<https://percona.service-now.com/percona?id=kb_article&sys_id=7baaecc01b9b46902aa6ed70604bcb0d>

<https://percona.service-now.com/percona?id=kb_article&sys_id=ca0dc7c697736d106a5a72200153aff0>

<https://percona.service-now.com/percona?id=kb_article&sys_id=2d5a96691ba7c2102aa6ed70604bcb75>

Auditing : pgaudit Extension for User Session Auditing

**Introduction**

Log-based auditing has a few advantages in PostgreSQL as it is a lightweight approach that does not directly affect transaction performance. Trigger-based auditing has the disadvantage of slowing down the transaction. pgaudit is an extension for the advanced session and object-level audit logging. PostgreSQL has some basic audit logging facilities by enabling the corresponding parameter. pgaudit enhances these features by more fine-grained control of the sessions and objects to be audited. This article explains how to install pgaudit and enable user session auditing.

**Process**

pgaudit allows the following classes of statements to be audited

1. **READ** (SELECT and COPY when the source is a relation or a query)
2. **WRITE** (INSERT, UPDATE, DELETE, TRUNCATE, and COPY when the destination is a relation)
3. **FUNCTION** (Functions and DO blocks)
4. **ROLE** (GRANT, REVOKE, CREATE/ALTER/DROP ROLE)
5. **DDL** (All DDL not included in ROLE)
6. **MISC** (DISCARD, FETCH, CHECKPOINT, VACUUM)

Thusly, if the "read" class is specified for auditing, all the SELECT and COPY statements will be audited.

**Installation**

Ready to Install rpm/deb packages are available from PGDG repo and [Percona Repo](https://docs.percona.com/percona-software-repositories/index.html).

On CentOS 7,  one can install pgaudit for your installed version of PostgreSQL from the PGDG repo.  For PostgreSQL 12, the command will be as follows:

$ sudo yum install pgaudit14\_12

The pgaudit version for each PostgreSQL version is different.  It is pgaudit14\_12 for PostgreSQL 12, pgaudit15\_13 for PostgreSQL 13, pgaudit16\_14 for PostgreSQL 14, pgaudit17\_15 for PostgreSQL 15, and pgaudit\_16 for PostgreSQL16.

If packages are installed from the repository for Percona Distribution, Package names will have "percona-" at the beginning, for example:

$ sudo yum install percona-pgaudit14\_12

We can check the available packages using sudo yum search pgaudit

Alternatively, Percona Distribution of PostgreSQL provides version-agnostic installation.

sudo yum install percona-pgaudit

**Installation for Ubuntu/Debian**

sudo apt install postgresql-12-pgaudit

Once the binary installation is completed, the extension can be loaded using shared\_preload\_libraries. If there are other extensions already specified, we can append pgaudit to the list. For example:

ALTER SYSTEM SET shared\_preload\_libraries=pg\_stat\_statements,pgaudit;

**Important:** This change requires a restart of the Instance

Once the restart is completed, It is important to make sure that the extension is loaded

postgres=# show shared\_preload\_libraries ;

shared\_preload\_libraries

-----------------------------

pg\_stat\_statements, pgaudit

Now we can proceed to install the pgaudit extension in the desired database.

**postgres=# CREATE EXTENSION pgaudit;**

CREATE EXTENSION

**Enabling and Testing the audit log**

Here is an example of how one can enable the audit by setting the pgaudit.log value and reloading the configuration

ALTER SYSTEM SET pgaudit.log TO 'read';

SELECT pg\_reload\_conf();

In this case, setting the value to 'read' result in writing an audit line for every SELECT and COPY statement executed against any table to the PostgreSQL log, as shown below:

2020-03-09 13:15:28.367 UTC [21397] LOG: AUDIT: SESSION,1,1,READ,SELECT,,,select \* from tblaudit;,<not logged>

2020-03-09 13:17:22.194 UTC [21397] LOG: AUDIT: SESSION,2,1,READ,SELECT,,,"select pid, datname,usename,query\_start,now()-query\_start as ""running since"" from pg\_stat\_activity order by query\_start;",<not logged>

We are allowed to set multiple options for pgaudit.log as comma separated list

For example, for auditing the read and write statements:

ALTER SYSTEM SET pgaudit.log TO 'read, write';

If we need to add functions also to auditing

ALTER SYSTEM SET pgaudit.log TO 'read, write, function';

If every statement except DDLs and MISC are to be audited, the setting should be as follows:

ALTER SYSTEM SET pgaudit.log TO 'all, -misc, -ddl';

**Scope of the auditing**

**Instead of auditing the complete PostgreSQL instance, it is possible to audit a specific user account**

ALTER USER super SET pgaudit.log TO 'read, write';

**Specific databases can be audited by setting the value at the database level**

ALTER DATABASE testdb SET pgaudit.log=read,write;

If the database level parameter is set, any user session connecting to that database will have session-level settings automatically and statements will be audited as per settings.

**Audit Policy Access**

A regular user won't be able to adjust his/her own audit settings. Only Superuser can specify the audit settings for other users. This protects the auditing policy from changes by a regular user connecting to the database.

postgres=> SET pgaudit.log=read,write;

ERROR: permission denied to set parameter "pgaudit.log"

**Checking pgaudit configuration**

**Instance level:**

SHOW pgaudit.log;

**Database level:**

select oid, datname, setconfig from pg\_db\_role\_setting join pg\_database on pg\_db\_role\_setting.setdatabase = pg\_database.oid where exists(select \* from unnest(setconfig) where unnest like 'pgaudit%') ;

**User level:**

select usename, useconfig from pg\_user where exists(select \* from unnest(useconfig) where unnest like 'pgaudit%'

**For our Demo we will be executing bellow command:**

**Prepare auditor user for the test demo, while connected as postgres user run bellow command;**

create user auditor with password auditor;

create schema audit;

grant usage, create on schema audit to auditor;

grant select,insert,update,delete on all tables in schema audit to auditor;

alter default privileges in schema audit grant select,insert,update,delete on tables to auditor;

grant usage on all sequences in schema audit to auditor;

alter default privileges in schema audit grant usage on sequences to auditor;

**Instead of auditing the complete PostgreSQL instance, it is possible to audit a specific user account in our case auditor user to be set up on DEV :**

ALTER USER auditor SET pgaudit.log TO 'read, write, ddl'; (for auditing read, write statements)

SELECT pg\_reload\_conf();

alter system set pgaudit.log\_parameter= on;

SELECT pg\_reload\_conf();

**Checking pgaudit configuration at User level:**

select usename, useconfig from pg\_user where exists(select \* from unnest(useconfig) where unnest like 'pgaudit%');

A black screen with white text

Description automatically generated

**Now connect to the database as auditor user and run bellow command ;**

psql -d citus -U auditor

CREATE TABLE audit.employees1 (

id SERIAL PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

department VARCHAR(50),

salary NUMERIC(10, 2)

);

INSERT INTO audit.employees1 (first\_name, last\_name, department, salary)

VALUES

('John', 'Doe', 'HR', 55000.00),

('Jane', 'Smith', 'Engineering', 75000.00),

('Mark', 'Taylor', 'Marketing', 60000.00),

('Lucy', 'Williams', 'Finance', 70000.00);

SELECT \* FROM audit.employees1;

UPDATE audit.employees1

SET salary = 60000.00

WHERE first\_name = 'John' AND last\_name = 'Doe';

SELECT \* FROM audit.employees1 WHERE first\_name = 'John' AND last\_name = 'Doe';

DELETE FROM audit.employees1

WHERE first\_name = 'Mark' AND last\_name = 'Taylor';

You should be able to see the above command/sql statements in the postgresql logs

**On UAT and QA servers we already have existing users for auditing to be implemented:**

**Read, Write statements will be audited from bellow users so we run bellow queries on UAT and QA**

ALTER USER diagmep SET pgaudit.log TO 'read, write, ddl';

ALTER USER mepinfauser SET pgaudit.log TO 'read, write, ddl';

ALTER USER mepmiguser SET pgaudit.log TO 'read, write, ddl';

ALTER USER mepuser SET pgaudit.log TO 'read, write, ddl';

ALTER USER mrgetluser SET pgaudit.log TO 'read, write, ddl';

ALTER USER mrguser SET pgaudit.log TO 'read, write, ddl';

ALTER USER ssretluser SET pgaudit.log TO 'read, write, ddl';

ALTER USER ssrmrgetluser SET pgaudit.log TO 'read, write, ddl';

ALTER USER cisuser SET pgaudit.log TO 'read, write, ddl';

ALTER USER diagcis SET pgaudit.log TO 'read, write, ddl';

SELECT pg\_reload\_conf();

**Checking pgaudit configuration at User level:**

A computer screen with white text

Description automatically generated

The users above enabled for pgAudit can now be seen on postgresql logs