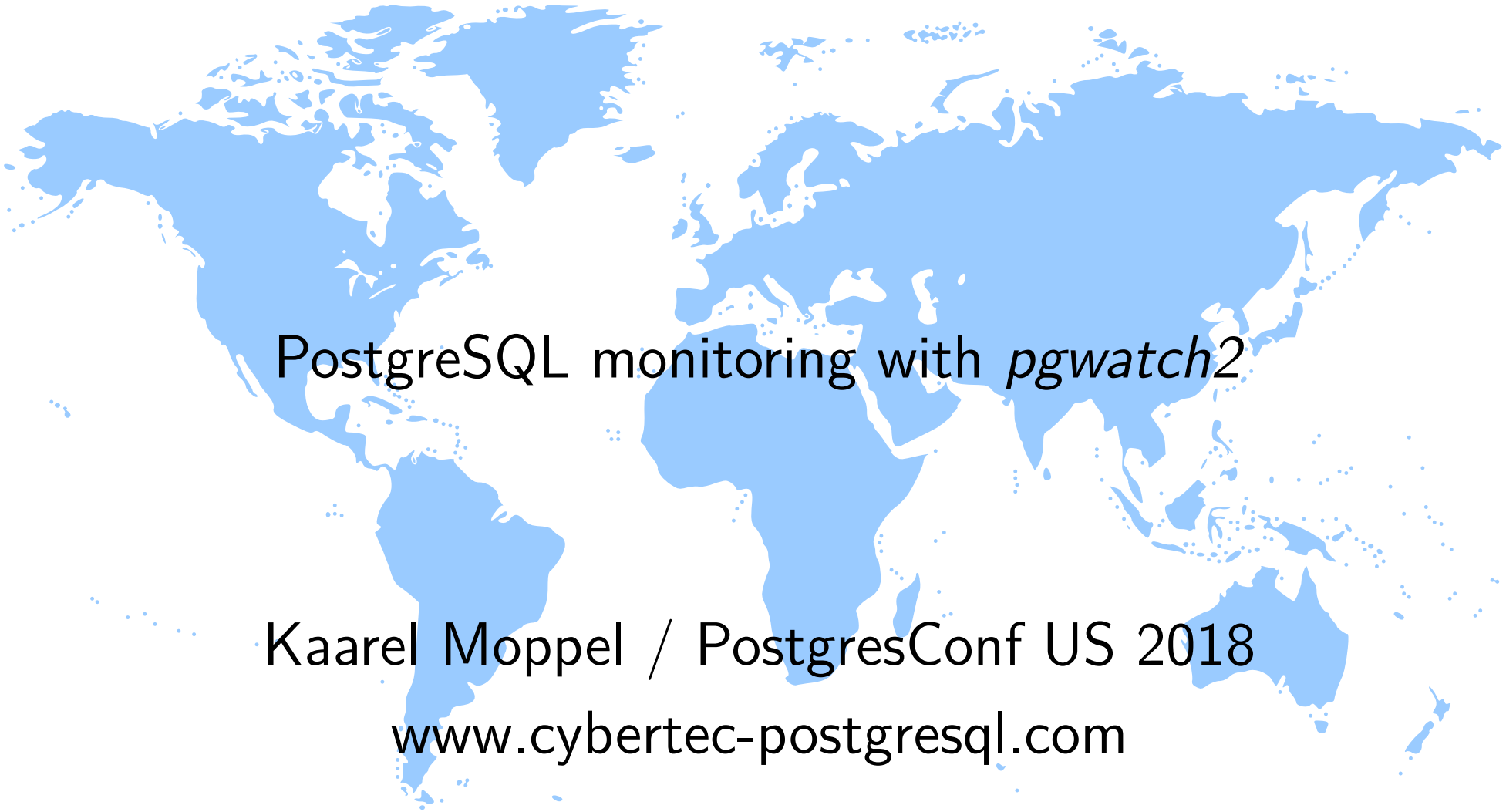


PostgreSQL monitoring with *pgwatch2*

Kaarel Moppel / PostgresConf US 2018

www.cybertec-postgresql.com

Why to monitor

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

Different levels of Database monitoring

- Service availability level
- System level monitoring
- Database
- Application level would be nice also

PostgreSQL land

- Log analysis
- Stats Collector
  - 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
  - NB! Not all track_* parameters enabled by default

PostgreSQL Monitoring Tools

No shortage of tools

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

Approaches

- Ad hoc
- Continuous monitoring frameworks
  - Cloud / SaaS / APM / Generic
  - DIY / Open Source

Postgres specific Open Source continuous monitoring frameworks

Postgres specific

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

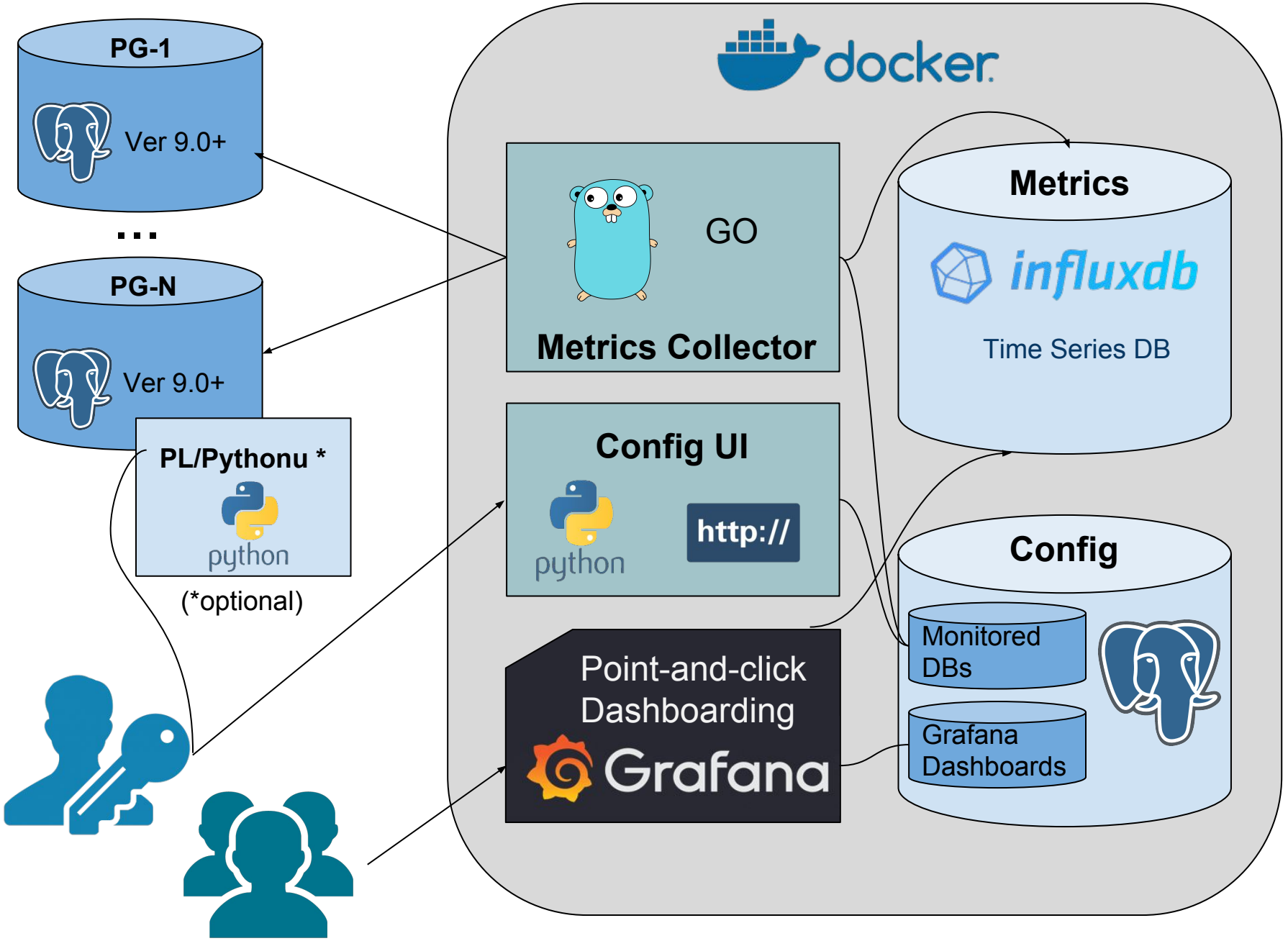
pgwatch2

Main principles - why another tool?

- 1-minute setup
  - Docker first
- User defined visuals / Dashboarding
- Non-invasive
  - No extensions for main functionality
- Easy extensibility
  - SQL defined metrics
- Do minimal work needed, use existing good software

Architecture components

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

Features (1)

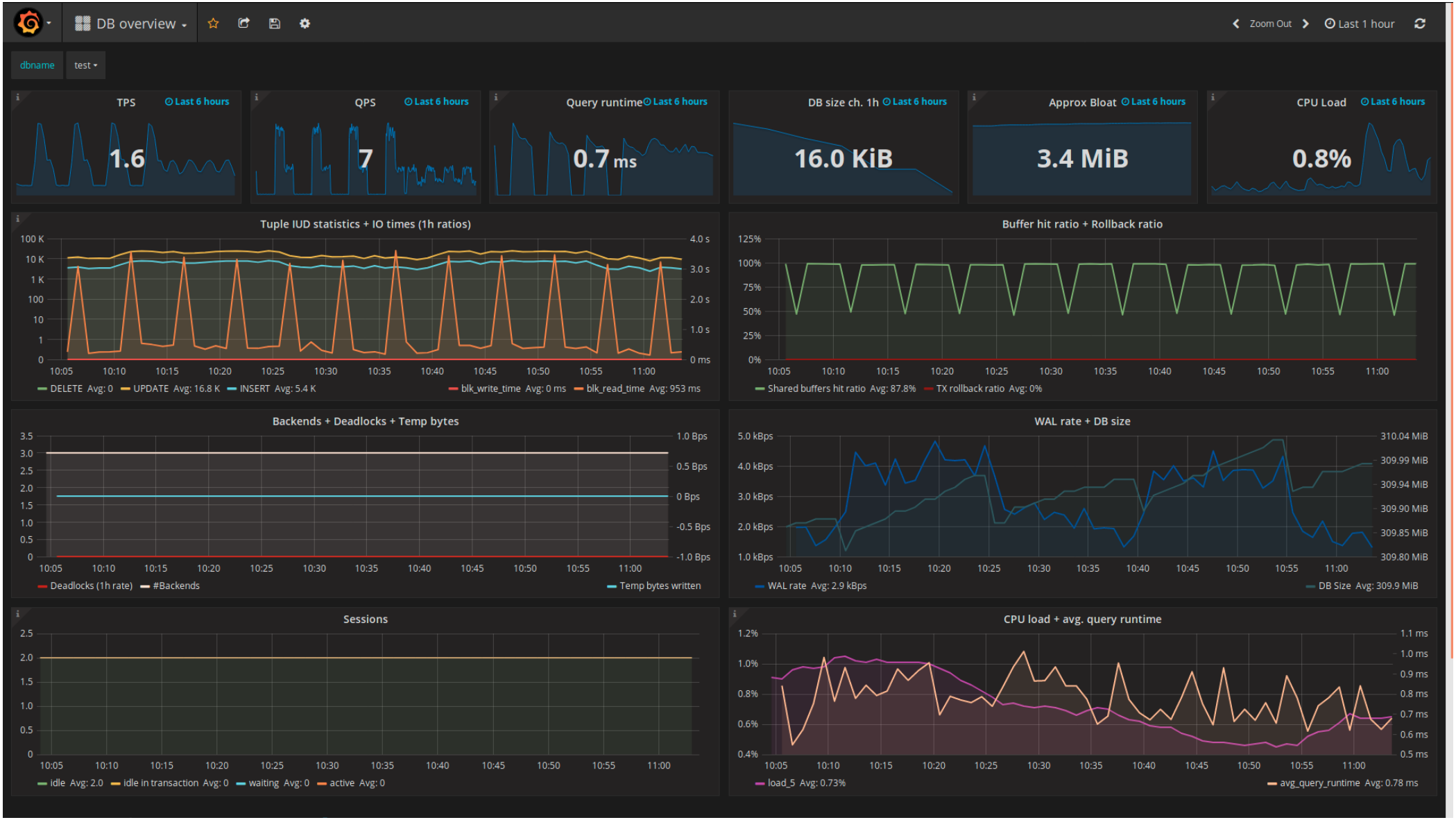
- "Ready to go"
  - Defaults cover almost all pg_stat* views
- Supports Postgres 9.0+ (older versions also possible)
- Security
- SSL between all components possible
- Admin / normal user separation
- Custom metrics via SQL, also for business domain!
- Reuse of existing Postgres, Grafana, InfluxDB installation possible
- Can be integrated with Kubernetes/OpenStack

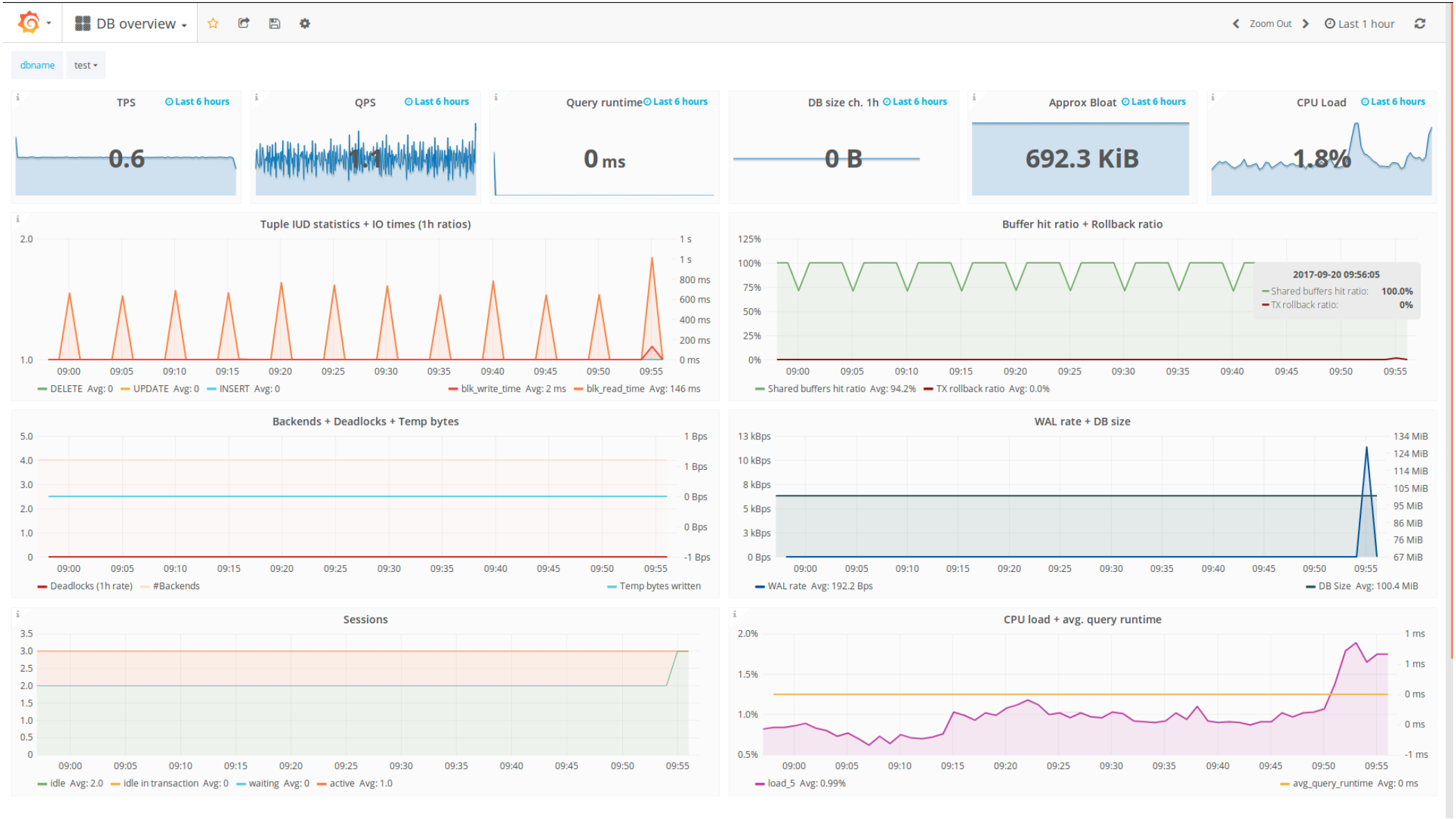
Features (2)

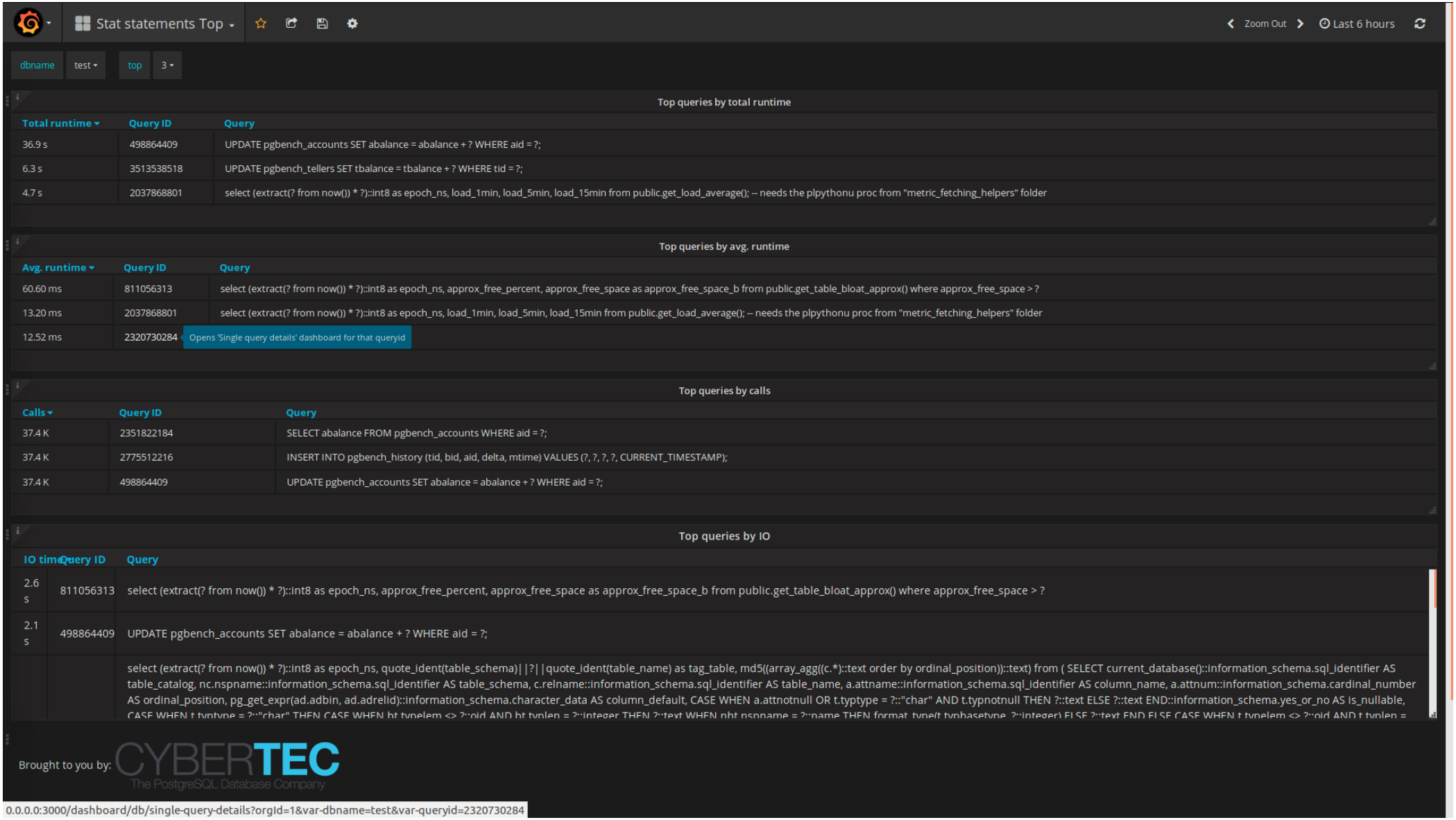
- Possible to monitor all databases of a cluster automatically
- Change detection
  - Added/changed/deleted table/index/sproc/config events
- PgBouncer statistics support
- AWS CloudWatch support
- Alerting easily possible with Grafana
  - Kapacitor ("K" from InfluxData's TICK stack)
- Extensible data presentation - Grafana has plugins!

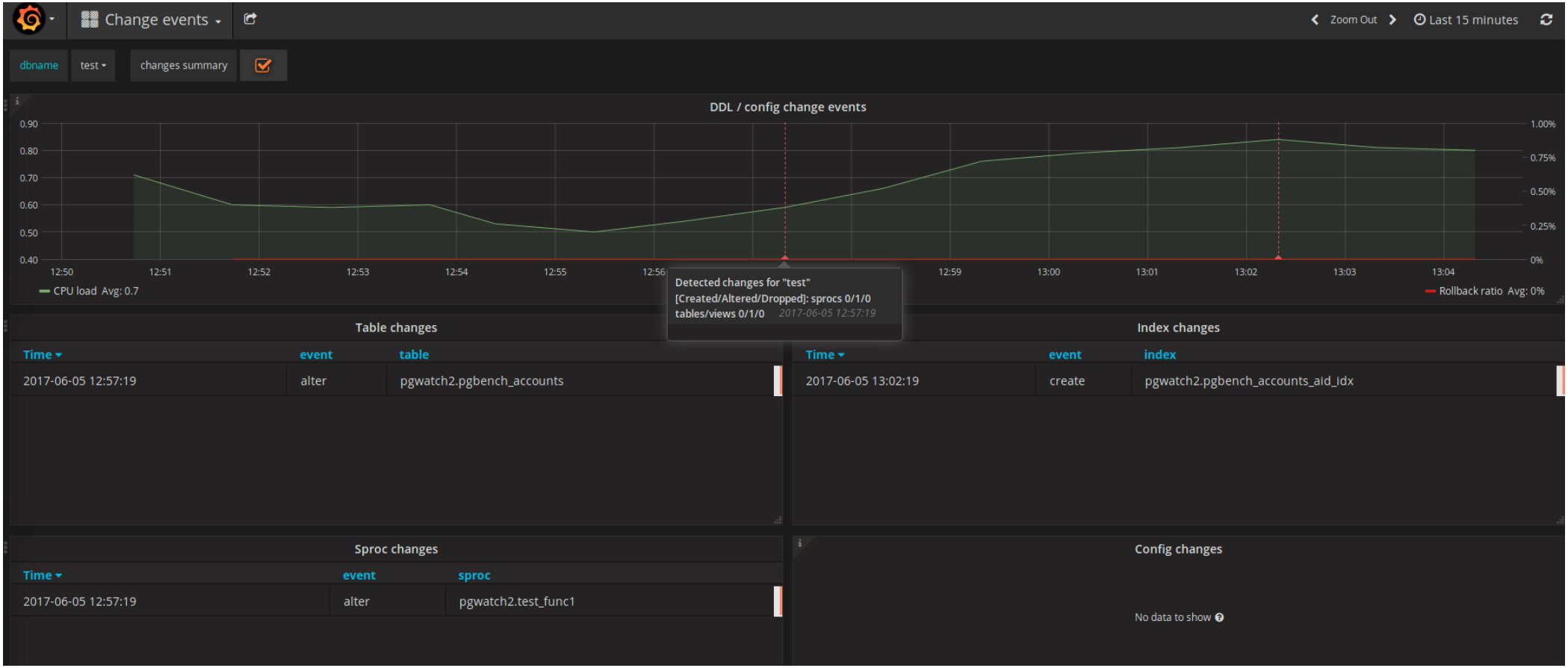
Getting started

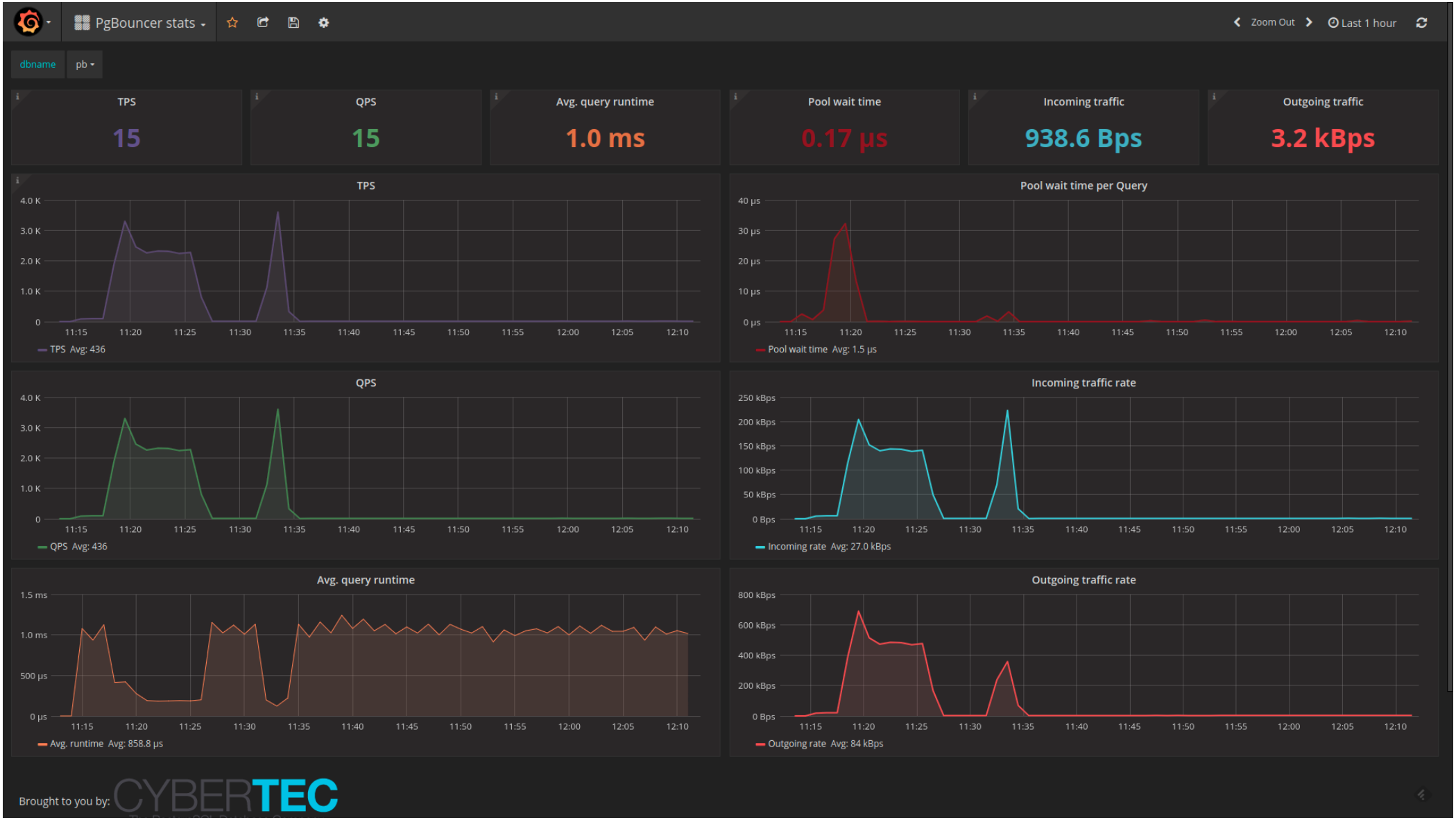
1. `docker run -d -p 3000:3000 -p 8080:8080 \`
   `--name pw2 cybertec/pgwatch2`
2. Wait some seconds and open browser at localhost:8080
3. Insert your DB connection strings and wait some minutes
4. Start viewing/editing dashboarding!

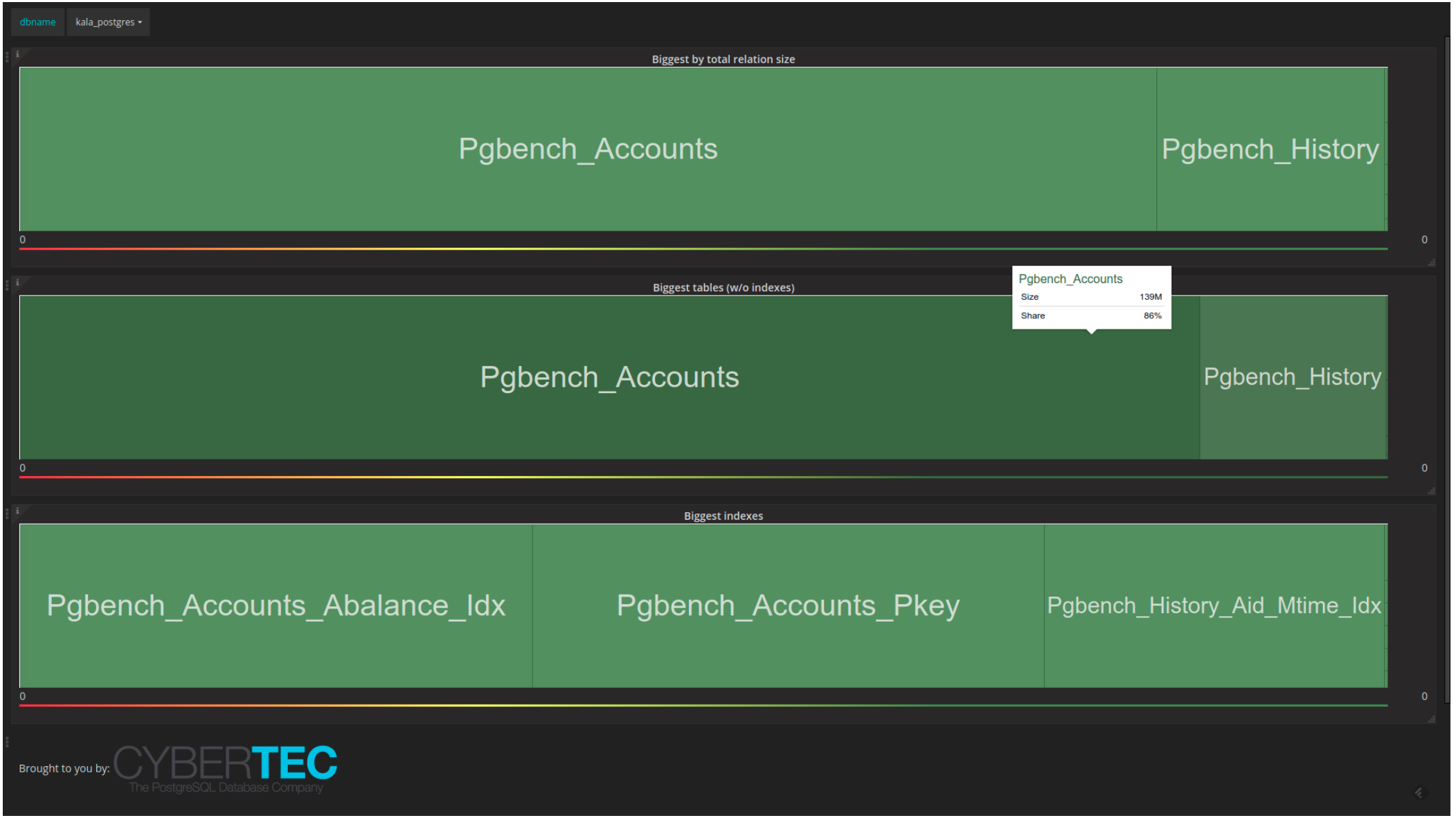

DB overview
Zoom Out
Last 1 hour
dbname
test
TPS
Last 6 hours
1.6
QPS
Last 6 hours
7
Query runtime
Last 6 hours
0.7 ms
DB size ch. 1h
Last 6 hours
16.0 KiB
Approx Bloat
Last 6 hours
3.4 MiB
CPU Load
Last 6 hours
0.8%
Tuple IUD statistics + IO times (1h ratios)
DELETE Avg: 0
UPDATE Avg: 16.8 K
INSERT Avg: 5.4 K
blk_write_time Avg: 0 ms
blk_read_time Avg: 953 ms
Buffer hit ratio + Rollback ratio
Shared buffers hit ratio Avg: 87.8%
TX rollback ratio Avg: 0%
Backends + Deadlocks + Temp bytes
Deadlocks (1h rate)
#Backends
Temp bytes written
WAL rate + DB size
WAL rate Avg: 2.9 kBps
DB Size Avg: 309.9 MiB
Sessions
idle Avg: 2.0
idle in transaction Avg: 0
waiting Avg: 0
active Avg: 0
CPU load + avg. query runtime
load_5 Avg: 0.73%
avg_query_runtime Avg: 0.78 ms


DB overview
Zoom Out
Last 1 hour
dbname
test
TPS
Last 6 hours
0.6
QPS
Last 6 hours
Query runtime
Last 6 hours
0 ms
DB size ch. 1h
Last 6 hours
0 B
Approx Bloat
Last 6 hours
692.3 KiB
CPU Load
Last 6 hours
1.8%
Tuple IUD statistics + IO times (1h ratios)
DELETE Avg: 0
UPDATE Avg: 0
INSERT Avg: 0
blk_write_time Avg: 2 ms
blk_read_time Avg: 146 ms
Buffer hit ratio + Rollback ratio
2017-09-20 09:56:05
Shared buffers hit ratio: 100.0%
TX rollback ratio: 0%
Shared buffers hit ratio Avg: 94.2%
TX rollback ratio Avg: 0.0%
Backends + Deadlocks + Temp bytes
Deadlocks (1h rate)
#Backends
Temp bytes written
WAL rate + DB size
WAL rate Avg: 192.2 Bps
DB Size Avg: 100.4 MiB
Sessions
idle Avg: 2.0
idle in transaction Avg: 0
waiting Avg: 0
active Avg: 1.0
CPU load + avg. query runtime
load_5 Avg: 0.99%
avg_query_runtime Avg: 0 ms


Stat statements Top
Zoom Out
Last 6 hours
dbname
test
top
3
Top queries by total runtime
Total runtime
Query ID
Query
36.9 s
498864409
UPDATE pgbench_accounts SET abalance = abalance + ? WHERE aid = ?;
6.3 s
3513538518
UPDATE pgbench_tellers SET tbalance = tbalance + ? WHERE tid = ?;
4.7 s
2037868801
select (extract(? from now()) * ?)::int8 as epoch_ns, load_1min, load_5min, load_15min from public.get_load_average(); -- needs the plpythonu proc from "metric_fetching_helpers" folder
Top queries by avg. runtime
Avg. runtime
Query ID
Query
60.60 ms
811056313
select (extract(? from now()) * ?)::int8 as epoch_ns, approx_free_percent, approx_free_space as approx_free_space_b from public.get_table_bloat_approx() where approx_free_space > ?
13.20 ms
2037868801
select (extract(? from now()) * ?)::int8 as epoch_ns, load_1min, load_5min, load_15min from public.get_load_average(); -- needs the plpythonu proc from "metric_fetching_helpers" folder
12.52 ms
2320730284
Opens 'Single query details' dashboard for that queryid
Top queries by calls
Calls
Query ID
Query
37.4 K
2351822184
SELECT abalance FROM pgbench_accounts WHERE aid = ?;
37.4 K
2775512216
INSERT INTO pgbench_history (tid, bid, aid, delta, mtime) VALUES (?, ?, ?, ?, CURRENT_TIMESTAMP);
37.4 K
498864409
UPDATE pgbench_accounts SET abalance = abalance + ? WHERE aid = ?;
Top queries by IO
IO time
Query ID
Query
2.6 s
811056313
select (extract(? from now()) * ?)::int8 as epoch_ns, approx_free_percent, approx_free_space as approx_free_space_b from public.get_table_bloat_approx() where approx_free_space > ?
2.1 s
498864409
UPDATE pgbench_accounts SET abalance = abalance + ? WHERE aid = ?;
select (extract(? from now()) * ?)::int8 as epoch_ns, quote_ident(table_schema)||?||quote_ident(table_name) as tag_table, md5((array_agg((c.*)::text order by ordinal_position))::text) from (SELECT current_database()::information_schema.sql_identifier AS table_catalog, nc.nspname::information_schema.sql_identifier AS table_schema, c.relname::information_schema.sql_identifier AS table_name, a.attname::information_schema.sql_identifier AS column_name, a.attnum::information_schema.cardinal_number AS ordinal_position, pg_get_expr(ad.adbin, ad.adrelid)::information_schema.character_data AS column_default, CASE WHEN a.attnotnull OR t.typtype = ?::"char" AND t.typnotnull THEN ?::text ELSE ?::text END::information_schema.yes_or_no AS is_nullable, CASE WHEN t.typtype = ?::"char" THEN CASE WHEN bt.typelem <> ?::oid AND bt.typlen = ?::integer THEN ?::text WHEN nbt.nspname = ?::name THEN format_type(t.typbasetype, ?::integer) ELSE ?::text END ELSE CASE WHEN t.typelem <> ?::oid AND t.typlen =
Brought to you by:
CYBERTEC
The PostgreSQL Database Company
0.0.0.0:3000/dashboard/db/single-query-details?orgId=1&var-dbname=test&var-queryid=2320730284


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

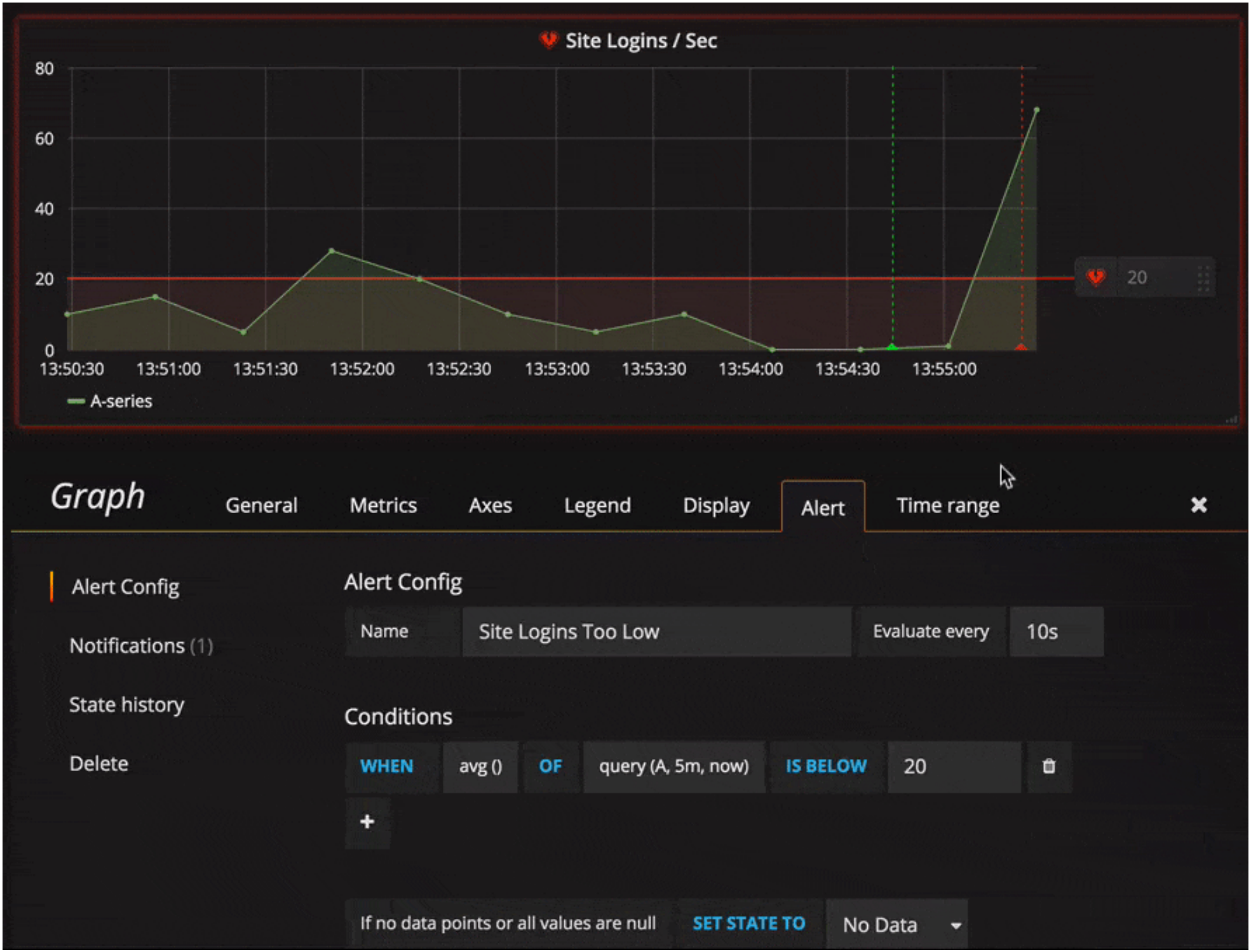

PgBouncer stats
Zoom Out
Last 1 hour
dbname
pb
TPS
15
QPS
15
Avg. query runtime
1.0 ms
Pool wait time
0.17 µs
Incoming traffic
938.6 Bps
Outgoing traffic
3.2 kBps
TPS
4.0 K
3.0 K
2.0 K
1.0 K
0
11:15 11:20 11:25 11:30 11:35 11:40 11:45 11:50 11:55 12:00 12:05 12:10
TPS Avg: 436
Pool wait time per Query
40 µs
30 µs
20 µs
10 µs
0 µs
Pool wait time Avg: 1.5 µs
QPS
QPS Avg: 436
Incoming traffic rate
250 kBps
200 kBps
150 kBps
100 kBps
50 kBps
0 Bps
Incoming rate Avg: 27.0 kBps
Avg. query runtime
1.5 ms
1.0 ms
500 µs
0 µs
Avg. runtime Avg: 858.8 µs
Outgoing traffic rate
800 kBps
600 kBps
400 kBps
200 kBps
0 Bps
Outgoing rate Avg: 84 kBps
Brought to you by: CYBERTEC


dbname
kala_postgres
Biggest by total relation size
Pgbench_Accounts
Pgbench_History
0
0
Biggest tables (w/o indexes)
Pgbench_Accounts
Size 139M
Share 86%
Pgbench_Accounts
Pgbench_History
0
0
Biggest indexes
Pgbench_Accounts_Abalance_Idx
Pgbench_Accounts_Pkey
Pgbench_History_Aid_Mtime_Idx
0
0
Brought to you by:
CYBERTEC
The PostgreSQL Database Company

Alerting

Grafana

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


Site Logins / Sec
80
60
40
20
0
13:50:30
13:51:00
13:51:30
13:52:00
13:52:30
13:53:00
13:53:30
13:54:00
13:54:30
13:55:00
A-series
20
Graph
General
Metrics
Axes
Legend
Display
Alert
Time range
Alert Config
Notifications (1)
State history
Delete
Alert Config
Name
Site Logins Too Low
Evaluate every
10s
Conditions
WHEN
avg ()
OF
query (A, 5m, now)
IS BELOW
20
+
If no data points or all values are null
SET STATE TO
No Data

Anomaly detection

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 (simplified)


```
|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
- Log parsing
- Per metric timeouts and activity times
- Fully automatic Docker updates
- ???

User input very much appreciated!

Contact us

github.com/cybertec-postgresql/pgwatch2

Austria / Switzerland / Estonia
Web: www.cybertec-postgresql.com
Github: github.com/cybertec-postgresql
Twitter: @PostgresSupport