Sustainable Database Performance Profiling in PostgreSQL

Dirk Krautschick
PG Day Paris 14.03.2024
# whoami

Dirk Krautschick
Solution Architect

with Aiven since Nov 2023

- 16 years
- DBA, Trainer, Consulting, Sales Engineering
- PostgreSQL, Oracle
- Married, 2 Junior DBAs
- Mountainbike, swimming, movies, music, hifi/home cinema, 8 bit computing
Disclaimer

Different audience, different perspectives
My experience, my honest opinion
Let’s stay open minded
Always open for discussions
What happened so far...

There was a talk...
“Pro-Active Performance Analysis in PostgreSQL”
https://www.youtube.com/watch?v=rgdA0FwVShI

About
- Performance problems overall
- Different analysis approaches
- Recommendations/usage of Extensions
Solid feedback, consensus in practical experiences
Many questions about “THAT LAST PART of the talk”
Sick of giving this talk after so many times ... :-)

Motivation to do a Spin-Off talk!
In PostgreSQL every relevant information is there...

...but only for NOW!

Obvious Sources

- Parameters, Sizing (at that time!)
- Information_schema, system catalogues

Main Challenge: How to handle, keep and collect all that stuff!
For sure, monitoring is essential, but...

...it shows mostly

that **something is slow**, sometimes maybe...

**what is exactly slow**, but almost never...

**why it is slow**!

PostgreSQL insights necessary

Deep dive or investigation as a next step anyway
What about logging...

PostgreSQL logging is awesome
Exhaustive possibilities
Straight and easy configuration
Be aware of storage and load
  High maintenance
Evaluate Logging strategies

```
log_line_prefix = '%t [%p]:
user=%u,db=%d,app=%a,client=%h ,
...
log_parser_stats = off
log_planner_stats = off
log_executor_stats = off
log_statement_stats = on
...
log_checkpoints = on
log_connections = on
log_disconnections = on
log_lock_waits = on
log_temp_files = 0
log_autovacuum_min_duration = 0
log_error_verbosity = default
...
log_min_messages = debug5
log_min_error_statement = debug5
log_min_duration_statement = 0
log_min_duration_sample = 0
...
log_statement = 'all'
```
Statement level statistics
Required by several monitoring tools
Statement based collection of e.g.
  Executions
  Execution times (min, max, average)
  Rows
  Blocks read/write
  …

# \d pg_stat_statements
View "public.pg_stat_statements"

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>userid</td>
<td>oid</td>
</tr>
<tr>
<td>dbid</td>
<td>oid</td>
</tr>
<tr>
<td>queryid</td>
<td>bigint</td>
</tr>
<tr>
<td>query</td>
<td>text</td>
</tr>
<tr>
<td>total_plan_time</td>
<td>double precision</td>
</tr>
<tr>
<td>calls</td>
<td>bigint</td>
</tr>
<tr>
<td>total_exec_time</td>
<td>double precision</td>
</tr>
<tr>
<td>min_exec_time</td>
<td>double precision</td>
</tr>
<tr>
<td>max_exec_time</td>
<td>double precision</td>
</tr>
<tr>
<td>mean_exec_time</td>
<td>double precision</td>
</tr>
<tr>
<td>stddev_exec_time</td>
<td>double precision</td>
</tr>
<tr>
<td>rows</td>
<td>bigint</td>
</tr>
<tr>
<td>blk_read_time</td>
<td>double precision</td>
</tr>
<tr>
<td>blk_write_time</td>
<td>double precision</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>
```sql
# SELECT
    substring(query, 1, 50) as short_query,
    round(total_exec_time) as total_exec_time, calls,
    round(mean_exec_time) as mean_exec_time,
    round(100 * total_exec_time / (SELECT sum(total_exec_time) FROM_stat_statements)) as percentage
FROM
    pg_stat_statements
ORDER BY
    percentage desc;
```

<table>
<thead>
<tr>
<th>short_query</th>
<th>total_exec_time</th>
<th>calls</th>
<th>mean_exec_time</th>
<th>percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPDATE pgbench_branches SET bbalance = bbalance +</td>
<td>7114</td>
<td>1500</td>
<td>5</td>
<td>63</td>
</tr>
<tr>
<td>UPDATE pgbench_tellers SET tbalance = tbalance + $</td>
<td>2506</td>
<td>1500</td>
<td>2</td>
<td>22</td>
</tr>
<tr>
<td>copy pgbench_accounts from stdin</td>
<td>664</td>
<td>1</td>
<td>664</td>
<td>6</td>
</tr>
<tr>
<td>UPDATE pgbench_accounts SET abalance = abalance +</td>
<td>194</td>
<td>1500</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>alter table pgbench_accounts add primary key (aid)</td>
<td>193</td>
<td>1</td>
<td>193</td>
<td>2</td>
</tr>
<tr>
<td>vacuum analyze pgbench_accounts</td>
<td>138</td>
<td>1</td>
<td>138</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Statistics about reads/writes on filesystem

Statistics about CPU usage

`pg_stat_statements` is required

```
postgres=# \d pg_stat_kcache_detail;
View "public.pg_stat_kcache"
<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Collation</th>
<th>Nullable</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>datname</td>
<td>name</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>plan_user_time</td>
<td>double precision</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>plan_system_time</td>
<td>double precision</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>plan_minflts</td>
<td>numeric</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>plan_majflts</td>
<td>numeric</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>plan_nswaps</td>
<td>numeric</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>plan_reads</td>
<td>numeric</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>plan_reads_blks</td>
<td>numeric</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>plan_writes</td>
<td>numeric</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>plan_writes_blks</td>
<td>numeric</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>numeric</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>plan_nivcsws</td>
<td>numeric</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>exec_user_time</td>
<td>double precision</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>exec_system_time</td>
<td>double precision</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>numeric</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>exec_nsignals</td>
<td>numeric</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>exec_nvcsws</td>
<td>numeric</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>exec_nivcsws</td>
<td>numeric</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
What’s still missing...?
...handling WAIT_EVENTS!!!
Wait events from `pg_stat_activity`
Sampled statistics of wait events
Combination with `pg_stat_statements`

[github.com/postgrespro/pg_wait_sampling](https://github.com/postgrespro/pg_wait_sampling)

Views
- `pg_wait_sampling_current`
- `pg_wait_sampling_history`
- `pg_wait_sampling_profile`

Functions
- `pg_wait_sampling_get_current(pid)`
- `pg_wait_sampling_reset_profile()`
```sql
postgres=# select * from pg_wait_sampling_profile order by pid, count desc;

<table>
<thead>
<tr>
<th>pid</th>
<th>event_type</th>
<th>event</th>
<th>queryid</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1689</td>
<td>IO</td>
<td>DataFileWrite</td>
<td>2862011717192834034</td>
<td>4010499</td>
</tr>
<tr>
<td>1685</td>
<td>IO</td>
<td>DataFileRead</td>
<td>2862011717192834034</td>
<td>4010097</td>
</tr>
<tr>
<td>1686</td>
<td>IO</td>
<td>DataFileSync</td>
<td>-4888004026240188267</td>
<td>4007477</td>
</tr>
<tr>
<td>1684</td>
<td>Activity</td>
<td>BgWriterHibernate</td>
<td>2862011717192834034</td>
<td>3991477</td>
</tr>
<tr>
<td>1683</td>
<td>Activity</td>
<td>CheckpointerMain</td>
<td>1511417639870010300</td>
<td>3927957</td>
</tr>
<tr>
<td>1684</td>
<td>Activity</td>
<td>BgWriterMain</td>
<td>-4888004026240188267</td>
<td>88494</td>
</tr>
<tr>
<td>3720</td>
<td>Client</td>
<td>ClientRead</td>
<td>-4888004026240188267</td>
<td>2393</td>
</tr>
<tr>
<td>1685</td>
<td>IO</td>
<td>WALSync</td>
<td>6648255685428052402</td>
<td>65</td>
</tr>
<tr>
<td>3546</td>
<td>Client</td>
<td>ClientRead</td>
<td>-4888004026240188267</td>
<td>1</td>
</tr>
<tr>
<td>3546</td>
<td>IO</td>
<td>DataFileRead</td>
<td>2862011717192834034</td>
<td>1</td>
</tr>
</tbody>
</table>

...
The Idea

Getting sustainable?
Let’s pick a random example...

... let’s say ... Oracle Database :-)

Collects almost everything per default (sometimes sampled)

Interpretation with

Querying Views (obviously!)

Statspack (basic, always available and “costless”)
Let’s pick a random example...

... let’s say ... Oracle Database :-)

Collects almost everything per default *(sometimes sampled)*

Interpretation with

- Querying Views *(obviously!)*
- Statspack *(basic, always available and “costless”)*
- Diagnostic and Tuning Pack *(expensive Option)*
  - Only for Enterprise Edition
  - Several Tools, like ASH, AWR,...
Think outside the Box

Frequent snapshots of performance data in a repository
Defined time periods and retention
Creation of nice reports based on those snapshots
Time frame between two or more snapshots
Frequent snapshots of performance data in a repository
Defined time periods and retention
Creation of nice reports based on those snapshots
Time frame between two or more snapshots
Advanced logging analysis reporting

https://pqbadger.darold.net/

Incremental daily/cumulative weekly reports

The right direction, but still

Massive logging necessary

Log file handling
The Idea – Getting sustainable?

Ring-Buffer-like settings in extensions are volatile
Several different retention
Several Views, Tables…but also volatile

How to handle the collection of all that information?

The Idea

putting all in a repository/database
while handling the retention of all information
A good or the actual only(?) Example

Sample collection of

System Catalogue, information_schema

- PG_STAT_STATEMENTS
- PG_STAT_KCACHE
- PG_WAIT_SAMPLING
Server Statistics
SQL query Statistics
Wait Event Statistics
Schema Object Statistics
User Function Statistics
Vacuum-related stats
Cluster settings
Initiated by Andrei Zubkov
Individual Open Source license
https://github.com/zubkov-andrei/pg_profile
Starting Release v0.0.7 (Nov 2019)
Actual Release v4.4 (Feb 2024)
Pure PL/PGsql-based
PG_PROFILE - Architecture

pg_stat_statements
pg_stat_kcache
pg_wait_sampling
...

Historic Repository
- Sample Management Engine
- Report Engine
- Administrative Functions
PG_PROFILE - Architecture
PG_PROFILE - Creating Reports
Extension `dblink` (part of contrib)

Repositories, e.g.

# sudo dnf install pg_profile_16

Direct from github

# curl -LJO https://github.com/zubkov-andrei/pg_profile/releases/download/4.4/pg_profile--4.4.tar.gz
# sudo tar xzf pg_profile--4.4.tar.gz --directory $(pg_config --sharedir)/extension
Create Schema for Repository (optional)

    CREATE SCHEMA profile;

Activate necessary Extensions

    CREATE EXTENSION pg_profile SCHEMA profile;
    CREATE EXTENSION dblink;
Preload Extensions (of your choice!)

Set few recommended Parameters

```
# vi $PGDATA/postgresql.conf
...
shared_preload_libraries = 'pg_stat_statements, pg_wait_sampling, pg_stat_kcache'
...
track_activities = on
track_counts = on
track_io_timing = on
track wal_io_timing = on
track_functions = all
```
Consider extension parameters

```
pg_profile.topn = 20
pg_profile.max_sample_age = 7
pg_profile.track_sample_timings = off
pg_profile.max_query_length = 20000
```

As well for the related extensions, like e.g.

```
pg_stat_statements.max = 10000
pg_stat_statements.track = 'top'
```
Add Server/Database

```
SELECT profile.create_server('core16','host=node0 dbname=postgres port=50160');
```

Other functions

```
profile.drop_server(server name)
profile.enable_server(server name)
profile.disable_server(server name)
profile.show_servers()
...```
Take a sample

```sql
select * from profile.take_sample();
select * from profile.take_sample('core16');
```

Check existing samples

```sql
select * from profile.show_samples();
select * from profile.show_samples('core16');
```
Best Practice Strategy

Frequented 30 Min Samples, starting point
Consider manual created Samples
Baselines

Putting into cron

*/30 * * * * psql -c 'SELECT profile.take_sample()' > /dev/null 2>&1
Tagged Group of Samples
Independent Retention
E.g. for bulk operations, load testings,...

Example handling

```sql
select * from profile.show_baselines();
select * from profile.create_baseline('core16', 'pgbench_run', 70, 71);
```
Standard Report

```bash
psql -Aqtc \
"SELECT profile.get_report('core16',8,9)" \
-o report_8_9.html
```

Diff Report Report

```bash
psql -Aqtc \
"SELECT profile.get_diffreport('core16', 8, 9, 11, 12)" \
-o diff_report_8_9-11_12.html
```
## List of relations

<table>
<thead>
<tr>
<th>Schema</th>
<th>Name</th>
<th>Type</th>
<th>Owner</th>
</tr>
</thead>
<tbody>
<tr>
<td>profile</td>
<td>baselines</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>bl_samples</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>funcs_list</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>import_queries</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>import_queries_version_order</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>indexes_list</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>last_stat_archiver</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>last_stat_cluster</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>last_stat_database</td>
<td>partitioned table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>last_stat_database_srv1</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>last_stat_database_srv2</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>last_stat_database_srv4</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>last_stat_indexes</td>
<td>partitioned table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>last_stat_indexes_srv1</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>last_stat_indexes_srv2</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>last_stat_indexes_srv4</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>last_stat_kcache</td>
<td>partitioned table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>last_stat_kcache_srv1</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>last_stat_kcache_srv2</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>last_stat_kcache_srv4</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>last_stat_statements</td>
<td>partitioned table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>last_stat_statements_srv1</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>last_stat_statements_srv2</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>last_stat_statements_srv4</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>last_stat_tables</td>
<td>partitioned table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>last_stat_tables_srv1</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>last_stat_tables_srv2</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>last_stat_tables_srv4</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>last_stat_tablespaces</td>
<td>partitioned table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>last_stat_tablespaces_srv1</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>last_stat_tablespaces_srv2</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>last_stat_tablespaces_srv4</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>last_stat_user_functions</td>
<td>partitioned table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>last_stat_user_functions_srv1</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>last_stat_user_functions_srv2</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>last_stat_user_functions_srv4</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>last_stat_wal</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>report</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>report_static</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>report_struct</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>roles_list</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>sample_kcache</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>sample_kcache_total</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>sample_settings</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>sample_stat_archiver</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>sample_stat_cluster</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>sample_stat_database</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>sample_stat_indexes</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>sample_stat_indexes_total</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>sample_stat_tables</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>sample_stat_tables_total</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>sample_stat_tablespaces</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>sample_stat_user_func_total</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>sample_stat_user_functions</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>sample_stat_statements</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>sample_statements_srv1</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>sample_statements_total</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>sample_timings</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>samples</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>servers</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>stmt_list</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>tables_list</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>tablespace_list</td>
<td>table</td>
<td>postgres</td>
</tr>
<tr>
<td>profile</td>
<td>wait_sampling_total</td>
<td>table</td>
<td>postgres</td>
</tr>
</tbody>
</table>

(64 rows)
Global Retention Policy

```
pg_profile.max_sample_age
```

Server Retention Policy

```
pg_profile.set_server_max_sample_age()
```
### Data Growth?

```sql
WITH schemas AS ( 
    SELECT
        schemaname as name,
        sum(pg_relation_size(quote_ident(schemaname) || '.' || quote_ident(tablename)))::bigint as size
    FROM pg_tables 
    GROUP BY schema_name
), db AS ( 
    SELECT pg_database_size(current_database()) AS size
) 
SELECT schemas.name, pg_size_pretty(schemas.size) as absolute_size,
       schemas.size::float / (SELECT size FROM db) * 100 as relative_size
FROM schemas;
```

<table>
<thead>
<tr>
<th>name</th>
<th>absolute_size</th>
<th>relative_size</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>41.00 MB</td>
<td>51.55</td>
</tr>
<tr>
<td>pg_catalog</td>
<td>0.50 MB</td>
<td>6.02</td>
</tr>
<tr>
<td>information_schema</td>
<td>0.09 MB</td>
<td>0.10</td>
</tr>
<tr>
<td>profile</td>
<td>18.00 MB</td>
<td>22.04</td>
</tr>
</tbody>
</table>

(4 rows)
Extensibility is a benefit, not a workaround!
Very clean, pragmatic way to handle performance data
Sustainable repository approach
Not exactly the amount like the Oracle packs
  But still a big point for considering Oracle folks on migrations
  But perfect for nearly all common problems
Conclusion – What is missing?

Own Job handling would be nice, but cron is fine!
Availability in DBaaS offerings or contrib
Still room for even more information pieces in reports
The trusted open source data platform for everyone
One data platform for your cloud needs

- Event streaming: Aiven for Apache Kafka® and Kafka® Connect
- Event stream processing: Aiven for Apache Flink®
- Relational databases: Aiven for PostgreSQL®, Aiven for MySQL
- Key-value database: Aiven for Redis®, Aiven for Apache Cassandra®
- Wide column database: Aiven for ClickHouse®, Aiven for M3
- Data warehouse: Aiven for OpenSearch®, Aiven for Grafana®
- Time series database: Aiven for M3
- Search engine: Aiven for OpenSearch®, Aiven for Grafana®
- Key-value database: Aiven for Redis®, Aiven for Apache Cassandra®
- Data visualization: Aiven for Grafana®

STREAM

- Host
  - AWS
  - Google Cloud
  - DigitalOcean
  - Microsoft Azure
  - Bring your own cloud

DEPLOY

- Terraform
- Kubernetes
- REST API
- Aiven CLI
- Aiven Console

ANALYZE

- Datadog
- Prometheus
- AWS CloudWatch
- GCP Monitoring
- MongoDB
- AWS S3
- Couchbase
- Snowflake
- Splunk
- Sumologic
- Debezium
- AWS Pub/Sub
- GCP Storage