CYBERTEC

The PostgreSQL Database Company

Professional PostgreSQL monitoring made easy

Kaarel Moppel - p2d2.cz 2019
Prague

Who?

Kaarel Moppel

Senior Database Consultant

km@cybertec.at

CYBERTEC
The PostgreSQL Database Company

PostgreSQL Database Services

CYBERTEC
The PostgreSQL Database Company

Client sectors

- University
- Automotive
- Government
- Industry
- Administration
- Finance
- Trade
- etc.

Agenda

- Different levels of database monitoring

Agenda

- Different levels of database monitoring
- PostgreSQL monitoring approaches

Agenda

- Different levels of database monitoring
- PostgreSQL monitoring approaches
- PostgreSQL monitoring tools

Agenda

- Different levels of database monitoring
- PostgreSQL monitoring approaches
- PostgreSQL monitoring tools
- pgwatch2
  - Main principles
  - Architecture
  - Features
  - Demo

Agenda

- Different levels of database monitoring
- PostgreSQL monitoring approaches
- PostgreSQL monitoring tools
- pgwatch2
  - Main principles
  - Architecture
  - Features
  - Demo
- Alerting / anomaly detection (if time)

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 themselves?

- 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
- Stats 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 (postgres_fdw, Graylog, ELK, ...)
  - pgBadger (JSON format)
  - Some cloud service (Loggly, Splunk, ...)

Logging configuration

Some settings to note

- log_destination (I recommend CSV format)
- log_statement = 'none' (default)
- log_min_duration_statement / log_duration
- log_min_messages / log_min_error_statement

```
krl@postgres=# SELECT count(*) FROM pg_settings
WHERE category LIKE 'Reporting and Logging%';
count
------
35
```

Stats 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_vacuum
- Cumulative views
  - Most pg_stat_* views
  - Long uptimes cause “lag” for problem detection
- Selective stats reset possible

Extensions

- Most notably **pg_stat_statments** (“top statements”)
- pgstattuple (bloat)
- pg_buffercache (what’s in the shared buffers)
- auto_explain (for jumping runtimes)
- …

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!)
- APM
  - A more high level concept, requires some trust / lock-in
  - AppDynamics, New Relic, DataDog, ...

PostgreSQL Monitoring Tools

- Ad hoc monitoring / troubleshooting
- Continuous monitoring frameworks

PostgreSQL Monitoring Tools

No shortage of tools!

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

Ad hoc monitoring / troubleshooting

Open Source “ad-hoc” tools

- pg_activity
- pgcenter
- pghero
- pgAdmin4
- ....

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)
- PgCluu (server side, with “sar” system info)
- pgwatch2 (client or server side)
- ...

pgwatch2

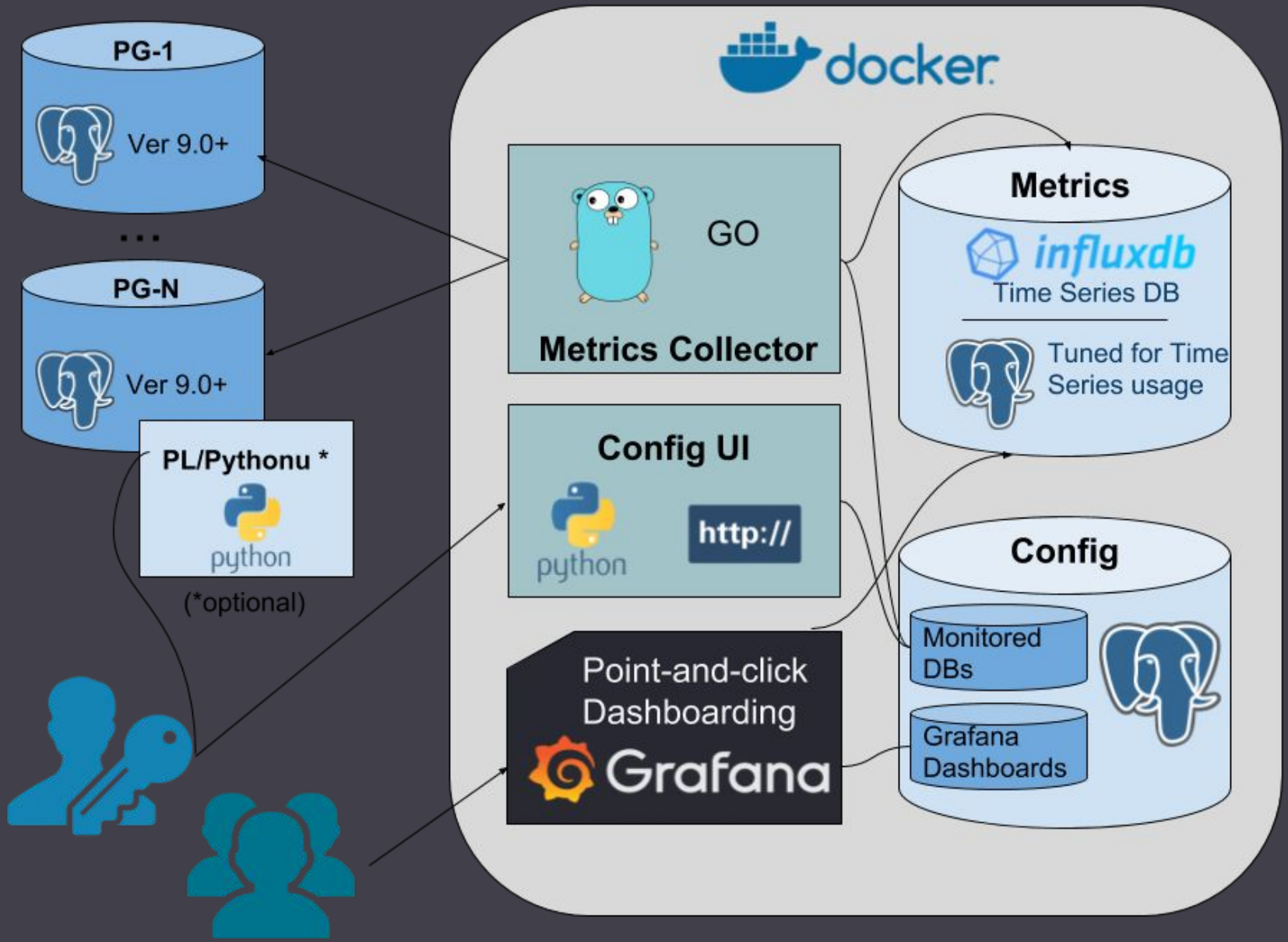
- Main principles
- Architecture
- Features
- Demo

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
- Easy alerting when needed

Architecture components

- Metrics gathering daemon
  - Go
- Config database / YAML files
- Metrics storage layer
  - PostgreSQL
  - InfluxDB
  - Graphite
- Optional simple Web UI for administration
- Easy dashboarding with data discovery and optional alerting
  - Grafana


PG-1
Ver 9.0+
...
PG-N
Ver 9.0+
PL/Pythonu *
python
(*optional)
docker
GO
Metrics Collector
Config UI
python
http://
Point-and-click
Dashboarding
Grafana
Metrics
influxdb
Time Series DB
Tuned for Time
Series usage
Config
Monitored
DBs
Grafana
Dashboards

Features

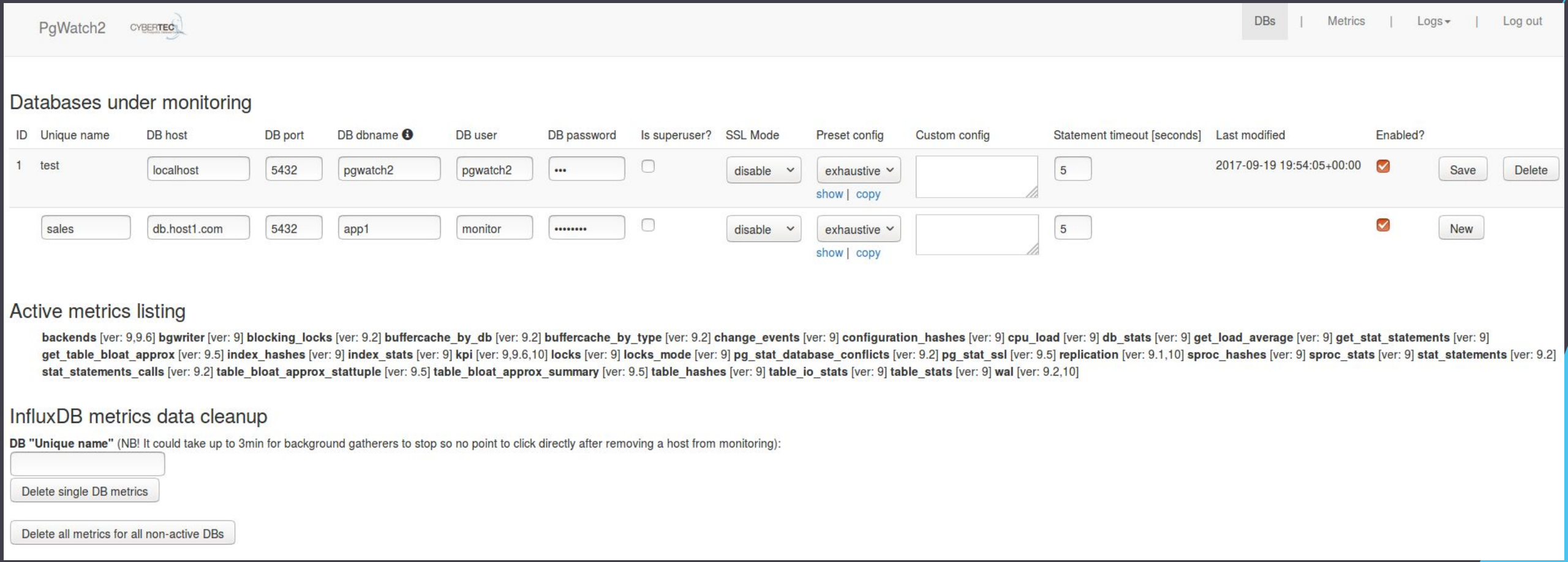
- “Ready to go”
  - Default metrics cover almost all pg_stat* views
  - Pre-configured dashboards for almost all metrics
- Supports Postgres 9.0+ out of the box
  - Older versions also possible
- Configurable security - admin login, SSL
- Reuse of existing Postgres, Grafana, InfluxDB installations possible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

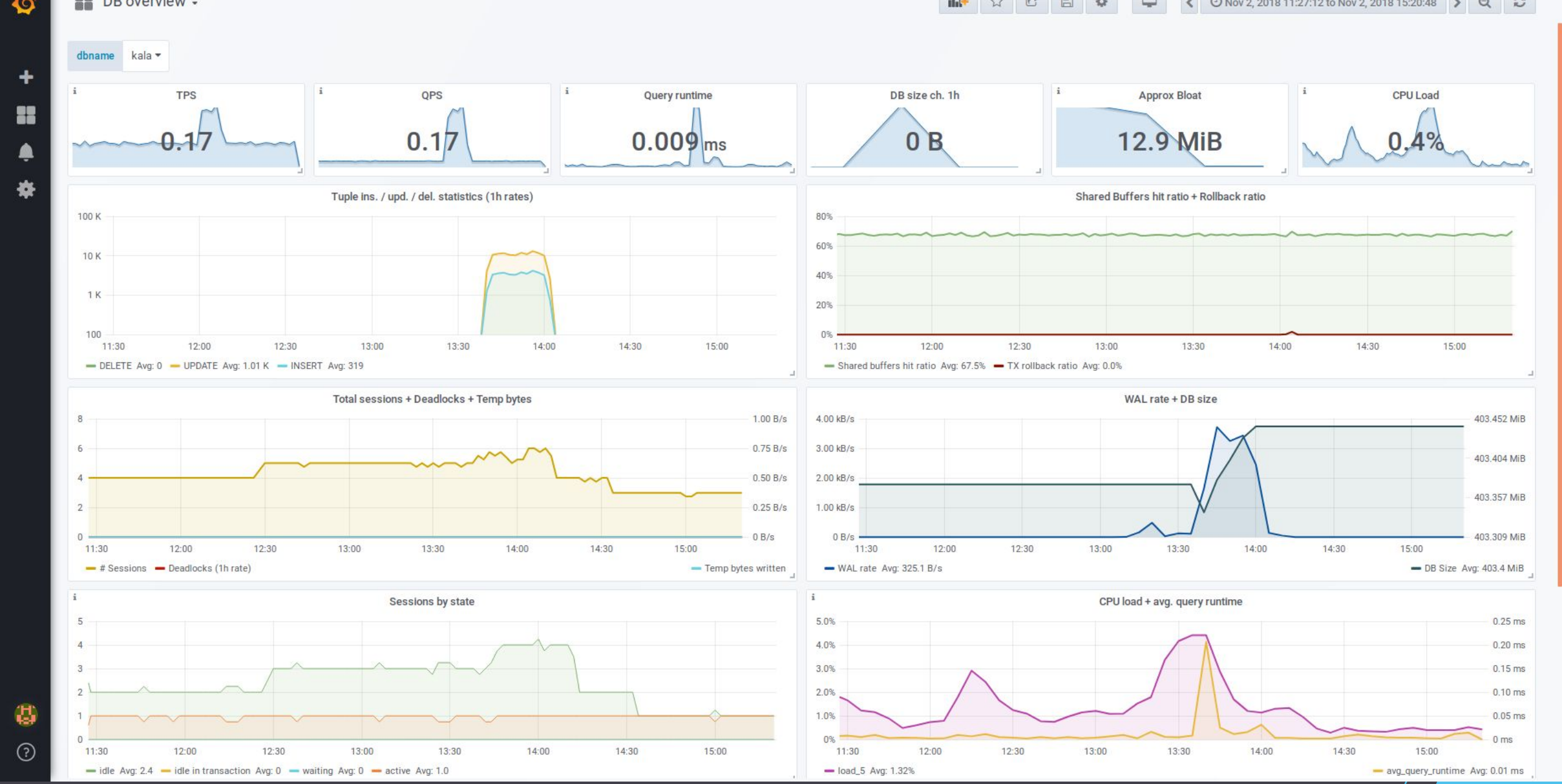
Features

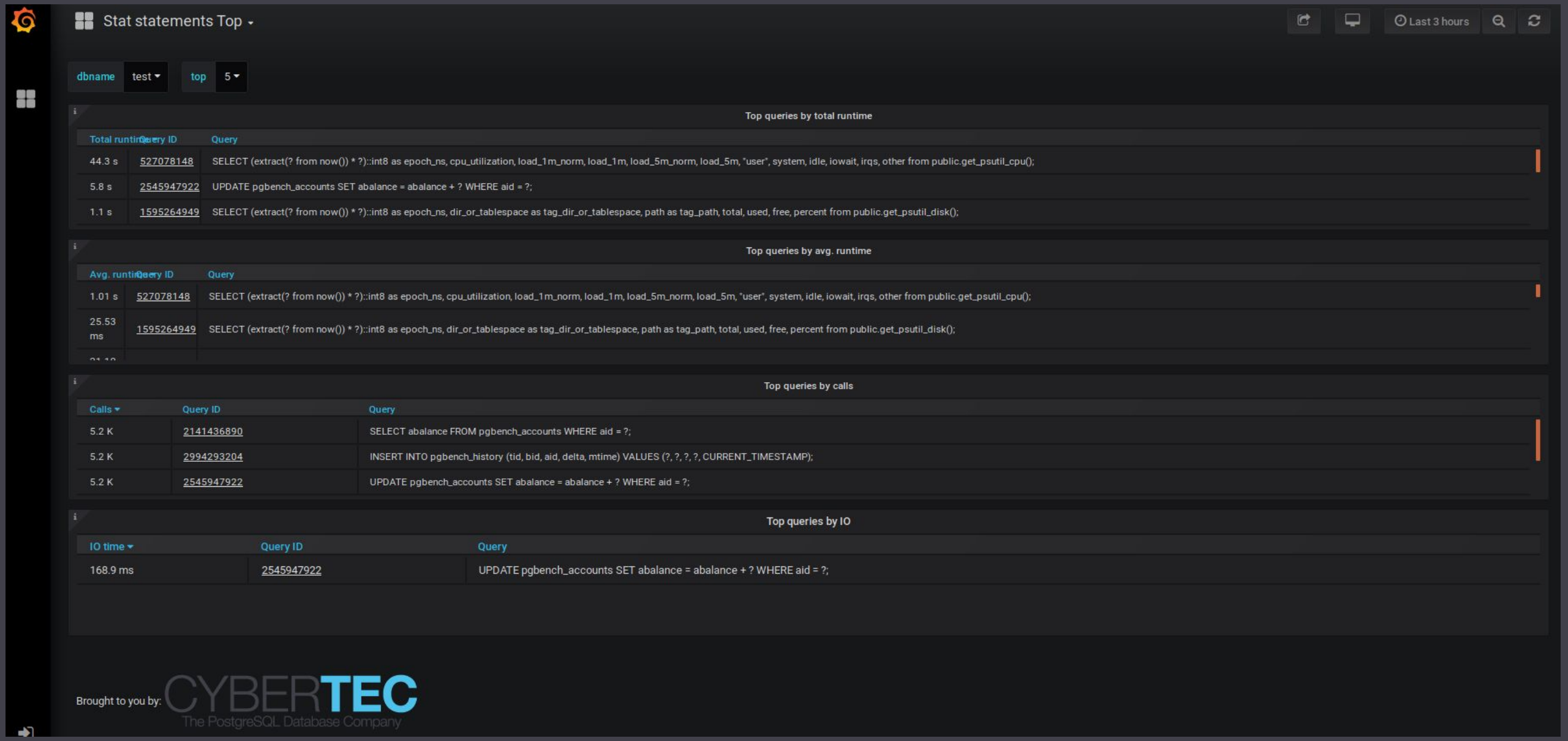
- Configured per DB, with optional auto-discovery of PG clusters
- Change detection
  - Added/changed/deleted table/index/sproc/config events
- AWS RDS CloudWatch metrics support
- PgBouncer metrics support
- Extensible
  - Custom metrics via SQL, i.e. usable also for business layer!
  - Grafana has plugins

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 5min...


PgWatch2
CYBERTEC
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 localhost 5432 pgwatch2 pgwatch2 ••• disable exhaustive show | copy 5 2017-09-19 19:54:05+00:00 Save Delete
sales db.host1.com 5432 app1 monitor •••••••• disable exhaustive show | copy 5 New
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):
Delete single DB metrics
Delete all metrics for all non-active DBs




Stat statements Top
Last 3 hours
dbname test
top 5
Top queries by total runtime
Total runtime Query ID Query
44.3 s 527078148 SELECT (extract(? from now()) * ?)::int8 as epoch_ns, cpu_utilization, load_1m_norm, load_1m, load_5m_norm, load_5m, "user", system, idle, iowait, irqs, other from public.get_psutil_cpu();
5.8 s 2545947922 UPDATE pgbench_accounts SET abalance = abalance + ? WHERE aid = ?;
1.1 s 1595264949 SELECT (extract(? from now()) * ?)::int8 as epoch_ns, dir_or_tablespace as tag_dir_or_tablespace, path as tag_path, total, used, free, percent from public.get_psutil_disk();
Top queries by avg. runtime
Avg. runtime Query ID Query
1.01 s 527078148 SELECT (extract(? from now()) * ?)::int8 as epoch_ns, cpu_utilization, load_1m_norm, load_1m, load_5m_norm, load_5m, "user", system, idle, iowait, irqs, other from public.get_psutil_cpu();
25.53 ms 1595264949 SELECT (extract(? from now()) * ?)::int8 as epoch_ns, dir_or_tablespace as tag_dir_or_tablespace, path as tag_path, total, used, free, percent from public.get_psutil_disk();
Top queries by calls
Calls Query ID Query
5.2 K 2141436890 SELECT abalance FROM pgbench_accounts WHERE aid = ?;
5.2 K 2994293204 INSERT INTO pgbench_history (tid, bid, aid, delta, mtime) VALUES (?, ?, ?, ?, CURRENT_TIMESTAMP);
5.2 K 2545947922 UPDATE pgbench_accounts SET abalance = abalance + ? WHERE aid = ?;
Top queries by IO
IO time Query ID Query
168.9 ms 2545947922 UPDATE pgbench_accounts SET abalance = abalance + ? WHERE aid = ?;
Brought to you by:
CYBERTEC
The PostgreSQL Database Company

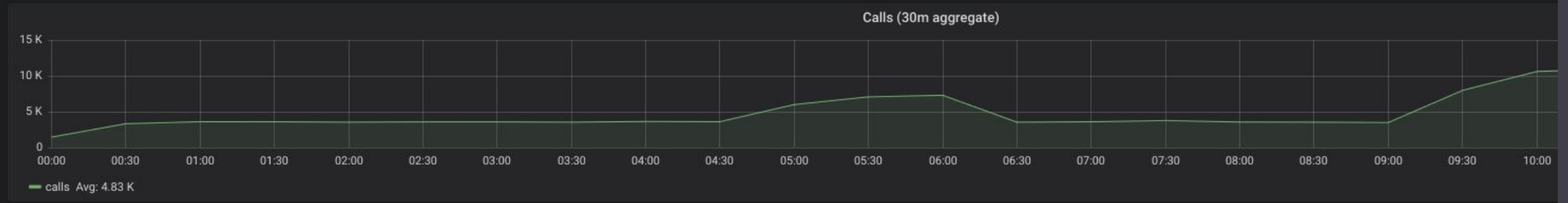
Single query details

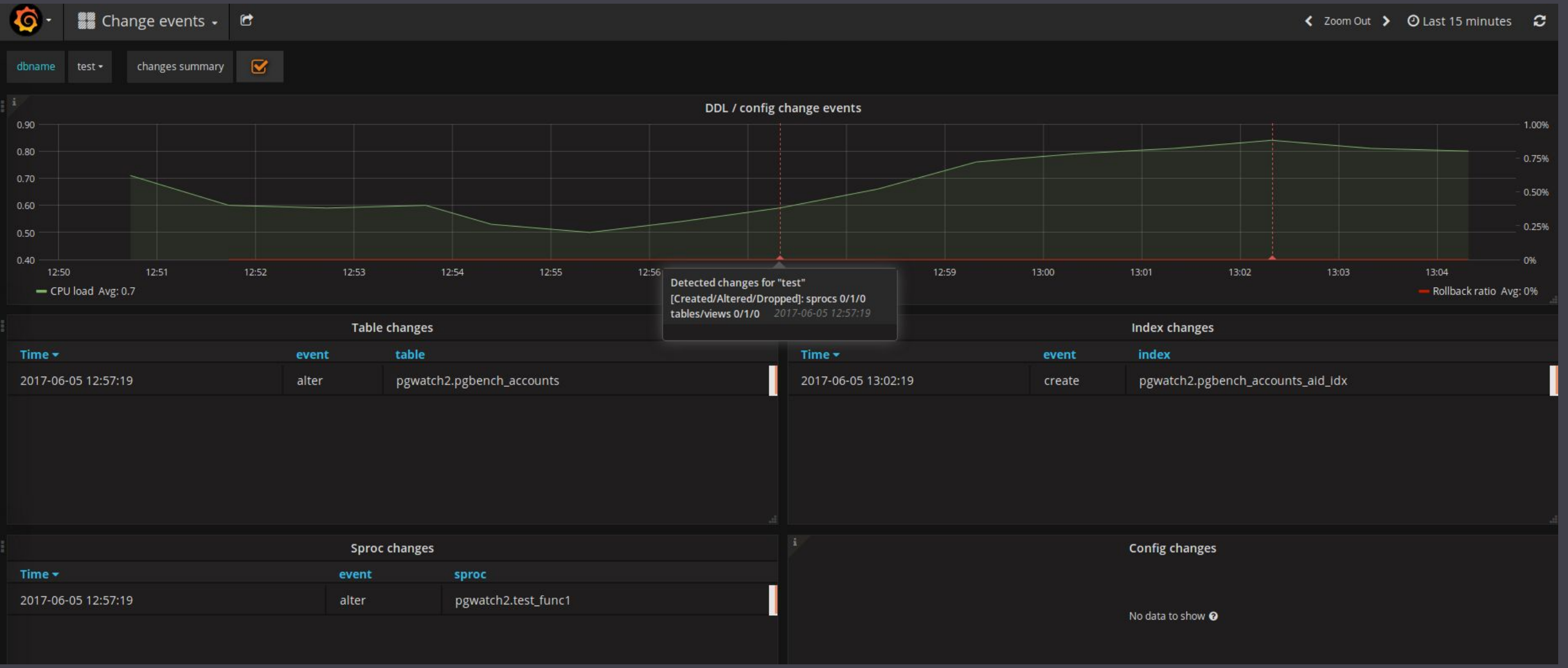
dbname: test | queryid: 1754591603769365576 | agg_interval: 30m

SQL

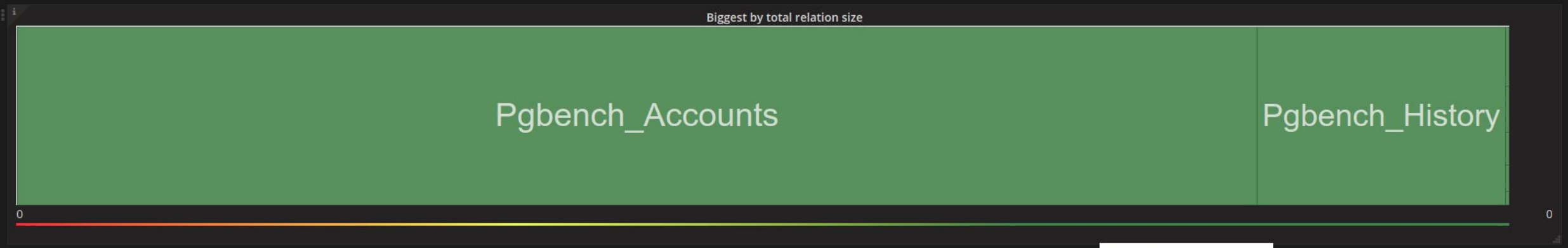
Value

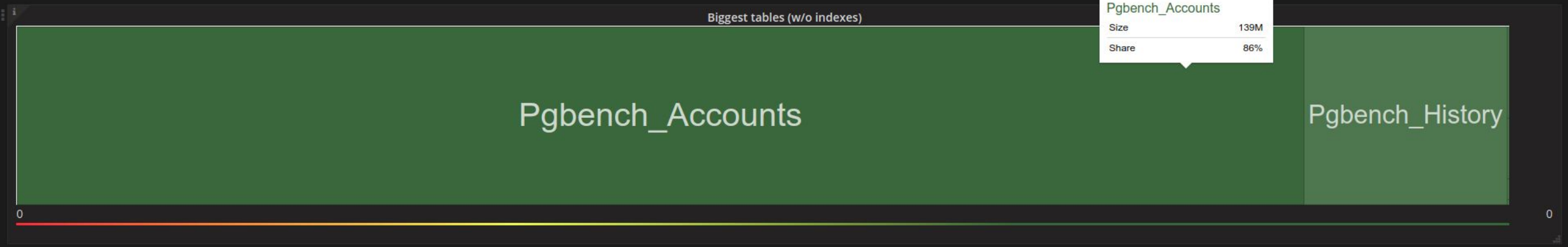
```
UPDATE pgbench_accounts SET abalance = abalance + $1 WHERE aid = $2
```

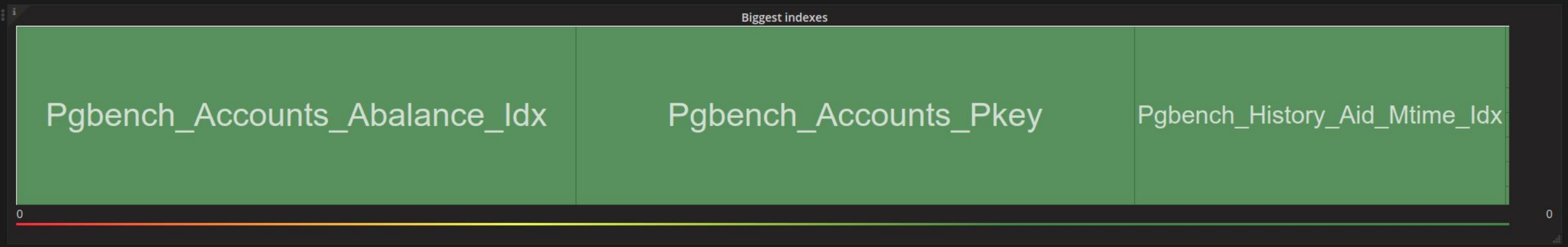



Change events
Zoom Out
Last 15 minutes
dbname
test
changes summary
DDL / config change events
0.90
0.80
0.70
0.60
0.50
0.40
1.00%
0.75%
0.50%
0.25%
0%
12:50
12:51
12:52
12:53
12:54
12:55
12:56
12:59
13:00
13:01
13:02
13:03
13:04
CPU load Avg: 0.7
Rollback ratio Avg: 0%
Detected changes for "test"
[Created/Altered/Dropped]: sprocs 0/1/0
tables/views 0/1/0
2017-06-05 12:57:19
Table changes
Time
event
table
2017-06-05 12:57:19
alter
pgwatch2.pgbench_accounts
Index changes
Time
event
index
2017-06-05 13:02:19
create
pgwatch2.pgbench_accounts_aid_idx
Sproc changes
Time
event
sproc
2017-06-05 12:57:19
alter
pgwatch2.test_func1
Config changes
No data to show

dbname kala_postgres







Brought to you by: CYBERTEC The PostgreSQL Database Company


Health-check2
Last 15 minutes
dbname
kala
Instance state
PRIMARY
Instance uptime
2 day
Backends
4
Blocked backends
0
Non-active repl. slots
0
TX error pct (avg.)
1.9%
"Idle in TX" count
0
CPU load (avg.)
0.9
TPS (avg.)
194.1
QPS (avg.)
1.2 K
Query runtime (avg.)
30.9 ms
Longest query runtime
0 seconds
Config change events
0
Table changes
0
Index changes
0
Sproc changes
0
DB size change
0 B
DB size
422.2 MiB
Tables size
129.5 MiB
Indexes size
43.0 MiB
Invalid indexes
1
Duplicate indexes
2
Approx. bloat
26.3 MiB
Avg. time from last autovacuum (tbls > 100MB)
19 hours
WAL per second (avg.)
242.9 B
Temp. bytes per second (avg.)
0 B
Seq. scans on >100 MB tables per minute (avg.)
0
Repl. slot max. XMIN horizon
no value
INSERT-s per minute (avg.)
191 K
UPDATE-s per minute (avg.)
7 K
DELETE-s per minute (avg.)
32
Backend max. XMIN horizon (in TX)
0
Brought to you by:
CYBERTEC
The PostgreSQL Database Company

Beyond basics

- Alerting
- Anomaly detection

Alerting

- Quite easy with Grafana, “point-and-click”
- Most important alerting services covered
  - Email
  - Slack
  - PagerDuty
  - Web hooks
  - …
- Based on graph panels only currently :/

Anomaly detection

Kapacitor - part of the InfluxData's TICK stack

- Harder to get going but powerful
- Extensive math/string processing support
- Statistical data mangling
- UDF-s
- Alert topics - pub/sub
- Stream caching (e.g. last 10min moving average)
- Stream redirection - store transformed data back into InfluxDB

Kapacitor sample -simplified

```
|from()
    .measurement('cpu')
|groupBy('service', 'datacenter')
|window()
    .period(10m)
|percentile('load_1min', 95.0)
|eval(lambda: sigma("percentile"))
    .as('sigma')
|alert()
    .crit(lambda: "sigma" > 3.0)
```

Improvement ideas?

User input very much expected

github.com/cybertec-postgresql/pgwatch2

kthxbye

Don't be a stranger:

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