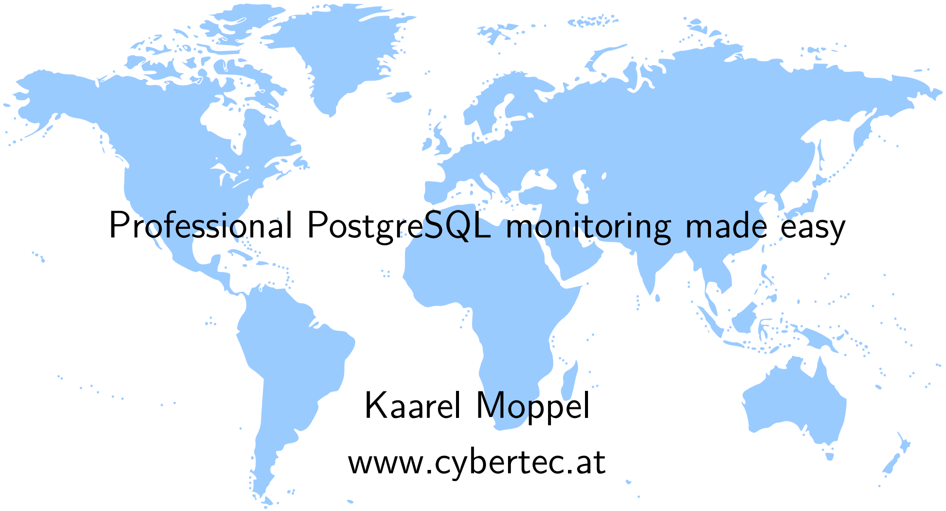


# Professional PostgreSQL monitoring made easy

Kaarel Moppel
www.cybertec.at

## Why to monitor

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

## Different levels of Database monitoring

## 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

### Operating System / Process monitoring

- DIY involving typically a TSDB and some graphing/alerting engine
  - Graphite, RRDtool, OpenTSDB
- Nagios / Icinga / ...
- Something provided out-of-the-box by cloud providers usually
  - Included in VM software like VMware vSphere etc

## System monitoring

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

- "Just in case" storing of logs for possible ad hoc needs. Moving logs to a central place makes sense.
  - Cron + rsync
  - (r)syslog(-ng), redislog
- Active parsing
  - DIY (Graylog, ELK, ...)
  - pgBadger
  - Some cloud service (Loggly, Splunk, ...)

## Logging configuration

### Settings to note

- log_destination (CSV format recommended)
- log_statement
- log_min_duration_statement
- log_min_messages / log_min_error_statement

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

## Stats Collector

- Not all track_* parameters enabled by default
- Dynamic views
  - pg_stat_activity, pg_stat_replication/pg_stat_wal_receiver, pg_stat_ssl
- Cumulative views
  - Most pg_stat_* views
  - Long uptimes cause “lag” for problem detection
- Selective stats reset possible

## Stats Collector

- pg_stat_database
- pg_stat(io)_user_tables
- pg_stat(io)_user_indexes
- pg_stat_user_functions
- ... (see “\dv pg_stat*“, 31 views for PG 10)

## Extensions

- Most notably pg_stat_statments
- pgstattuple
- pg_buffercache
- auto_explain
- ...

## Locks

### Separate from Stats Collector

- pg_locks
- pg_stat_activity
  - wait_event_type/wait_event (9.6+, very detailed info)
- log_lock_waits (uses deadlock_timeout)

## Autovacuum

- For busy databases monitor also Autovacuum
  - pg_stat_progress_vacuum
  - pg_stat_activity WHERE query LIKE ‘autovacuum%’
- If Autovacuum is lagging behind you’ll end up with unecessary bloat
  - Tip: idle_in_transaction_session_timeout / old_snapshot_threshold

## Real life

### Mixed approach for bigger setups

- 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

## No shortage of tools

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

## Approaches

- Ad hoc
- Continuous monitoring frameworks
  - Cloud / SaaS
  - DIY

## Ad hoc monitoring / troubleshooting

## Open Source Ad hoc tools

- pgAdmin4
- pg_activity
- pg_view
- pgcenter
- pghero

## Continuous monitoring frameworks

## Commercial

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

Most also have some free version with basic features

## Open Source

### Genral Monitoring Frameworks

- Nagios
- Icinga
- Munin
- Zabbix

*check_postgres* script

## Open Source

### Postgres specific

- pghero
- PoWa (server side, quite advanced - pg_qualstats, pg_stat_kcache)
- PgObserver (client side + ad hoc)
- pgwatch2 (client side)
- ...

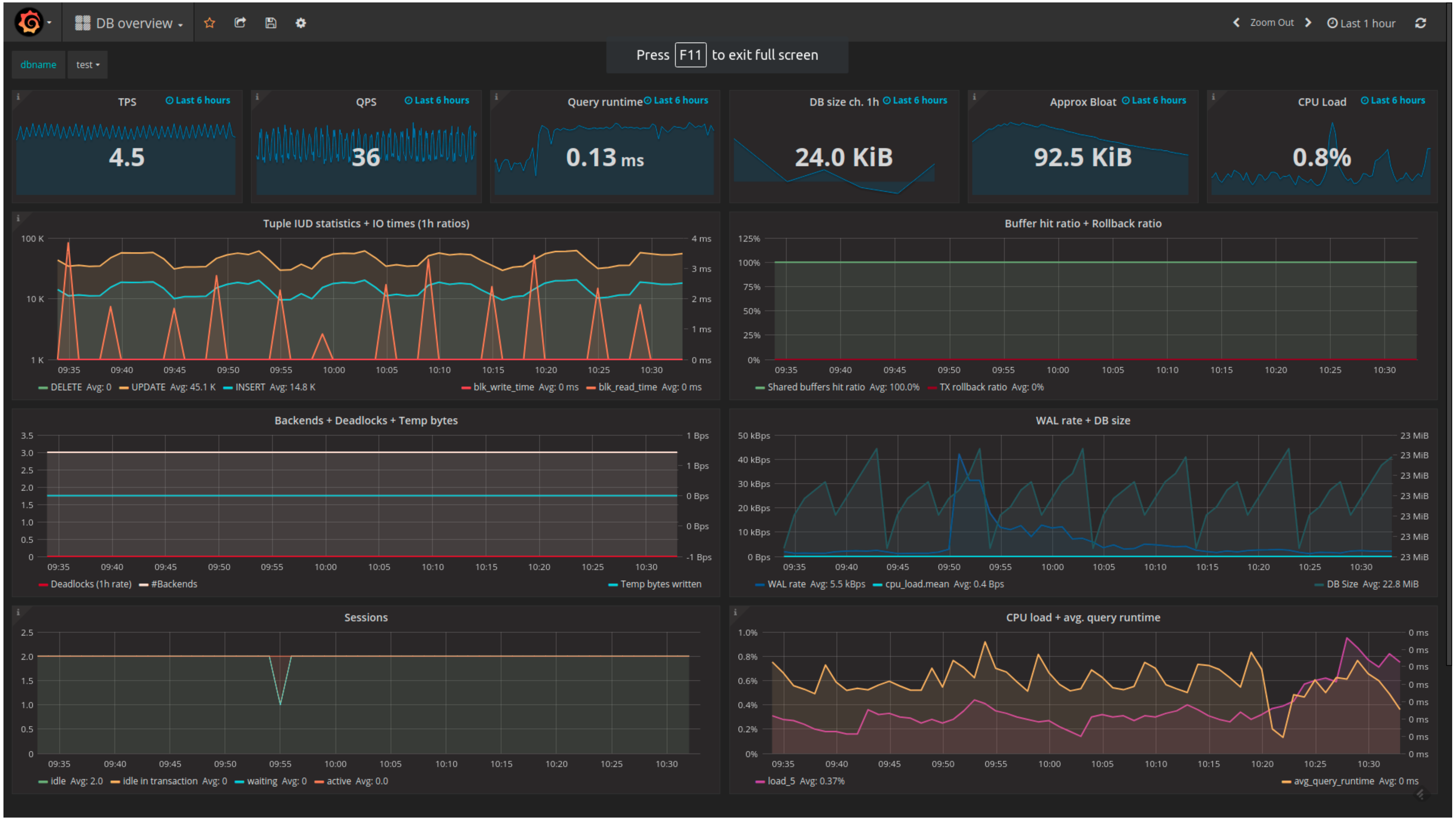
## pgwatch2

## Main principles - why another tool?

- 1-minute setup
  - Docker
- Custom visuals / Dashboarding
- Non-invasive
  - No extensions for main functionality
- Easy extensibility
  - SQL metrics
- Do minimal work needed, use existing SW

## Architecture components

- Metrics gathering daemon
  - Go
- Config database
  - Postgres
- Metrics storage layer
  - InfluxDB (Graphite possible)
- Web UI for administration
  - Python / Bootstrap
- Easy dashboarding with data discovery
  - Grafana


DB overview
Zoom Out
Last 1 hour
Press F11 to exit full screen
dbname
test
TPS
Last 6 hours
4.5
QPS
Last 6 hours
36
Query runtime
Last 6 hours
0.13 ms
DB size ch. 1h
Last 6 hours
24.0 KiB
Approx Bloat
Last 6 hours
92.5 KiB
CPU Load
Last 6 hours
0.8%
Tuple IUD statistics + IO times (1h ratios)
DELETE Avg: 0
UPDATE Avg: 45.1 K
INSERT Avg: 14.8 K
blk_write_time Avg: 0 ms
blk_read_time Avg: 0 ms
Buffer hit ratio + Rollback ratio
Shared buffers hit ratio Avg: 100.0%
TX rollback ratio Avg: 0%
Backends + Deadlocks + Temp bytes
Deadlocks (1h rate)
#Backends
Temp bytes written
WAL rate + DB size
WAL rate Avg: 5.5 kBps
cpu_load.mean Avg: 0.4 Bps
DB Size Avg: 22.8 MiB
Sessions
idle Avg: 2.0
idle in transaction Avg: 0
waiting Avg: 0
active Avg: 0.0
CPU load + avg. query runtime
load_5 Avg: 0.37%
avg_query_runtime Avg: 0 ms]

## Features

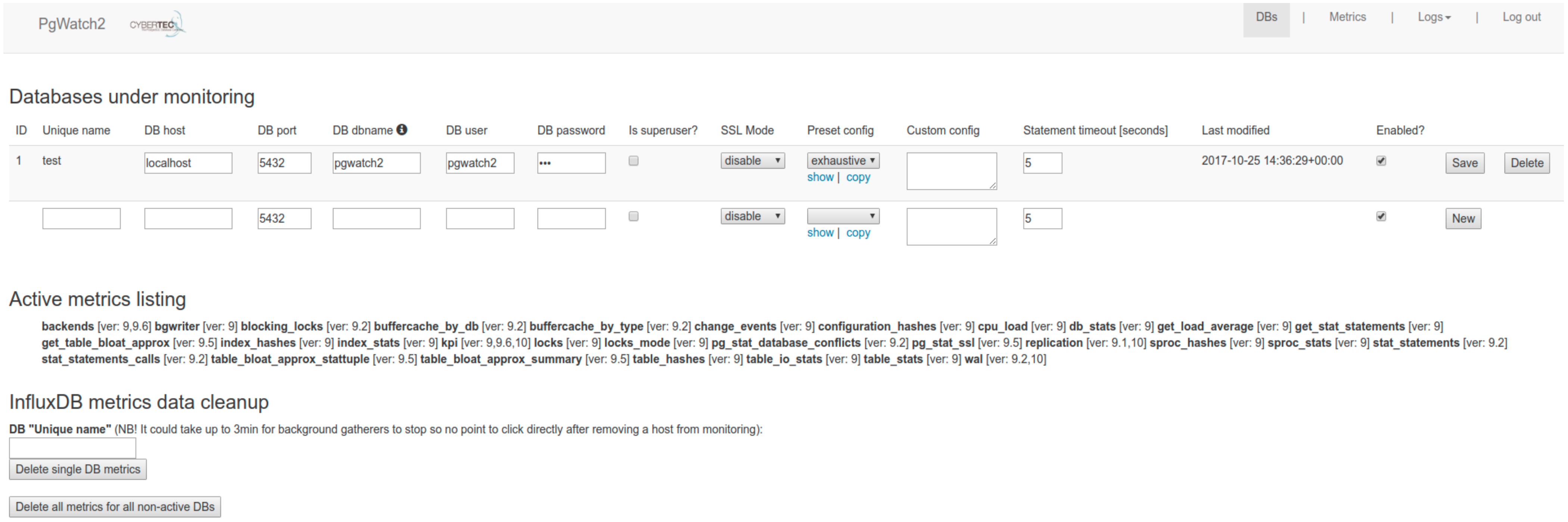
- Ready to go
  - Default cover almost all pg_stat* views
  - Test database (possible to disable) as playground
- Supports Postgres 9.0+ (older versions also possible)
- Security (SSL)
- Custom metrics via SQL, also for business layer!
- Reuse of existing components (Postgres, Grafana, InfluxDB) possible
- Can be integrated with your “cloud”

## Features

- Component logs available via Web UI for troubleshooting
- Possible to monitor all databases of a cluster automatically
- Change detection
  - Added/changed/deleted table/index/sproc/config events
- Alerting easily possible
  - Grafana
  - Kapacitor ("K" from InfluxData's TICK stack)
- Extensible - Grafana has plugins!

## Getting started

1. `docker run -d -p 3000:3000 -p 8080:8080 \`
   `--name pw2 cybertec/pgwatch2`
2. Wait some seconds and open browser at localhost:8080
3. Insert your DB connection strings and wait some minutes
4. Start Dashboarding!



## Databases under monitoring

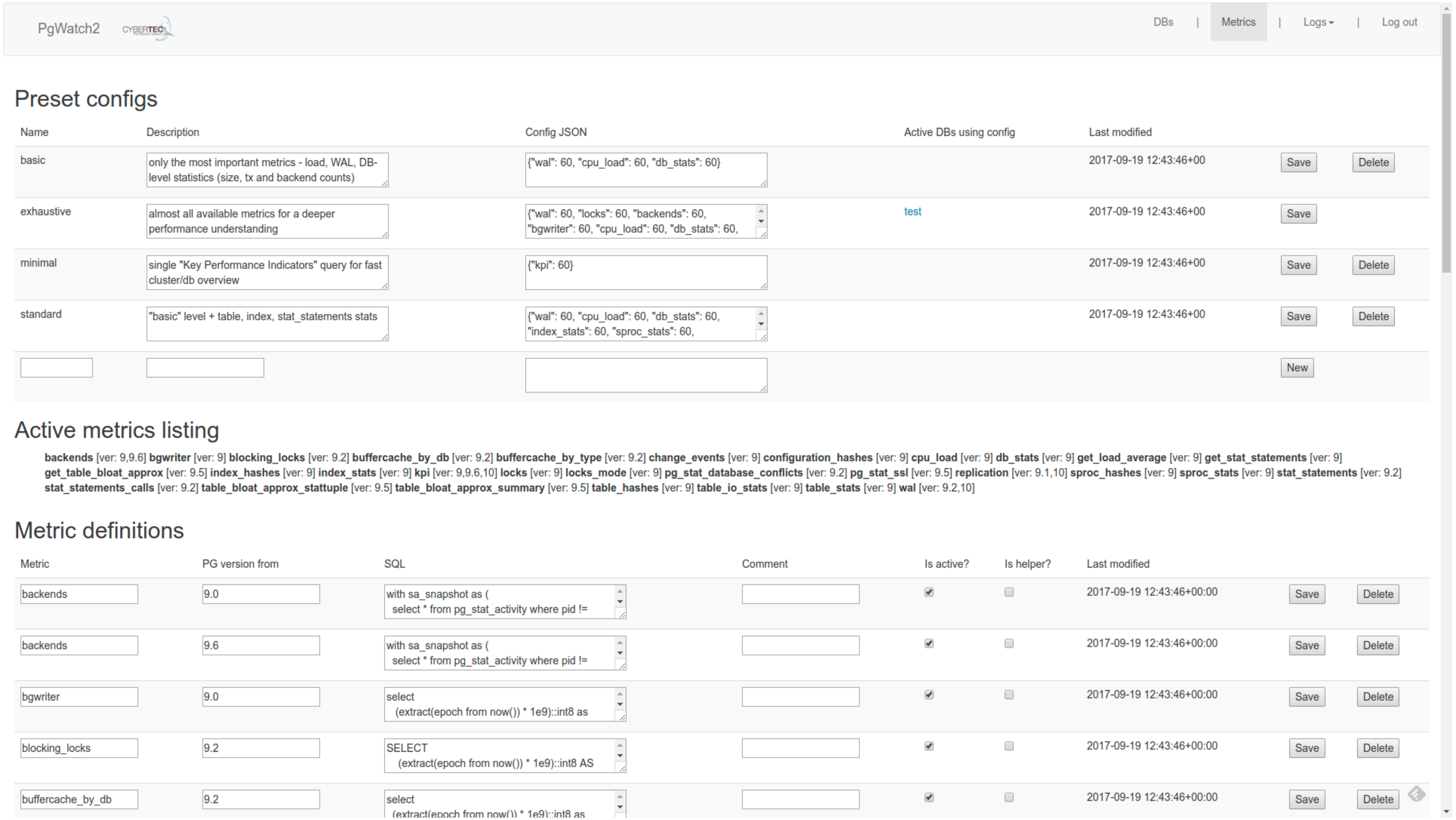
| ID | Unique name | DB host | DB port | DB dbname | DB user | DB password | Is superuser? | SSL Mode | Preset config | Custom config | Statement timeout [seconds] | Last modified | Enabled? | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | test | localhost | 5432 | pgwatch2 | pgwatch2 | ••• | ☐ | disable | exhaustive show \| copy | | 5 | 2017-10-25 14:36:29+00:00 | ✓ | Save | Delete |
| | | | 5432 | | | | ☐ | disable | 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):



## Preset configs

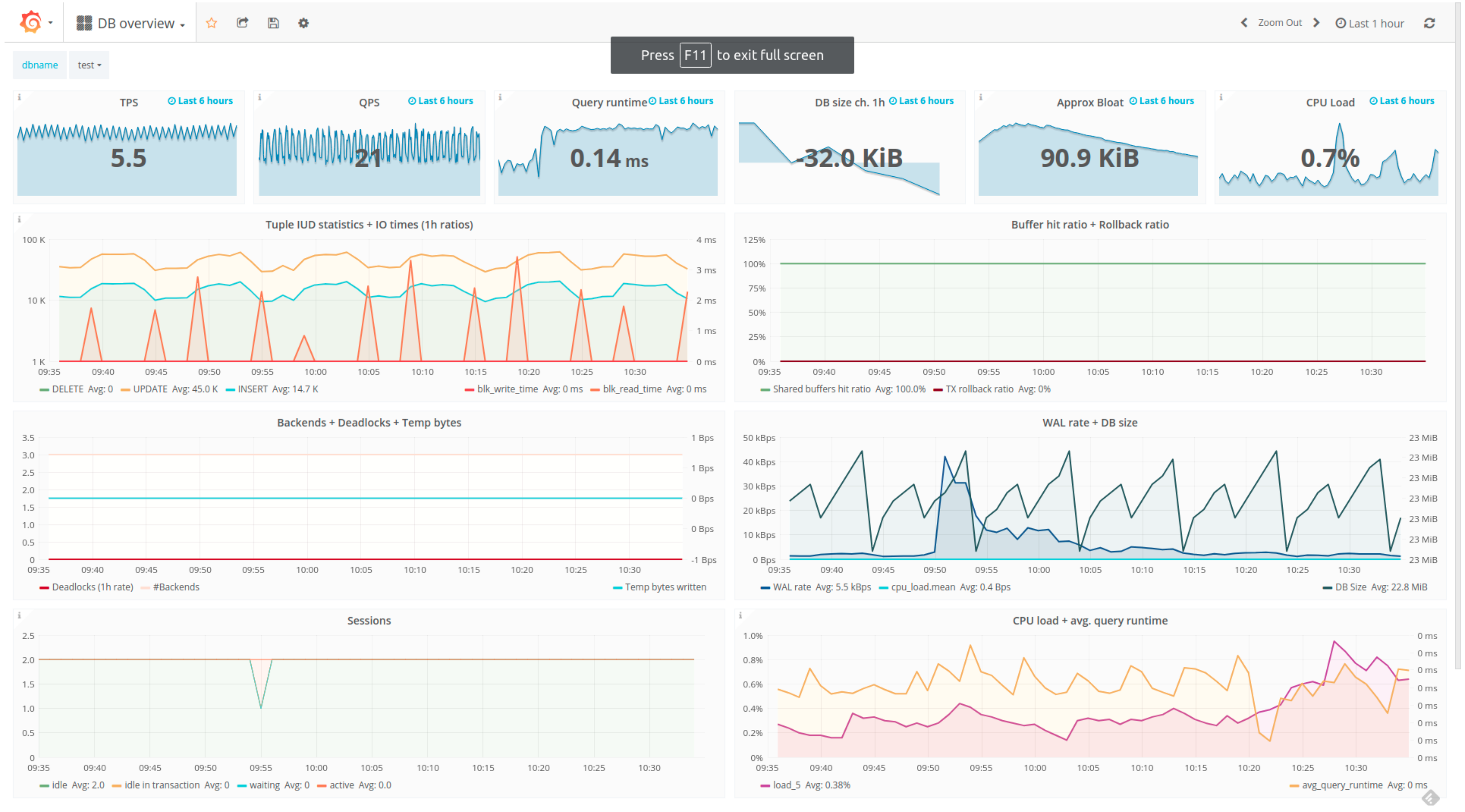
| Name | Description | Config JSON | Active DBs using config | Last modified | | |
|---|---|---|---|---|---|---|
| basic | only the most important metrics - load, WAL, DB-level statistics (size, tx and backend counts) | {"wal": 60, "cpu_load": 60, "db_stats": 60} | | 2017-09-19 12:43:46+00 | Save | Delete |
| exhaustive | almost all available metrics for a deeper performance understanding | {"wal": 60, "locks": 60, "backends": 60, "bgwriter": 60, "cpu_load": 60, "db_stats": 60, | test | 2017-09-19 12:43:46+00 | Save | |
| minimal | single "Key Performance Indicators" query for fast cluster/db overview | {"kpi": 60} | | 2017-09-19 12:43:46+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, | | 2017-09-19 12:43:46+00 | Save | Delete |
| | | | | | New | |

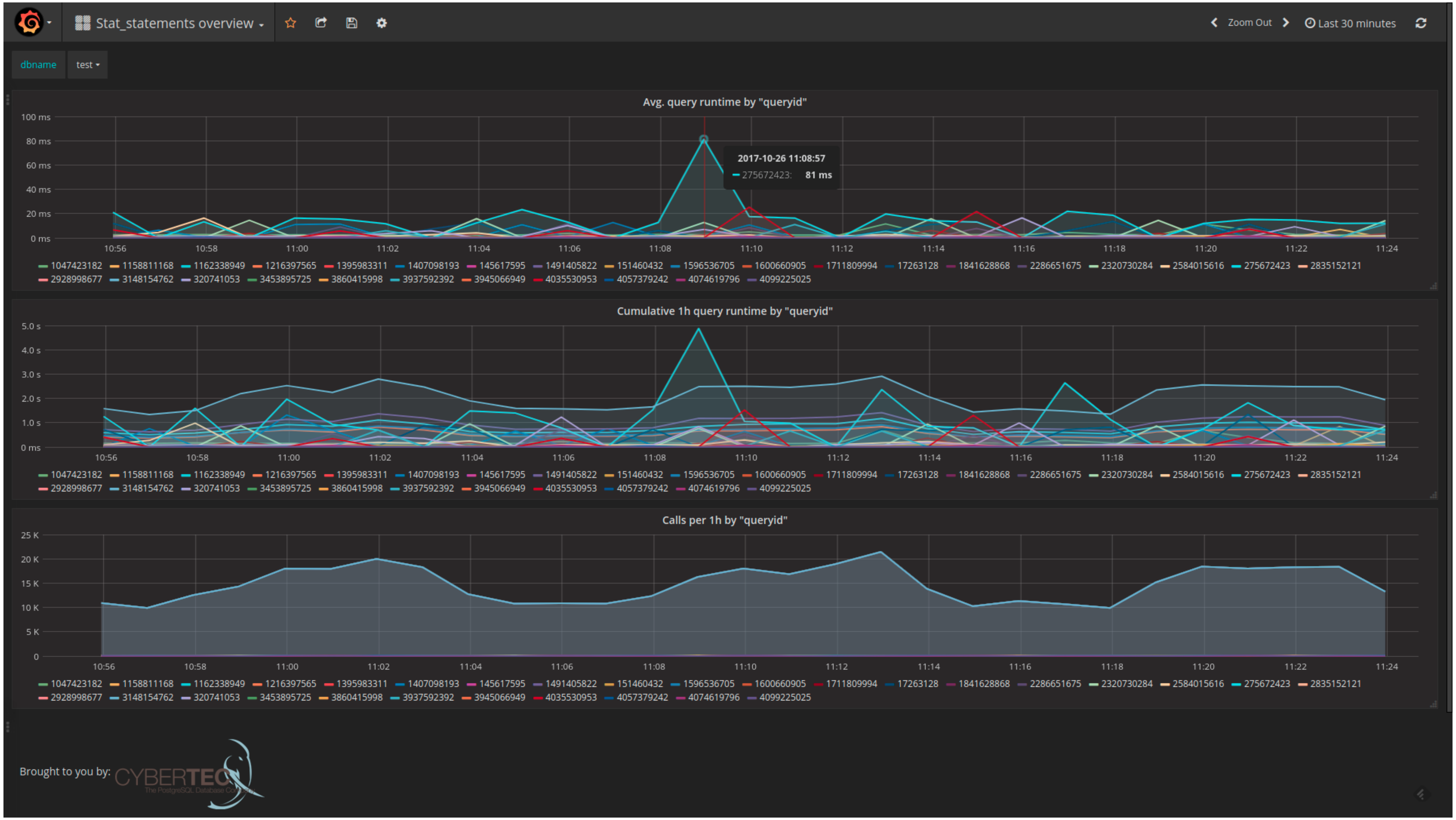
## Active metrics listing

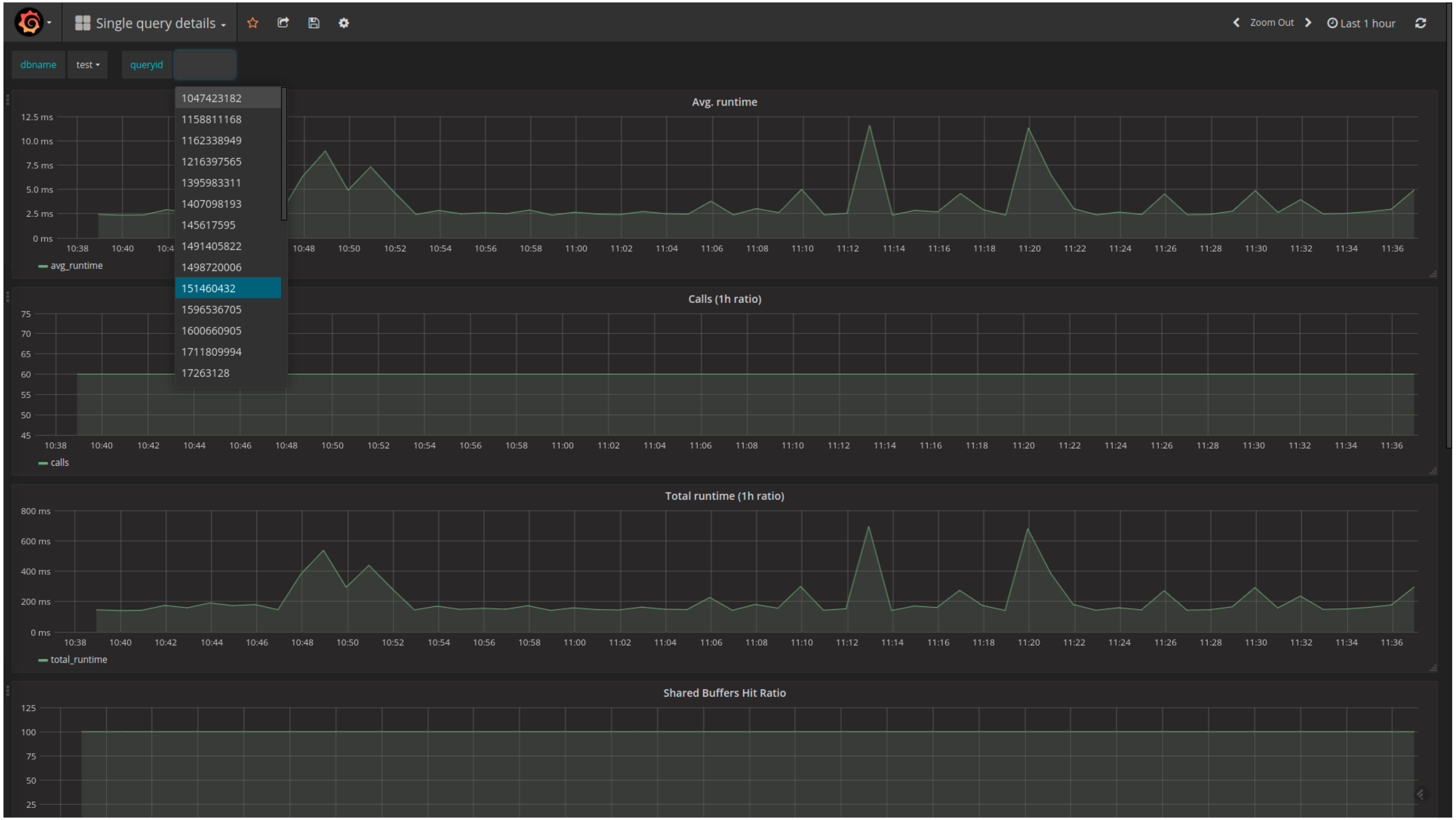
**backends** [ver: 9,9.6] **bgwriter** [ver: 9] **blocking_locks** [ver: 9.2] **buffercache_by_db** [ver: 9.2] **buffercache_by_type** [ver: 9.2] **change_events** [ver: 9] **configuration_hashes** [ver: 9] **cpu_load** [ver: 9] **db_stats** [ver: 9] **get_load_average** [ver: 9] **get_stat_statements** [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]

## Metric definitions

| Metric | PG version from | SQL | Comment | Is active? | Is helper? | Last modified | | |
|---|---|---|---|---|---|---|---|---|
| backends | 9.0 | with sa_snapshot as ( select * from pg_stat_activity where pid != | | ✓ | ☐ | 2017-09-19 12:43:46+00:00 | Save | Delete |
| backends | 9.6 | with sa_snapshot as ( select * from pg_stat_activity where pid != | | ✓ | ☐ | 2017-09-19 12:43:46+00:00 | Save | Delete |
| bgwriter | 9.0 | select (extract(epoch from now()) * 1e9)::int8 as | | ✓ | ☐ | 2017-09-19 12:43:46+00:00 | Save | Delete |
| blocking_locks | 9.2 | SELECT (extract(epoch from now()) * 1e9)::int8 AS | | ✓ | ☐ | 2017-09-19 12:43:46+00:00 | Save | Delete |
| buffercache_by_db | 9.2 | select (extract(epoch from now()) * 1e9)::int8 as | | ✓ | ☐ | 2017-09-19 12:43:46+00:00 | Save | Delete |


DB overview
Zoom Out
Last 1 hour
Press F11 to exit full screen
dbname
test
TPS Last 6 hours 5.5
QPS Last 6 hours 21
Query runtime Last 6 hours 0.14 ms
DB size ch. 1h Last 6 hours -32.0 KiB
Approx Bloat Last 6 hours 90.9 KiB
CPU Load Last 6 hours 0.7%
Tuple IUD statistics + IO times (1h ratios)
DELETE Avg: 0
UPDATE Avg: 45.0 K
INSERT Avg: 14.7 K
blk_write_time Avg: 0 ms
blk_read_time Avg: 0 ms
Buffer hit ratio + Rollback ratio
Shared buffers hit ratio Avg: 100.0%
TX rollback ratio Avg: 0%
Backends + Deadlocks + Temp bytes
Deadlocks (1h rate)
#Backends
Temp bytes written
WAL rate + DB size
WAL rate Avg: 5.5 kBps
cpu_load.mean Avg: 0.4 Bps
DB Size Avg: 22.8 MiB
Sessions
idle Avg: 2.0
idle in transaction Avg: 0
waiting Avg: 0
active Avg: 0.0
CPU load + avg. query runtime
load_5 Avg: 0.38%
avg_query_runtime Avg: 0 ms


Stat_statements overview
Zoom Out
Last 30 minutes
dbname
test
Avg. query runtime by "queryid"
100 ms
80 ms
60 ms
40 ms
20 ms
0 ms
2017-10-26 11:08:57
275672423: 81 ms
10:56
10:58
11:00
11:02
11:04
11:06
11:08
11:10
11:12
11:14
11:16
11:18
11:20
11:22
11:24
1047423182 1158811168 1162338949 1216397565 1395983311 1407098193 145617595 1491405822 151460432 1596536705 1600660905 1711809994 17263128 1841628868 2286651675 2320730284 2584015616 275672423 2835152121
2928998677 3148154762 320741053 3453895725 3860415998 3937592392 3945066949 4035530953 4057379242 4074619796 4099225025
Cumulative 1h query runtime by "queryid"
5.0 s
4.0 s
3.0 s
2.0 s
1.0 s
0 ms
Calls per 1h by "queryid"
25 K
20 K
15 K
10 K
5 K
0
Brought to you by:
CYBERTEC
The PostgreSQL Database Company


Single query details
Zoom Out
Last 1 hour
dbname
test
queryid
1047423182
1158811168
1162338949
1216397565
1395983311
1407098193
145617595
1491405822
1498720006
151460432
1596536705
1600660905
1711809994
17263128
Avg. runtime
avg_runtime
Calls (1h ratio)
calls
Total runtime (1h ratio)
total_runtime
Shared Buffers Hit Ratio

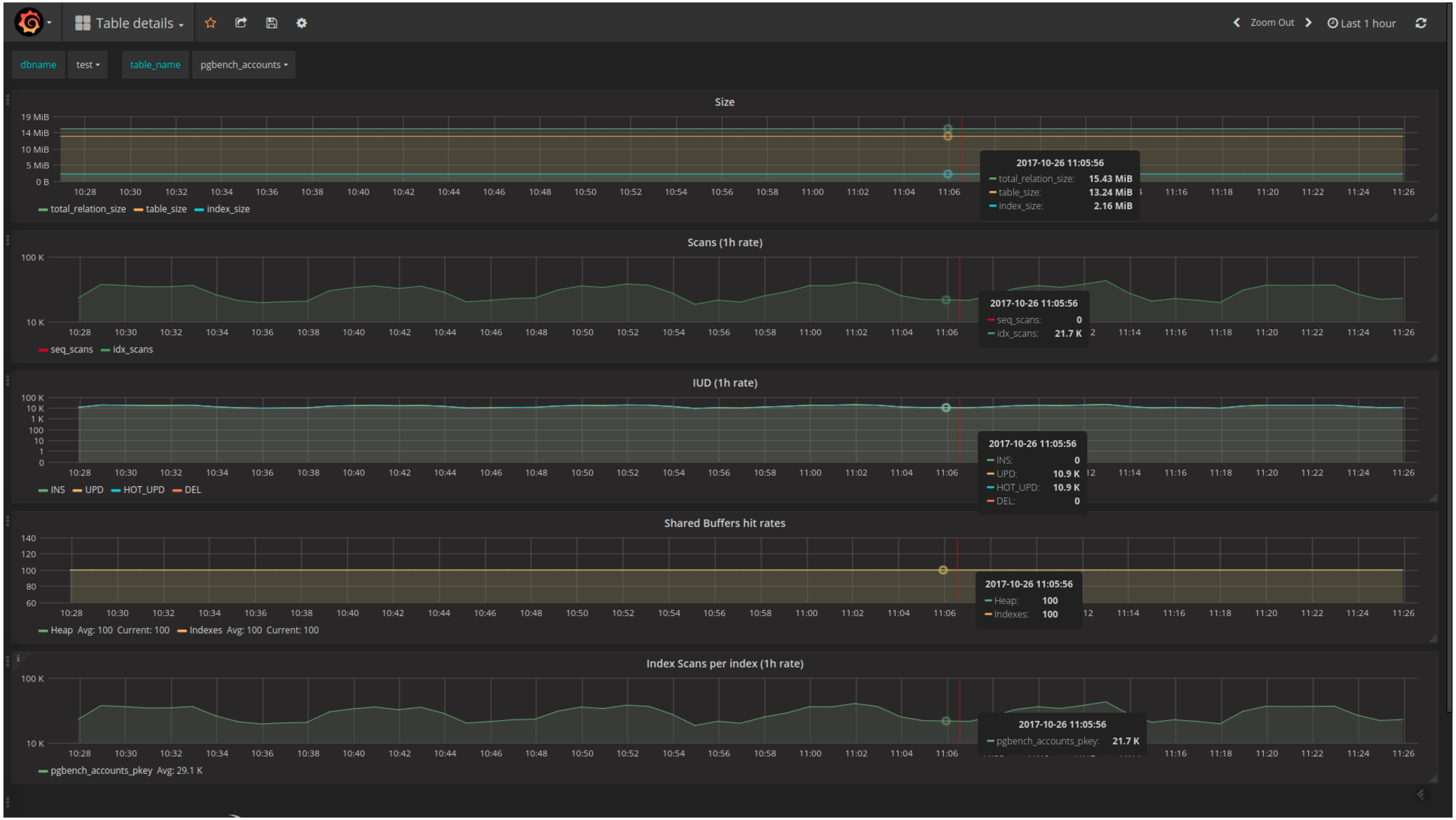
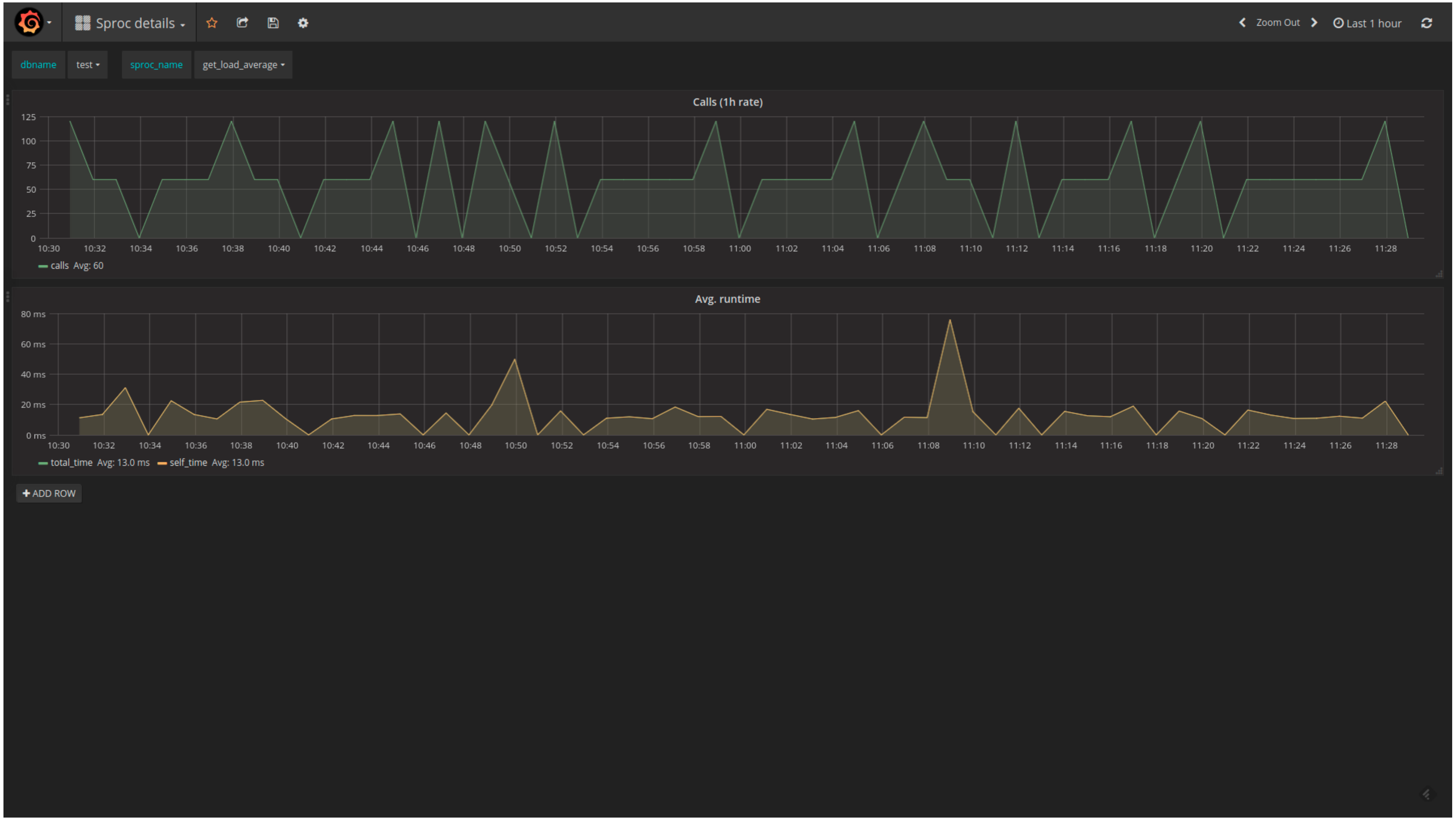
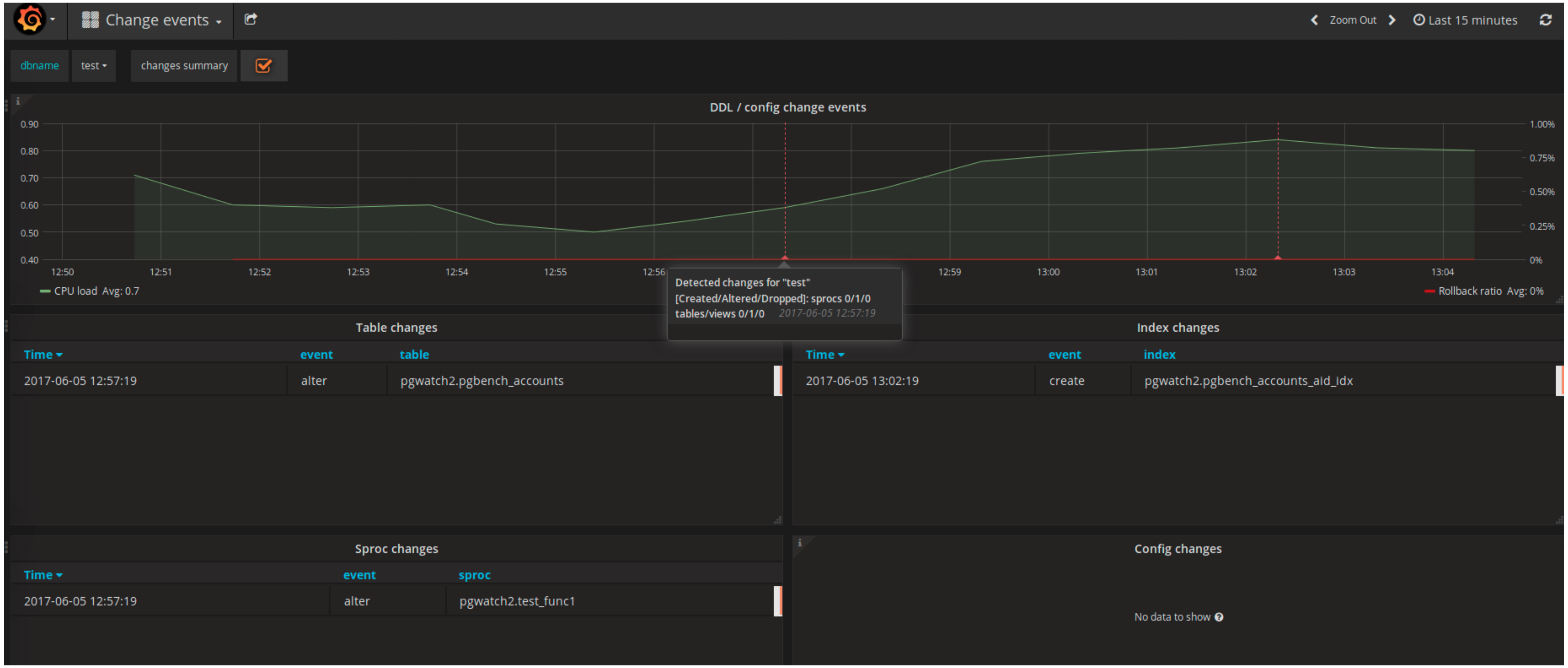


Table details
Zoom Out
Last 1 hour
dbname
test
table_name
pgbench_accounts
Size
2017-10-26 11:05:56
total_relation_size: 15.43 MiB
table_size: 13.24 MiB
index_size: 2.16 MiB
total_relation_size
table_size
index_size
Scans (1h rate)
2017-10-26 11:05:56
seq_scans: 0
idx_scans: 21.7 K
seq_scans
idx_scans
IUD (1h rate)
2017-10-26 11:05:56
INS: 0
UPD: 10.9 K
HOT_UPD: 10.9 K
DEL: 0
INS
UPD
HOT_UPD
DEL
Shared Buffers hit rates
2017-10-26 11:05:56
Heap: 100
Indexes: 100
Heap Avg: 100 Current: 100
Indexes Avg: 100 Current: 100
Index Scans per index (1h rate)
2017-10-26 11:05:56
pgbench_accounts_pkey: 21.7 K
pgbench_accounts_pkey Avg: 29.1 K


Sproc details
Zoom Out
Last 1 hour
dbname
test
sproc_name
get_load_average
Calls (1h rate)
calls Avg: 60
Avg. runtime
total_time Avg: 13.0 ms
self_time Avg: 13.0 ms
+ ADD ROW



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 |

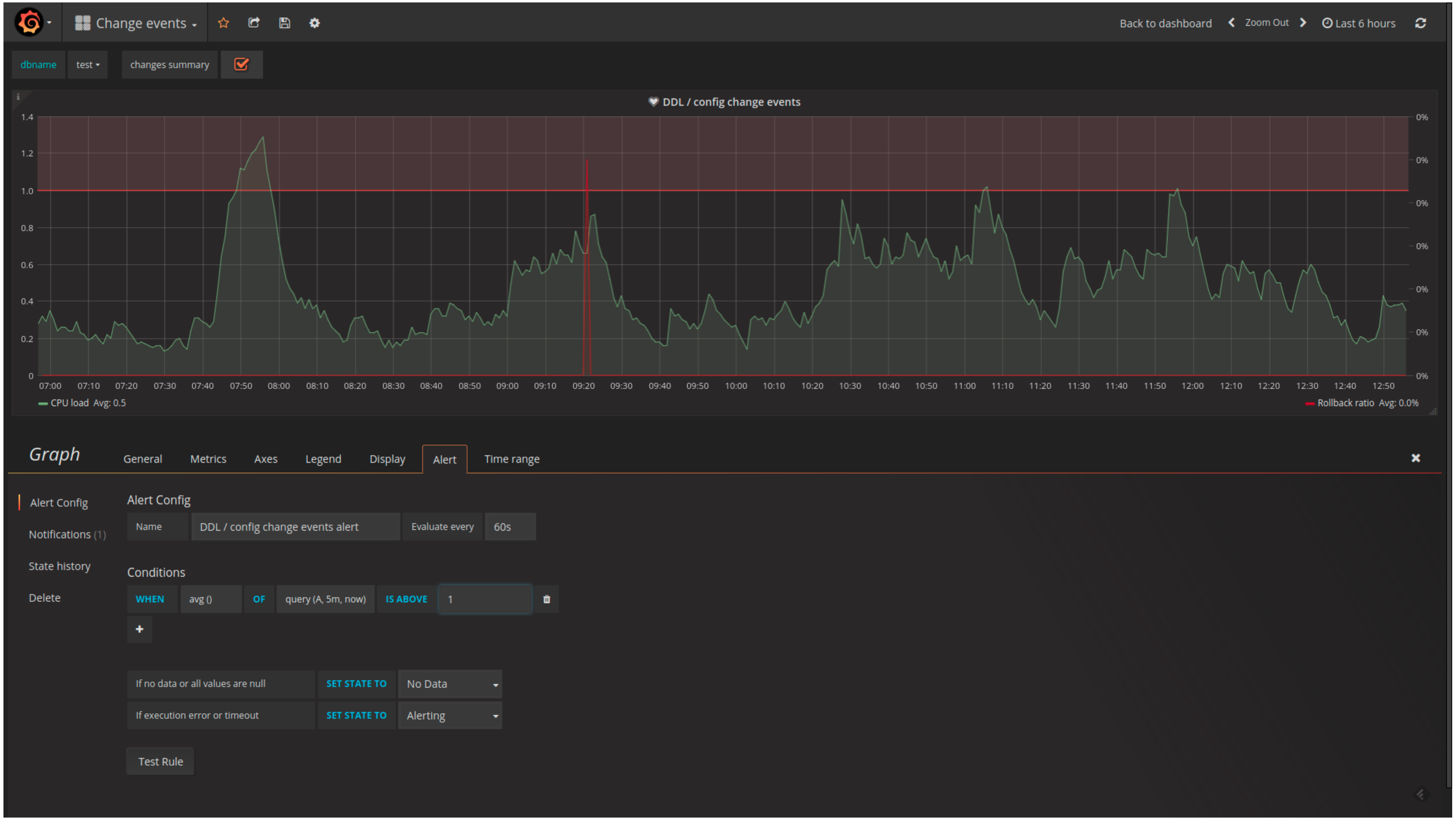
**Config changes**

No data to show

## Alerting / Anomaly detection

## Grafana

- Eeasy setup, point and click
- Most important alerting services covered
  - Email
  - PagerDuty
  - Slack
  - Web hooks
  - Kafka
  - ...
- Graph panel only



Alert Config

Name: DDL / config change events alert
Evaluate every: 60s

Conditions

WHEN avg () OF query (A, 5m, now) IS ABOVE 1

If no data or all values are null SET STATE TO No Data

If execution error or timeout SET STATE TO Alerting

## Kapacitor

Part of the InfluxData's TICK stack

- Harder to get going but very powerful!
- Features
  - 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 - simple

```
stream
    |from()
        .measurement('cpu')
    |alert()
        .crit(lambda: "usage_idle" <  70)
        .log('/tmp/alerts.log')
        .email()
```

## Kapacitor sample - complex

```
|from()
    .measurement('cpu')
|groupBy('service', 'datacenter')
|window()
    .period(1m)
|percentile('load_1min', 95.0)
|eval(lambda: sigma("percentile"))
    .as('sigma')
|alert()
 .id('{{ .Name }}/{{ index .Tags "service" }}/{{ index .T
 .message('{{ .ID }} is {{ .Level }} cpu-95th:{{ index .F
    .crit(lambda: "sigma" > 3.0)
```

## pgwatch2 - What's next?

## Improvement areas

- More system level metrics
    - Better wrappers for cpu, disk, mem
- Better query text handling
    - Web UI has pg_stat_statements overview
- Fully automatic Docker updates
- Log parsing?

User input expected @ github.com/cybertec-postgresql/pgwatch2

## Contact us

```
Web: www.cybertec.at
Github: github.com/cybertec-postgresql
Twitter: @PostgresSupport
```