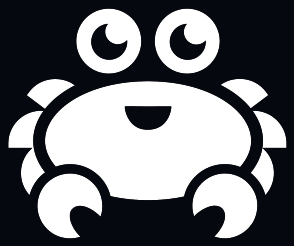


Sustainable Database Performance Profiling in PostgreSQL

25.06.2026

Dirk Krautschick

Swiss PG Day

#whoami

Dirk Krautschick
Senior Solution Architect

With
aiven
since Nov 2023

19 years
DBA, Trainer, Consulting,
Sales Engineering

PostgreSQL, Oracle,...
...and Kafka, Clickhouse, OpenSearch,...

Married, 2 Junior DBAs

mountainbike, swimming,
movies, music,
hifi/homecinema, 8 bit
computing

Before we start…

Disclaimer

Different audience, different perspectives

My experience, my honest opinion

Let‘s stay open minded

Always open for discussions

Before we start…

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

Before we start…

What happened so far…?

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!

Before we start…

Performance Problems

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!

Before we start…

What about monitoring…?

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

Before we start…

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'
```

PG_STAT_STATEMENTS

- 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"
       Column        |        Type       | ...
---------------------+-------------------+-...
 userid              | oid               | ...
 dbid                | oid               | ...
 queryid             | bigint            | ...
 query               | text              | ...
 total_plan_time     | double precision  | ...
…
 calls               | bigint            | ...
 total_exec_time     | double precision  | ...
 min_exec_time       | double precision  | ...
 max_exec_time       | double precision  | ...
 mean_exec_time      | double precision  | ...
 stddev_exec_time    | double precision  | ...
 rows                | bigint            | ...
…
 blk_read_time       | double precision  | ...
 blk_write_time      | double precision  | ...
…
```

Before we start…

PG_STAT_STATEMENTS


```
# 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;
```

short_query	total_exec_time	calls	mean_exec_time	percentage
UPDATE pgbench_branches SET bbalance = bbalance +	7114	1500	5	63
UPDATE pgbench_tellers SET tbalance = tbalance + $	2506	1500	2	22
copy pgbench_accounts from stdin	664	1	664	6
UPDATE pgbench_accounts SET abalance = abalance +	194	1500	0	2
alter table pgbench_accounts add primary key (aid)	193	1	193	2
vacuum analyze pgbench_accounts	138	1	138	1

…

Before we start…

PG_STAT_KCACHE

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"
      Column      |       Type       | Collation | Nullable | Default
------------------+------------------+-----------+----------+---------
 datname          | name             |           |          |
 plan_user_time   | double precision |           |          |
 plan_system_time | double precision |           |          |
 plan_minflts     | numeric          |           |          |
 plan_majflts     | numeric          |           |          |
 plan_nswaps      | numeric          |           |          |
 plan_reads       | numeric          |           |          |
 plan_reads_blks  | numeric          |           |          |
 plan_writes      | numeric          |           |          |
 plan_writes_blks | numeric          |           |          |
…
 plan_nivcsws     | numeric          |           |          |
 exec_user_time   | double precision |           |          |
 exec_system_time | double precision |           |          |
…
 exec_nsignals    | numeric          |           |          |
 exec_nvcsws      | numeric          |           |          |
 exec_nivcsws     | numeric          |           |          |
```

Before we start…

What is still missing…?

Before we start…

Handling WAIT_EVENTS!!!

RECAP

Before we start…

PG_WAIT_SAMPLING

Wait events from pg_stat_activity

Sampled statistics of wait events

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()`

Before we start…

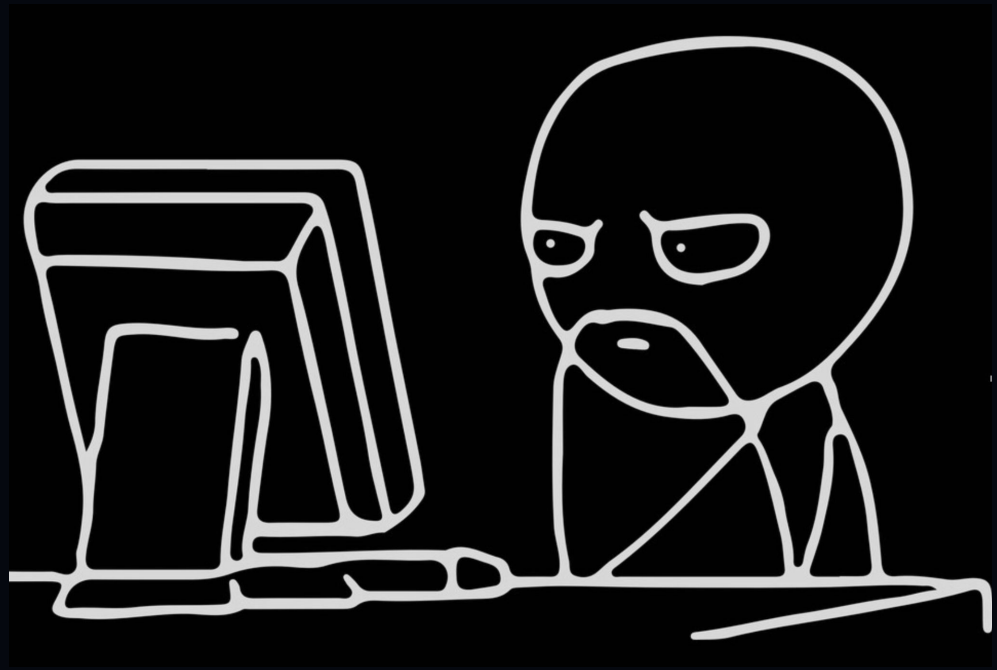
PG_STAT_STATEMENTS


```
postgres=# select * from pg_wait_sampling_profile order by pid, count desc;

pid  | event_type |     event        |     queryid          |  count
-----+------------+------------------+----------------------+---------
1689 | IO         | DataFileWrite    |  2862011717192834034 | 4010499
1685 | IO         | DataFileRead     |  2862011717192834034 | 4010097
1686 | IO         | DataFileSync     | -4888004026240188267 | 4007477
1684 | Activity   | BgWriterHibernate|  2862011717192834034 | 3991477
1683 | Activity   | CheckpointerMain|  1511417639870010300 | 3927957
1684 | Activity   | BgWriterMain     | -4888004026240188267 |   88494
3720 | Client     | ClientRead       | -4888004026240188267 |    2393
1685 | IO         | WALSync          |  6648255685428052402 |      65
3546 | Client     | ClientRead       | -4888004026240188267 |       1
3546 | IO         | DataFileRead     |  2862011717192834034 |       1

...
```

Getting sustainable?



Think outside the box

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”)

Think outside the box

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

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

But hey, there was already…

RECAP

Advanced logging analysis reporting

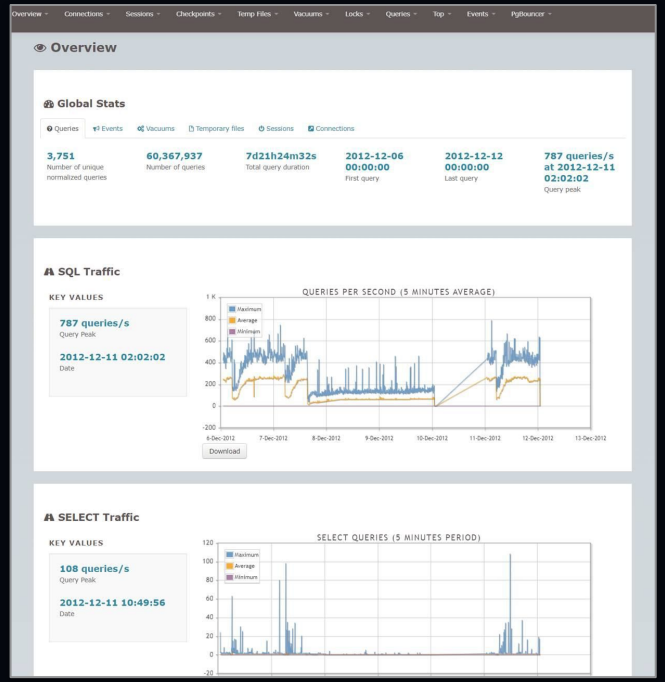
https://pgbadger.darold.net/

Incremental daily/cumulative weekly reports

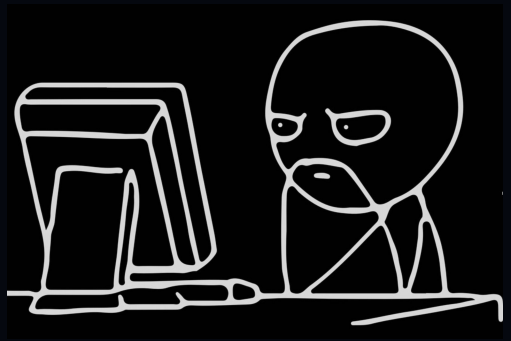
The right direction, but still

Massive logging necessary

Log file handling



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

Developement

Initiated by Andrei Zubkov

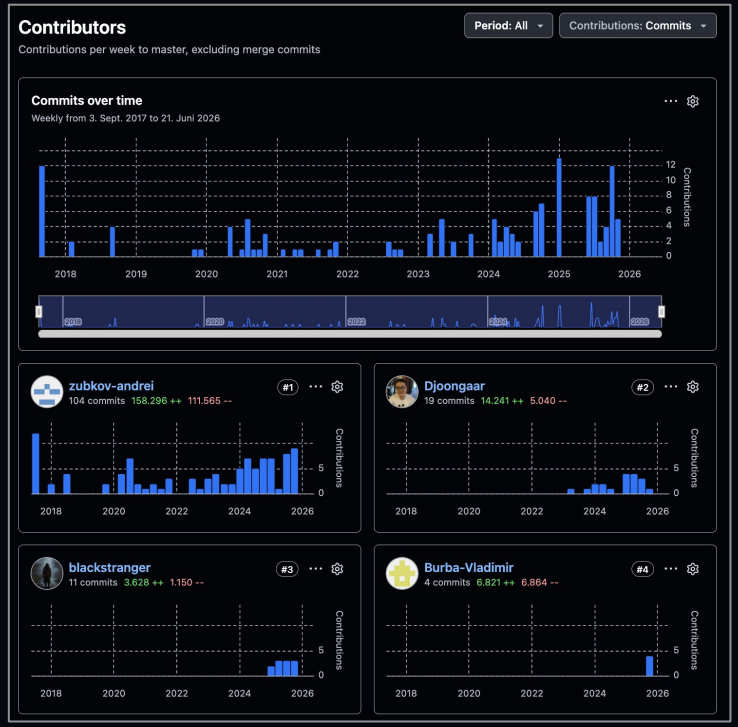
Individual Open Source license

https://github.com/zubkov-andrei/pg_profile

Starting Release v0.0.7 (Nov 2019)

Actual Release v4.11 (Nov 2025)

Pure PL/PGsql-based



Introduction

A good or the actual only(?) Example

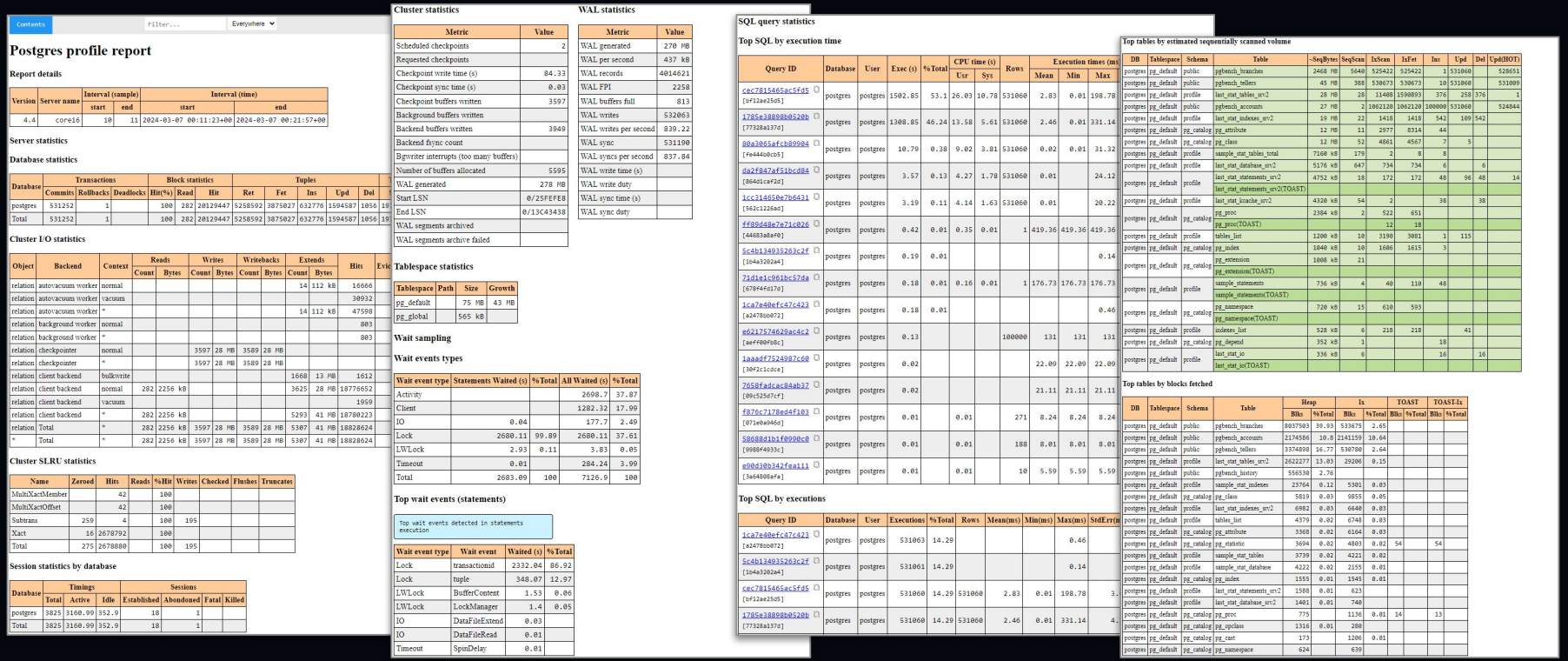
Sample collection of

- System Catalogue, information_schema
- `PG_STAT_STATEMENTS`
- `PG_STAT_KCACHE`
- `PG_WAIT_SAMPLING`

Report Examples

- Server Statistics
- SQL query Statistics
- Wait Event Statistics
- Schema Object Statistics
- User Function Statistics
- Vacuum-related stats
- Cluster settings

Report Examples



Contents

Filter... Everywhere

Postgres profile report

Report details

Version	Server name	Interval (sample)		Interval (time)	
		start	end	start	end
4.4	core16	10	11	2024-03-07 00:11:23+00	2024-03-07 00:21:57+00

Server statistics

Database statistics

Database	Transactions			Block statistics			Tuples					
	Commits	Rollbacks	Deadlocks	Hit(%)	Read	Hit	Ret	Fet	Ins	Upd	Del	
postgres	531252	1		100	282	20129447	5258592	3875027	632776	1594587	1056	19
Total	531252	1		100	282	20129447	5258592	3875027	632776	1594587	1056	19

Cluster I/O statistics

Object	Backend	Context	Reads		Writes		Writebacks		Extends		Hits	Evic
			Count	Bytes	Count	Bytes	Count	Bytes	Count	Bytes		
relation	autovacuum worker	normal							14	112 kB	16666	
relation	autovacuum worker	vacuum									30932	
relation	autovacuum worker	*							14	112 kB	47598	
relation	background worker	normal									803	
relation	background worker	*									803	
relation	checkpointer	normal			3597	28 MB	3589	28 MB				
relation	checkpointer	*			3597	28 MB	3589	28 MB				
relation	client backend	bulkwrite							1668	13 MB	1612	
relation	client backend	normal	282	2256 kB					3625	28 MB	18776652	
relation	client backend	vacuum									1959	
relation	client backend	*	282	2256 kB					5293	41 MB	18780223	
relation	Total	*	282	2256 kB	3597	28 MB	3589	28 MB	5307	41 MB	18828624	
*	Total	*	282	2256 kB	3597	28 MB	3589	28 MB	5307	41 MB	18828624	

Cluster SLRU statistics

Name	Zeroed	Hits	Reads	%Hit	Writes	Checked	Flushes	Truncates
MultiXactMember		42		100				
MultiXactOffset		42		100				
Subtrans	259	4		100	195			
Xact	16	2678792		100				
Total	275	2678880		100	195			

Session statistics by database

Database	Timings			Sessions			
	Total	Active	Idle	Established	Abondoned	Fatal	Killed
postgres	3825	3160.99	352.9	18	1		
Total	3825	3160.99	352.9	18	1		

Cluster statistics

Metric	Value
Scheduled checkpoints	2
Requested checkpoints	
Checkpoint write time (s)	84.33
Checkpoint sync time (s)	0.03
Checkpoint buffers written	3597
Background buffers written	
Backend buffers written	3949
Backend fsync count	
Bgwriter interrupts (too many buffers)	
Number of buffers allocated	5595
WAL generated	278 MB
Start LSN	0/25FEFE8
End LSN	0/13C43438
WAL segments archived	
WAL segments archive failed	

WAL statistics

Metric	Value
WAL generated	270 MB
WAL per second	437 kB
WAL records	4014621
WAL FPI	2258
WAL buffers full	813
WAL writes	532063
WAL writes per second	839.22
WAL sync	531190
WAL syncs per second	837.84
WAL write time (s)	
WAL write duty	
WAL sync time (s)	
WAL sync duty	

Tablespace statistics

Tablespace	Path	Size	Growth
pg_default		75 MB	43 MB
pg_global		565 kB	

Wait sampling

Wait events types

Wait event type	Statements Waited (s)	%Total	All Waited (s)	%Total
Activity			2698.7	37.87
Client			1282.32	17.99
IO	0.04		177.7	2.49
Lock	2680.11	99.89	2680.11	37.61
LWLock	2.93	0.11	3.83	0.05
Timeout	0.01		284.24	3.99
Total	2683.09	100	7126.9	100

Top wait events (statements)

Top wait events detected in statements execution

Wait event type	Wait event	Waited (s)	%Total
Lock	transactionid	2332.04	86.92
Lock	tuple	348.07	12.97
LWLock	BufferContent	1.53	0.06
LWLock	LockManager	1.4	0.05
IO	DataFileExtend	0.03	
IO	DataFileRead	0.01	
Timeout	SpinDelay	0.01	

SQL query statistics

Top SQL by execution time

Query ID	Database	User	Exec (s)	%Total	CPU time (s)		Rows	Execution times (ms)		
					Usr	Sys		Mean	Min	Max
cec7815465ac5fd5 [bf12ae25d5]	postgres	postgres	1502.85	53.1	26.03	10.78	531060	2.83	0.01	198.78
1785a38898b0520b [77328a137e]	postgres	postgres	1308.85	46.24	13.58	5.61	531060	2.46	0.01	331.14
80a3065afcb89904 [fe444b0cb5]	postgres	postgres	10.79	0.38	9.02	3.81	531060	0.02	0.01	31.32
da2f847af51bcd84 [864d1caf2d]	postgres	postgres	3.57	0.13	4.27	1.78	531060	0.01		24.12
1cc3146502e7b643 [562c1226ad]	postgres	postgres	3.19	0.11	4.14	1.63	531060	0.01		20.22
ff89d48e7e71c026 [44683a8af0]	postgres	postgres	0.42	0.01	0.35	0.01	1	419.36	419.36	419.36
5c4b134935263c2f [1b4a3202e4]	postgres	postgres	0.19	0.01						0.14
71d1e1c961bc57da [678f4fd17d]	postgres	postgres	0.18	0.01	0.16	0.01	1	176.73	176.73	176.73
1ca7e40efc47c423 [a2478bb072]	postgres	postgres	0.18	0.01						0.46
e6217574629ac4c2 [aef00f08c]	postgres	postgres	0.13				100000	131	131	131
1aaadf7524987c60 [30f2c1cdce]	postgres	postgres	0.02					22.09	22.09	22.09
7658fadcac84ab37 [09c52567cf]	postgres	postgres	0.02					21.11	21.11	21.11
f876c7178ed4f103 [071e0a946d]	postgres	postgres	0.01		0.01		271	8.24	8.24	8.24
58688d1b1f0090c0 [9988f4933c]	postgres	postgres	0.01		0.01		188	8.01	8.01	8.01
e90d30b342fea111 [3e64808afa]	postgres	postgres	0.01		0.01		10	5.59	5.59	5.59

Top SQL by executions

Query ID	Database	User	Executions	%Total	Rows	Mean(ms)	Min(ms)	Max(ms)	StdErr(ms)
1ca7e40efc47c423 [a2478bb072]	postgres	postgres	531063	14.29				0.46	
5c4b134935263c2f [1b4a3202e4]	postgres	postgres	531061	14.29				0.14	
cec7815465ac5fd5 [bf12ae25d5]	postgres	postgres	531060	14.29	531060	2.83	0.01	198.78	3.1
1785a38898b0520b [77328a137e]	postgres	postgres	531060	14.29	531060	2.46	0.01	331.14	4.2

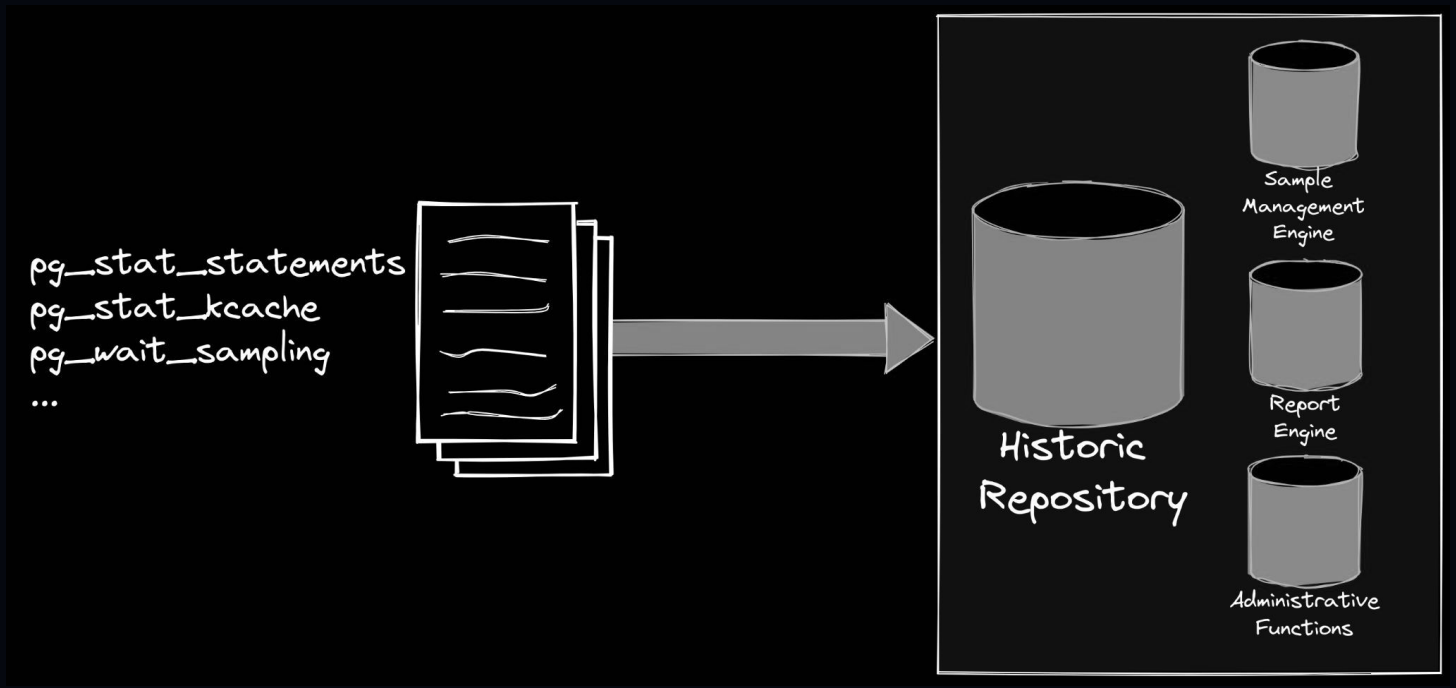
Top tables by estimated sequentially scanned volume

DB	Tablespace	Schema	Table	~SeqBytes	SeqScan	IxScan	IxFet	Ins	Upd	Del	Upd(HOT)
postgres	pg_default	public	pgbench_branches	2468 MB	5640	525422	525422	1	531060		528651
postgres	pg_default	public	pgbench_tellers	45 MB	388	530673	530673	10	531060		531009
postgres	pg_default	profile	last_stat_tables_srv2	28 MB	28	11408	1598893	376	258	376	1
postgres	pg_default	public	pgbench_accounts	27 MB	2	1062120	1062120	100000	531060		524844
postgres	pg_default	profile	last_stat_indexes_srv2	19 MB	22	1418	1418	542	109	542	
postgres	pg_default	pg_catalog	pg_attribute	12 MB	11	2977	8314	44			
postgres	pg_default	pg_catalog	pg_class	12 MB	52	4861	4567	7	5		
postgres	pg_default	profile	sample_stat_tables_total	7168 kB	179	2	8	8			
postgres	pg_default	profile	last_stat_database_srv2	5176 kB	647	734	734	6		6	
postgres	pg_default	profile	last_stat_statements_srv2	4752 kB	18	172	172	48	96	48	14
			last_stat_statements_srv2(TOAST)								
postgres	pg_default	profile	last_stat_kcache_srv2	4320 kB	54	2		38		38	
postgres	pg_default	pg_catalog	pg_proc	2384 kB	2	522	651				
			pg_proc(TOAST)			12	18				
postgres	pg_default	profile	tables_list	1200 kB	10	3190	3081	1	115		
postgres	pg_default	pg_catalog	pg_index	1040 kB	10	1606	1615	3			
postgres	pg_default	pg_catalog	pg_extension	1008 kB	21						
			pg_extension(TOAST)								
postgres	pg_default	profile	sample_statements	736 kB	4	40	110	48			
			sample_statements(TOAST)								
postgres	pg_default	pg_catalog	pg_namespace	720 kB	15	610	593				
			pg_namespace(TOAST)								
postgres	pg_default	profile	indexes_list	528 kB	6	218	218		41		
postgres	pg_default	pg_catalog	pg_depend	352 kB	1			18			
postgres	pg_default	profile	last_stat_io	336 kB	6			16		16	
			last_stat_io(TOAST)								

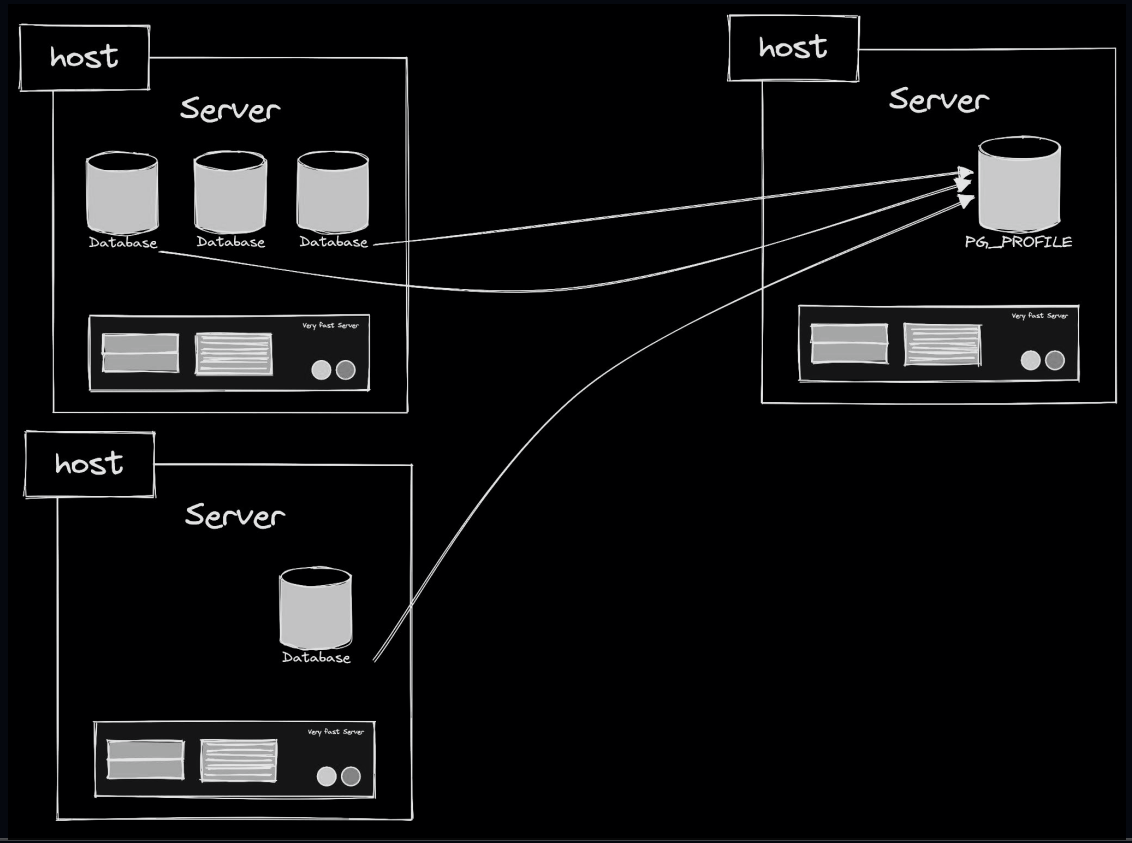
Top tables by blocks fetched

DB	Tablespace	Schema	Table	Heap		Ix		TOAST		TOAST-Ix	
				Blks	%Total	Blks	%Total	Blks	%Total	Blks	%Total
postgres	pg_default	public	pgbench_branches	8037503	39.93	533675	2.65				
postgres	pg_default	public	pgbench_accounts	2174586	10.8	2141159	10.64				
postgres	pg_default	public	pgbench_tellers	3374898	16.77	530780	2.64				
postgres	pg_default	profile	last_stat_tables_srv2	2622277	13.03	29206	0.15				
postgres	pg_default	public	pgbench_history	556530	2.76						
postgres	pg_default	profile	sample_stat_indexes	23764	0.12	5301	0.03				
postgres	pg_default	pg_catalog	pg_class	5819	0.03	9855	0.05				
postgres	pg_default	profile	last_stat_indexes_srv2	6982	0.03	6640	0.03				
postgres	pg_default	profile	tables_list	4379	0.02	6748	0.03				
postgres	pg_default	pg_catalog	pg_attribute	3368	0.02	6164	0.03				
postgres	pg_default	pg_catalog	pg_statistic	3694	0.02	4803	0.02	54		54	
postgres	pg_default	profile	sample_stat_tables	3739	0.02	4221	0.02				
postgres	pg_default	profile	sample_stat_database	4222	0.02	2155	0.01				
postgres	pg_default	pg_catalog	pg_index	1555	0.01	1545	0.01				
postgres	pg_default	profile	last_stat_statements_srv2	1588	0.01	623					
postgres	pg_default	profile	last_stat_database_srv2	1401	0.01	740					
postgres	pg_default	pg_catalog	pg_proc	775		1136	0.01	14		13	
postgres	pg_default	pg_catalog	pg_opclass	1316	0.01	280					
postgres	pg_default	pg_catalog	pg_cast	173		1206	0.01				
postgres	pg_default	pg_catalog	pg_namespace	624		639					

Architecture



Architecture



Prerequisite and Setup

Extension `dblink` (part of contrib)

Repositories, e.g.

```
# sudo dnf install pg_profile_18
```

Direct from github

```
# curl -LJO
https://github.com/zubkov-andrei/pg_profile/releases/download/4.11/pg_profile--4.11.tar.gz
# sudo tar xzf pg_profile--4.11.tar.gz --directory $(pg_config --sharedir)/extension
```

Prerequisites

Create Schema for Repository (optional)

```
CREATE SCHEMA profile;
```

Activate necessary Extensions

```
CREATE EXTENSION pg_profile SCHEMA profile;
CREATE EXTENSION dblink;
```

Prerequisites on source DBs

Preload Extensions (of your choice!)

Set few recommended Parameters

```
# vi $PGDATA/postgresql.conf
…
shared_preload_libraries = 'pg_stat_statements',...
…
track_activities = on
track_counts = on
track_io_timing = on
track_wal_io_timing = on
track_functions = all
```

Configuration

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 = 'all'
```

Adding Clusters/Servers for Collection

Add Server/Database

```
SELECT profile.create_server('cluster18','host=node0,dbname=postgres
                             port=5432');
```

Other functions

```
profile.drop_server(server name)
profile.enable_server(server name)
profile.disable_server(server name)
profile.show_servers()
…
```

Collecting Data

Take a sample

```
select * from profile.take_sample();
select * from profile.take_sample('core18');
```

Check existing samples

```
select * from profile.show_samples();
select * from profile.show_samples('core18');
```

Baselines

Tagged Group of Samples

Independent Retention
e.g. for bulk operations, load testings,...

Example handling

```
select * from profile.show_baselines();
select * from profile.create_baseline('cluster18', 'pgbench_run' ,70,71);
```

Collecting Data

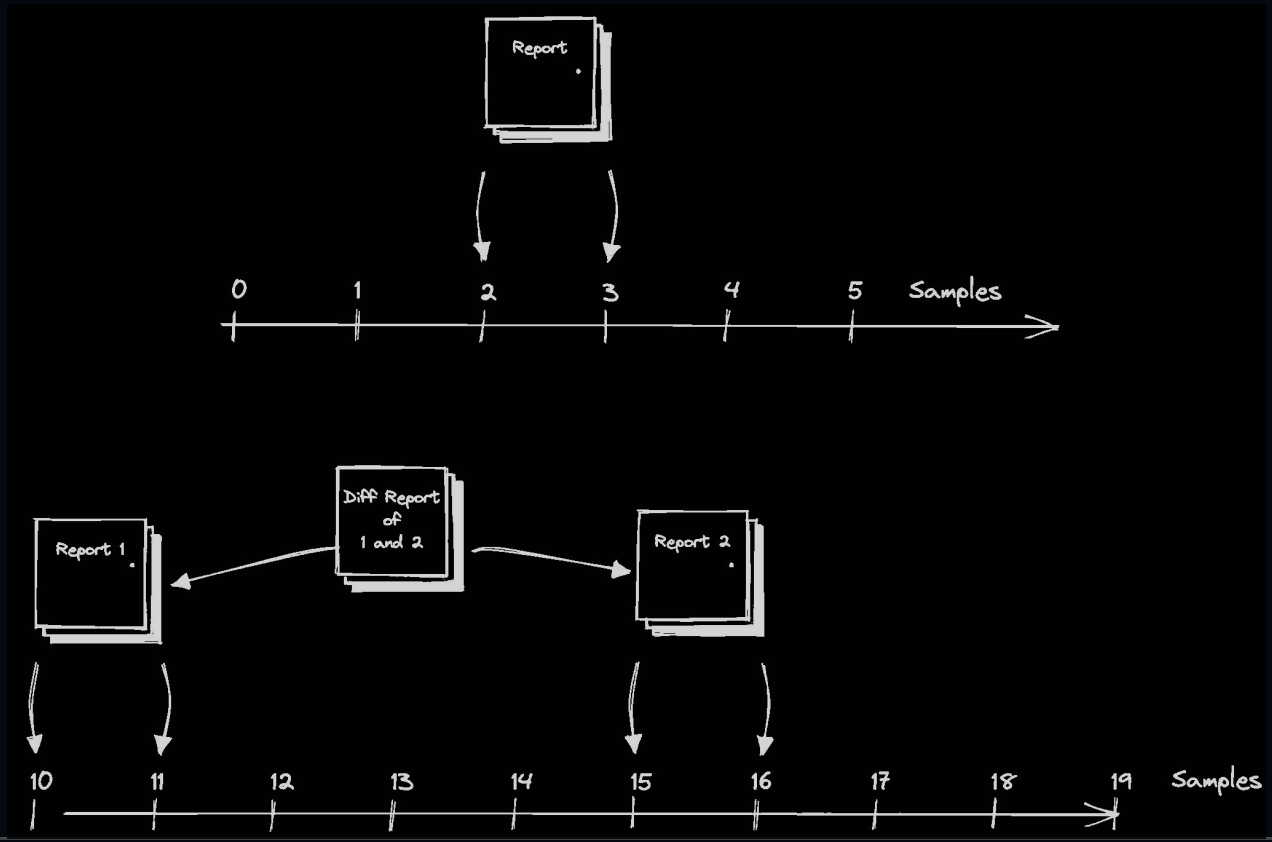
Best Practice Strategy

- 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
```

Creating Reports



Creating Reports

Standard Report

```
psql -Aqtc \
"SELECT profile.get_report('core18',8, 9)" \
-o report_8_9.html
```

Diff Report Report

```
psql -Aqtc \
"SELECT profile.get_diffreport('core18', 8, 9, 11, 12)" \
-o diff_report_8_9-11_12.html
```

A look into the repository schema

```
# \dt profile.*
List of relations
Schema | Name | Type | Owner
---------+-------------------------------+-------------------+----------
profile | baselines | table | postgres
profile | bl_samples | table | postgres
profile | funcs_list | table | postgres
profile | import_queries | table | postgres
profile | import_queries_version_order | table | postgres
profile | indexes_list | table | postgres
profile | last_stat_archiver | table | postgres
profile | last_stat_cluster | table | postgres
profile | last_stat_database | partitioned table | postgres
profile | last_stat_database_srv1 | table | postgres
profile | last_stat_database_srv2 | table | postgres
profile | last_stat_database_srv4 | table | postgres
profile | last_stat_indexes | partitioned table | postgres
profile | last_stat_indexes_srv1 | table | postgres
profile | last_stat_indexes_srv2 | table | postgres
profile | last_stat_indexes_srv4 | table | postgres
profile | last_stat_kcache | partitioned table | postgres
profile | last_stat_kcache_srv1 | table | postgres
profile | last_stat_kcache_srv2 | table | postgres
profile | last_stat_kcache_srv4 | table | postgres
profile | last_stat_statements | partitioned table | postgres
profile | last_stat_statements_srv1 | table | postgres
profile | last_stat_statements_srv2 | table | postgres
profile | last_stat_statements_srv4 | table | postgres
profile | last_stat_tables | partitioned table | postgres
profile | last_stat_tables_srv1 | table | postgres
profile | last_stat_tables_srv2 | table | postgres
profile | last_stat_tables_srv4 | table | postgres
profile | last_stat_tablespaces | partitioned table | postgres
profile | last_stat_tablespaces_srv1 | table | postgres
profile | last_stat_tablespaces_srv2 | table | postgres
profile | last_stat_tablespaces_srv4 | table | postgres
…
…
profile | last_stat_user_functions | partitioned table | postgres
profile | last_stat_user_functions_srv1 | table | postgres
profile | last_stat_user_functions_srv2 | table | postgres
profile | last_stat_user_functions_srv4 | table | postgres
profile | last_stat_wal | table | postgres
profile | report | table | postgres
profile | report_static | table | postgres
profile | report_struct | table | postgres
profile | roles_list | table | postgres
profile | sample_kcache | table | postgres
profile | sample_kcache_total | table | postgres
profile | sample_settings | table | postgres
profile | sample_stat_archiver | table | postgres
profile | sample_stat_cluster | table | postgres
profile | sample_stat_database | table | postgres
profile | sample_stat_indexes | table | postgres
profile | sample_stat_indexes_total | table | postgres
profile | sample_stat_tables | table | postgres
profile | sample_stat_tables_total | table | postgres
profile | sample_stat_tablespaces | table | postgres
profile | sample_stat_user_func_total | table | postgres
profile | sample_stat_user_functions | table | postgres
profile | sample_stat_wal | table | postgres
profile | sample_statements | table | postgres
profile | sample_statements_total | table | postgres
profile | sample_timings | table | postgres
profile | samples | table | postgres
profile | servers | table | postgres
profile | stmt_list | table | postgres
profile | tables_list | table | postgres
profile | tablespaces_list | table | postgres
profile | wait_sampling_total | table | postgres
(64 rows)
```

A look into the repository schema

Global Retention Policy

```
pg_profile.max_sample_age
```

Server Retention Policy

```
pg_profile.set_server_max_sample_age()
```

A look into the repository schema

Data Growth?

```
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;

      name                | absolute_size | relative_size
      --------------------+---------------+---------------------
      public              | 41.00 MB      | 51.55
      pg_catalog          | 0.50 MB       | 6.02
      information_schema  | 0.09 MB       | 0.10
      profile             | 18.00 MB      | 22.04
      (4 rows)
```

Conclusion - My 2 Cents

Own Job handling would be nice, but cron is fine!

Availability in DBaaS offerings or contrib
…but usage always possible with self maintained repo db

Still room for even more information pieces in reports

Conclusion - What is missing?

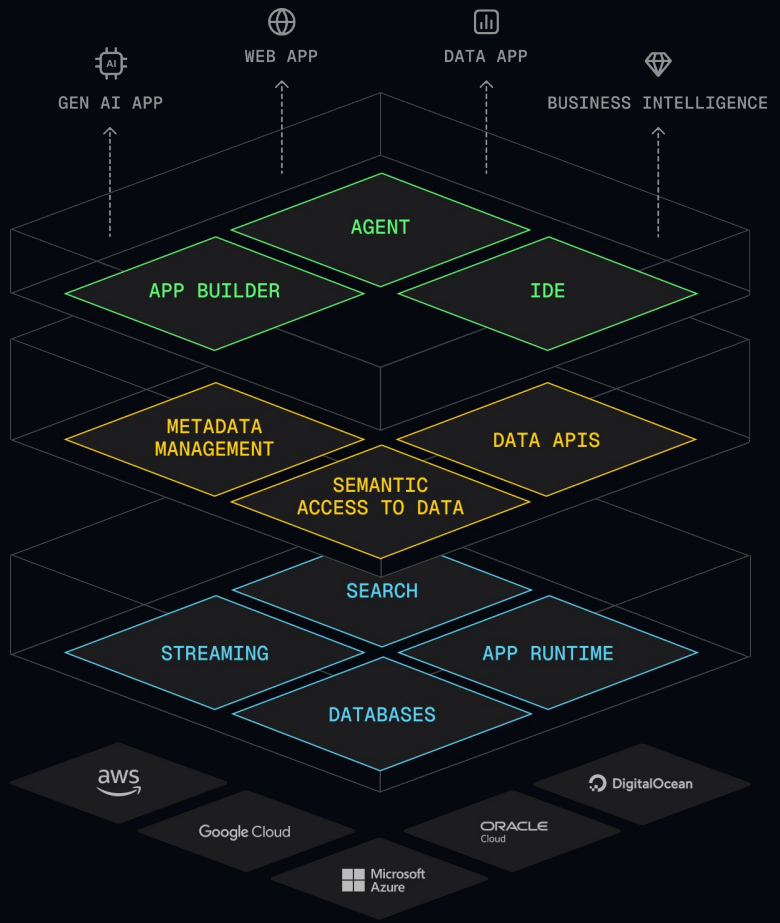
Extensibility is a benefit, not a workaround!
Very tidy, pragmatic way to handle performance data
Sustainable repository approach
Not exactly the amount like the Oracle packs
But still a big point for considering Oracle folks on migrations
But perfect for nearly all common problems

Thanks!

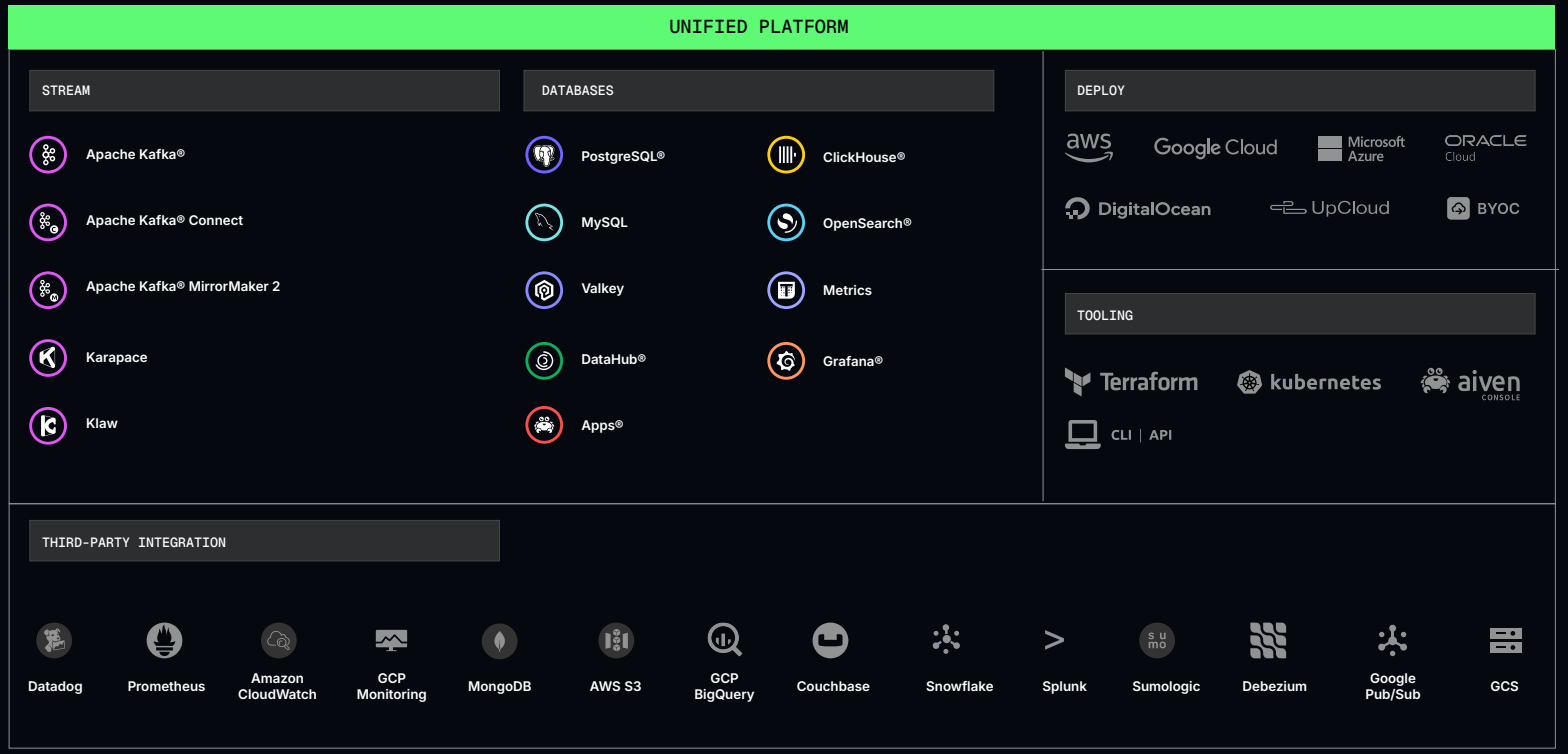
Questions?

YOUR AI-READY OPEN SOURCE DATA PLATFORM

Streaming | Database Optimization | Analytics | Search | Data Warehousing | In-Memory Caching



AIVEN PLATFORM



TRUSTED BY OVER 1000 CUSTOMERS WORLDWIDE