



# Elevate your DBA toolset with pg collector

**Mohamed Ali**

Senior database engineer

# Agenda :

- Who am I ?
- What is PG Collector ?
- Why did I create PG Collector ?
- Who can use PG Collector ?
- What is PG Collector Versioning Policy ?
- How to download PG Collector that match your PostgreSQL major version ?
- How to run PG Collector script ( `pg_collector.sql` )
- PG Collector's Report name & location
- How to check PG Collector version ?
- Use Cases & Scenarios
- How to edit PG Collector script ?
- Q&A

# Who am I?

Senior database engineer at Amazon Web Services.

I has been working for over 14 years with databases ( Oracle and PostgreSQL ) and for over 7 years at AWS.

At AWS, I works on PostgreSQL engine including Amazon RDS PostgreSQL and Aurora PostgreSQL. i contributed to many Database Engineering projects including the design and integration between Babelfish and PostgreSQL, Aurora PostgreSQL serverless V2, and Aurora PostgreSQL Limitless database .

I am the author of two open source tools pg-collector ( <https://github.com/awslabs/pg-collector> ) and PG Counter Metrics ( PGCM <https://github.com/awslabs/pg-counter-metrics> )

# What is PG Collector ?

PG Collector for [Postgresql](#) is a sql script that gathers valuable database information and presents it in a consolidated HTML file which provides a convenient way to view and navigate between different sections of the report.

PG Collector is safe to run on production environments and does not create any database objects to produce the output.

PG Collector published Under [MIT-0 license](#) part of [awslabs](#) Github .

<https://github.com/awslabs/pg-collector>



# Software Catalogue - Administration/development tools

<https://www.postgresql.org/download/products/1-administrationdevelopment-tools/>

## PG Collector

Description	License	Pricing	Publisher
<p>PG Collector for PostgreSQL is a SQL script that gathers valuable database information and presents it in a consolidated HTML file which provides a convenient way to view and navigate between different sections of the report.</p> <p>PG Collector is safe to run on production environments and does not create any database objects to produce the output.</p>	Open source		<a href="#">Amazon Web Services</a> <a href="#">View</a>

With PG Collector an operator gains insights on various aspects of the database, such as:

- Database size
- Configuration parameters
- Installed extensions
- Vacuum & Statistics
- Unused Indexes & invalid indexes
- Users & Roles Info
- Toast tables mapping
- Database schemas
- Bloat
- Tablespaces Info
- Memory setting
- Tables and Indexes size and info
- Transaction ID TXID (Wraparound) & Multixact ID MXID
- Replication slots
- Public schema info
- Unlogged tables.
- and more

# PG Collector report header

## PG COLLECTOR V2.9

For more information about PG Collector, visit the [project github repository](#)

### DB INFO

PG Host Name / PG RDS ENDPOINT: `pgbench.cyzxyxpyxw.us-west-2.rds.amazonaws.com`  
 RDS-12.5 Primary/writer DB (Read write)

Date	DB_START_DATE	UP_TIME	DB_connected	user_name	DB_PORT	DB_Version	block_size
2021-08-02 18:13:48.107489+00	2021-06-10 20:12:52.071047+00	52 days 22:00:56.036442	postgres	postgres	8200	PostgreSQL 12.5 on x86_64-pc-linux-gnu, compiled by gcc (GCC) 4.8.5 20150623 (Red Hat 4.8.5-11), 64-bit	8192

List of databases

Name	Owner	Encoding	Collate	Ctype	Access privileges	Size	Tablespace	Description
pg2	postgres	UTF8	en_US.UTF-8	en_US.UTF-8		8265 kB	pg_default	
postgres	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	=Tc/postgres postgres=CtC/postgres pwd_pgcm=c/postgres test=c/postgres	39 GB	pg_default	default administrative connection database
rdsadmin	rdsadmin	UTF8	en_US.UTF-8	en_US.UTF-8	rdsadmin=CtC/rdsadmin	No Access	pg_default	
template0	rdsadmin	UTF8	en_US.UTF-8	en_US.UTF-8	=c/rdsadmin rdsadmin=CtC/rdsadmin	8137 kB	pg_default	unmodifiable empty database
template1	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	postgres=CtC/postgres =c/postgres	8121 kB	pg_default	default template for new databases

database_name	database_is_template	database_allow_connections	database_connection_limit	datlastsysoid	datfrozensid	datminmxid
postgres	f	t	-1	14313	48841645	130
pg2	f	t	-1	14313	549	1
rdsadmin	f	t	-1	14313	549	1
template1	t	t	-1	14313	549	1
template0	t	f	-1	14313	2716	1

### INFO

Database size	DB parameters	Transaction ID TXID (Wraparound)	Table Size
<a href="#">Index Size</a>	<a href="#">Vacuum &amp; Statistics</a>	<a href="#">Extensions</a>	<a href="#">Memory setting</a>
<a href="#">pg_stat_statements extension</a>	<a href="#">Users &amp; Roles Info</a>	<a href="#">schema info</a>	<a href="#">Tablespaces Info</a>
<a href="#">Table Access Profile</a>	<a href="#">Unused indexes</a>	<a href="#">Index Access Profile</a>	<a href="#">Fragmentation (Bloat)</a>
<a href="#">Toast Tables Mapping</a>	<a href="#">Replication</a>	<a href="#">Sessions/Connections Info</a>	<a href="#">Orphaned prepare transactions</a>
<a href="#">PK or FK using numeric or integer data type</a>	<a href="#">public Schema</a>	<a href="#">invalid indexes</a>	<a href="#">Default access privileges</a>
<a href="#">pgaudit extension</a>	<a href="#">Unlogged Tables</a>	<a href="#">Access privileges</a>	<a href="#">ssl</a>
<a href="#">Background processes</a>	<a href="#">Multixact ID MXID (Wraparound)</a>	<a href="#">Temp tables</a>	<a href="#">Large objects</a>
<a href="#">Partition tables</a>	<a href="#">pg_shdepend</a>	<a href="#">FK without index</a>	<a href="#">sequences</a>
<a href="#">pg_hba.conf</a>	<a href="#">Duplicate indexes</a>	<a href="#">Functions statistics</a>	<a href="#">DB Load</a>
<a href="#">Triggers</a>	<a href="#">pg_config</a>	*****	*****

## INFO

<u>Database size</u>	<u>DB parameters</u>	<u>Transaction ID TXID (Wraparound)</u>	<u>Table Size</u>
<u>Index Size</u>	<u>Vacuum &amp; Statistics</u>	<u>Extensions</u>	<u>Memory setting</u>
<u>pg_stat_statements extension</u>	<u>Users &amp; Roles Info</u>	<u>schema Info</u>	<u>Tablespaces Info</u>
<u>Table Access Profile</u>	<u>Unused Indexes</u>	<u>Index Access Profile</u>	<u>Fragmentation (Bloat)</u>
<u>Toast Tables Mapping</u>	<u>Replication</u>	<u>Sessions/Connections Info</u>	<u>Orphaned prepare transactions</u>
<u>PK or FK using numeric or integer data type</u>	<u>public Schema</u>	<u>invalid indexes</u>	<u>Default access privileges</u>
<u>pgaudit extension</u>	<u>Unlogged Tables</u>	<u>Access privileges</u>	<u>ssl</u>
<u>Background processes</u>	<u>Multixact ID MXID (Wraparound)</u>	<u>Temp tables</u>	<u>Large objects</u>
<u>Partition tables</u>	<u>pg_shdepend</u>	<u>FK without index</u>	<u>sequences</u>
<u>pg_hba.conf</u>	<u>Duplicate indexes</u>	<u>Functions statistics</u>	<u>DB Load</u>
<u>Triggers</u>	<u>pg_config</u>	<u>*****</u>	<u>*****</u>



# *Example of PG Collector report*

- [http://pg-collector.s3-website-us-west-2.amazonaws.com/pg\\_collector\\_postgres-2021-08-02\\_181348.html](http://pg-collector.s3-website-us-west-2.amazonaws.com/pg_collector_postgres-2021-08-02_181348.html)



# Why did I created PG Collector ?

- Because I found that the time needed to collect data about the issue is more than time needed to identify the issue.

Reduce your **MTR** (Mean Time to Repair)



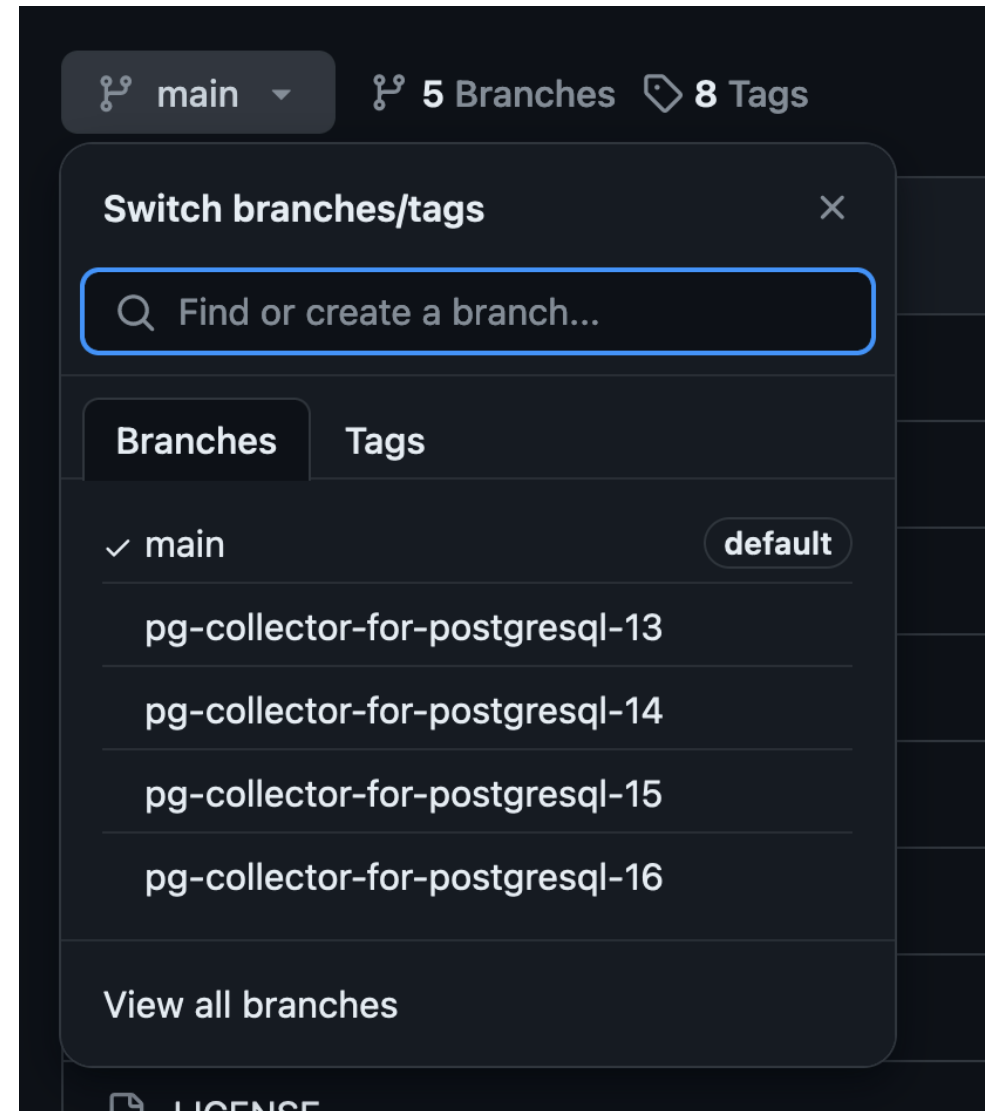
**BEST OPPORTUNITY TO REALIZE EFFICIENCIES!**

# *Who can use PG Collector ?*

- Any Database user wants to check his PostgreSQL Database:
- Developers
- Support engineers
- database administrators

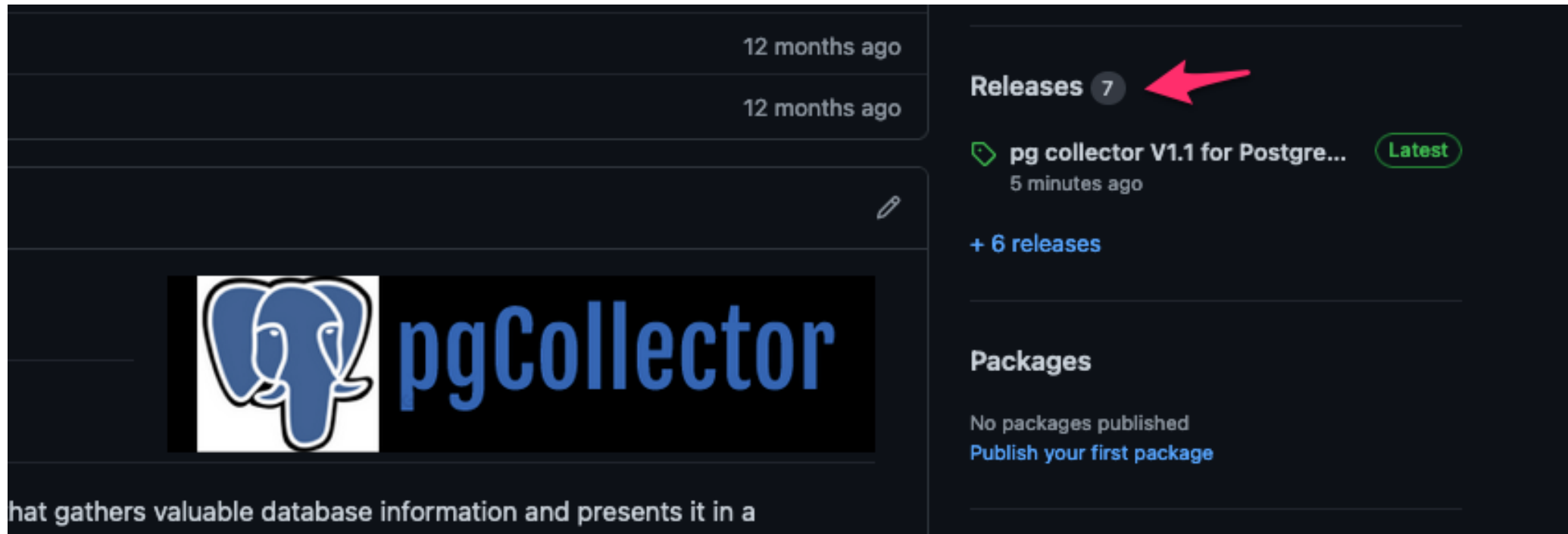
# What is PG Collector Versioning Policy ?

- Starting from PostgreSQL 13, PG Collector will have a dedicated script for each PostgreSQL major version.
- Each PG Collector major version will have its own branch and the main branch will be for PG Collector that supports PostgreSQL 12 and older versions.



# How to download PG Collector that match your PostgreSQL major version ?

- From [pg-collector releases](https://github.com/awslabs/pg-collector/releases) <https://github.com/awslabs/pg-collector/releases>



12 months ago

12 months ago

Releases 7

pg collector V1.1 for Postgre... Latest

5 minutes ago

+ 6 releases

Packages

No packages published

Publish your first package

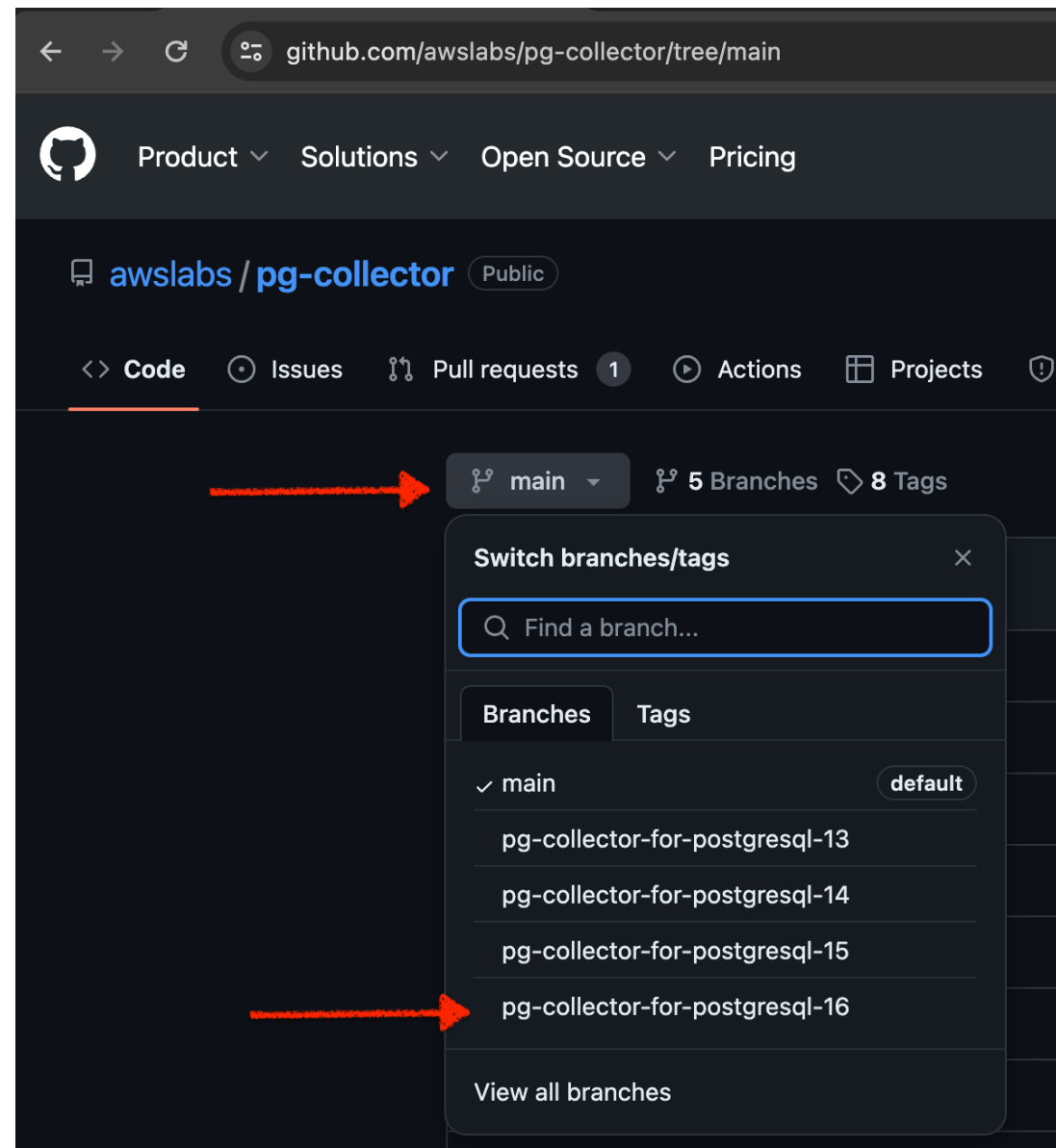
hat gathers valuable database information and presents it in a

# Select the PG Collector version that matches your PostgreSQL major version .

The screenshot shows the GitHub Releases page for the 'pg-collector' repository. It displays two recent releases:

- pg collector V1.1 for PostgreSQL 13** (Latest):
  - 8 minutes ago by mmohali
  - Changes: 1- Add logical\_decoding\_work\_mem parameter to Replication Parameters; 2- Add Replication Slot wal status to Replication section.
  - Assets: Source code (zip) and Source code (tar.gz), both from 16 hours ago.
- pg collector V1.1 for PostgreSQL 14**:
  - 6 minutes ago by mmohali
  - Changes: 1- Add new section for COPY command progress; 2- Add pg\_stat\_statements\_info View to pg\_stat\_statements\_extension section; 3- Add vacuum\_failsafe\_age and vacuum\_multixact\_failsafe\_age parameters to vacuum and Statistics section; 4- Add pg\_stat\_wal veiw to DB load section inder pg\_stat\_\* views; 5- Add pg\_stat\_replication\_slots View to Replication section; 6- Add logical\_decoding\_work\_mem parameter to Replication Parameters; 7- Add n\_ins\_since\_vacuum and n\_tup\_ins columns to pg\_stat\_all\_tables's queries in vacuum and Statistics section; 8- Add Sessions statistics ( session\_time,active\_time,idle\_in\_transaction\_time,sessions,sessions\_abandoned,sessions\_fatal,sessions\_killed ) to Sessions/Connections Info section; 9- Add Replication Slot wal status to Replication section.
  - Assets: Source code (zip) and Source code (tar.gz), both from yesterday. A red arrow points to the 'Source code (zip)' asset.

- Or select the branch that match your PostgreSQL version



# How to run PG Collector script ( *pg\_collector.sql* )

- 1- you need [psql](#) to be able to connect to the postgresql DB and run the pg\_collector.sql script
- 2- Download pg\_collector.sql in your laptop or the host that want to access the database from
- 3- login to the database using psql

```
psql -h [hostname or RDS endpoint] -p [Port] -d [Database name ] -U [user name]
```

- 4- run the pg\_collector.sql script

```
\i pg_collector.sql
```

or use -f option in psql

```
psql -h [hostname or RDS endpoint] -p [Port] -d [Database name ] -U [user name] -f  
pg_collector.sql
```



```
[mohamed@mydevhost ~]$ export PGHOST=testdb.us-east-1.rds.amazonaws.com
[mohamed@mydevhost ~]$ export PGPORT=5432
[mohamed@mydevhost ~]$ export PGUSER=postgres
[mohamed@mydevhost ~]$ export PGPASSWORD=postgres
[mohamed@mydevhost ~]$ export PGDATABASE=postgres
[mohamed@mydevhost ~]$ psql
psql (13.4, server 15.6)
WARNING: psql major version 13, server major version 15.
         Some psql features might not work.
Type "help" for help.

postgres=>
postgres=> \i /tmp/pg_collector.sql
Output format is html.
Report name and location: /tmp/pg_collector_postgres-2024-04-15_165704.html
postgres=> \q
[mohamed@mydevhost ~]$
```

# ***PG Collector's Report name & location***

- PG Collector script will generate HTML file using the following naming convention
- `pg_colletcor_[DB Name]-[timestamp].html`
- [DB Name] : is the database name that you are connected to.
- Example : `pg_collector_testdb-2020-10-10_030920.html`
- Report location:
- PG Collector script will generate HTML file under /tmp directory

# How to check PG Collector version ?

- from the PG Collector report header



The screenshot shows the header of a PG Collector report. At the top, there is a yellow bar with the text "PG COLLECTOR V1.1 for PostgreSQL 13". Below this bar, there is a link: "For more information about PG Collector, visit the [project github repository](#)". Underneath the link, the text "DB INFO" is displayed in a bold, black font, followed by a horizontal line.

- from the PG Collector script header

```
-- +-----+
-- |  -- Script Name: pg_collector.sql
-- |  -- Author : Mohamed Ali
-- |  -- Create Date : 16 SEPT 2019
-- |  -- Description : Script to Collect PostgreSQL Database Informations
-- |                    and generate HTML Report
-- |  -- version : V1.1 for PostgreSQL 13
-- |  -- Changelog : https://github.com/awslabs/pg-collector/blob/pg-collector-for-postgresql-13/CHANGELOG.md
-- | Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
-- | SPDX-License-Identifier: MIT-0
-- +-----+
```

# Use Cases & Scenarios

[\[Top\]](#)

## Orphaned prepared transactions

---

▼ Details

gid	prepared	owner	database	xmin
-----	----------	-------	----------	------

Note:

During two-phase commit, a distributed transaction is first prepared with the PREPARE statement and then committed with the COMMIT PREPARED statement

Once a transaction has been prepared, it is kept hanging around until it is committed or aborted. It

even has to survive a server restart! Normally, transactions don't remain in the prepared state for long,

but sometimes things go wrong and a prepared transaction has to be removed manually by an administrator.

any Orphaned prepared transactions will prevent the VACUUM from removing the dead rows

EXAMPLE: DETAIL: 50000 dead row versions cannot be removed yet, oldest xmin: 22300

Use the ROLLBACK PREPARED transaction\_id SQL statement to remove prepared transactions

[\[Top\]](#)

# Replication

▼ Details

Active replication slots order by age\_xmin:

slot_name	plugin	slot_type	datoid	database	temporary	active	active_pid	xmin	catalog_xmin	restart_lsn	confirmed_flush_lsn	age_xmin	age_catalog_xmi
-----------	--------	-----------	--------	----------	-----------	--------	------------	------	--------------	-------------	---------------------	----------	-----------------

Replication Slot Lag:

slot_name	slot_type	database	active	lag_mb_behind	lag_gb_behind
-----------	-----------	----------	--------	---------------	---------------

Inactive replication slots order by age\_xmin::

slot_name	plugin	slot_type	datoid	database	temporary	active	active_pid	xmin	catalog_xmin	restart_lsn	confirmed_flush_lsn	age_xmin	age_catalog_xmi
-----------	--------	-----------	--------	----------	-----------	--------	------------	------	--------------	-------------	---------------------	----------	-----------------

Note:

RDS Postgres instance storage may get full because inactive replication slots were not removed after DMS task completed

If replication slot is created and it becomes in-active, then transaction logs wont recycle from master instance. So eventually storage gets full

These replication slots can be cleaned as below

Drop inactive replication slot :

Use the below SQL to Generate SQL to drop the inactive slots

```
select select pg_drop_replication_slot('||slot_name||'); from pg_replication_slots where active = false;
```

then Verify the CCloudWatch metrics Free Storage Space to confirm that disk space was released

Replication Parameters :

parameter_name	setting	unit	short_desc
max_logical_replication_workers	4		Maximum number of logical replication worker processes.
max_replication_slots	10		Sets the maximum number of simultaneously defined replication slots.
max_sync_workers_per_subscription	2		Maximum number of table synchronization workers per subscription.
max_wal_senders	10		Sets the maximum number of simultaneously running WAL sender processes.
max_worker_processes	8		Maximum number of concurrent worker processes.
wal_level	replica		Set the level of information written to the WAL.
wal_receiver_status_interval	10	s	Sets the maximum interval between WAL receiver status reports to the sending server.
wal_receiver_timeout	30000	ms	Sets the maximum wait time to receive data from the sending server.
wal_retrieve_retry_interval	5000	ms	Sets the time to wait before retrying to retrieve WAL after a failed attempt.

[\[Top\]](#)

---

## Unused Indexes

---

▶ Details

## PK or FK using numeric or integer data type

---

▶ Details

## Duplicate indexes

---

▶ Details

## Invalid indexes

---

▼ Details

count_of_invalid_indexes
0

indx_id	index_name	table_name	schema_name	owner_id	indx_is_valid
---------	------------	------------	-------------	----------	---------------

# pg\_stat\_statements extension

pg\_stat\_statements installed version:

▼ Details

Extension Name	Version	Schema	owner	Description	relocatable to another schema	extconfig	extcondition
pg_stat_statements	1.7	public	rdsadmin	track execution statistics of all SQL statements executed	t		

Parameters values:

parameter_name	setting
pg_stat_statements.max	5000
pg_stat_statements.save	on
pg_stat_statements.track	top
pg_stat_statements.track_utility	on
shared_preload_libraries	rdsutils,pg_stat_statements,plprofiler

Available versions that are available to upgrade:

extension_name	version	installed
----------------	---------	-----------

Latest Extension version that is available to upgrade:

extension_name	latest_version
----------------	----------------

Top SQL order by total\_time:

► Details

Top SQL order by avg\_time:

► Details

Top SQL order by percent of total DB time percent:

► Details

Top SQL order by number of execution (CALLs):

► Details

Top SQL order by shared blocks read (physical reads):

► Details

[\[Top\]](#)



# Toast Tables Mapping

---

Note:

When a column is written to the toast table, an OID is used to identify the chunk to be toasted. When a toast table grows very large, and contains chunk\_ids that are nearing the value of  $2^{32}$ , it can lead to performance degradation when writing to toast. This is because PostgreSQL must check if an OID is available for assignment by scanning the table

A large Toast table can be a good indication that your toast can face OID wraparound

over time the insert statement will be slower as the Database will be searching for an unused OID and it will have to read from the disk , you will see the insert statements is waiting on IPC:BufferIO or IO:DataFileRead

you can use below SQL to check the toast table

```
select COUNT(DISTINCT chunk_id), 2^31 - COUNT(DISTINCT chunk_id) as remaining_OID , ROUND(100*((2^31 - COUNT(DISTINCT chunk_id))/2^31)::float) AS remaining_OID_PCT , ROUND(100*(COUNT(DISTINCT chunk_id)/2^31)::float) as percent_towards_Toast_oid_wraparound from pg_toast.pg_toast_{number};
```

when the toast hits the OID wraparound, you will see the following wait event LWLock:OidGen and the insert statements will fail and you will see below error in the log file

:LOG: still searching for an unused OID in relation "pg\_toast\_{number}"

:DETAIL: OID candidates have been checked 1000000 times, but no unused OID has been found yet.

Toast Tables Mapping and sizes:

► Details

[\[Top\]](#)



## Transaction ID TXID (Wraparound)

### ▼ Details

#### oldest xid:

oldest_xid
50075394

#### oldest xid per database:

database_name	oldest_xid_per_db
postgres	1234298
template0	50073227
pg2	50075394
rdsadmin	50075394
template1	50075394

#### percent\_towards\_emergency\_autovac & percent\_towards\_wraparound :

oldest_xid	percent_towards_wraparound	percent_towards_emergency_autovac
50075394	3	25

#### current running autovacuum process:

datname	username	state	query	duration	wait_event
---------	----------	-------	-------	----------	------------

#### current running vacuum process:

datname	username	state	query	duration	wait_event
---------	----------	-------	-------	----------	------------

#### vacuum progress process:

pid	duration	waiting	mode	database	table	phase	query	table_size	total_size	scanned	vacuumed	scanned_pct	vacuumed_pct	index_vacuum_count	max_dead_tuples_per_cycle	total_num_dead_tuples	index_cycles_required
-----	----------	---------	------	----------	-------	-------	-------	------------	------------	---------	----------	-------------	--------------	--------------------	---------------------------	-----------------------	-----------------------

#### Inactive replication slots order by age\_xmin:

slot_name	plugin	slot_type	datoid	database	temporary	active	active_pid	xmin	catalog_xmin	restart_lsn	confirmed_flush_lsn	age_xmin	age_catalog_xmin
-----------	--------	-----------	--------	----------	-----------	--------	------------	------	--------------	-------------	---------------------	----------	------------------

#### active replication slots order by age\_xmin:

slot_name	plugin	slot_type	datoid	database	temporary	active	active_pid	xmin	catalog_xmin	restart_lsn	confirmed_flush_lsn	age_xmin	age_catalog_xmin
-----------	--------	-----------	--------	----------	-----------	--------	------------	------	--------------	-------------	---------------------	----------	------------------

#### Orphaned prepared transactions:

gid	prepared	owner	database	ag_xmin
-----	----------	-------	----------	---------

#### MAX XID held:

oldest_running_xact	oldest_prepared_xact	oldest_replication_slot	oldest_replica_xact
0			

#### Autovacuum , vacuum and maintenance\_work\_mem Parameters:

name	setting	source	sourcefile
autovacuum	on	default	
autovacuum_analyze_scale_factor	0.05	configuration file	/rdsdbdata/config/postgresql.conf

# Functions statistics

---

The `pg_stat_user_functions` view will contain one row for each tracked function, showing statistics about executions of that function.

The `track_functions` parameter controls exactly which functions are tracked.

`track_functions` parameter Enables tracking of function call counts and time used.

Specify `pl` to track only procedural-language functions, `all` to also track SQL and C language functions.

## ▼ Details

name	setting
<code>track_functions</code>	<code>all</code>

func_name	calls	total_time	mean_time	self_time
<code>public.pg_stat_statements</code>	147061	3978029.451	27.05	3978029.451
<code>aws_oracle_ext.next_day</code>	438682	128288.553	0.29	128288.553
<code>oracle.next_day</code>	2954578	114420.466	0.04	112154.599
<code>pg_catalog.next_day</code>	2954578	2262.821	0.00	2262.821
<code>pg_catalog.shobj_description</code>	240	175.048	0.73	175.048
<code>pg_catalog.obj_description</code>	1056	86.259	0.08	86.259
<code>information_schema.pg_expandarray</code>	2726	33.526	0.01	33.526
<code>pg_catalog.col_description</code>	12	0.91	0.08	0.91

[\[Top\]](#)

# How to edit PG Collector script ?

		INFO
<a href="#">Database size</a>		<a href="#">DB parameters</a>
<a href="#">Index Size</a>		<a href="#">Vacuum &amp; Statistics</a>
<a href="#">pg_stat_statements extension</a>		<a href="#">Users &amp; Roles Info</a>
<a href="#">Table Access Profile</a>		<a href="#">Unused Indexes</a>
<a href="#">Toast Tables Mapping</a>		<a href="#">Replication</a>
<a href="#">PK or FK using numeric or integer data type</a>		<a href="#">public Schema</a>
<a href="#">pgaudit extension</a>		<a href="#">Unlogged Tables</a>
<a href="#">Background processes</a>		<a href="#">Multixact ID MXID (Wraparound)</a>
<a href="#">Partition tables</a>		<a href="#">pg_shdepend</a>
<a href="#">pg_hba.conf</a>		<a href="#">Duplicate indexes</a>
<a href="#">Triggers</a>		<a href="#">pg_config</a>

```
\qecho <br>
select datname as Database_name , datistemplate as database_is_template , datallowconn as database_allow_connections, datconnlimit as databa
\qecho <br>
\qecho <br>
\qecho <table width="90%" border="1">
\qecho <tr><th colspan="4"><div align="center"><font color="#16191f"><b>INFO</b></font></div></th></tr>
\qecho <tr>
\qecho <td nowrap align="center" width="25%"><a class="link" href="#Database_size">Database size</a></td>
\qecho <td nowrap align="center" width="25%"><a class="link" href="#DB_parameters">DB parameters</a></td>
\qecho <td nowrap align="center" width="25%"><a class="link" href="#Transaction_ID_TXID">Transaction ID TXID (Wraparound)</a></td>
\qecho <td nowrap align="center" width="25%"><a class="link" href="#Table_Size">Table Size</a></td>
\qecho </tr>
\qecho <tr>
\qecho <td nowrap align="center" width="25%"><a class="link" href="#index_Size">Index Size</a></td>
\qecho <td nowrap align="center" width="25%"><a class="link" href="#vacuum_Statistics">Vacuum & Statistics</a></td>
\qecho <td nowrap align="center" width="25%"><a class="link" href="#Extensions">Extensions</a></td>
\qecho <td nowrap align="center" width="25%"><a class="link" href="#Memory_setting">Memory setting</a></td>
\qecho </tr>
\qecho <tr>
\qecho <td nowrap align="center" width="25%"><a class="link" href="#pg_stat_statements_extension">pg_stat_statements extension</a></td>
\qecho <td nowrap align="center" width="25%"><a class="link" href="#Users_Roles_Info">Users & Roles Info</a></td>
```

# Database size

database_name	database_size
postgres	39 GB
pg2	8265 kB
rdsadmin	8345 kB
template1	8121 kB
template0	8137 kB

```
179
180  -- +-----+
181  -- |      - Database_size      |
182  -- +-----+
183
184  \qecho <a name="Database_size"></a>
185  \qecho <font size="+2" face="Arial,Helvetica,Geneva,sans-serif" color="#16191f"><b>Database size</b></font><hr align="left" width="460">
186  SELECT pg_database.datname Database_Name , pg_size_pretty(pg_database_size(pg_database.datname)) AS Database_Size FROM pg_database;
187
188  \qecho <center>[<a class="noLink" href="#top">Top</a>]</center><p>
189
```

# Issue/question/feedback/idea/New SQL

- <https://github.com/awslabs/pg-collector/issues>

# Q&A