

How to Install PostgreSQL Database Server on Ubuntu 22.04

Learn how to install PostgreSQL database server on Ubuntu 22.04 with step-by-step instructions for setup, configuration, and basic administration tasks.

Contents

01	Introduction	3
02	Prerequisites	3
03	Install PostgreSQL Database Server on Ubuntu 22.04	4
04	Use the PostgreSQL Database Command Line Interface (CLI)	5
05	Secure the PostgreSQL Database Server with Password Authentication	7
06	Conclusion	10

Introduction

PostgreSQL is an open-source relational database management system known for its advanced features, reliability, and strong support for SQL and JSON data types. It is trusted across industries, from web applications and real-time analytics to Geographic Information Systems (GIS) and healthcare platforms, because of its flexibility, extensibility, and data integrity. When paired with Ubuntu 22.04, a long-term support (LTS) Linux distribution recognized for its stability, security, and performance, it becomes an ideal foundation for deploying scalable and production-grade database systems. Ubuntu's lightweight and developer-friendly environment, combined with PostgreSQL's robust capabilities, provides a reliable and secure platform for managing data in modern applications across cloud, virtual, and physical servers.

This article explains how to install PostgreSQL on Ubuntu 22.04 and deploy a PostgreSQL database server while securing it for access by trusted users on the server. If you're working with a different system, you can also check out our article on how to install PostgreSQL on FreeBSD 12.2.

Prerequisites

Before you begin:

- [Deploy a Ubuntu 22.04](#) on Vultr.
- Access the server using SSH.
- Create a non-root user with sudo privileges and switch to the user.
- [Update the Server](#).

Install PostgreSQL Database Server on Ubuntu 22.04

This section shows you how to install the PostgreSQL database server package, ideal for anyone installing PostgreSQL on Ubuntu 22.04, whether for development or production use.

1. Add the PostgreSQL repository to your server's APT sources.

CONSOLE

```
$ sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" > /etc/apt/sources.list.d/postgresql.list'
```

2. Import the PostgreSQL repository key to your server using the `wget` utility.

CONSOLE

```
$ wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo tee /etc/apt/trusted.gpg.d/postgresql.asc > /dev/null
```

3. Update the server packages to synchronize the new PostgreSQL repository.

CONSOLE

```
$ sudo apt update
```

4. Install PostgreSQL on your server.

CONSOLE

```
$ sudo apt install postgresql
```

5. Start the PostgreSQL database server.

CONSOLE

```
$ sudo systemctl restart postgresql
```

6. View the PostgreSQL service status and verify that it's active.

CONSOLE

```
$ sudo systemctl status postgresql
```

Output:

```
● postgresql.service - PostgreSQL RDBMS
   Loaded: loaded (/lib/systemd/system/postgresql.service; enabled; vendor preset: enabled)
   Active: active (exited) since Sun 2024-04-21 16:08:10 UTC; 13s ago
   Process: 6756 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
   Main PID: 6756 (code=exited, status=0/SUCCESS)
   CPU: 1ms
```

Use the PostgreSQL Database Command Line Interface (CLI)

The `psql` client tool creates a connection to the PostgreSQL database server with support for operations such as SQL statement execution and remote database access. Follow the steps below to use the PostgreSQL client tool to connect and access your database server.

1. Log in to the PostgreSQL Database server using the default `postgres` user.

CONSOLE

```
$ sudo -u postgres psql
```

2. Create a new sample database `hospital`.

```
PLPGSQL
```

```
psql > CREATE DATABASE hospital;
```

3. Switch to the new database.

```
PLPGSQL
```

```
psql > \c hospital
```

4. Create a new sample `doctors` table

```
PLPGSQL
```

```
psql > CREATE TABLE doctors(  
  doctor_id SERIAL PRIMARY KEY,  
  first_name VARCHAR(50),  
  last_name VARCHAR(50),  
  appointment_date DATE  
);
```

In the above SQL statement:

- The `doctor_id` column is a `PRIMARY KEY` that uniquely identifies each doctor in the `doctors` table.
- `first_name` and `last_name` columns store names in the `doctors` table.
- `appointment_date` stores the doctor's appointment date with a patient in the hospital.
- `SERIAL` generates a new `doctor_id` for each new record.

5. Insert sample data into the `doctors` table

```
PLPGSQL
```

```
psql > INSERT INTO doctors ( first_name, last_name,  
  appointment_date)  
VALUES  
( 'Ben', 'Joe', '2023-11-15'),  
( 'Carson', 'Smith', '2023-02-28'),  
( 'Donald', 'James', '2023-04-10');
```

6. Query the `doctors` table to view all available data.

```
PLPGSQL
```

```
psql > SELECT * FROM doctors;
```

Output:

```
doctor_id | first_name | last_name | appointment_date
-----+-----+-----+-----
1 | Ben       | Joe       | 2023-11-15
2 | Carson    | Smith     | 2023-02-28
3 | Donald    | James     | 2023-04-10
(3 rows)
```

7. Exit the PostgreSQL console.

```
PLPGSQL
```

```
psql > exit
```

Secure the PostgreSQL Database Server with Password Authentication

Follow the steps below to secure the PostgreSQL database server with password authentication to enable only authorized users to access databases.

1. Log in to the PostgreSQL database server.

```
CONSOLE
```

```
$ sudo -u postgres psql
```

2. Modify the default `postgres` user with a new strong password.

```
PLPGSQL
```

```
psql > ALTER USER postgres WITH PASSWORD 'strong_password';
```

3. Exit the PostgreSQL console

```
PLPGSQL
```

```
psql > exit
```

4. Back up the default `pg_hba.conf` configuration file for recovery purposes. Replace `16` with your actual database server version.

```
CONSOLE
```

```
$ sudo cp /etc/postgresql/16/main/pg_hba.conf /etc/postgresql/16/main/pg_hba.conf.0RIG
```

5. Open the `pg_hba.conf` configuration file using a text editor such as Nano

```
CONSOLE
```

```
$ sudo nano /etc/postgresql/16/main/pg_hba.conf
```

6. Find the following section within the file.

```
local all postgres peer
# TYPE DATABASE USER ADDRESS METHOD
```

7. Change the `peer` value to `password` to enable password authentication on the PostgreSQL database server.

```
local all postgres password
# TYPE DATABASE USER ADDRESS METHOD
```

8. Restart the PostgreSQL server to apply the new configuration changes.

```
CONSOLE
```

```
$ sudo systemctl restart postgresql
```

9. Log in to the PostgreSQL Database server using the `postgres` user to test the new authentication.

```
CONSOLE
```

```
$ sudo -u postgres psql
```

Enter the `postgres` user password you set earlier and press Enter to access the database server.

10. Create a new user role `db_manager` with a secure password.

```
PLPGSQL
```

```
psql > CREATE ROLE db_manager WITH LOGIN PASSWORD 'strong-  
password';
```

11. Grant the new user full privileges to the `hospital` database.

```
PLPGSQL
```

```
psql > GRANT ALL PRIVILEGES ON DATABASE hospital TO  
db_manager;
```

12. Exit the PostgreSQL database console.

```
PLPGSQL
```

```
psql > exit
```

13. Log in to the PostgreSQL database as the new user `db_manager` user to test access to the `hospital` database.

```
PLPGSQL
```

```
psql > psql -h localhost -U db_manager -d hospital
```

Enter the user password when prompted and press Enter to access the database.

Conclusion

You have installed the PostgreSQL server on Ubuntu 22.04 and used the `psql` utility to access the database server. In addition, you created a new database table with sample data, a new role with database privileges, and enabled secure user password authentication. For more information and database configurations, visit the official [PostgreSQL documentation](#).



VULTR

