



**Monitoring PostgreSQL made simple**

pgconf.de 2024

# Senior Consultant/Developer

## Pavlo Golub

### MAIL

[pavlo.golub@cybertec.at](mailto:pavlo.golub@cybertec.at)

### Social

[pashagolub.github.io](https://github.com/pashagolub)

### WEB

[www.cybertec-postgresql.com](http://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

# CUSTOMER SECTORS

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



Lufthansa

amazon



skype™

Raiffeisen  
Meine Bank



PORSCHE

HEROLD.at



Allianz

hims



Auswärtiges Amt

Klarna.

voestalpine

NOKIA



ProSiebenSat.1  
Media AG

ÖBB

Immer in Bewegung



BOSCH

Invented for life

SPAR



METRO

DANONE



Bundeskanzleramt

BRZ



TARGET



NOVOMATIC

TOMTOM



MAGNA



Audi



Atos

LEONI

Rappi



SBB CFF FFS

ALCATEL



JOHN DEERE

ZEISS



Magenta

SIEMENS

VOR  
DER VERKEHRSVERBUND

# DATABASE - PRODUCTS



# WHY

## PostgreSQL?



ADVANCED OPEN  
SOURCE DATABASE  
SYSTEM



25 YEARS OF  
DEVELOPMENT



NO  
LICENSE COSTS



EXTENSIVE  
FUNCTIONALITY



RELIABILITY



LOW  
SUPPORT COSTS



SCALABILITY



[www.cybertec-postgresql.com](http://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”

- 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

## 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

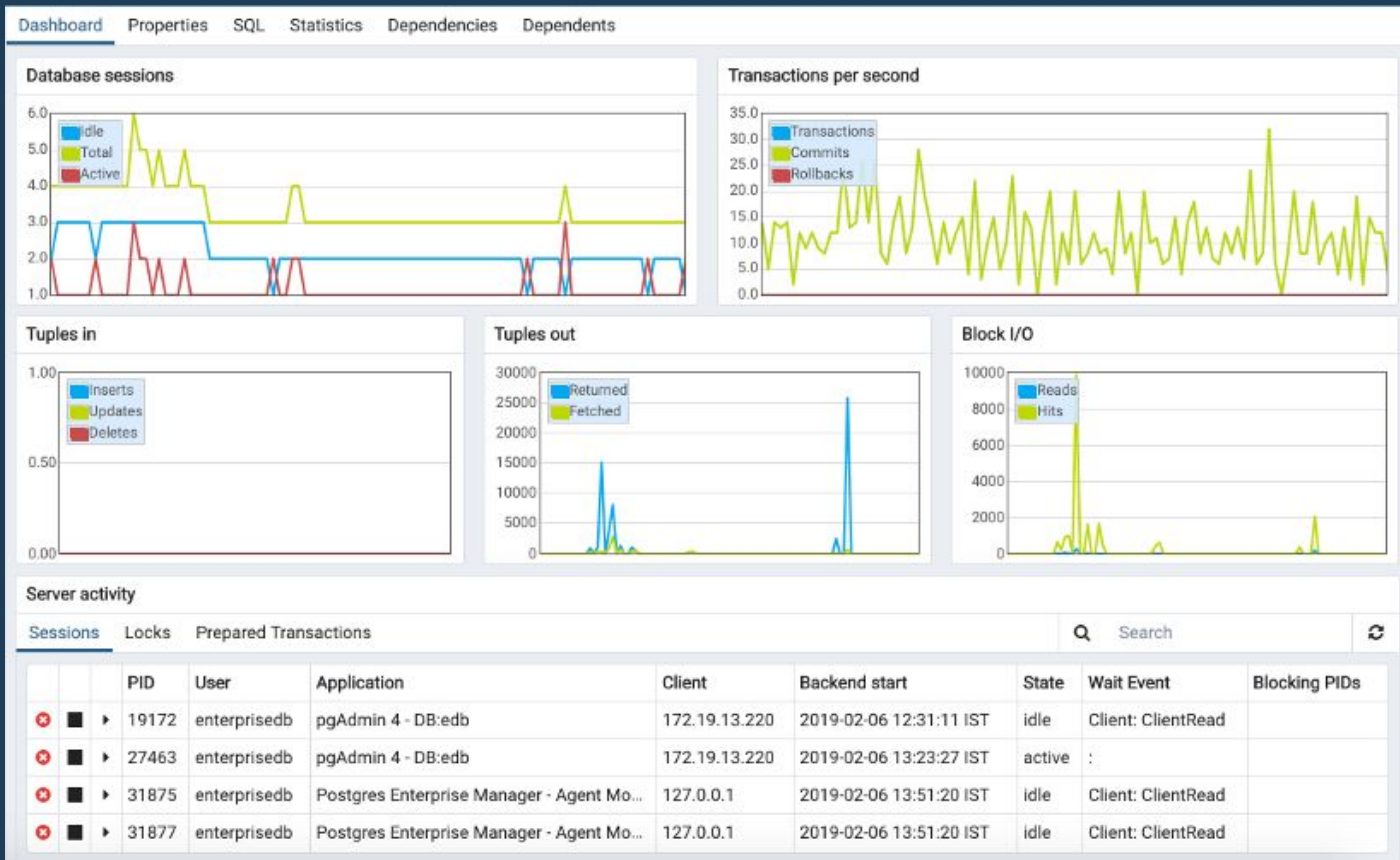
```
PostgreSQL 14.5 - dbserver - postgres@var/run/postgresql:5432/postgres - Ref.: 2s - Duration mode: query
* Global: 10 minutes uptime - 1.75G dbs size - 113.85K/s growth - 99.98% cache hit ratio
Sessions: 31/100 total - 31 active - 0 idle - 0 idle in txn - 0 idle in txn abrt - 28 waiting
Activity: 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 bbalanc
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 bbalanc
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

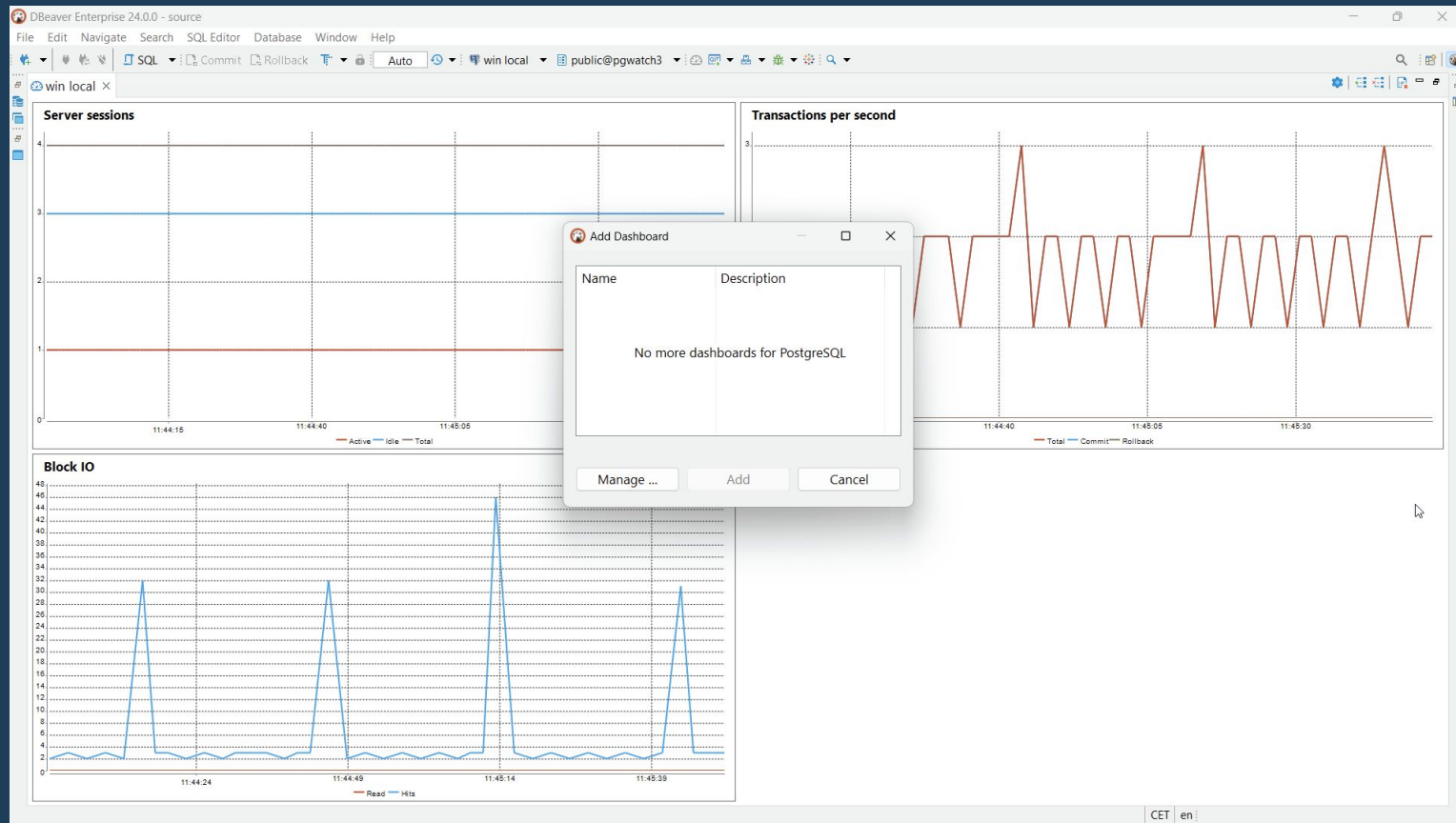
F1/1 Running queries F2/2 Waiting queries F3/3 Blocking queries Space Pause/unpause q Quit h Help

# pgAdmin 4





# DBeaver



# Continuous monitoring frameworks

Commercial (mostly “agent” based)

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

# Continuous monitoring frameworks

## Generic Open Source

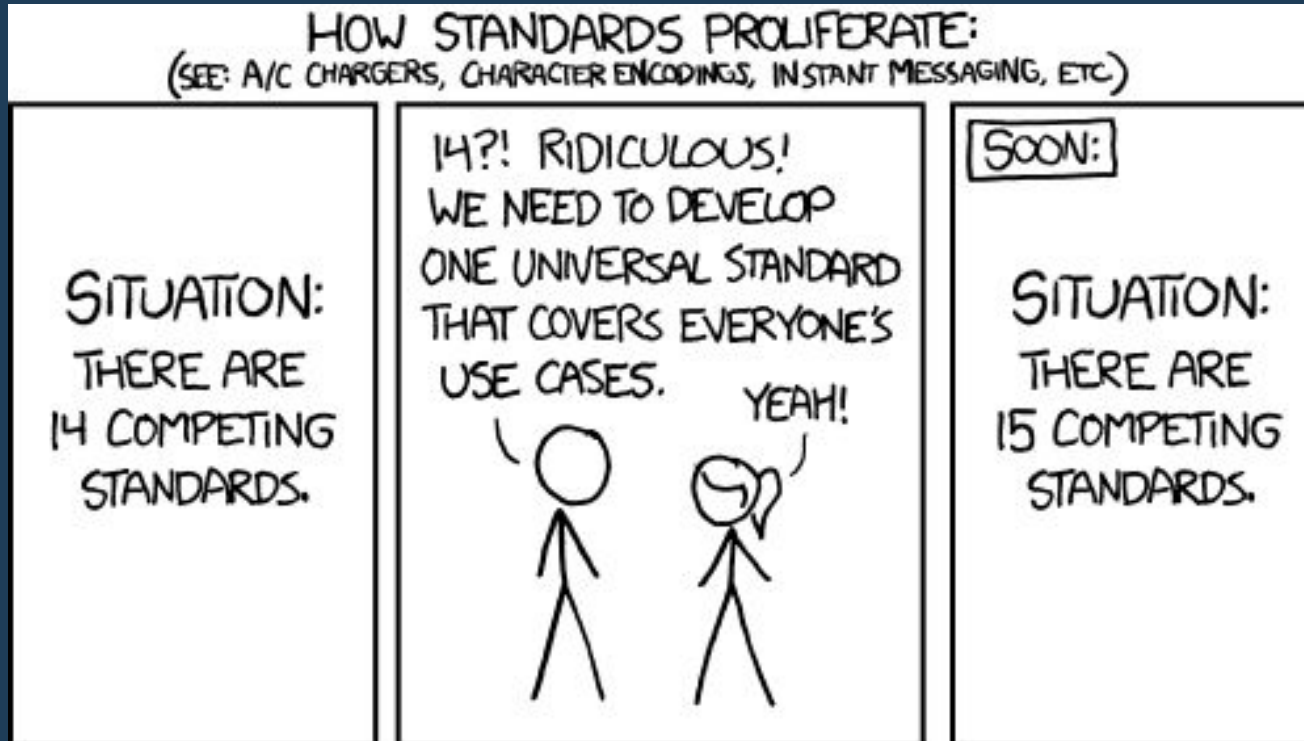
- Nagios
- Icinga
- Munin
- Zabbix

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
- pgwatch2 (client or server side)
- ...

# pgwatch2?



# Monitoring PostgreSQL made simple



pgwatch2

# Thanks!



Kaarel Moppel

The author of pgwatch2

# ● Main principles - why another tool?

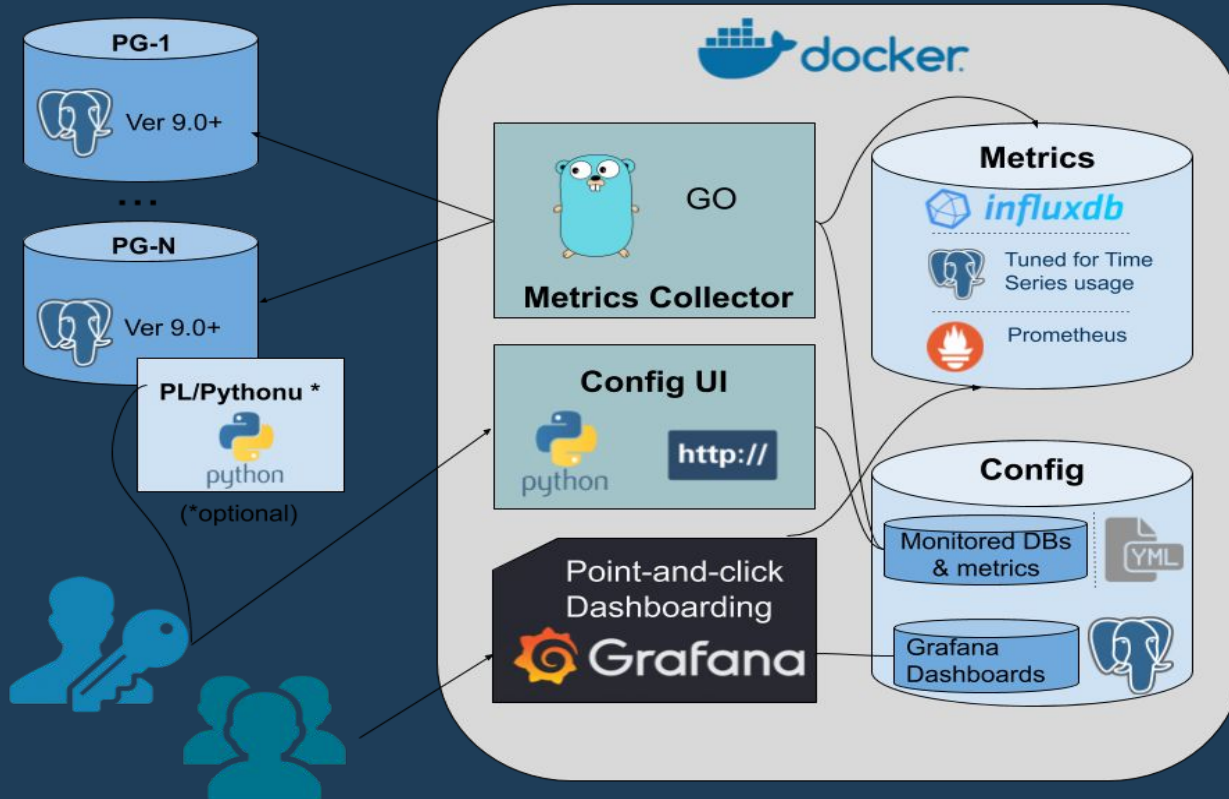
- 1-minute setup
  - Docker (custom setup also possible)
- User changeable visuals / dashboarding
- Non-invasive
  - No extensions or superuser needed for base functionality
- Easy extensibility, do minimal work needed
  - SQL metrics
- Simple alerting via Grafana possible



# Architecture components

- Metrics gathering daemon (Golang)
- Configuration database / YAML files / ENV vars
- Measurement 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

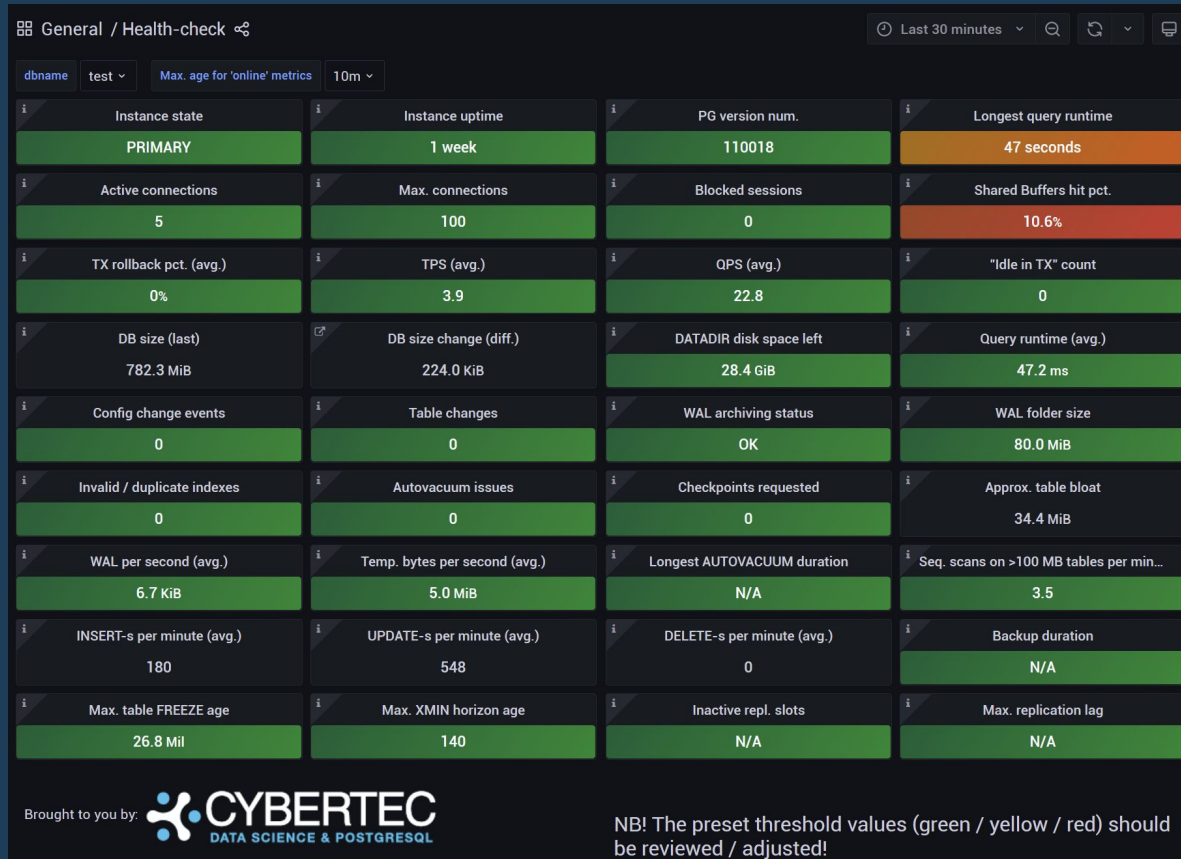
- Quite easy with Grafana, “point-and-click”
  - Most important alerting services covered
  - Email
  - Slack
  - PagerDuty
  - Web hooks
  - ...
- Based on graph panels only currently
- Use **pg\_timetable** scheduler to analyze and send alerts

# Getting started

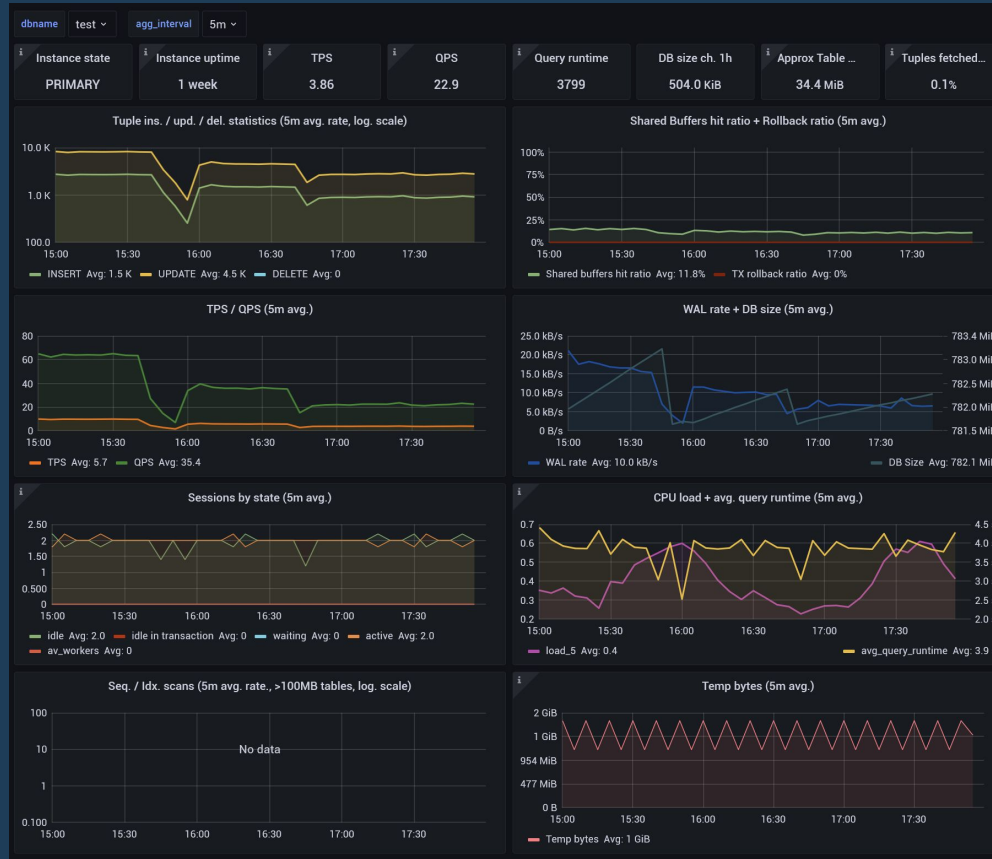
1. `docker run -d --restart=unless-stopped \`  
`-p 3000:3000 -p 8080:8080 \`  
`--name pw2 cybertec/pgwatch2-postgres`
2. Wait some seconds and open browser at `localhost:8080`
3. Insert your DB connection strings
4. Start viewing/editing dashboards in 5 min...

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**


**YO DAWG! HEARD YOU LIKE MONITORING**



**SO I'VE MADE YOU PGWATCH MAINTAINER.  
NOW YOU CAN MONITOR WHILE YOU'RE PROGRAMMING**



# Web UI: Before

PgWatch2 

DBs | Metrics | Logs ▾ | Log out

### Databases under monitoring

ID	Unique name	DB host	DB port	DB dbname	DB user	DB password	Is superuser?	SSL Mode	Preset config	Custom config	Statement timeout [seconds]	Last modified	Enabled?	
1	test	<input type="text" value="localhost"/>	<input type="text" value="5432"/>	<input type="text" value="pgwatch2"/>	<input type="text" value="pgwatch2"/>	<input type="password" value="..."/>	<input type="checkbox"/>	<input type="text" value="disable"/>	<input type="text" value="exhaustive"/>	<input type="text" value=""/>	<input type="text" value="5"/>	2017-09-19 19:54:05+00:00	<input checked="" type="checkbox"/>	<input type="button" value="Save"/> <input type="button" value="Delete"/>
	<input type="text" value="sales"/>	<input type="text" value="db.host1.com"/>	<input type="text" value="5432"/>	<input type="text" value="app1"/>	<input type="text" value="monitor"/>	<input type="password" value="*****"/>	<input type="checkbox"/>	<input type="text" value="disable"/>	<input type="text" value="exhaustive"/>	<input type="text" value=""/>	<input type="text" value="5"/>		<input checked="" type="checkbox"/>	<input type="button" value="New"/>

[show](#) | [copy](#)

### Active metrics listing

[backends](#) [ver: 9.9.6] [bgwriter](#) [ver: 9] [blocking\\_locks](#) [ver: 9.2] [buffercache\\_by\\_db](#) [ver: 9.2] [buffercache\\_by\\_type](#) [ver: 9.2] [change\\_events](#) [ver: 9] [configuration\\_hashes](#) [ver: 9] [cpu\\_load](#) [ver: 9] [db\\_stats](#) [ver: 9] [get\\_load\\_average](#) [ver: 9] [get\\_stat\\_statements](#) [ver: 9] [get\\_table\\_bloat\\_approx](#) [ver: 9.5] [index\\_hashes](#) [ver: 9] [index\\_stats](#) [ver: 9] [kpi](#) [ver: 9.9.6,10] [locks](#) [ver: 9] [locks\\_mode](#) [ver: 9] [pg\\_stat\\_database\\_conflicts](#) [ver: 9.2] [pg\\_stat\\_ssl](#) [ver: 9.5] [replication](#) [ver: 9.1,10] [sproc\\_hashes](#) [ver: 9] [sproc\\_stats](#) [ver: 9] [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] [table\\_io\\_stats](#) [ver: 9] [table\\_stats](#) [ver: 9] [wal](#) [ver: 9.2,10]




### InfluxDB metrics data cleanup

DB "Unique name" (NB! It could take up to 3min for background gatherers to stop so no point to click directly after removing a host from monitoring):


# Web UI: After

## 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	<input type="checkbox"/>	04.10.2023 17:58:07	<input checked="" type="checkbox"/>	 EDIT  DUPLICATE  DELETE

# Web UI: Before

PgWatch2  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)	<pre>{"wal": 60, "cpu_load": 60, "db_stats": 60}</pre>		2018-03-11 22:15:15+00	Save	Delete
exhaustive	almost all available metrics for a deeper performance understanding	<pre>{"wal": 60, "locks": 60, "backends": 60, "bgwriter": 60, "cpu_load": 60, "db_stats": 60, "locks_mode": 60, "index_stats": 60, "replication": 60, "sproc_stats": 60, "table_stats": 60}</pre>	test	2018-03-11 22:15:15+00	Save	
minimal	single "Key Performance Indicators" query for fast cluster/db overview	<pre>{"kpi": 60}</pre>		2018-03-11 22:15:15+00	Save	Delete
pgbouncer	per DB stats	<pre>{"pgbouncer_stats": 60}</pre>		2018-03-11 22:15:15+00	Save	Delete
standard	"basic" level + table, index, stat_statements stats	<pre>{"wal": 60, "cpu_load": 60, "db_stats": 60, "index_stats": 60, "sproc_stats": 60, "table_stats": 60, "stat_statements": 60}</pre>		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\_approxstattuple [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

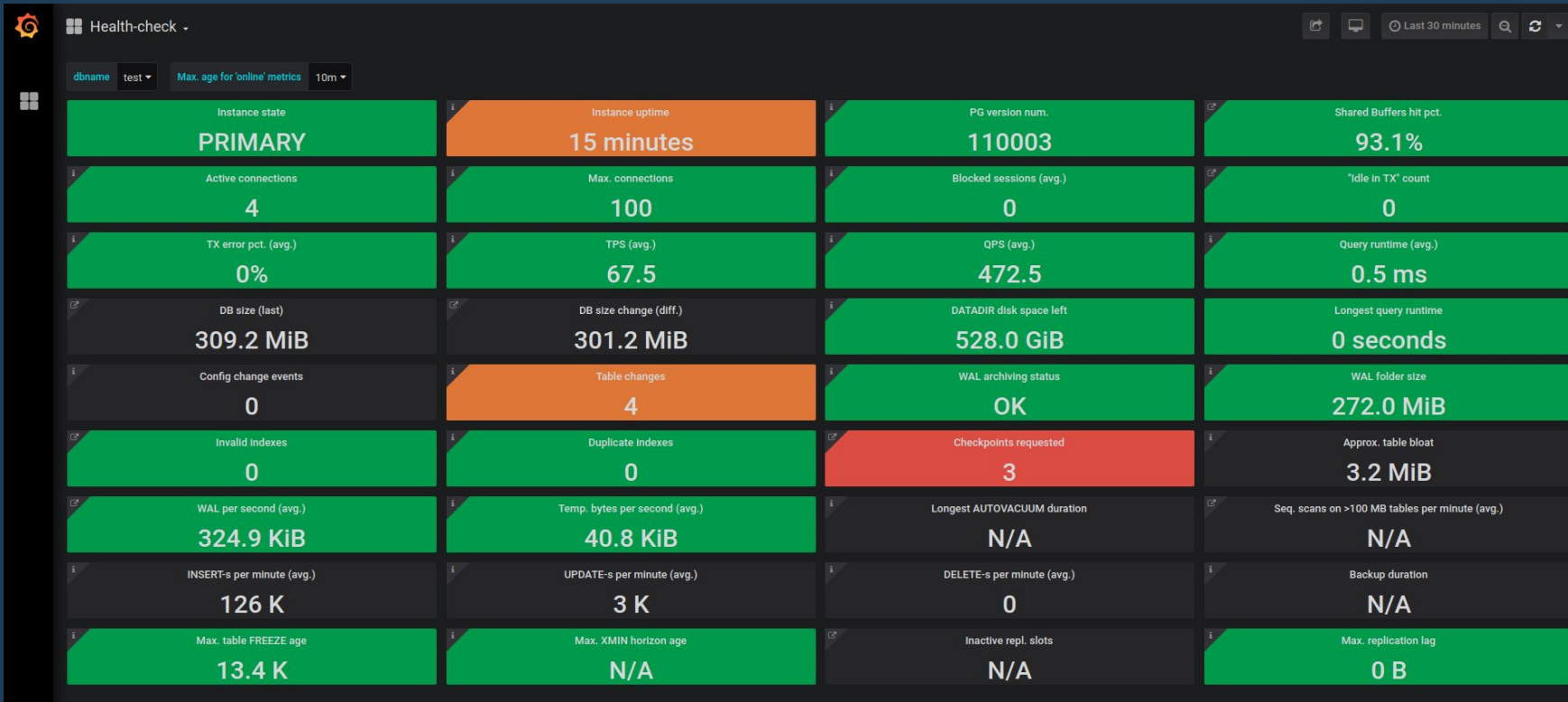
PRESET CONFIGS

## 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	<a href="#">EDIT</a> <a href="#">DELETE</a>
basic	only the most important metrics - WAL, DB-level statistic...	3 ↗	04.10.2023 17:56:35	<a href="#">EDIT</a> <a href="#">DELETE</a>
standard	"basic" level + table, index, stat_statements stats	9 ↗	04.10.2023 17:56:35	<a href="#">EDIT</a> <a href="#">DELETE</a>
pgbouncer	per DB stats	1 ↗	04.10.2023 17:56:35	<a href="#">EDIT</a> <a href="#">DELETE</a>
pgpool	pool global stats, 1 row per node ID	1 ↗	04.10.2023 17:56:35	<a href="#">EDIT</a> <a href="#">DELETE</a>
exhaustive	all important metrics for a deeper performance understand...	23 ↗	04.10.2023 17:56:35	<a href="#">EDIT</a> <a href="#">DELETE</a>

# Dashboards: Before



# Dashboards: After

🔧 Search or jump to... ctrl+k Sign in

Home > Dashboards > Health-check 🔗 Last 5 minutes 🔍 ↺ ↻ ↕

dbname test Max. age for 'online' metrics 5m

Instance state ⓘ	Instance uptime ⓘ	PG version num. ⓘ ↗	Longest query runtime ⓘ
PRIMARY	16 minutes	150004	2 seconds
Active connections ⓘ ↗	Max. connections ⓘ ↗	Blocked sessions ⓘ ↗	Shared Buffers hit pct. ⓘ ↗
15	100	8	92.8%
TX rollback pct. (avg.) ⓘ ↗	TPS (avg.) ⓘ ↗	QPS (avg.) ⓘ ↗	"Idle in TX" count ⓘ ↗
3.8%	287.5	2.0 K	2
DB size (last) ⓘ ↗	DB size change (diff.) ↗	⚠️ DATADIR disk space left ⓘ ↗	Query runtime (avg.) ⓘ ↗
9.6 MiB	0.0 B	0	1.6 ms
Config change events ⓘ ↗	Table changes ⓘ ↗	WAL archiving status ⓘ	WAL folder size ⓘ ↗
0	0	N/A	32.0 MiB
Invalid / duplicate indexes ⓘ ↗	Autovacuum issues ⓘ	Checkpoints requested ⓘ ↗	Approx. table bloat ⓘ
No data	0	0	208.0 KiB
WAL per second (avg.) ⓘ ↗	Temp. bytes per second (avg.) ⓘ ↗	Longest AUTOVACUUM duration ⓘ	Seq. scans on >100 MB tables per mi... ⓘ ↗
4.3 MiB	407.8 KiB	N/A	N/A

# 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 measurement storages

```
src | master 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 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 PAGE

[pashagolub.github.io](https://pashagolub.github.io)



## PERSONAL GITHUB

[www.github.com/pashagolub](https://www.github.com/pashagolub)



## CYBERTEC BLOG

[www.cybertec-postgresql.com/en/blog/](https://www.cybertec-postgresql.com/en/blog/)



## CYBERTEC GITHUB

[www.github.com/cybertec-postgresql](https://www.github.com/cybertec-postgresql)



# #StandWithUkraine

