How to Install and Setup PostgreSQL server on Ubuntu 20.04

by Mantas Levinas • October 8th, 2021 • LINUX

Introduction

PostgreSQL is a fully featured database management system (DBMS) with a strong emphasis on extensibility and SQL compliance. It is backed by 20 years of open-source development, and supports both SQL (relational) and JSON (non-relational) querying.

PostgreSQL is one of the most popular databases in the industry that is used for various web, mobile and analytics applications. Let's now go through a step-by-step guide of how to install and setup PostgreSQL on your Ubuntu 20.04 machine.

Add Official Repository

You may want to install PostgreSQL from an official repository, since it is updated more frequently than official Ubuntu sources.

First, you should install prerequisite software packages that will be used to download and install software certificates for a secure SSL connection.

```
sudo apt install wget ca-certificates
```

Then, get the certificate, add it to apt-key management utility and create a new configuration file with an official PostgreSQL repository address inside.

```
wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -
```

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

Install PostgreSQL

It is always a good idea to download information about all packages available for installation from your configured sources before the actual installation.

```
sudo apt update
```

Now is the time to do the actual PostgreSQL installation. This will install the latest PostgreSQL version along with the newest extensions and additions that are not yet officially part of the PostgreSQL core.

```
apt install postgresql postgresql-contrib
```

Check PostgreSQL status

After the installation you may double-check that postgresql daemon is active.

service postgresql status

The output should look like this:

```
root@ubuntu-sandbox:~ > service postgresql status
● postgresql.service - PostgreSQL RDBMS
     Loaded: loaded (/lib/systemd/system/postgresql.service; enabled; vendor preset: enabled)
     Active: active (exited) since Tue 2022-03-15 09:56:35 EET; 46s ago
   Main PID: 3046735 (code=exited, status=0/SUCCESS)
      Tasks: 0 (limit: 19175)
     Memory: 0B
     CGroup: /system.slice/postgresql.service

Mar 15 09:56:35 ubuntu-sandbox systemd[1]: Starting PostgreSQL RDBMS...
Mar 15 09:56:35 ubuntu-sandbox systemd[1]: Finished PostgreSQL RDBMS.
```

Start Using PostgreSQL Command Line Tool

When you install PostgreSQL a default admin user "postgres" is created by the default. You must use it to log-in to your PostgreSQL database for the first time.

A "psql" command-line client tool is used to interact with the database engine. You should invoke it as a "postgres" user to start an interactive session with your local database.

```
sudo -u postgres psql
```

In addition to creating a postgres admin user for you, PostgreSQL installation also creates a default database named "postgres" and connects you to it automatically when you first launch psql.

After first launching psql, you may check the details of your connection by typing \conninfo into the interpreter.

```
postgres=# \conninfo
You are connected to database "postgres" as user "postgres" via socket in "/var/run/postgresql" at port "5432".
```

You are now connected to database "postgres" as user "postgres".

If you want to see a list of all the databases that are available on a server, use \l command.

```
postgres=# \l
                                 List of databases
   Name    |  Owner   | Encoding | Collate |  Ctype  |   Access privileges
-----------+----------+----------+---------+---------+-----------------------
 postgres  | postgres | UTF8     | C.UTF-8 | C.UTF-8 |
 template0 | postgres | UTF8     | C.UTF-8 | C.UTF-8 | =c/postgres          +
           |          |          |         |         | postgres=CTc/postgres
 template1 | postgres | UTF8     | C.UTF-8 | C.UTF-8 | =c/postgres          +
           |          |          |         |         | postgres=CTc/postgres
(3 rows)
```

And to see a list of all the users with their privileges use \du command.

```
postgres=# \du
                             List of roles
 Role name |                         Attributes                         | Member of
-----------+------------------------------------------------------------+-----------
 postgres  | Superuser, Create role, Create DB, Replication, Bypass RLS | {}
```

Since the default "postgres" user does not have a password, you should set it yourself.

```
\password postgres
```

Create and Populate a New Database

You are now connected to your database server through psql command line tool with full access rights, so it's time to create a new database.

```
CREATE DATABASE test_erp;
```

After the new "test_erp" database is created, connect to it.

```
\c test_erp
```

Now you are ready to start creating tables where your data will be stored. Let's create your first table with a primary key, and three client attributes.

```
CREATE TABLE clients (id SERIAL PRIMARY KEY, first_name VARCHAR, last_name VARCHAR, role VARCHAR);
```

You may double check that your new table is created successfully by typing a \dt command.

```
test_erp=# \dt
         List of relations
 Schema |  Name   | Type  |  Owner
--------+---------+-------+----------
 public | clients | table | postgres
(1 row)
```

Let's now insert the first row into your newly created "clients" table.

```
INSERT INTO clients (first_name, last_name, role) VALUES ('John', 'Smith', 'CEO');
```

And query the table to get all its rows.

```
SELECT * FROM clients;
```

```
test_erp=# SELECT * FROM clients;
 id | first_name | last_name | role
----+------------+-----------+------
  1 | John       | Smith     | CEO
(1 row)
```

As you can see, John Smith has been successfully added to the "clients" table of the "test_erp" database.

Setup PostgreSQL server

It's fun to play with the database locally, but eventually you will need to connect to it through a remote server.

When you install a PostgreSQL server, it is only accessible locally through the loopback IP address of your machine. However, you may change this setting in the PostgreSQL configuration file to allow remote access.

Let's now exit the interactive psql session by typing exit, and access `postgresql.conf` configuration file of PostgreSQL version 14 by using vim text editor.

```
vim /etc/postgresql/14/main/postgresql.conf
```

Uncomment and edit the listen_addresses attribute to start listening to start listening to all available IP addresses.

```
listen_addresses = '*'
```

Now edit the PostgreSQL access policy configuration file.

```
vim /etc/postgresql/14/main/pg_hba.conf
```

Append a new connection policy (a pattern stands for `[CONNECTION_TYPE][DATABASE][USER] [ADDRESS][METHOD]`) in the bottom of the file.

```
host all all 0.0.0.0/0 md5
```

We are allowing TCP/IP connections (`host`) to all databases (`all`) for all users (`all`) with any IPv4 address (`0.0.0.0/0`) using an MD5 encrypted password for authentication (`md5`).

It is now time to restart your PostgreSQL service to load your configuration changes.

```
systemctl restart postgresql
```

And make sure your system is listening to the 5432 port that is reserved for PostgreSQL.

```
ss -nlt | grep 5432
```

```
root@ubuntu-sandbox:~ > ss -nlt | grep 5432
LISTEN  0        244              0.0.0.0:5432          0.0.0.0:*
LISTEN  0        244                 [::]:5432             [::]:*
```

If everything is OK, you should see this output.

Connect to PostgreSQL database through a remote host

Your PostgreSQL server is now running and listening for external requests. It is now time to connect to your database through a remote host.

Connect via Command Line Tool

A psql command line tool also allows you to connect to a remote database. If you don't have it on your remote machine yet, follow the steps 1 – 3 for a full PostgreSQL installation or install a command line tool only by using sudo apt install postgresql-client command.

You may now connect to a remote database by using the following command pattern:

```
psql -h [ip address] -p [port] -d [database] -U [username]
```

Let's now connect to a remote PostgreSQL database that we have hosted on one of the Cherry Servers machines.

```
psql -h 5.199.162.56 -p 5432 -d test_erp -U postgres
```

To double check your connection details use the `\conninfo` command.

```
test_erp=# \conninfo
You are connected to database "test_erp" as user "postgres" on host "5.199.162.56" at port "5432".
```

Now you can start writing SQL queries to retrieve data from your database tables.

```
SELECT * FROM clients;
```

We can see that our previously created entry is safely stored in the "clients" table.

Article contents

```
test_erp=# SELECT * FROM clients;
 id | first_name | last_name | role
----+------------+-----------+------
  1 | John       | Smith     | CEO
(1 row)
```

Connect via Application Code

To connect to a database through your application code you need a database driver that allows you to connect to a specific database from your chosen programming language.

If you are using Python, a standard PostgreSQL driver is psycopg2. Let's install this library using pip packet manager.

```
pip install psycopg2-binary
```

You can now import psycopg2 into your code and start using PostgreSQL natively.

```
import psycopg2

# Connect to your PostgreSQL database on a remote server
conn = psycopg2.connect(host="5.199.162.56", port="5432", dbname="test_erp", user="postgres", password

# Open a cursor to perform database operations
cur = conn.cursor()

# Execute a test query
cur.execute("SELECT * FROM clients")

# Retrieve query results
records = cur.fetchall()

# Finally, you may print the output to the console or use it anyway you like
print(records)
```

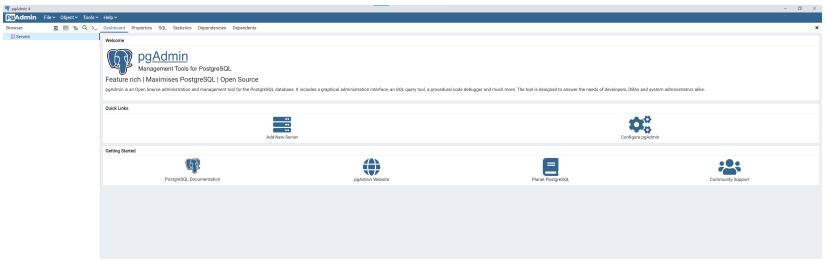
You will get the following output when using an ipython3 interpreter:

```
┌─brex@DESKTOP-DPAC4SS in ~
└$ ipython3
Python 3.8.10 (default, Sep 28 2021, 16:10:42)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.13.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: import psycopg2

In [2]: conn = psycopg2.connect(host="5.199.162.56", port="5432", dbname="test_erp", user="postgres", password="test123")

In [3]: cur = conn.cursor()

In [4]: cur.execute("SELECT * FROM clients")

In [5]: records = cur.fetchall()

In [6]: print(records)
[(1, 'John', 'Smith', 'CEO')]
```

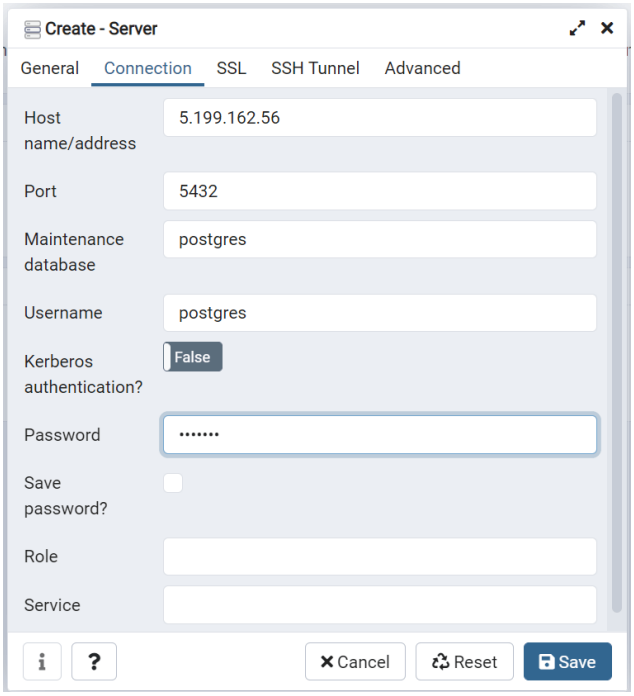
Connect via GUI Client

Although there are many GUI clients that can help you connect to a database and manage it, pgAdmin is probably the most popular option for PostgreSQL, and we highly recommend using it.

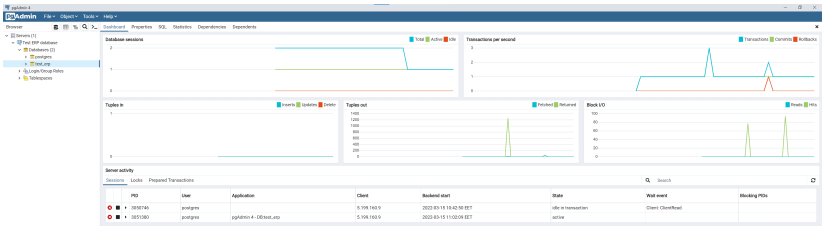
After installing pgAdmin 4 and running it you will get to a standard pgAdmin 4 darshboard.



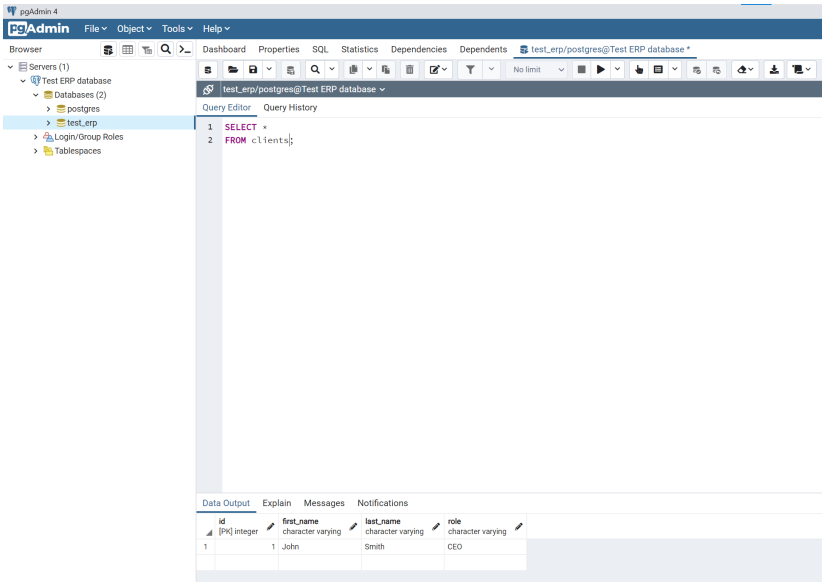
Press Add New Server button and enter the information of your remote server.



After saving your credentials you will be automatically connected to a remote database server.



You may now open a Query Tool for your selected database test_erp and start writing your queries.



Conclusion

Congrats! You have successfully installed a PostgreSQL database, set-up a database server and started interacting with it through a remote machine. It is now time to dive deeper into the official PostgreSQL documentation to build your application.

Mantas Levinas
Marketing Coordinator

Helping engineers learn 💡 about new technologies and ingenious IT automation use cases to build better systems 💻

Share this article

Twitter Facebook LinkedIn

Join Cherry Servers Community

Get monthly practical guides about building more secure, efficient and easier to scale systems on an open cloud ecosystem.

Enter your email

Subscribe

I agree to receive marketing communication as per privacy policy

Related Articles

LINUX

Can DDoS attacks harm your server and how to prevent them?

Aug 21, 2019 • by Marius Rimkus

DDoS attacks can cause real harm to your system. Learn some practical examples of how to lower the risk and how to prevent ddos attacks.

LINUX

How To Back Up A Linux Server

Jan 21, 2020 • by Marius Rimkus

Short tutorial how to backup and restore your Linux server using TAR method and Bera backup tools.

LINUX

openSUSE Leap Review: What's New in 15.2?

May 17, 2021 • by Mantas Levinas

openSUSE Leap 15.2 is one of the most versatile and stable distributions available today, offering powerful tools for system developers and home users alike.

Ready to get started?

Create an Account