

Open in app ↗

Medium

 Search Member-only story

# Postgres Security 101: Logging and Auditing (3/8)



Oz · Following

9 min read · Oct 4, 2024



Listen



Share



More

Logging and auditing are critical components of any robust security strategy for PostgreSQL. Proper logging enables database administrators to monitor activity, detect anomalies, and maintain an audit trail for security and compliance purposes. In this article, we'll delve into PostgreSQL's logging capabilities, how to configure them for effective auditing, and the best practices for retaining and analyzing logs to help you stay ahead of potential security threats. Whether it's tracking user actions or monitoring system performance, effective logging is essential to ensuring the integrity of your database.



### 3.1 PostgreSQL Logging

- **3.1.1 Logging Rationale:** Understand the importance of logging for monitoring and auditing. Having an audit trail is an important feature of any relational database system.

You want enough detail **to** describe **when** an **event of** interest has started **and** stopped, what the **event is/was**, the **event's cause**, and **what the event** did/**is** doing **to** the system. Ideally, the logged information **is in** a format permitting further analysis giving us **new perspectives and** insight.

- **3.1.2 Ensure the Log Destinations Are Set Correctly:** Configure log destinations to capture all relevant logs. The log destinations should comply with your organization's policies on logging. If all the expected log destinations are not set, this is a fail.

```

show log_destination;

log_destination
-----
stderr
/*
stderr log/postgresql.log
csvlog log/postgresql.csv
jsonlog log/postgresql.json /*required version at least 15*/
*/
alter system set log_destination = 'csvlog';
/* Reload configuration file */
select pg_reload_conf();

```

- **3.1.3 Ensure the Logging Collector Is Enabled:** Enable the logging collector to manage log files. When enabling this parameter, a background process is started, and it captures all the messages sent to the standard error (stderr) and redirects them into log files. The default value for this parameter is off; however, it is highly recommended to set it to on.

```

show logging_collector;

logging_collector
-----
off
alter system set logging_collector = 'on';
/* Reload configuration file */
select pg_reload_conf();

```

- **3.1.4 Ensure the Log File Destination Directory Is Set Correctly:** Specify an appropriate directory for log files. under high workloads, avoid causing an impact on the disk for the database operations by moving the log files to a different disk

```

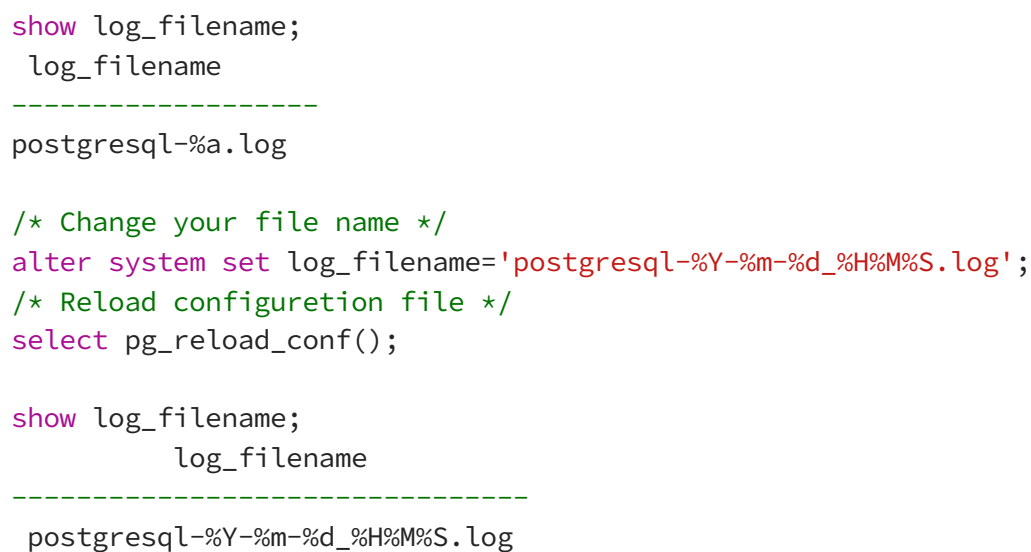
show log_directory;

log_directory
-----

```

```
log
alter system set log_directory='/pg_log/log';
/* Reload configuration file */
select pg_reload_conf();
/*
Please do not forget logical volume is not enough. Also, physical volume is re
*/
```

- **3.1.5 Ensure the Filename Pattern for Log Files Is Set Correctly (Manual):** Set a clear and consistent pattern for log file names.



```
show log_filename;
log_filename
-----
postgresql-%a.log

/* Change your file name */
alter system set log_filename='postgresql-%Y-%m-%d_%H%M%S.log';
/* Reload configuration file */
select pg_reload_conf();

show log_filename;
log_filename
-----
postgresql-%Y-%m-%d_%H%M%S.log
```

- **3.1.6 Ensure the Log File Permissions Are Set Correctly:** Restrict access to log files to prevent tampering.

```
show log_file_mode;
log_file_mode
-----
0600

/* if you have correct mode, please alter your mode. Default Value: 0600*/
alter system set log_file_mode = '0600';
/* Reload configuration file */
select pg_reload_conf();
```

- **3.1.7 Ensure 'log\_truncate\_on\_rotation' Is Enabled:** Enable log truncation on rotation to manage log file sizes.

```
show log_truncate_on_rotation;

log_truncate_on_rotation
-----
on
/* Default Value:on */
alter system set log_truncate_on_rotation = 'on';
/* Reload configuration file */
select pg_reload_conf();
```

- **3.1.8 Ensure the Maximum Log File Lifetime Is Set Correctly (Manual):** Define a suitable log file lifetime. When the logging\_collector is enabled, this parameter controls if the messages sent to an existing log file will be appended (off) or the file will be truncated/overwritten (on). As we saw before, this can be combined with the log\_filename to control the number of files to store. For example, if the log\_filename is configured as postgresql-%H.log, the log\_rotation\_age is on its default of 1440 minutes (1 day), and the log\_truncate\_on\_rotation is on, then every hour, a new empty file will be created, and PostgreSQL will keep only 24 files. This is only related your company strategies.

```
show log_rotation_age;
log_rotation_age
-----
1d

/* Default Value: 1d (one day) */
alter system set log_rotation_age='1h';
/* Reload configuration file */
select pg_reload_conf();
```

- **3.1.9 Ensure the Maximum Log File Size Is Set Correctly (Manual):** Set an appropriate maximum size for log files. This parameter can be used when the logging\_collector is enabled. If the value has no units specified, it is calculated as kilobytes. The default value is 0MB, and setting it to 0 (zero) disables the size-based rotation.

```
show log_rotation_size;
log_rotation_size
-----
0

/* Default Value: 0 */
alter system set log_rotation_size = '1GB';
/* Reload configuration file */
select pg_reload_conf();
```

- **3.1.10 Ensure the Correct Syslog Facility Is Selected :** Configure the correct syslog facility for PostgreSQL logs.

```
show syslog_facility;
syslog_facility
-----
local0

/* Default Value: LOCAL0 */
alter system set syslog_facility = 'LOCAL1';
/* Reload configuration file */
select pg_reload_conf();
```

- **3.1.11 Ensure Syslog Messages Are Not Suppressed:** Allow all relevant syslog messages to be captured.

```
show syslog_sequence_numbers;

syslog_sequence_numbers
-----
on
/* Default Value: on */
alter system set syslog_sequence_numbers = 'on';
/* Reload configuration file */
select pg_reload_conf();
```

- **3.1.12 Ensure Syslog Messages Are Not Lost Due to Size:** Prevent loss of syslog messages by configuring appropriate size limits.

```
show syslog_split_messages;

syslog_split_messages
-----
on
/* Default Value: on */
alter system set syslog_split_messages = 'on';
/* Reload configuration file */
select pg_reload_conf();
```

- **3.1.13 Ensure the Program Name for PostgreSQL Syslog Messages Is Correct (Manual):** Set the correct program name for syslog messages. If this is not set correctly, it may be difficult or impossible to distinguish PostgreSQL messages from other messages in Syslog logs.

```
show syslog_ident;

syslog_ident
-----
postgres
/* Default Value: postgres */
alter system set syslog_ident = 'proddb';
/* Reload configuration file */
select pg_reload_conf();
```

- **3.1.14 Ensure the Correct Messages Are Written to the Server Log:** Ensure all necessary log messages are captured. If this is not set to the correct value, too many or too few messages may be written to the server log.

```
/*
• DEBUG5 <-- exceedingly chatty
• DEBUG4
• DEBUG3
• DEBUG2
• DEBUG1
• INFO
• NOTICE
• WARNING <-- default
• ERROR
• LOG
```

- FATAL
- PANIC <-- practically muteshow syslog\_ident;

WARNING is considered the best practice unless indicated otherwise by your organization's logging policy.

```
*/
show log_min_messages;
log_min_messages
-----
warning
/* Default Value: warning */
alter system set log_min_messages = 'warning';
/* Reload configuration file */
select pg_reload_conf();
```

- **3.1.15 Ensure the Correct SQL Statements Generating Errors Are Recorded:** Log SQL statements that cause errors.

```
show log_min_error_statement;

log_min_error_statement
-----
error
/* Default Value: error */
alter system set log_min_error_statement = 'error';
/* Reload configuration file */
select pg_reload_conf();
```

- **3.1.16 Ensure 'debug\_print\_parse' Is Disabled:** Disable debug printing of parse trees.

```
show debug_print_parse;

debug_print_parse
-----
off
/* Default Value: off */
alter system set debug_print_parse='off';
/* Reload configuration file */
select pg_reload_conf();
```



- **3.1.17 Ensure 'debug\_print\_rewritten' Is Disabled:** Disable debug printing of rewritten queries.

```
show debug_print_rewritten;

debug_print_rewritten
-----
off
/* Default Value: off */
alter system set debug_print_rewritten = 'off';
/* Reload configuration file */
select pg_reload_conf();
```

- **3.1.18 Ensure 'debug\_print\_plan' Is Disabled:** Disable debug printing of execution plans.

```
show debug_print_plan;

debug_print_plan
-----
off
/* Default Value: off */
alter system set debug_print_plan = 'off';
/* Reload configuration file */
select pg_reload_conf();
```

- **3.1.19 Ensure 'debug\_pretty\_print' Is Enabled:** Enable pretty printing for easier debugging.

```
show debug_pretty_print;

debug_pretty_print
-----
on
/* Default Value: on */
alter system set debug_pretty_print = 'on';
/* Reload configuration file */
select pg_reload_conf();
```

- **3.1.20 Ensure 'log\_connections' Is Enabled: Log connection attempts.**

```
show log_connections;

log_connections
-----
on
/* Default Value: on */
alter system set log_connections = 'on';
/* Reload configuration file */
select pg_reload_conf();
```

- **3.1.21 Ensure 'log\_disconnections' Is Enabled: Log disconnections.**

```
show log_disconnections;

log_disconnections
-----
on
/* Default Value: on */
alter system set log_disconnections = 'on';
/* Reload configuration file */
select pg_reload_conf();
```

- **3.1.22 Ensure 'log\_error\_verbosity' Is Set Correctly:** Configure the verbosity level of error logs. The log\_error\_verbosity setting specifies the verbosity (amount of detail) of logged messages. with each containing the fields of the level above it as well as additional fields.  
TERSE excludes the logging of DETAIL, HINT, QUERY, and CONTEXT error information. VERBOSE output includes the SQLSTATE, error code, and the source code file name, function name, and line number that generated the error. The appropriate value should be set based on your organization's logging policy.

Valid values are:

- TERSE
- DEFAULT
- VERBOSE

```

show log_error_verbosity;
log_error_verbosity
-----
default
/* Default Value: default */
alter system set log_error_verbosity = 'verbose';
/* Reload configuration file */
select pg_reload_conf();

```

- **3.1.23 Ensure 'log\_hostname' Is Set Correctly:** Log hostnames in addition to IP addresses.

```

show log_hostname;

log_hostname
-----
off
/* Default Value: off */
alter system set log_hostname='off';
/* Reload configuration file */
select pg_reload_conf();

```

- **3.1.24 Ensure 'log\_line\_prefix' Is Set Correctly:** Define a clear and informative log line prefix.

```

%a = application name
%u = user name
%d = database name
%r = remote host and port
%h = remote host
%b = backend type
%p = process ID
%P = process ID of parallel group leader
%t = timestamp without milliseconds
%m = timestamp with milliseconds
%n = timestamp with milliseconds (as a Unix epoch)
%Q = query ID (0 if none or not computed)
%i = command tag
%e = SQL state
%c = session ID
%l = session line number
%s = session start timestamp

```

```

%v = virtual transaction ID
%x = transaction ID (0 if none)
%q = stop here in non-session processes
%% = '%'

show log_line_prefix;
log_line_prefix
-----

%m [%p]
/* Default Value: %m [%p]*/
alter system set log_line_prefix = '%m [%p]: [%l-1] db=%d,user=%u,app=%a,client=%i';
/* Reload configuration file */
select pg_reload_conf();

```

- 3.1.25 Ensure 'log\_statement' Is Set Correctly:** Configure which SQL statements to log. It is recommended this be set to ddl unless otherwise directed by your organization's logging policy. PREPARE, EXECUTE, and EXPLAIN ANALYZE statements are also logged if their contained command is of an appropriate type.) For clients using extended query protocol, logging occurs when an Execute message is received, and values of the Bind parameters are included (with any embedded singlequote marks doubled).

```

/*
The log_statement setting specifies the types of SQL statements that are logged
Valid values are:
• none (off)
• ddl
• mod
• all (all statements)
ddl logs all data definition statements:
• CREATE
• ALTER
• DROP
mod logs all ddl statements, plus data-modifying statements:
• INSERT
• UPDATE
• DELETE
• TRUNCATE
• COPY FROM
*/
show log_statement;
log_statement
-----

none
/* Default Value: none */

```

```
alter system set log_statement='ddl';
/* Reload configuration file */
select pg_reload_conf();
```

- **3.1.26 Ensure 'log\_timezone' Is Set Correctly:** Set the correct timezone for logs. If log\_timezone is not set to GMT, UTC, or as defined by your organization's logging policy this is a fail.

```
show log_timezone;

log_timezone
-----
Europe/Istanbul
/* Default Value: By default, the PGDG packages will set this to match the serv
Operating System.*/
alter system set log_timezone = 'GMT +3';
/* Reload configuration file */
select pg_reload_conf();
```

- **3.1.27 Ensure That Log Directory Is Outside the PGDATA:** Place the log directory outside the main data directory. Best practice is to not write PostgreSQL logs into the PGDATA for performances reason and disk space use. Please do not forget logical volume is not enough.

```
show data_directory;

data_directory
-----
/pg_data/data

show log_directory;

log_directory
-----
/pg_log/log
```

## 3.2 Ensure the PostgreSQL Audit Extension (pgAudit) Is Enabled

- Enable the pgAudit extension for detailed auditing of database activities. The PostgreSQL Audit Extension ([pgAudit](#)) provides detailed session and/or object audit logging via the standard PostgreSQL logging facility. The goal of pgAudit is to provide PostgreSQL users with the capability to produce audit logs often required to comply with government, financial, or ISO certifications. Basic statement logging can be provided by the standard logging facility with `log_statement = all`. This is acceptable for monitoring and other uses but does not provide the level of detail generally required for an audit. It is not enough to have a list of all the operations performed against the database, it must also be possible to find particular statements that are of interest to an auditor. The standard logging facility shows what the user requested, while pgAudit focuses on the details of what happened while the database was satisfying the request.

#Firstly, you can instal pgaudit package

```
dnf -y install pgaudit15_13
alter system set shared_preload_libraries = 'pgaudit';
ALTER SYSTEM SET pgaudit.log TO 'ddl, write';
#Reload configuration file
select pg_reload_conf();
```

```
show pgaudit.log;
pgaudit.log
```

-----  
ddl, write

- READ: SELECT and COPY when the `source` is a relation or a query.
- WRITE: INSERT, UPDATE, DELETE, TRUNCATE, and COPY when the destination is a r
- FUNCTION: Function calls and DO blocks.
- ROLE: Statements related to roles and privileges: GRANT, REVOKE, CREATE/ALTER
- DDL: All DDL that is not included in the ROLE class.
- MISC: Miscellaneous commands, e.g. DISCARD, FETCH, CHECKPOINT, VACUUM

By configuring PostgreSQL's logging and auditing features correctly, you can gain valuable insights into your database activity and maintain a secure environment. These tools not only help in detecting and preventing unauthorized access but also play a key role in meeting compliance requirements. Regularly reviewing and analyzing logs is a best practice that will bolster your overall security posture and help in responding swiftly to incidents. For more insights on strengthening your PostgreSQL security, I recommend checking out my next article: "[Postgres Security](#)"

**101: User Access and Authorization (4/8)**”, where we explore how to manage user roles and permissions to ensure only authorized individuals can access your database. For more detailed and technical articles like this, keep following our blog on Medium. If you have any questions or need further assistance, feel free to reach out in the comments below and directly.

Database Security

Postgres Security

Security

Cybersecurity

Technology



Following

## Written by Oz

149 Followers · 13 Following

Database Administrator 

## Responses (1)



Gvadakte

What are your thoughts?



Dodocat

Feb 21



Excellent



1

[Reply](#)

More from Oz

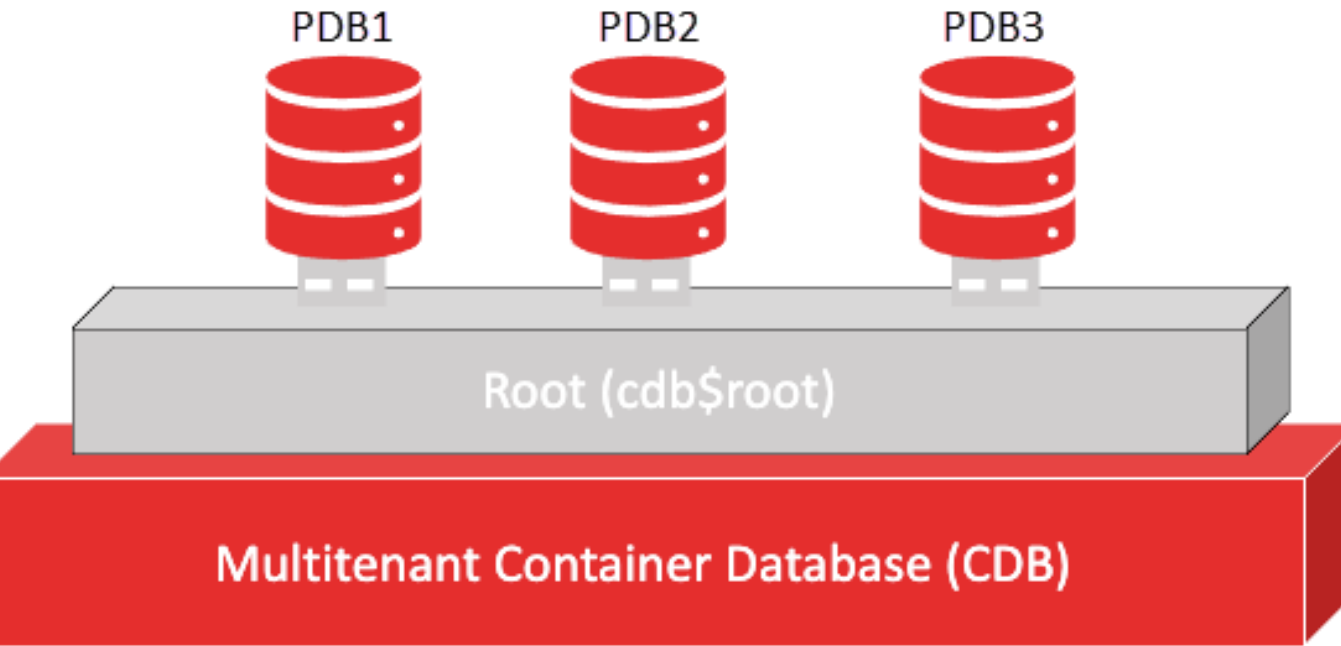


Oz

Installing Percona Monitoring & Management (PMM) with Postgres

Introduction:

★ Sep 26, 2024 54 1





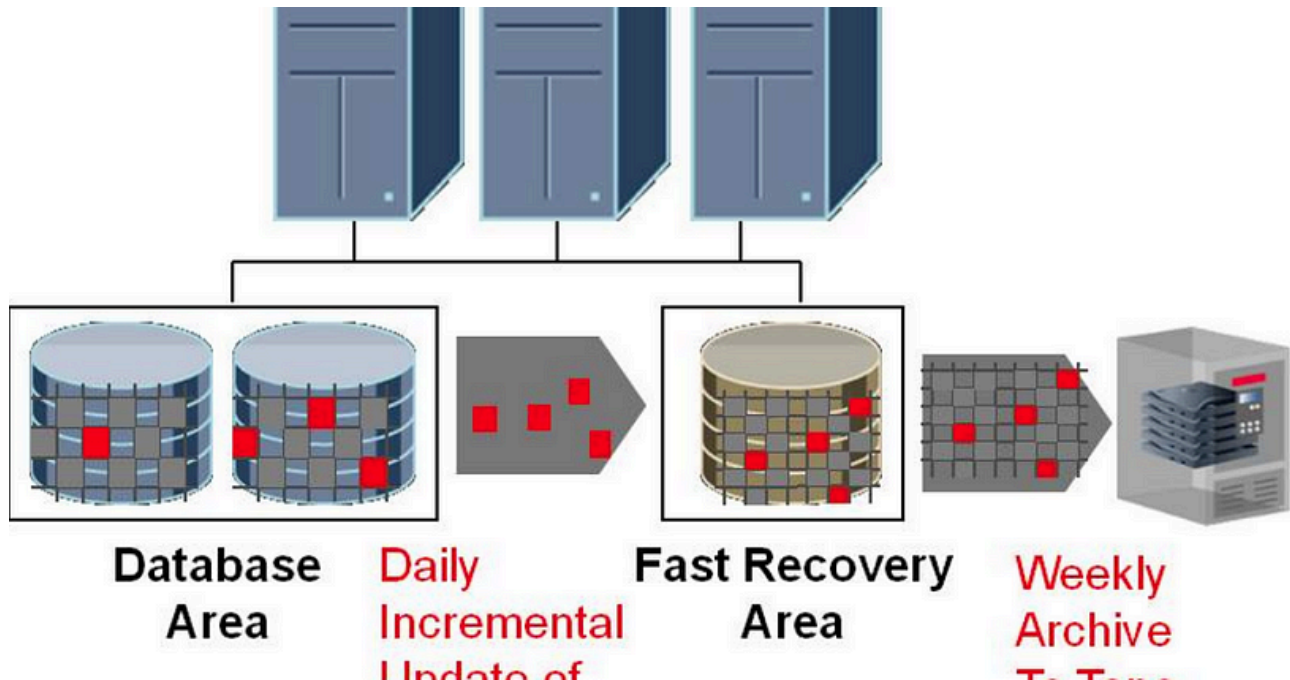


Oz

## Pluggable Database Command

----- - create pluggable database pdb1 admin user root identified by test123; alter pluggable database...

★ May 12, 2023



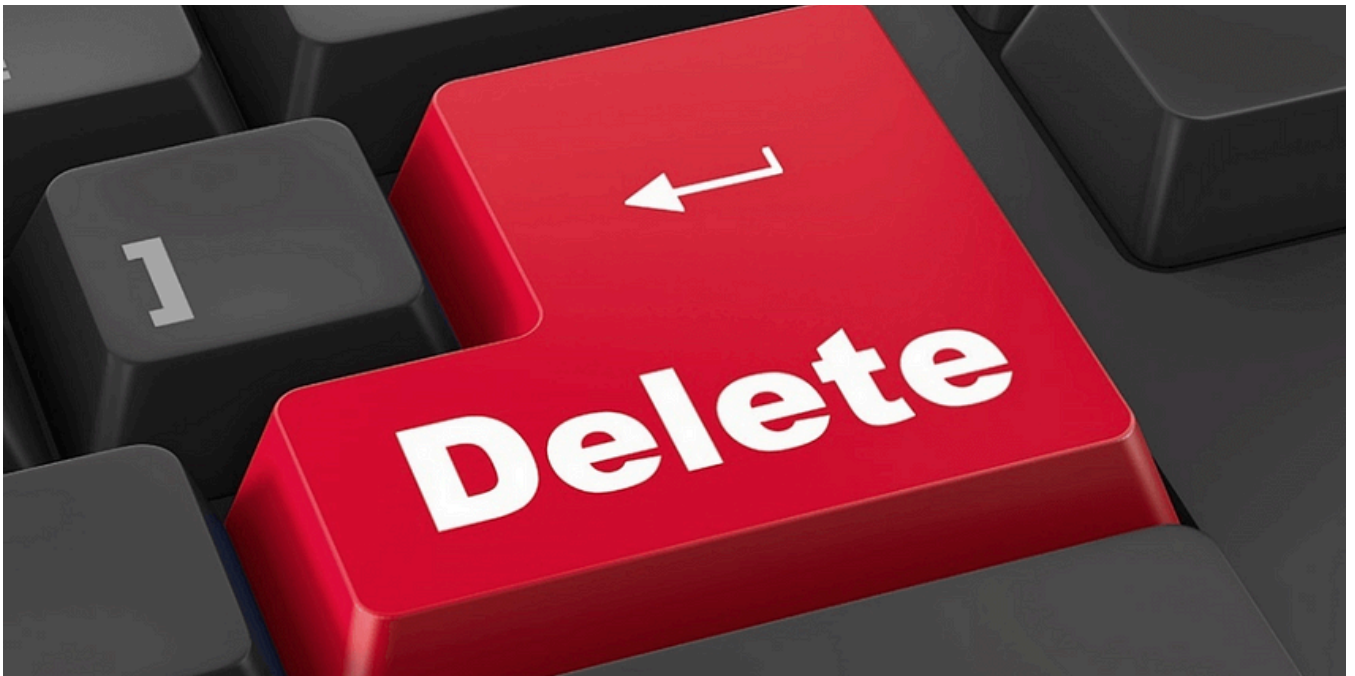
Oz

## RMAN Backup Basic Commands

rman target / rman target sys/password@YDKTST; backup database; backup database format '/backup/path/%d\_%t\_%s.rman'; backup tablespace...

★ May 11, 2023 🖱 1





Oz

## delete jobs


✦ May 8, 2023



See all from Oz

## Recommended from Medium

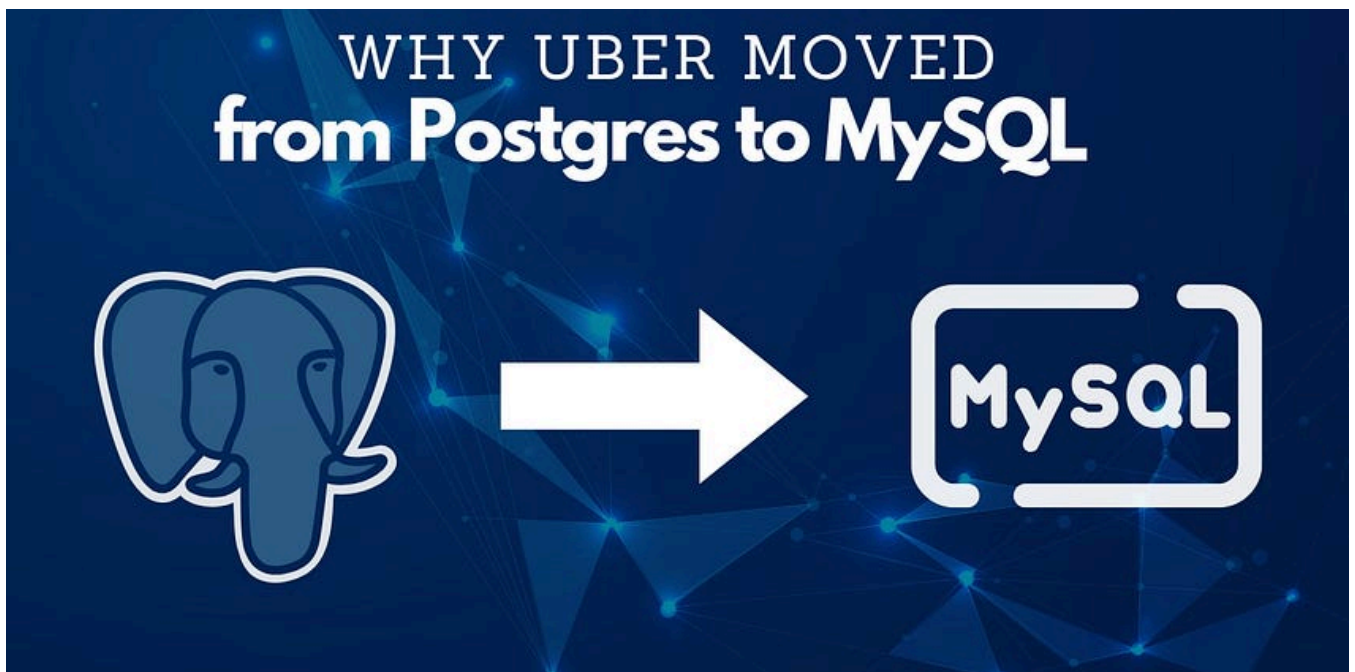


 Tihomir Manushev

## Vector Search with pgvector in PostgreSQL

Simple AI-powered similarity search

★ Mar 9



 In Databases by Sergey Egorenkov


## Why Uber Moved from Postgres to MySQL

How PostgreSQL's architecture clashed with Uber's scale—and why MySQL offered a better path forward

Mar 29 🖱️ 228 💬 7



What it Means	Best Used For
Store directly in the row	Simple data like INT
Store in the row (unless large)	Larger types, but try
Compress + store out-of-row	Long texts, large ob
Store out-of-row, no compression	When compression

 Udbhav Singh

## Storages in PostgreSQL

What Are Storage Types in PostgreSQL — And Why Should You Care?

6d ago 🖱️ 18

 In Hack the Stack by Coders Stop

## 9 Database Optimization Tricks SQL Experts Are Hiding From You



Most developers learn enough SQL to get by—SELECT, INSERT, UPDATE, DELETE, and maybe a few JOINS. They might even know how to create...

★ Mar 27 🖱 185 💬 5



```
3. nodeAPP 4. nodeTWO
b/postgresql/16/main/*

t patroni

/etc/patroni.yml list
21665717) -----+-----+-----+
Role      | State      | TL | Lag in MB |
-----+-----+-----+
Leader    | running    | 1  |           |
Replica   | streaming  | 1  | 0         |
Replica   | streaming  | 1  | 0         |
-----+-----+-----+

```

 Dickson Gathima

## Building a Highly Available PostgreSQL Cluster with Patroni, etcd, and HAProxy

Achieving high availability in PostgreSQL requires the right combination of tools and architecture.

Mar 14 🖱 4





In Towards Dev by Nakul Mitra

## PostgreSQL Performance Optimization—Cleaning Dead Tuples & Reindexing

Performance optimization is crucial in PostgreSQL to ensure efficient query execution and minimal resource consumption.

Mar 28  1[See more recommendations](#)