

22. Health check

One of the key tasks for any PostgreSQL DBA is conducting daily health checks to ensure the database remains healthy. Performing these checks is critical as it helps identify potential issues and resolve them before they impact production. PostgreSQL provides several views to monitor various metrics such as wait states, CPU usage, I/O, and more.

However, today we'll focus on a tool called **pgmetrics**.

pgmetrics is an open-source, zero-dependency, single-binary tool that can collect over **350 metrics** from a running PostgreSQL server. It presents the data in an **easy-to-read text format** or allows you to **export it as JSON or CSV** for further analysis. This tool will help us monitor various aspects, including replication, slow queries, buffer usage, WAL status, and CPU & RAM usage.

Installing pgmetrics

Installing pgmetrics is quite simple and involves just downloading the binary.

1. download binary

Make sure to download the binary using the `postgres` user and save it in a directory owned by PostgreSQL. For my lab setup, I've already created a directory for backups, which I will use to download pgmetrics.

```
sudo su - postgres
```

```
wget https://github.com/rapidloop/pgmetrics/releases/download/v1.17.0/pgmetrics_1.17.0_linux_amd64.tar.gz
```

```
postgres@postgresql-stg-15:/db_backup$ whoami
postgres
postgres@postgresql-stg-15:/db_backup$ ls -l /
total 2097228
lrwxrwxrwx 1 root root 7 Feb 16 2024 bin -> usr/bin
drwxr-xr-x 4 root root 4096 Sep 25 06:32 boot
dr-xr-xr-x 2 root root 4096 Feb 16 2024 cdrom
drwxr-xr-x 2 postgres postgres 4096 Sep 25 08:26 db_backup
drwxr-xr-x 20 root root 4040 Sep 19 20:11 dev
drwxr-xr-x 100 root root 4096 Sep 20 06:53 etc
drwxr-xr-x 3 root root 4096 Jun 14 09:56 home
lrwxrwxrwx 1 root root 7 Feb 16 2024 lib -> usr/lib
lrwxrwxrwx 1 root root 9 Feb 16 2024 lib32 -> usr/lib32
lrwxrwxrwx 1 root root 9 Feb 16 2024 lib64 -> usr/lib64
lrwxrwxrwx 1 root root 10 Feb 16 2024 libx32 -> usr/libx32
drwx----- 2 root root 16384 Jun 13 19:54 lost+found
drwxr-xr-x 2 root root 4096 Feb 16 2024 media
drwxr-xr-x 2 root root 4096 Feb 16 2024 mnt
drwxr-xr-x 2 root root 4096 Feb 16 2024 opt
dr-xr-xr-x 174 root root 0 Sep 19 20:11 proc
drwx----- 5 root root 4096 Sep 24 19:34 root
drwxr-xr-x 33 root root 1020 Sep 25 08:16 run
lrwxrwxrwx 1 root root 8 Feb 16 2024 sbin -> usr/sbin
drwxr-xr-x 6 root root 4096 Feb 16 2024 snap
drwxr-xr-x 2 root root 4096 Feb 16 2024 srv
-rw----- 1 root root 2147483648 Jun 13 20:09 swap.img
dr-xr-xr-x 13 root root 0 Sep 19 20:11 sys
drwxrwxrwt 13 root root 4096 Sep 25 08:25 tmp
drwxr-xr-x 14 root root 4096 Feb 16 2024 usr
drwxr-xr-x 13 root root 4096 Feb 16 2024 var
postgres@postgresql-stg-15:/db_backup$ wget https://github.com/rapidloop/pgmetrics/releases/download/v1.17.0/pgmetrics_1.17.0_linux_amd64.tar.gz
--2024-09-25 08:27:26-- https://github.com/rapidloop/pgmetrics/releases/download/v1.17.0/pgmetrics_1.17.0_linux_amd64.tar.gz
Resolving github.com (github.com)... 20.233.83.145
Connecting to github.com (github.com)|20.233.83.145|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/122923888/32e95b5e-ec02-4135-8bd6-26e8995cfa55?X-Amz-Algorithm=
```

2. extract the binary.

```
tar xvf pgmetrics_1.17.0_linux_amd64.tar.gz
```

```
postgres@postgresql-stg-15:/db_backup$ tar xvf pgmetrics_1.17.0_linux_amd64.tar.gz
pgmetrics_1.17.0_linux_amd64/LICENSE
pgmetrics_1.17.0_linux_amd64/README.md
pgmetrics_1.17.0_linux_amd64/pgmetrics
postgres@postgresql-stg-15:/db_backup$
```

3. change directory to extracted binary and start check various option option using `--help`

```
cd pgmetrics_1.17.0_linux_amd64/
./pgmetrics --help
```

```
postgres@postgresql-stg-15:/db_backup/pgmetrics_1.17.0_linux_amd64$ ./pgmetrics --help
pgmetrics collects PostgreSQL information and metrics.
```

Usage:

```
pgmetrics [OPTION]... [DBNAME]
```

General options:

```
-t, --timeout=SECS      individual query timeout in seconds (default: 5)
--lock-timeout=MILLIS  lock timeout in milliseconds (default: 50)
-i, --input=FILE        don't connect to db, instead read and display
                        this previously saved JSON file
-V, --version           output version information, then exit
-?, --help[=options]   show this help, then exit
--help=variables       list environment variables, then exit
```

Collection options:

```
-S, --no-sizes          don't collect tablespace and relation sizes
-c, --schema=REGEXP    collect only from schema(s) matching POSIX regexp
-C, --exclude-schema=REGEXP do NOT collect from schema(s) matching POSIX regexp
-a, --table=REGEXP     collect only from table(s) matching POSIX regexp
-A, --exclude-table=REGEXP do NOT collect from table(s) matching POSIX regexp
--omit=WHAT            do NOT collect the items specified as a comma-separated
                        list of: "tables", "indexes", "sequences",
                        "functions", "extensions", "triggers",
                        "statements", "log", "citus", "indexdefs",
                        "bloat"
--sql-length=LIMIT     collect only first LIMIT characters of all SQL
                        queries (default: 500)
--statements-limit=LIMIT collect only utmost LIMIT number of row from
                        pg_stat_statements (default: 100)
--only-listed          collect info only from the databases listed as
                        command-line args (use with Heroku)
--all-dbs              collect info from all user databases
--log-file             location of PostgreSQL log file
--log-dir              read all the PostgreSQL log files in this directory
--log-span=MINS        examine the last MINS minutes of logs (default: 5)
--aws-rds-dbid         AWS RDS/Aurora database instance identifier
--az-resource          Azure resource ID
--pgpool              collect only Pgpool metrics
```

Using pgmetrics

to view the report i will use `--all-dbs` which will show status of all database

```
./pgmetrics --all-dbs
```

here are the metric you will get when executing the command

- **Server:** version, system identifier, timeline, transaction id wraparound, checkpoint lag
- **Replication:** primary-side, standby-side, physical and logical replication slots
- **WAL Archiving:** archive rate, wal and ready file counts, last success and fail time
- **BG Writer:** checkpoint rate, total checkpoints (sched+req), buffers

- **Checkpoint:** checkpoints and restartpoints performed, buffers written, time taken
- **Vacuum-related:** ongoing auto/manual vacuum progress, last analyze/vacuum, settings
- **Tablespaces:** location, size, disk and inode usage of filesystem
- **Database:** size, bloat, disabled triggers, installed extensions, temp files, transaction id wraparound, deadlocks, conflicts
- **Roles:** users, groups, membership
- **Active backends:** transaction running too long, idling in transaction, waiting for locks
- **Tables:** vacuum, analyze, row estimates, idx and seq scans, cache hit ratio, HOT update ratio, size, bloat
- **Indexes:** cache hit ratio, scans, rows read/scan, rows fetched/scan
- **Sequences:** cache hit ratio
- **System metrics:** cores, load average, memory and disk usage
- **Settings:** current values, and default ones where different
- **Slow queries:** from pg_stat_statements, if available
- **Locks:** granted and waiting locks, from pg_locks
- **Job Progress:** progress of analyze, backup, cluster, copy, create index and vacuum jobs
- **Blocked queries:** blocked queries, along with the queries that they are waiting for

PostgreSQL Cluster:

```

Name: 15/main
Server Version: 15.7 (Ubuntu 15.7-1.pgdg22.04+1)
Server Started: 24 Sep 2024 6:43:32 PM (13 hours ago)
System Identifier: 7380300028322386333
Timeline: 1
Last Checkpoint: 24 Sep 2024 6:58:03 PM (13 hours ago)
REDO LSN: 0/8E404BD8
Checkpoint LSN: 0/9E3D1F28 (256 MiB since REDO)
Transaction IDs: oldest = 716, next = 782, range = 66
Notification Queue: 0.0% used
Active Backends: 1 (max 100)
Recovery Mode? no

```

System Information:

```

Hostname: postgresql-stg-15
CPU Cores: 2 x Common KVM processor
Load Average: 0.02
Memory: used=144 MiB, free=541 MiB, buff=35 MiB, cache=1.1 GiB
Swap: used=16 MiB, free=2.0 GiB

```

Setting	Value
shared_buffers	16384 (128 MiB)
work_mem	4096 (4.0 MiB)
maintenance_work_mem	65536 (64 MiB)
temp_buffers	1024 (8.0 MiB)
autovacuum_work_mem	-1
temp_file_limit	-1
max_worker_processes	8
autovacuum_max_workers	3
max_parallel_workers_per_gather	2
effective_io_concurrency	1

pg_stat_statements

`pg_stat_statements` is an extension for PostgreSQL that tracks and records statistics about SQL queries executed in the database. It provides a way to monitor query performance, helping DBAs identify slow or expensive queries, optimize resource usage, and improve overall performance.

when using `pgmetrics` it advisable to enable `pg_stat_statements` extension to get report of slow query which is very useful when performing health check or even try to identify slow running query

Enable `pg_stat_statements`

`pg_stat_statements` should be already installed with postgresql but only change that you have to update the parameter `shared_preload_libraries` and include .

```
sudo nano /etc/postgresql/15/main/postgresql.conf
```

```
shared_preload_libraries = 'pg_stat_statements'
```

```
#local_preload_libraries = ''
#session_preload_libraries = ''
shared_preload_libraries = 'pgaudit,pg_stat_statements' # (change requires restart)
#jit_provider = 'llvmjit' # JIT library to use

# - Other Defaults -

#dynamic_library_path = '$libdir'
#extension_destdir = '' # prepend path when loading extensions
#                        # and shared objects (added by Debian)
#gin_fuzzy_search_limit = 0
```

After that restart postgresql services

```
sudo systemctl restart postgresql@15-main.service
```

now login to `psql` change to database you want to enable the extension

```
sudo -u postgres psql
```

```
\c production
```

```
CREATE EXTENSION pg_stat_statements;
```

```

postgres=# \c prodction
connection to server on socket "/var/run/postgresql/.s.PGSQL.5432" failed: FATAL:  database "prodction" does not exist
Previous connection kept
postgres=# \c production
You are now connected to database "production" as user "postgres".
production=# CREATE EXTENSION pg_stat_statements;
ERROR:  extension "pg_stat_statements" already exists
production=# \dt+

```

Schema	Name	Type	Owner	Persistence	Access method	Size	Description
public	pgbench_accounts	table	postgres	permanent	heap	641 MB	
public	pgbench_branches	table	postgres	permanent	heap	40 kB	
public	pgbench_history	table	postgres	permanent	heap	0 bytes	
public	pgbench_tellers	table	postgres	permanent	heap	56 kB	

(4 rows)

```

production=# \dx

```

Name	Version	Schema	Description
pg_stat_statements	1.10	public	track planning and execution statistics of all SQL statements executed
plpgsql	1.0	pg_catalog	PL/pgSQL procedural language

(2 rows)

```

production=#

```

Now, run the command `./pgmetrics --all-dbs` and scroll down until you reach the database where you enabled the extension. You should see the top most expensive queries listed there.

```

Database #2:
Name:          production
Owner:         postgres
Tablespace:    pg_default
Connections:   1 (no max limit)
Frozen Xid Age: 72
Transactions:  3717 (99.9%) commits, 5 (0.1%) rollbacks
Cache Hits:    23.7%
Rows Changed:  ins 100.0%, upd 0.0%, del 0.0%
Total Temp:    382 MiB in 16 files
Problems:      0 deadlocks, 0 conflicts
Totals Since:
Size:          755 MiB
Installed Extensions:

```

Name	Version	Comment
pg_stat_statements	1.10	track planning and execution statistics of all SQL statements executed
plpgsql	1.0	PL/pgSQL procedural language

```

Slow Queries:

```

Calls	Avg Time	Total Time	Rows/Call	Query
3	11.411s	34.235s	5000000	copy pgbench_accounts from stdin with (freeze on)
3	3.973s	11.921s	0	alter table pgbench_accounts add primary key (aid)
3	673ms	2.02s	0	vacuum analyze pgbench_accounts
1	730ms	730ms	0	ANALYZE pgbench_accounts
5	41ms	209ms	7	SELECT current_database() AS db, schemaname, tab
6	31ms	187ms	2	SELECT name, current_database(), COALESCE(default
12	5ms	67ms	1	SELECT pg_tablespace_size(\$1)
1	61ms	61ms	0	CREATE EXTENSION IF NOT EXISTS pg_stat_statements
3	14ms	43ms	0	alter table pgbench_branches add primary key (bid)
12	2ms	27ms	1	SELECT pg_database_size(\$1)
5	4ms	23ms	98	SELECT userid, dbid, queryid, LEFT(COALESCE(query,
6	3ms	18ms	368	SELECT name, setting, COALESCE boot_val,\$1), sourc
3	5ms	16ms	0	drop table if exists pgbench_accounts, pgbench_bra