different levels of database monitoring

Why monitor?

- Failure / Downtime detection
- Slowness / Performance analysis
- Proactive predictions
- Possibly 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 things...

System monitoring

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

- Do you know what the CPU load number actually means?
  - Is it a good metric?
- What's the difference between VIRT, RES and 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
  - pgweasel
  - 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

```
1| postgres=# SELECT count(*) FROM pg_settings
2| WHERE category LIKE 'Reporting and Logging%';
3|  count
4| -------
5|    44
```

Statistics Collector

- Not all track_* parameters enabled by default
- Dynamic views (13 views)
  - `pg_stat_activity`, `pg_stat_(replication|wal_receiver)`,
  - `pg_locks`, `pg_stat_ssl`, `pg_stat_progress_*`
- Cumulative views (21 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)
- **pg_stat_plans** (“top plans”)
- **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”

- DIY
  - 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

FIND MORE

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

No shortage of tools!

Real-time troubleshooting

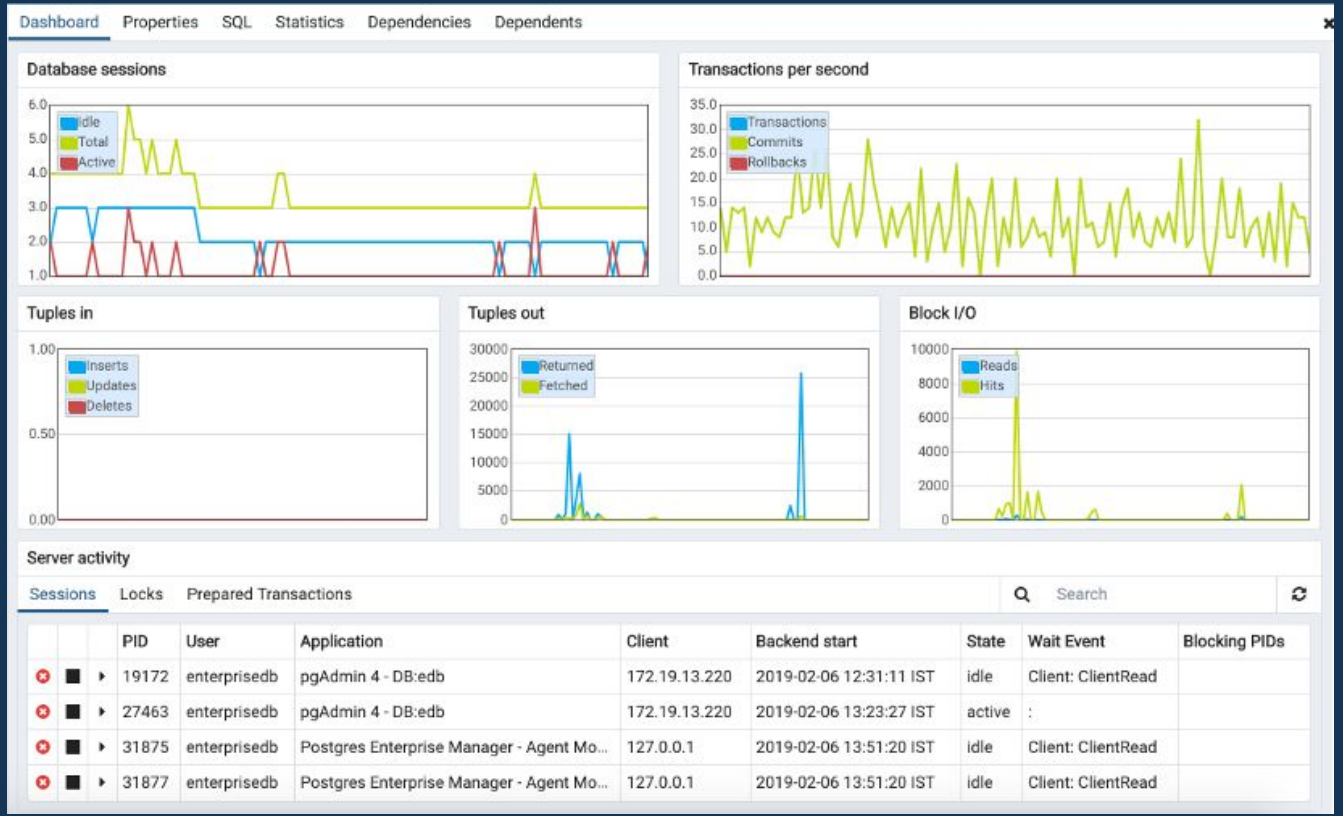
Open Source real-time tools

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

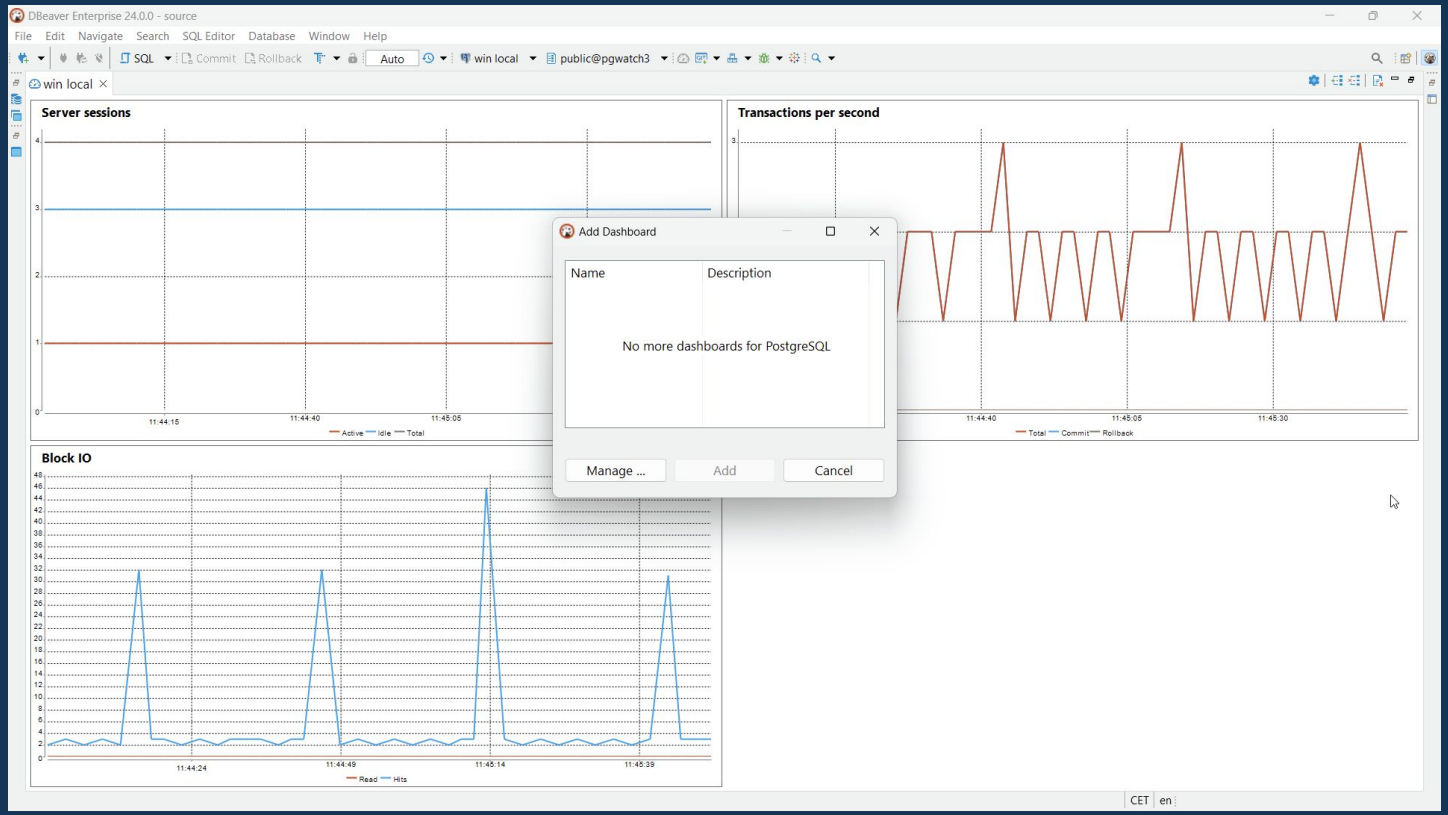
pg_activity

```
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

DBeaver


DBeaver Enterprise 24.0.0 - source
File Edit Navigate Search SQL Editor Database Window Help
win local
Server sessions
Transactions per second
Block IO
Add Dashboard
Name
Description
No more dashboards for PostgreSQL
Manage ...
Add
Cancel
CET
en

Continuous monitoring frameworks

Commercial (mostly “agent” based)

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

Continuous monitoring frameworks

Generic Open Source

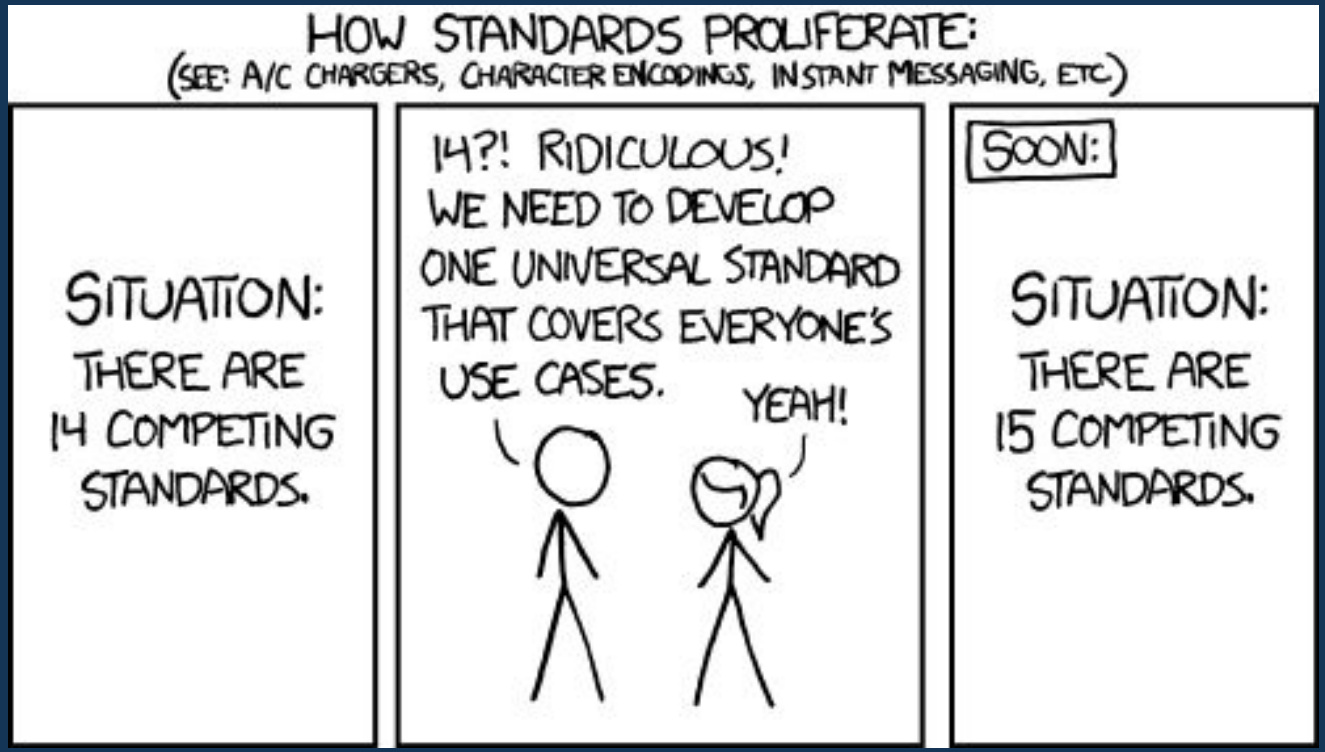
- Nagios
- Icinga
- Munin
- Zabbix

Based 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
- pgwatch (client or server side)
- ...

pgwatch?


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

pgwatch

Thanks

Kaarel Moppel

The author of pgwatch v2

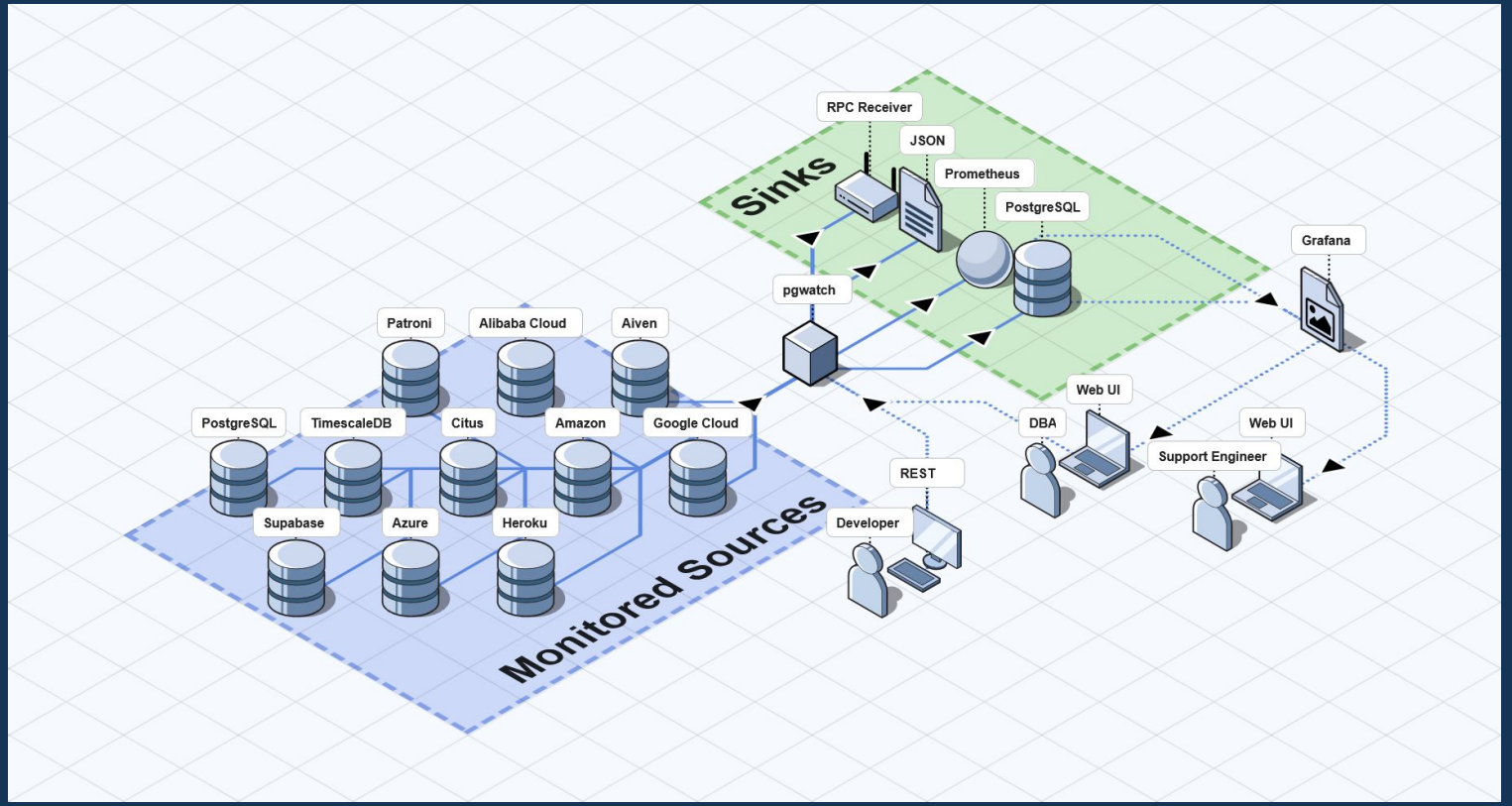
Main principles - why another tool?

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

Architecture components

- Metrics gathering daemon (Golang)
- Configuration
  - PostgreSQL compatible database/YAML files/ENV vars
- Measurement sinks
  - PostgreSQL (with TimescaleDB optionally)
  - Prometheus
  - Local file
  - RPC custom sink
- Optional simple Web UI for administration
- Grafana for intuitive point-and-click dashboarding

Architecture components



Features

- “Ready to go”
  - Default metrics cover all pg_stat* views
  - Pre-configured dashboards for almost all metrics
- Supports Postgres 11+ out of the box
- Configurable security - passwords, SSL
- Reuse of existing Postgres, Grafana, Prometheus 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 a lot of plugins as well

Alerting

- Quite easy with Grafana, “point-and-click”
- Alertmanager if Prometheus involved
- Use **pg_timetable** scheduler to analyze and send alerts

Among other things

- TimescaleDB as a metric storage is available out of the box
  - 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 measurement storages

```
src  master ≡ ?1  1.22.0  ./pgwatch3 --config=sources/sample.sources.yaml
  --sink=jsonfile://metrics.json  --sink=postgresql://pgwatch3@localhost/pgwatch3_metrics -
-sink=prometheus://:9127
2024-04-11 13:00:03.544 [INFO] [sink:jsonfile://metrics.json] measurements sink actviated
2024-04-11 13:00:03.683 [INFO] [sink:postgresql://pgwatch3@localhost/pgwatch3_metrics] init
ialising the measurement database ...
2024-04-11 13:00:03.690 [INFO] [sink:postgresql://pgwatch3@localhost/pgwatch3_metrics] meas
urements sink actviated
2024-04-11 13:00:03.690 [INFO] [sink:prometheus://:9127] measurements sink actviated
2024-04-11 13:00:03.691 [INFO] [metrics:73] [sources:1] host info refreshed
2024-04-11 13:00:03.867 [INFO] Connect OK. [test1] is on version 16.1 (in recovery: false)
2024-04-11 13:00:03.867 [INFO] Trying to create helper functions if missing for "test1" ...
2024-04-11 13:00:03.952 [INFO] [sink:postgres] [rows:1] [db:pgwatch3_metrics] [elapsed:5.83
72ms] measurements written
```

Among other things

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

```
1| docker images -a
2| REPOSITORY                          TAG          IMAGE ID       SIZE
3| cybertec/pgwatch2-postgres          latest       7a889edafe50   1.17GB
4| cybertecpostgresql/pgwatch-demo     latest       5e3b24fb5a6a   753MB
5| cybertecpostgresql/pgwatch          latest       2cbd8fc36e28   44.1MB
```

Getting started

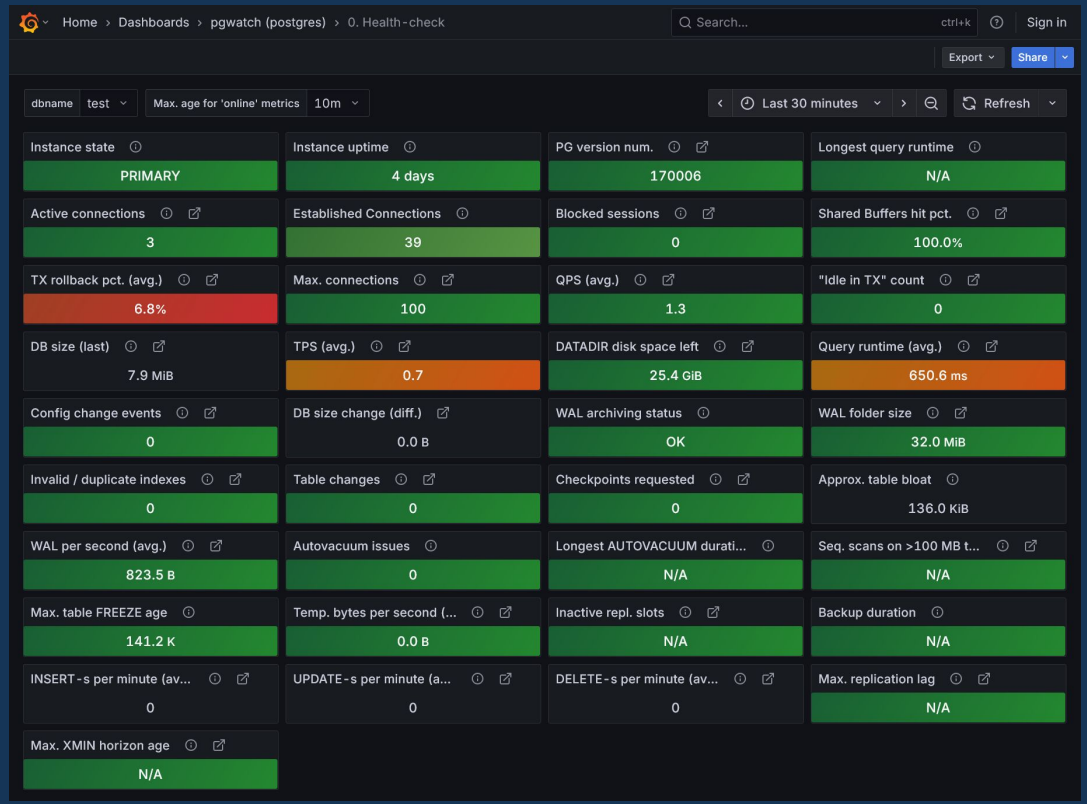
1. docker run -d --name pgw \

   ```
   -p 5432:5432 -p 3000:3000 -p 8080:8080 \
   -e PW_TESTDB=true \
   cybertecpostgresql/pgwatch-demo
   ```

2. Wait a few seconds and open browser at localhost:8080
3. Insert your DB connection strings
4. Start viewing/editing dashboards in 5 min…

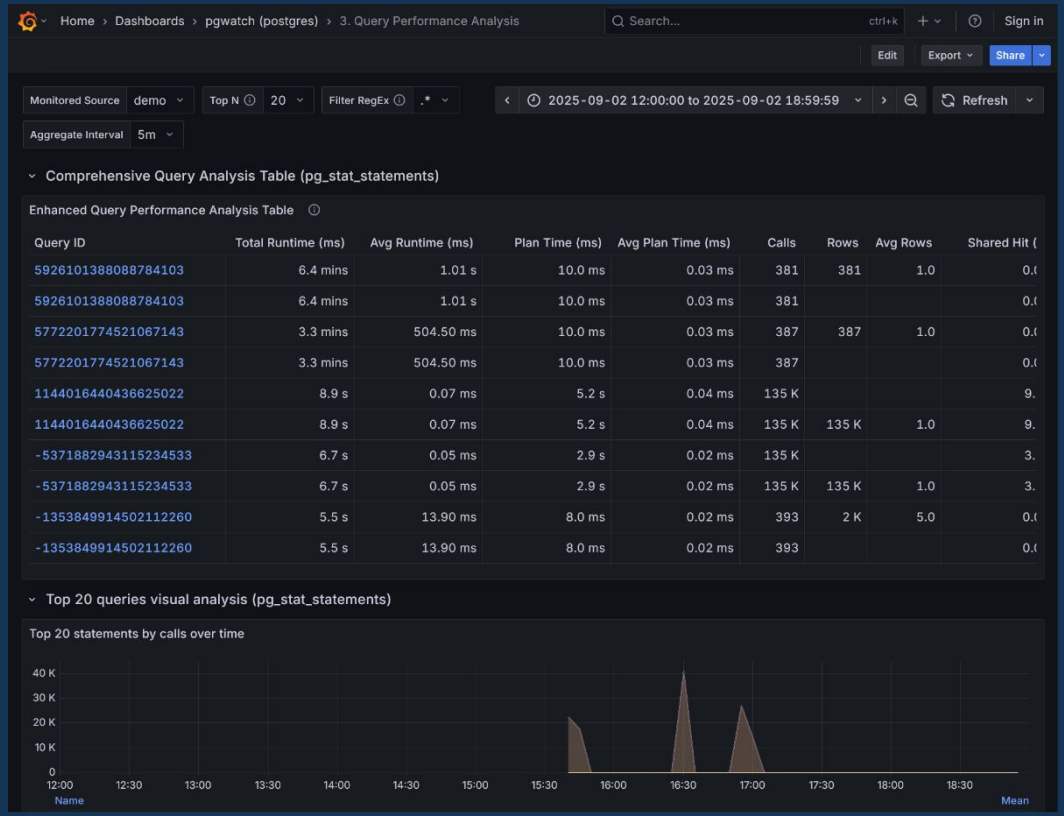
or check online demo at https://demo.pgwatch.com

Health Check Dashboard



Database Overview Dashboard

Stat Statements Dashboard



“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 Page

pashagolub.github.io

Personal GitHub

www.github.com/pashagolub

CYBERTEC Blog

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

CYBERTEC GitHub

www.github.com/cybertec-postgresql

#StandWithUkraine