

PostgreSQL auf LXC installieren, konfigurieren & absichern

PostgreSQL ist ein leistungsfähiges, objektrelationales Open-Source-Datenbanksystem mit Fokus auf Stabilität, Erweiterbarkeit und strikte SQL-Standardkonformität.

Warum PostgreSQL und nicht MariaDB/MySQL?

Während Systeme wie MariaDB (oder MySQL) für einfache Webanwendungen (z. B. den LAMP-Stack) eine ausgezeichnete und schnelle Wahl sind, stellt das **PDAL-Projekt** höhere Anforderungen an die Datenintegrität, Flexibilität und erweiterte Analysefunktionen.

PostgreSQL wird oft als das technisch fortschrittlichere System angesehen und ist die bevorzugte Wahl für komplexe Datenanalyse- und kritische Unternehmensanwendungen.

Feature	PostgreSQL	MariaDB/MySQL	Relevanz für PDAL
Erweiterbarkeit	Überragend (z.B. PostGIS, JSONB, TimescaleDB, FDWs)	Basisfunktionen (Plugins)	ENTSCHEIDEND: Ermöglicht komplexe Analysen (GIS, Zeitreihen, NoSQL-Daten).
Datenintegrität	Sehr strikt (volle ACID-Compliance, robuste MVCC)	Gut, aber weniger strikt in einigen Konfigurationen	HOHE Priorität: Stellt die Zuverlässigkeit der Analysedaten sicher.
Moderne Datentypen	Native Unterstützung für JSONB (indizierbar), Arrays, HStore.	JSON-Unterstützung ist weniger flexibel und indizierbar.	WICHTIG: Umgang mit semi-strukturierten Daten ohne externe NoSQL-DB.
Lizenz	Liberale BSD-Lizenz	GPL/Kommerzielle Lizzenzen (gemischtes Modell)	GARANTIE: Langfristige Stabilität und 100% Open Source.

Fazit: Die Entscheidung für PostgreSQL stellt sicher, dass das PDAL-Projekt eine **Datenplattform** nutzt, die nicht nur relationale Standardaufgaben bewältigt, sondern auch für die zukünftigen Anforderungen der Datenanalyse – einschließlich komplexer Geodaten, Zeitreihen und unstrukturierter Daten – bestens gerüstet ist.

Aber wie bei allen Entscheidungen: Es ist abhängig von den jeweigen Anforderungen.

Voraussetzungen

- LXC-Container mit Ubuntu 20.04, 22.04 oder 24.04
- Root- oder Sudo-Zugriff auf den Container
- Netzwerkzugriff (z. B. statische IP), in diesem Fall **192.168.137.160**

- Optional: TLS-Zertifikate oder CA-Setup für Verschlüsselung (siehe separate CA-Doku) Bitte immer das aktuellste Release installieren, nach Möglichkeit immer aus der Paketverwaltung `apt` direkt.

❖ 1. Installation

```
sudo apt update  
sudo apt install -y postgresql-16 postgresql-contrib
```

```
pdal@postgresql160:~$ sudo apt install postgresql-16 postgresql-contrib  
[sudo] password for pdal:  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following additional packages will be installed:  
  libcommon-sense-perl libgdbm-compat4t64 libjson-perl libjson-xs-perl libldap-comm  
  libtypes-serialiser-perl libxslt1.1 perl perl-modules-5.38 postgresql-client-16 p  
Suggested packages:  
  perl-doc libterm-readline-gnu-perl | libterm-readline-perl-perl make libtap-harne  
The following NEW packages will be installed:  
  libcommon-sense-perl libgdbm-compat4t64 libjson-perl libjson-xs-perl libldap-comm  
  libtypes-serialiser-perl libxslt1.1 perl perl-modules-5.38 postgresql-16 postgresq  
  postgresql-contrib  
0 upgraded, 18 newly installed, 0 to remove and 0 not upgraded.  
Need to get 52.2 MB of archives.  
After this operation, 227 MB of additional disk space will be used.  
Do you want to continue? [Y/n] y
```

Nach

der Installation wird der PostgreSQL-Dienst automatisch gestartet.

Mit der Option `postgresql-16` installieren wir die Version 16 von PostgreSQL. Die Standard-Version - `apt install -y postgresql` - im Ubuntu-Repository kann abweichen. `postgresql-contrib` ist ein Paket, das zusätzliche Erweiterungen und Funktionen für PostgreSQL enthält, die nicht Teil der Kerninstallation sind, aber die Datenbank erweitert.

Zum testen der Version `psql --version` eingeben.

👤 2. PostgreSQL-Benutzer & -Zugriff

Der Standard-User bei der Erstinstallation ist `postgres`.

Zugriff auf die PostgreSQL-Konsole:

```
sudo -u postgres psql
```

```
fixing permissions on existing directory /var/lib/postgresql/16/main ... ok
creating subdirectories ... ok
selecting dynamic shared memory implementation ... posix
selecting default max_connections ... 100
selecting default shared_buffers ... 128MB
selecting default time zone ... Europe/Berlin
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... ok
syncing data to disk ... ok
Setting up postgresql-contrib (16+257build1.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for libc-bin (2.39-0ubuntu8.4) ...
pdal@postgresql160:~$ sudo -u postgres psql
psql (16.9 (Ubuntu 16.9-0ubuntu0.24.04.1))
Type "help" for help.

postgres=#
```

2. PostgreSQL Benutzerrollen – Strukturierte Zuweisung von Rechten

Für eine saubere Trennung von Verantwortlichkeiten und erhöhte Sicherheit im PostgreSQL-Datenbanksystem werden drei verschiedene Benutzerrollen eingerichtet: Diese Rollen können einzelnen Usern zugewiesen werden.

1. Datenbank-Systemadministrator (z.B. **pdal**)

Dieser Benutzer besitzt **vollständige Superuser-Rechte** für das gesamte PostgreSQL-System. Er darf:

- Datenbanken systemweit anlegen und löschen
- Benutzer und Rollen verwalten
- Systemkonfiguration ändern
- Auf alle Datenbanken und deren Inhalte zugreifen

Grund:

Zentrale und übergeordnete Verwaltung des gesamten DBMS – vergleichbar mit einem root-Nutzer im Betriebssystem.

2. Datenbankadministrator je Datenbank (z. B. **dbadmin_<dbname>**, **dbadmin_<dbname>**)

Dieser Benutzer hat **vollständige Rechte innerhalb einer bestimmten Datenbank**. Er darf:

- Benutzer für seine Datenbank anlegen und verwalten
- Backups der Datenbank erstellen
- Tabellen, Views, Prozeduren und Trigger erstellen, ändern und löschen
- Daten in der Datenbank bearbeiten

Grund:

Verantwortung liegt bei der jeweiligen Fachabteilung, ohne Zugriff auf systemweite Ressourcen oder andere Datenbanken.

3. Anwendungs- bzw. Standardnutzer (z. B. **appuser_<dbname>**, **appuser_<dbname>**)

Dieser Benutzer darf **nur mit den Daten innerhalb der ihm zugewiesenen Datenbank arbeiten**. Er hat folgende Rechte:

- **SELECT**: Daten lesen
- **INSERT**: Neue Daten einfügen
- **UPDATE**: Bestehende Daten ändern
- **DELETE**: Daten löschen

Grund:

Minimale Rechte für Anwendungen oder Endnutzer, um Daten zu bearbeiten, ohne administrative Funktionen auszuführen. Dies erhöht die Sicherheit und verhindert unbeabsichtigte Strukturänderungen.

Vorteile dieser Struktur

- Klare Trennung zwischen Systemadministration, fachlicher Datenbankpflege und einfacher Datenverwendung
- Sicherheitsprinzipien wie „Least Privilege“ werden eingehalten
- Rollen lassen sich klar zuordnen und dokumentieren
- Gute Skalierbarkeit und Nachvollziehbarkeit in Multi-User-Umgebungen
- Rechte zentral über Rollen verwalten und an Benutzer vergeben
- Vermeidung unnötiger Superuser-Rechte für Datenbank-Admins

PostgreSQL: Rollenbasierte Benutzerverwaltung mit Benutzer-/Rollen-Trennung

Es werden Benutzer und Rollen **sauber getrennt**. Es werden drei funktionale Rollen erstellt und anschließend einzelnen Benutzern zugewiesen:

PostgreSQL Benutzer- und Rollenverwaltung mit differenzierten Rechten

- **db_system_admin**: Systemweiter PostgreSQL-Superuser (Rolle ohne Login)
 - **db_admin**: Datenbank-Administrator für eine einzelne Datenbank, mit Recht zur Benutzerverwaltung und Objektverwaltung, aber **ohne** Datenbankerstellung
 - **standard_user**: Normaler Datenbanknutzer mit Lese- und Schreibrechten auf Daten
-

1. Rolle und Benutzer für Systemadministrator

```
CREATE ROLE db_system_admin WITH
    SUPERUSER
    CREATEROLE
    CREATEDB
    NOLOGIN;

CREATE ROLE pdal WITH
```

```
LOGIN  
PASSWORD 'JadeHS20';  
  
GRANT db_system_admin TO pdal;
```

```
pdal@postgresql160:~$ sudo -u postgres psql  
psql (16.2 (Ubuntu 16.2-0ubuntu0.24.04.1))  
Type "help" for help.
```

```
postgres=# CREATE ROLE db_system_admin WITH  
postgres-#      SUPERUSER  
postgres-#      CREATEROLE  
postgres-#      CREATEDB  
postgres-#      NOLOGIN;  
CREATE ROLE  
postgres=# ■
```

```
postgres=# CREATE ROLE pdal WITH  
postgres-#      LOGIN  
postgres-#      PASSWORD 'JadeHS20';  
CREATE ROLE  
postgres=# ■
```

```
postgres=# GRANT db_system_admin TO pdal;  
GRANT ROLE  
postgres=# ■
```

2. Rolle und Benutzer für Datenbank-Administrator (für Datenbank pdal)

```
CREATE ROLE db_admin WITH  
CREATEROLE  
NOLOGIN;  
  
CREATE ROLE dbadmin_pdal WITH  
LOGIN  
PASSWORD '1234';  
  
GRANT db_admin TO dbadmin_pdal;
```

```
postgres=# CREATE ROLE db_admin WITH
postgres-#           CREATEROLE
postgres-#           NOLOGIN;
CREATE ROLE
postgres=# CREATE ROLE dbadmin_pdal WITH
postgres-#           LOGIN
postgres-#           PASSWORD '1234';
CREATE ROLE
postgres=# GRANT db_admin TO dbadmin_pdal;
GRANT ROLE
postgres=#
```

Datenbank anlegen und Rechte vergeben (als Systemadministrator)

```
CREATE DATABASE pdal OWNER dbadmin_pdal;

GRANT ALL PRIVILEGES ON DATABASE pdal TO dbadmin_pdal;
```

```
postgres=# CREATE DATABASE pdal OWNER dbadmin_pdal;
CREATE DATABASE
postgres=# GRANT ALL PRIVILEGES ON DATABASE pdal TO dbadmin_pdal;
GRANT
postgres=#
```

Innerhalb der Datenbank pdal (Rechte für Objektverwaltung)

```
\c pdal
```

3. Standardrechte für neue Tabellen, Funktionen, Sequenzen

```
ALTER DEFAULT PRIVILEGES IN SCHEMA public
GRANT ALL ON TABLES TO dbadmin_pdal;

ALTER DEFAULT PRIVILEGES IN SCHEMA public
GRANT ALL ON SEQUENCES TO dbadmin_pdal;

ALTER DEFAULT PRIVILEGES IN SCHEMA public
GRANT ALL ON FUNCTIONS TO dbadmin_pdal;
```

```
pdal=# ALTER DEFAULT PRIVILEGES IN SCHEMA public
GRANT ALL ON TABLES TO dbadmin pdal;
ALTER DEFAULT PRIVILEGES
pdal=# ALTER DEFAULT PRIVILEGES IN SCHEMA public
GRANT ALL ON SEQUENCES TO dbadmin pdal;
ALTER DEFAULT PRIVILEGES
pdal=# ALTER DEFAULT PRIVILEGES IN SCHEMA public
GRANT ALL ON FUNCTIONS TO dbadmin pdal;
ALTER DEFAULT PRIVILEGES
pdal=# █
```

Hinweis: Das CREATEROLE Recht erlaubt dem Benutzer dbadmin_pdal, weitere Rollen (Benutzer) anzulegen und zu verwalten, jedoch keine Datenbanken.

4. Rolle und Benutzer für Standardnutzer mit Datenzugriff

```
CREATE ROLE standard_user WITH NOLOGIN;

CREATE ROLE appuser_pdal WITH
    LOGIN
    PASSWORD 'user_passwd';

GRANT standard_user TO appuser_pdal;

GRANT CONNECT ON DATABASE pdal TO appuser_pdal;
```

```
pdal=# CREATE ROLE standard_user WITH NOLOGIN;
CREATE ROLE
pdal=# CREATE ROLE appuser_pdal WITH
pdal-#     LOGIN
pdal-#     PASSWORD 'user_passwd';
CREATE ROLE
pdal=# Grant standard_user TO appuser_pdal;
GRANT ROLE
pdal=# Grant connect on database pdal to appuser_pdal;
GRANT
pdal=# █
```

Rechte für Datenzugriff in Datenbank pdal

```
\c pdal
```

-- Beispielhafte Tabelle:

```
CREATE TABLE beispiel (
    id SERIAL PRIMARY KEY,
    name TEXT,
    wert INTEGER
);
```

```
pdal=# CREATE TABLE beispiel (
pdal(#     id SERIAL PRIMARY KEY,
pdal(#     name TEXT,
pdal(#     wert INTEGER
pdal(# );
CREATE TABLE
pdal=# █
```

-- Datenrechte vergeben

```
GRANT SELECT, INSERT, UPDATE, DELETE ON beispiel TO appuser_pdal;
```

```
pdal=# GRANT SELECT, INSERT, UPDATE, DELETE ON beispiel TO appuser_pdal;
GRANT
pdal=# █
```

-- Standardrechte für neue Tabellen

```
ALTER DEFAULT PRIVILEGES IN SCHEMA public
GRANT SELECT, INSERT, UPDATE, DELETE ON TABLES TO appuser_pdal;
```

```
pdal=# ALTER DEFAULT PRIVILEGES IN SCHEMA public
GRANT SELECT, INSERT, UPDATE, DELETE ON TABLES TO appuser_pdal;
ALTER DEFAULT PRIVILEGES
pdal=# █
```

Zusammenfassung

Benutzer	Zugewiesene Rolle	Beschreibung
pdal	db_system_admin	Voller Systemzugriff (Superuser)
dbadmin_pdal	db_admin	Benutzer- und Objektverwaltung in DB pdal, keine DB-Erstellung
appuser_pdal	standard_user	Daten lesen, schreiben, löschen in DB pdal

3. nützliche PostgreSQL-Befehle

-- Benutzer anzeigen

```
\du
```

-- Datenbanken anzeigen

```
\l
```

-- Tabellen anzeigen

```
\dt
```

-- Verbindung beenden

```
\q
```

4. Passwortauthentifizierung aktivieren

Öffne:

```
sudo nano /etc/postgresql/*/main/pg_hba.conf
```

Das * in dem Befehl steht für die Versionsnummer; in diesem Fall für die 16.

```
# Database administrative login by Unix domain socket
local   all      postgres          peer

# TYPE  DATABASE        USER        ADDRESS             METHOD
# "local" is for Unix domain socket connections only
local   all      all            peer
# IPv4 local connections:
host    all      all      127.0.0.1/32      scram-sha-256
# IPv6 local connections:
host    all      all      ::1/128           scram-sha-256
# Allow replication connections from localhost, by a user with the
# replication privilege.
local   replication all            peer
host   replication all      127.0.0.1/32      scram-sha-256
host   replication all      ::1/128           scram-sha-256
```

Ändere am Ende:

```
# alter Eintrag local    all          all
peer
local  all            all          md5

# alter Eintrag host    all          all          127.0.0.1/32
scram-sha-256
host   all            all          192.168.137.0/24        md5

# alter Eintrag host    all          all          ::1/128
scram-sha-256
host   all            all          ::1/128        md5
```

```
# Database administrative login by Unix domain socket
local  all            postgres      peer

# TYPE  DATABASE      USER          ADDRESS         METHOD
# "local" is for Unix domain socket connections only
# local  all            all          peer
local  all            all          md5
# IPv4 local connections:
# host   all            all          127.0.0.1/32      scram-sha-256
host   all            all          192.168.137.0/24  md5
# IPv6 local connections:
# host   all            all          ::1/128         scram-sha-256
host   all            all          ::1/128         md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
local  replication    all          peer
```

Alternativ kann man auch scram-sha-256 verwenden, aber dann muss das Passwort entsprechend gespeichert werden.

🌐 5. Netzwerzugriff aktivieren

Öffne:

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

Ändere oder ergänze:

```
listen_addresses = '*'
```

```

#-----#
# CONNECTIONS AND AUTHENTICATION
#-----#
# - Connection Settings -
listen_addresses = '*'          # what IP address(es) to listen on;
                                # comma-separated list of addresses;
                                # defaults to 'localhost'; use '*' for
                                # (change requires restart)
                                # begin with 0 to use octal notation
port = 5432
max_connections = 100
#reserved_connections = 0
#superuser_reserved_connections = 3      # (change requires restart)
unix_socket_directories = '/var/run/postgresql' # comma-separated list of directo
                                                # (change requires restart)
#unix_socket_group = ''                  # (change requires restart)
#unix_socket_permissions = 0777

```

⌚ 6. Dienst neu starten

```

sudo systemctl restart postgresql
sudo systemctl status postgresql

```

```

pdal@postgresql160:~$ sudo systemctl restart postgresql
pdal@postgresql160:~$ sudo systemctl status postgresql
* postgresql.service - PostgreSQL RDBMS
  Loaded: loaded (/usr/lib/systemd/system/postgresql.service; enabled; preset: enabled)
  Active: active (exited) since Mon 2025-07-21 14:20:48 CEST; 13s ago
    Process: 1777 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
   Main PID: 1777 (code=exited, status=0/SUCCESS)
     CPU: 3ms

Jul 21 14:20:48 postgresql160 systemd[1]: Starting postgresql.service - PostgreSQL RDBMS..
Jul 21 14:20:48 postgresql160 systemd[1]: Finished postgresql.service - PostgreSQL RDBMS.
pdal@postgresql160:~$ 

```

🔒 6. TLS-Verschlüsselung (optional)

📄 Zertifikate vorbereiten:

Platzieren von:

CA-Zertifikat: /etc/ssl/certs/ca.cert.pem

Server-Zertifikat: /etc/ssl/certs/server.cert.pem

Private Key: /etc/ssl/private/server.key.pem

Weitere Infos: [[0650 CA-sslmitSANZertifikat]] In dieser Dokumentation ist es anhand des Beispiels Apache2 Server genau beschrieben, gleiche Vorgehensweise auch bei [PostgreSQL](#).

```

pdal@postgresql160:~$ sudo chown postgres:postgres /etc/ssl/private/server.key.pem
pdal@postgresql160:~$ sudo chmod 600 /etc/ssl/private/server.key.pem
pdal@postgresql160:~$ 

```

🔧 Konfiguration anpassen

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

```
# - SSL -
```

```
ssl = on
ssl_ca_file = '/etc/ssl/certs/ca.cert.pem'
ssl_cert_file = '/etc/ssl/certs/server.cert.pem'
#ssl_crl_file =
#ssl_crl_dir =
ssl_key_file = '/etc/ssl/private/server.key.pem'
#ssl_ciphers = 'HIGH:MEDIUM:+3DES::aNULL' # allowed SSL ciphers
#ssl_prefer_server_ciphers = on
#ssl_ecdh_curve = 'prime256v1'
#ssl_min_protocol_version = 'TLSv1.2'
#ssl_max_protocol_version = ''
#ssl_dh_params_file =
#ssl_passphrase_command =
#ssl_passphrase_command_supports_reload = off
```

Ergänzen oder aktivieren:

```
ssl = on
ssl_ca_file = '/etc/ssl/certs/ca.cert.pem'
ssl_cert_file = '/etc/ssl/certs/server.cert.pem'
ssl_key_file = '/etc/ssl/private/server.key.pem'
```

```
sudo chown postgres:postgres /etc/ssl/private/server.key.pem
sudo chmod 600 /etc/ssl/private/server.key.pem
```

⚡ Neustarten:

```
sudo systemctl restart postgresql
```

🔒 7. Authentifizierung mit Zertifikaten (Client-CA)(optional)

Auf dem Server eigene CA einrichten. siehe: [[0650 CA-sslmitSANZertifikat]]

Signierte Client-Zertifikate verteilen

In pg_hba.conf ergänzen:

```
hostssl all all 192.168.137.0/24 cert clientcert=verify-full
```

PostgreSQL-Dienst neustarten:

8. Verbindung testen (lokal & remote)

```
psql -h localhost -U appuser_pdal -d pdal
```

Remote (z. B. über Workstation):

```
psql -h 192.168.137.160 -U appuser_pdal -d pdal
```

Falls nötig, installiere:

```
sudo apt install postgresql-client-16
```

10. Automatischer Start (systemd)

Bereits aktiviert nach Installation:

```
sudo systemctl enable postgresql
```

Manuell neu starten:

```
sudo systemctl restart postgresql
```

11. Sicherheitsmaßnahmen

- *Starke Passwörter verwenden oder Public-Key-Auth*
- *Firewall aktivieren, z. B. mit ufw:*

```
sudo ufw allow from 192.168.137.0/24 to any port 5432 proto tcp
```

Mit **192.168.137.0/24** wird dem gesamten Netzwerk erlaubt auf die Datenbank zu zugreifen. Alternativ kann auch jeweils eine konkrete IP-Adresse frei gegeben werden.

- *Keine externen Verbindungen zulassen, außer explizit notwendig*

- *Zertifikatsbasierte Authentifizierung nutzen (siehe oben)*
- *Datenbank-Backups regelmäßig automatisieren mit pg_dump*

Quellen

- „PostgreSQL 16.x Documentation“, PostgreSQL Documentation. Zugegriffen: 22. Juli 2025. [Online]. Verfügbar unter: [PostgreSQL Doc](#)
 - „18.9. Secure TCP/IP Connections with SSL“, PostgreSQL Documentation. Zugegriffen: 22. Juli 2025. [Online]. Verfügbar unter: [PostgreSQL SSL](#)
-

Lizenz

Dieses Werk ist lizenziert unter der **Creative Commons - Namensnennung - Weitergabe unter gleichen Bedingungen 4.0 International Lizenz**.

[Zum Lizenztext auf der Creative Commons Webseite](#)