Member-only story

Postgres Security 101: Installation and Patching Checklist (1/8)

Oz · Following

8 min read · Oct 4, 2024

Listen Share More

This article is the first part of an eight-part series on PostgreSQL security. In this series, we'll explore various aspects of securing a PostgreSQL database. We begin with the foundation: installation and patching. A secure installation process sets the stage for a protected database environment, and keeping your system up to date with patches is key to maintaining that security over time. This checklist will guide you through best practices to follow during the installation and patching of PostgreSQL to ensure your system is as secure as possible from day one.

1.1 Ensure Packages Are Obtained from Authorized Repositories

- **1.1.1 PostgreSQL Packages Installed (Manual):** Ensure PostgreSQL packages are installed correctly.

```
# Check Installed PostgreSQL Packages for On Red Hat Based Systems
rpm -qa | grep postgresql

# You can check your rpm your outputs
postgresql13-libs-13.13-1PGDG.rhel9.x86_64
postgresql13-13.13-1PGDG.rhel9.x86_64
postgresql13-server-13.13-1PGDG.rhel9.x86_64
postgresql13-contrib-13.13-1PGDG.rhel9.x86_64
postgresql13-devel-13.13-1PGDG.rhel9.x86_64

/*
This command lists all RPM packages with "postgresql" in their names.
You should see entries like above this;
postgresql-server
postgresql-libs
and postgresql-contrib.
*/
```

- **1.1.2 Ensure Packages Are Obtained from PGDG:** Verify that packages are sourced from the PostgreSQL Global Development Group (PGDG) repository.

1.2 Ensure Systemd Service Files Are Enabled

- Enable and configure systemd service files to manage PostgreSQL services.

```
# Check direcktly postgresql service if you do not use service that run postgre
systemctl status postgresql-13.service

# In case using other service such us patroni, you can check other service
systemctl status patroni.service
# Do not forget patroni default preset is disable
# so you have to enable your service due to the fact that machine boot
# Service running and active
● patroni.service - Runners to orchestrate a high-availability PostgreSQL
     Loaded: loaded (/usr/lib/systemd/system/patroni.service; enabled; preset:
     Active: active (running) since Wed 2024-04-24 14:55:46 +03; 1 month 5 days
   Main PID: 1151 (patroni)
      Tasks: 24 (limit: 203328)
     Memory: 16.1G
        CPU: 2d 11h 58min 56.969s
     CGroup: /system.slice/patroni.service
```

1.3 Ensure Data Cluster Initialized Successfully

- **1.3.1 Check Initialization of the PGDATA:** Verify that the data directory (PGDATA) is initialized properly.

```
#directory list command for On Red Hat Based Systems.

sudo ls -l /pg_data/data/ #you can chechk your directory with sql "SHOW data_di

# You can see like this output
-rw-------  1 postgres postgres   179 Mar 25 23:03 backup_label.old
drwx------ 63 postgres postgres  4096 May 30 09:24 base
-rw-------  1 postgres postgres    52 May 30 00:00 current_logfiles
drwx------  2 postgres postgres  4096 May 30 09:31 global
drwx------  2 postgres postgres    32 Feb  7 15:30 log
-rw-------  1 postgres postgres  1896 May 27 15:53 patroni.dynamic.json
drwx------  2 postgres postgres     6 Feb  7 15:30 pg_commit_ts
drwx------  2 postgres postgres     6 Feb  7 15:30 pg_dynshmem
-rw-------  1 postgres postgres  5996 May  6 14:51 pg_hba.conf
-rw-------  1 postgres postgres  5996 May  7 11:33 pg_hba.conf.backup
-rw-------  1 postgres postgres  1636 Mar 25 23:01 pg_ident.conf
```

```
-rw-------  1 postgres postgres  1636 May  7 11:33 pg_ident.conf.backup
drwx------  4 postgres postgres    84 May 30 09:59 pg_logical
drwx------  4 postgres postgres    48 Feb  7 15:30 pg_multixact
drwx------  2 postgres postgres     6 Feb  7 15:30 pg_notify
drwx------  4 postgres postgres    50 May  8 13:43 pg_replslot
drwx------  2 postgres postgres     6 Feb 12 14:42 pg_serial
drwx------  2 postgres postgres     6 Feb 12 14:33 pg_snapshots
drwx------  2 postgres postgres     6 May  7 11:33 pg_stat
drwx------  2 postgres postgres  4096 May 30 10:20 pg_stat_tmp
drwx------  2 postgres postgres    26 May 25 11:50 pg_subtrans
drwx------  2 postgres postgres     6 Feb 27 15:42 pg_tblspc
drwx------  2 postgres postgres     6 Feb 12 14:41 pg_twophase
-rw-------  1 postgres postgres     3 Feb  7 15:30 PG_VERSION
lrwxrwxrwx  1 postgres postgres     7 Feb  7 15:30 pg_wal -> /pg_wal
drwx------  2 postgres postgres    26 Feb 12 14:37 pg_xact
-rw-------  1 postgres postgres    88 Mar 25 23:03 postgresql.auto.conf
-rw-------  1 postgres postgres 28098 Mar 25 23:03 postgresql.base.conf
-rw-------  1 postgres postgres 28098 May  7 11:33 postgresql.base.conf.backup
-rw-r--r--  1 postgres postgres  2475 May 27 15:53 postgresql.conf
-rw-r--r--  1 postgres postgres  2475 May  7 11:33 postgresql.conf.backup
-rw-------  1 postgres postgres   433 May  7 11:33 postmaster.opts
-rw-------  1 postgres postgres    99 May  8 13:43 postmaster.pid
```

- **1.3.2 Check Version in PGDATA:** Ensure the PostgreSQL version in the data directory matches the installed version.

```
cat /pg_data/data/PG_VERSION
# Output
13

psql --version
# Output
psql (PostgreSQL) 13.13
```

- **1.3.3 Ensure Data Cluster Has Checksum Enabled:** Enable checksums for data integrity.

```
#Please do not forget to use postgres user
pg_controldata /pg_data/data/ | grep "Data page checksum version"

#Output
```

```
Data page checksum version:           1
# If it says 1, checksums are enabled. If it says 0, checksums are not enabled.
```

- **1.3.4 Ensure WALs and Temporary Files Are Not on the Same Partition as the PGDATA:** Separate write-ahead logs (WALs) and temporary files from the data directory.

```
lvs #Logical volume size command
# Output
  LV         VG     Attr       LSize    Pool Origin Data%  Meta%  Move Log Cpy%
  pg_data_lv datavg -wi-ao----   <2.25t
  pg_log_lv  datavg -wi-ao----  300.00g
  pg_temp_lv datavg -wi-ao----  300.00g
  pg_wal_lv  datavg -wi-ao---- <172.00g
  homelv     rootvg -wi-ao----   10.00g
  rootlv     rootvg -wi-ao----   20.00g
  swaplv     rootvg -wi-ao----   20.00g
  varlv      rootvg -wi-ao----   10.00g

pvs #Physical volume size command
# Output
  PV         VG     Fmt  Attr PSize   PFree
  /dev/sda2  rootvg lvm2 a--  <99.00g <39.00g
  /dev/sdb1  datavg lvm2 a--   <3.00t      0

lsblk # list block devices
# Output
NAME                  MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda                     8:0    0  100G  0 disk
├─sda1                  8:1    0    1G  0 part /boot
└─sda2                  8:2    0   99G  0 part
  ├─rootvg-rootlv     253:0    0   20G  0 lvm  /
  ├─rootvg-swaplv     253:1    0   20G  0 lvm  [SWAP]
  ├─rootvg-varlv      253:6    0   10G  0 lvm  /var
  └─rootvg-homelv     253:7    0   10G  0 lvm  /home
sdb                     8:16   0    3T  0 disk
└─sdb1                  8:17   0    3T  0 part
  ├─datavg-pg_wal_lv  253:2    0  172G  0 lvm  /pg_wal
  ├─datavg-pg_log_lv  253:3    0  300G  0 lvm  /pg_log
  ├─datavg-pg_temp_lv 253:4    0  300G  0 lvm  /pg_temp
  └─datavg-pg_data_lv 253:5    0  2.2T  0 lvm  /pg_data
sr0                    11:0    1 1024M  0 rom

#This example did not pass our exam due to logical volume partitioning, but not
```

- **1.3.5 Ensure the PGDATA Partition Is Encrypted (Manual):** Encrypt the partition containing the data directory.

```
lsblk -o NAME,FSTYPE,FSVER,LABEL,UUID,FSAVAIL,FSUSE%,MOUNTPOINT

NAME                  FSTYPE      FSVER    LABEL UUID
sda
├─sda1                xfs                       3cd2b87c-ddbe-49b1-a770-80001f
└─sda2                LVM2_member LVM2 001      dptPYc-9Hxh-GD5C-Jaa1-1zhI-E3L
  ├─rootvg-rootlv     xfs                       fdd0848b-a7fe-4754-a726-c86253
  ├─rootvg-swaplv     swap        1             18b5352c-7a6a-4d5b-b9ac-2ad02a
  ├─rootvg-varlv      xfs                       03ea089a-1a53-48b9-b9cc-042a54
  └─rootvg-homelv     xfs                       dba1b3ee-c531-4701-a793-c55915
sdb
└─sdb1                LVM2_member LVM2 001      I2xKSO-853B-eu6f-HMIc-lsbT-f5C
  ├─datavg-pg_wal_lv  xfs                       03ce8167-0476-4581-9e95-633e3b
  ├─datavg-pg_log_lv  xfs                       e48f8a8e-75ba-41f1-93a8-1978e8
  ├─datavg-pg_temp_lv xfs                       74438755-9e98-4bfb-8baf-e837d1
  └─datavg-pg_data_lv xfs                       d8bfaded-6e81-48b2-a32a-2eb6e4
sr0

"""
Check the "FSTYPE" column to see the file system type of the partition.
If it's encrypted
it will likely be a type such as "crypto_LUKS" or "crypt"
instead of a specific file system like "ext4" or "xfs".
"""
#This example did not pass our exam
#We may need to encrypt it manually using tools like LUKS
#(Linux Unified Key Setup) or other encryption mechanisms supported by your Lir

"""
To manually encrypt a partition using LUKS (Linux Unified Key Setup)
"""
sudo dnf install cryptsetup
sudo umount /mnt/data
sudo cryptsetup luksFormat /dev/sdXn
#Open the Encrypted Partition
sudo cryptsetup open /dev/sdXn cryptdata
#Create a File System on the Encrypted Partition
sudo mkfs.ext4 /dev/mapper/cryptdata
sudo mkdir /mnt/cryptdata
#Mount the Encrypted Partition
sudo mount /dev/mapper/cryptdata /mnt/cryptdata
#Update /etc/fstab and /etc/crypttab
sudo nano /etc/crypttab
#Add a line like this:
cryptdata /dev/sdXn none luks
sudo nano /etc/fstab
#Edit /etc/fstab to include the mount point:
```

```
sudo nano /etc/fstab
# Add a line like this:
/dev/mapper/cryptdata /mnt/cryptdata ext4 defaults 0 2
sudo systemctl start cryptsetup.target
sudo mount -a
```

1.4 Ensure PostgreSQL Versions Are Up-to-Date

- Regularly update PostgreSQL to the latest stable version to mitigate vulnerabilities.

```
if you do not have internet connection you can skip this step
```

1.5 Ensure Unused PostgreSQL Extensions Are Removed

- Remove any unnecessary extensions to reduce the attack surface.

```
postgres=# \dx
                                    List of installed extensions
        Name         | Version |   Schema   |                              Descr
---------------------+---------+------------+---------------------------------
 btree_gist          | 1.5     | public     | support for indexing common datat
 dblink              | 1.2     | public     | connect to other PostgreSQL databa
 pg_profile          | 4.3     | profile    | PostgreSQL load profile repository
 pg_qualstats        | 2.1.0   | public     | An extension collecting statistics
 pg_stat_kcache      | 2.2.2   | public     | Kernel statistics gathering
 pg_stat_statements | 1.8     | public     | track planning and execution stati
 pg_wait_sampling    | 1.1     | public     | sampling based statistics of wait
 plpgsql             | 1.0     | pg_catalog | PL/pgSQL procedural language
(8 rows)


postgres=# SELECT * FROM pg_extension;
  oid   |      extname       | extowner | extnamespace | extrelocatable | extve
--------+--------------------+----------+--------------+----------------+------
  13486 | plpgsql            |       10 |           11 | f              | 1.0
 707724 | dblink             |       10 |         2200 | t              | 1.2
 707770 | pg_profile         |       10 |       707723 | f              | 4.3
  17717 | btree_gist         |       10 |         2200 | t              | 1.5
  18340 | pg_stat_statements |       10 |         2200 | t              | 1.8
  18354 | pg_stat_kcache     |       10 |         2200 | t              | 2.2.2
  18367 | pg_qualstats       |       10 |         2200 | f              | 2.1.0
  18407 | pg_wait_sampling   |       10 |         2200 | t              | 1.1
```

```
(8 rows)
#you can check your extension two method. If you do not use anymore please drop
```

1.6 Ensure Tablespace Location Is Not Inside the PGDATA

- Configure tablespaces to reside outside the main data directory.

```
/*
Tablespaces can be used mainly for two purposes:
(1) “extend” the capacity of the disk where the data directory was created by a
(2) to improve the performance of some operations and leverage the capacities c
*/

\db+ /*Chech your tablespaces*/

CREATE TABLESPACE newspace LOCATION '/ssd1/postgres/newspace';
CREATE DATABASE cbsm TABLESPACE newspace;
```

Proper installation and timely patching are the first lines of defense in protecting your PostgreSQL environment. By following a structured checklist, you can minimize the risk of security gaps that attackers could exploit. Staying diligent with these foundational security practices ensures your database remains resilient against known vulnerabilities and helps maintain a secure environment. To further enhance your understanding of PostgreSQL security, I recommend reading the next article in this series, “*Postgres Security 101: Directory and File Permissions (2/8)*”, where we discuss how file permissions play a critical role in database security. 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

No responses yet

Gvadakte

What are your thoughts?

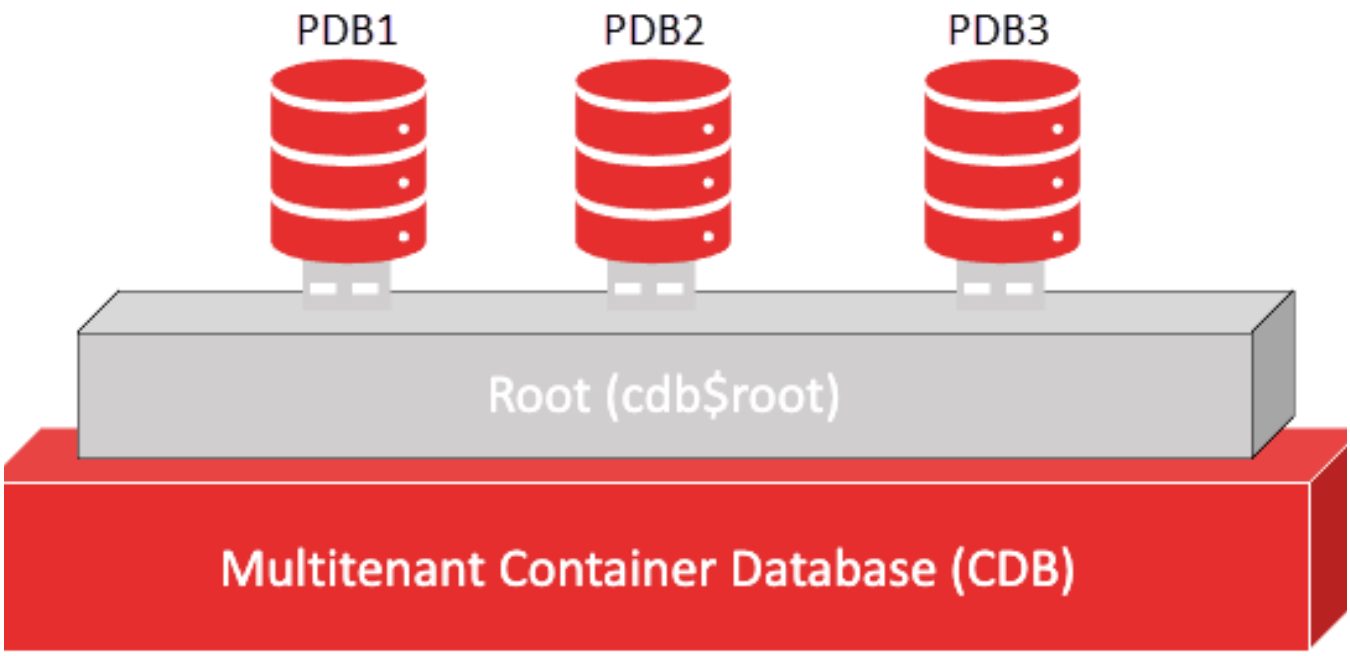
More from Oz

Oz

Installing Percona Monitoring & Management (PMM) with Postgres

Introduction:

Sep 26, 2024 54 1

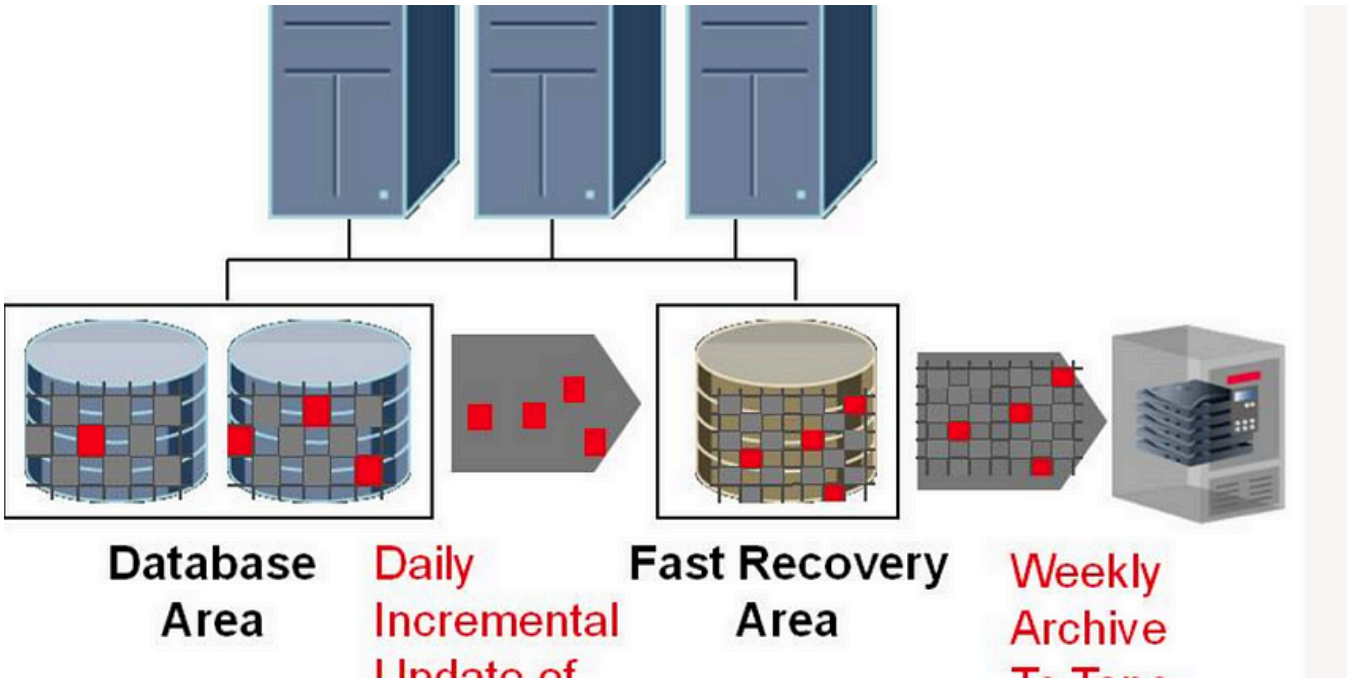


Oz

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

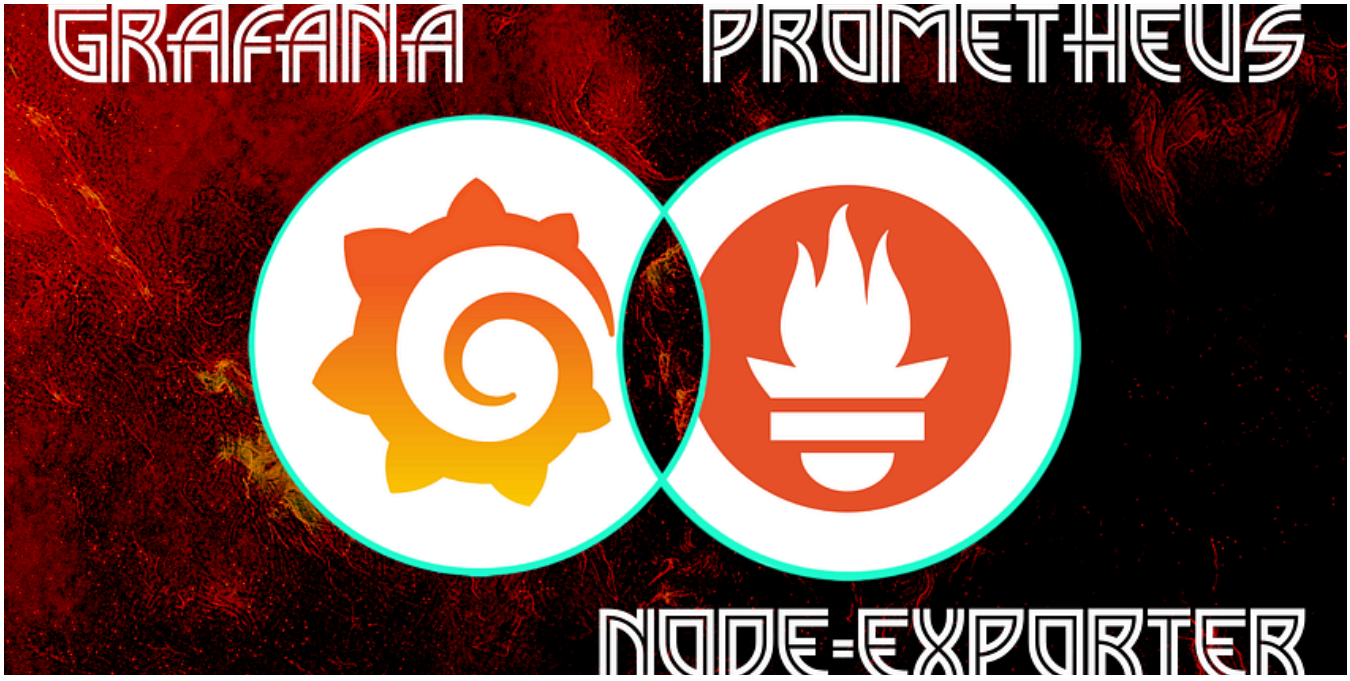
Recommended from Medium

Tihomir Manushev

Vector Search with pgvector in PostgreSQL

Simple AI-powered similarity search

Mar 9



crptcpchk

Grafana, Prometheus & Node-Exporter | Setup Guide

Grafana open source software enables you to query, visualize, alert on, and explore your metrics, logs, and traces wherever they are stored...

Oct 30, 2024 8 1

mehmetozanguven

Running PostgreSQL with Podman

Instead of running PostgresSQL locally, we can easily run with Podman. Here are the basic steps you should follow.

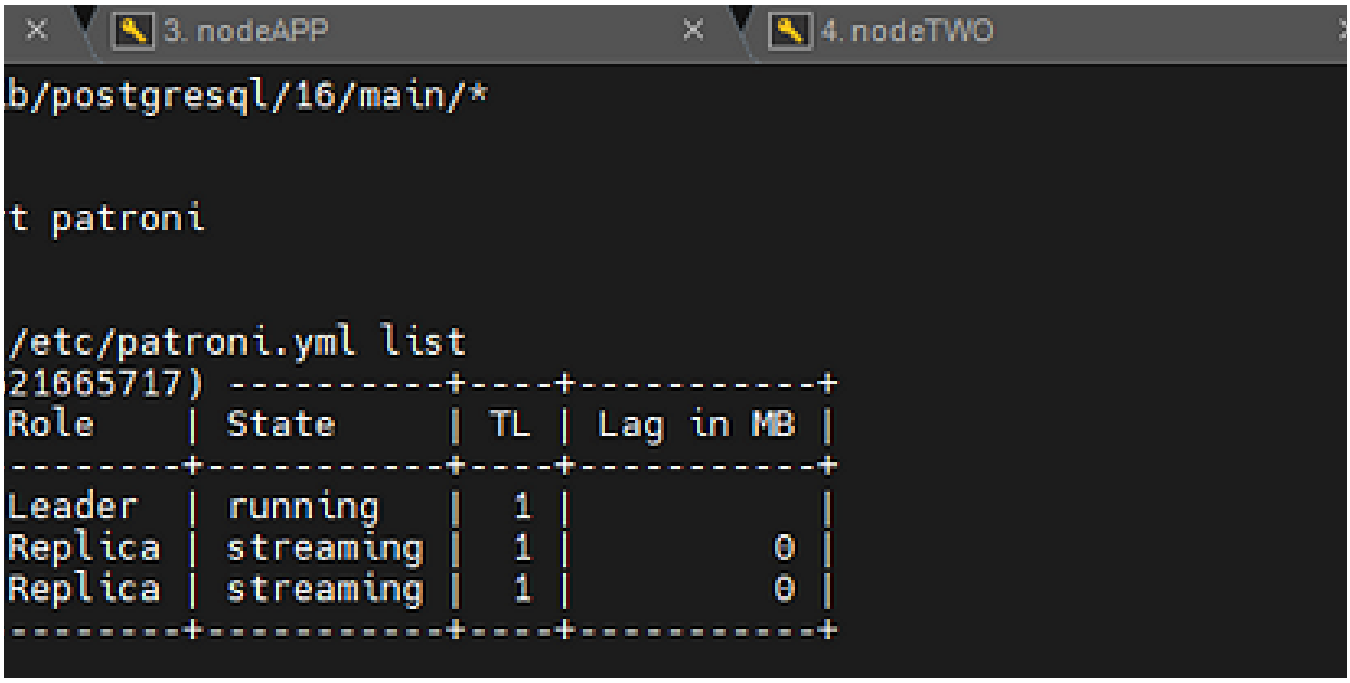
Mar 28 2

In Timescale by Team Timescale

State of PostgreSQL 2024: PostgreSQL and AI

A sneak peek into how PostgreSQL developers are using AI, courtesy of the 2024 State of PostgreSQL survey.

Dec 26, 2024



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