Install PostgreSQL 13 on Arch | Manjaro | Garuda Linux

By**Frankline Bett**- September 8, 2021 Modified date: September 8, 2021

PostgreSQL is an open-source object-relational database system that can reliably store and grow even the most complex data demands. Users may rely on the PostgreSQL database system for its dependability, data integrity, extensive feature set, and flexibility. To assure innovation and a solid reputation, it has a huge community behind it. Thousands of businesses rely on PostgreSQL to support financial transactions, massive website traffic, and e-commerce systems, among other things. Custom functions written in programming languages such as Java, Python, C/C++, and others can also be added.

In this tutorial, we'll look at how to install PostgreSQL 13 on Arch Linux | Manjaro | Garuda Linux.

Where is PostgreSQL used?

The following list illustrates some of the ways PostgreSQL is or may be utilized in the real world scenarios.

- **GIS data from the government:** PostgreSQL offers a sophisticated GIS extension called "PostGIS" that contains hundreds of functions for processing geometric data in various formats. PostGIS is one of the power standards in the Open Source GIS industry, as it is extremely standard compliant.
- **Financial sector:** PostgreSQL is well-suited to the financial sector. PostgreSQL is completely ACID compliant, making it excellent for OLTP workloads (Online Transaction Processing).
- Data generated by scientific initiatives might amount to terabytes, which must be managed in the most useful and effective manner feasible. PostgreSQL has excellent analytical capabilities and a robust SQL engine that makes handling huge volumes of data a breeze.
- **Web technologies and NoSQL workloads:** To service your consumers, modern websites may demand thousands or even hundreds of thousands of requests per second. Scalability is a key concern, and the PostgreSQL community has been working hard to solve such concerns in recent years.
- **Manufacturing:** Many world-class industrial manufacturers utilize PostgreSQL as a storage backend to speed up innovation and drive growth through customer-centric operations, as well as to enhance supply chain performance. PostgreSQL is a long-term data store that provides you with secure storage at a cheap cost.

Top Features of PostgreSQL database

The following are some of PostgreSQL's key features:

- Different data kinds, as well as user-defined types, are supported.
- Inheritance in tables.
- Ensures data integrity by ensuring the integrity of foreign key referential.
- High levels of protection.
- Extensibility.
- Transactions nested.
- Supports asynchronous replication and point-in-time recovery for reliability and disaster recovery.
- Locking mechanism with a high level of sophistication.

Install PostgreSQL on Arch | Manjaro | Garuda Linux

With the following steps, we will install PostgreSQL 13 on Arch | Manjaro | Garuda Linus OS.

Step 1: Update System Packages

Linux users should always upgrade their operating system packages to the most recent versions available from upstream sources.

Run the following command to update and upgrade system packages before installation:

```
sudo pacman -Syu
```

Reboot the system:

```
sudo reboot
```

Step 2: Install PostgreSQL 13 on Arch | Manjaro | Garuda Linux

Following the reboot, install PostgreSQL from the AUR on Arch | Manjaro | Garuda Linux:

```
sudo pacman -S postgresql vim
```

Accept the installation prompt to proceed:

```
resolving dependencies...
looking for conflicting packages...

Packages (2) postgresql-libs-13.3-3  postgresql-13.3-3

Total Download Size:   18.32 MiB
Total Installed Size:  60.00 MiB

:: Proceed with installation? [Y/n] y
```

We can confirm PostgreSQL installed version:

```
$ postgres --version
postgres (PostgreSQL) 13.3
```

Set applicable entries in */etc/locale.gen*:

```
echo "en_US.UTF-8 UTF-8" | sudo tee /etc/locale.gen
```

Then run ***locale-gen*** to generate locale settings:

```
$ sudo locale-gen
Generating locales...
  en_US.UTF-8... done
Generation complete.
```

Step 3: PostgreSQL Service Management

Now check PostgreSQL status with the following command:

```
$ sudo systemctl status postgresql
[sudo] password for frank:
○ postgresql.service - PostgreSQL database server
     Loaded: loaded (/usr/lib/systemd/system/postgresql.service;
disabled; vendor preset: disabled)
     Active: inactive (dead)
```

In the above output we have seen that PostgreSQL is inactive and dead. Before we can start and enable PostgreSQL, we need to initialize PostgreSQL data directory.

You must first login as the **postgres** user using the following command before you can initialize PostgreSQL's data directory:

```
sudo su - postgres
```

With the following command, you can now initialize PostgreSQL's data directory:

```
$ initdb --locale en_US.UTF-8 -D /var/lib/postgres/data
The files belonging to this database system will be owned by user
"postgres".
This user must also own the server process.

The database cluster will be initialized with locale "en_US.UTF-8".
The default database encoding has accordingly been set to "UTF8".
The default text search configuration will be set to "english".

Data page checksums are disabled.

fixing permissions on existing directory /var/lib/postgres/data ... ok
creating subdirectories ... ok
selecting dynamic shared memory implementation ... posix
selecting default max_connections ... 100
selecting default shared_buffers ... 128MB
selecting default time zone ... Africa/Nairobi
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... ok
syncing data to disk ... ok

initdb: warning: enabling "trust" authentication for local connections
You can change this by editing pg_hba.conf or using the option -A, or
--auth-local and --auth-host, the next time you run initdb.

Success. You can now start the database server using:

    pg_ctl -D /var/lib/postgres/data -l logfile start
```

Now logout postgres use and start PostgreSQL:

```
$ exit
logout
```

Start PostgreSQL:

```
sudo systemctl start postgresql
```

Then enable PostgreSQL:

```
sudo systemctl enable postgresql
```

Check if PostgreSQL is active and running:

```
$ sudo systemctl status postgresql
● postgresql.service - PostgreSQL database server
     Loaded: loaded (/usr/lib/systemd/system/postgresql.service; enabled;
vendor preset: disabled)
     Active: active (running) since Wed 2021-09-08 15:14:51 UTC; 9s ago
   Main PID: 6214 (postgres)
      Tasks: 7 (limit: 4682)
     Memory: 15.6M
     CGroup: /system.slice/postgresql.service
             ├─6214 /usr/bin/postgres -D /var/lib/postgres/data
             ├─6217 "postgres: checkpointer " "" "" "" "" "" "" "" "" ""
"" "" "" "" "" "" "" "" "" "" ""
             ├─6218 "postgres: background writer " "" "" "" "" "" "" ""
"" "" "" "" "" "" "" ""
             ├─6219 "postgres: walwriter " "" "" "" "" "" "" "" "" "" ""
"" "" "" "" "" "" "" "" "" "" "" "" ""
             ├─6220 "postgres: autovacuum launcher " "" "" "" "" "" "" ""
```

```
"" "" "" "" "" ""
            ├─6221 "postgres: stats collector " "" "" "" "" "" "" "" ""
"" "" "" "" "" "" "" "" ""
            └─6222 "postgres: logical replication launcher " "" "" "" ""

Sep 08 15:14:50 arch-linux systemd[1]: Starting PostgreSQL database
server...
Sep 08 15:14:51 arch-linux postgres[6214]: 2021-09-08 15:14:51.132 UTC
[6214] LOG:  starting PostgreSQL 13.3 on x86_64-pc-linux-gnu, compiled by
gcc (GCC) 11.1.0, 64-bit
Sep 08 15:14:51 arch-linux postgres[6214]: 2021-09-08 15:14:51.136 UTC
[6214] LOG:  listening on IPv6 address "::1", port 5432
Sep 08 15:14:51 arch-linux postgres[6214]: 2021-09-08 15:14:51.136 UTC
[6214] LOG:  listening on IPv4 address "127.0.0.1", port 5432
Sep 08 15:14:51 arch-linux postgres[6214]: 2021-09-08 15:14:51.138 UTC
[6214] LOG:  listening on Unix socket "/run/postgresql/.s.PGSQL.5432"
Sep 08 15:14:51 arch-linux postgres[6216]: 2021-09-08 15:14:51.145 UTC
[6216] LOG:  database system was shut down at 2021-09-08 15:14:30 UTC
Sep 08 15:14:51 arch-linux postgres[6214]: 2021-09-08 15:14:51.154 UTC
[6214] LOG:  database system is ready to accept connections
Sep 08 15:14:51 arch-linux systemd[1]: Started PostgreSQL database
server.
```

Now that PostgreSQL is active and running we can create databases and users as shown below.

Create Database in PostgreSQL

To create a database, connect to PostgreSQL. A default user named ‘**postgres**‘ is established when PostgreSQL is installed. Connect to this user first. We will create a database called **mydb**.

```
$ sudo su - postgres
[postgres@frank-bett ~]$
```

Now you are logged in as **postgres** user, secure this default postgres user with a strong password using the following command:

```
$ psql -c "alter user postgres with password 'StrongPassword'"
ALTER ROLE
```

To execute Postgresql commands, type the following command into the PostgreSQL command prompt:

```
$ psql
psql (13.3)
Type "help" for help.
```

Run the following command to create a database:

```
# CREATE DATABASE mydb;
CREATE DATABASE
```

Confirm created database above as well listing available databases:

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

Create User in PostgreSQL

Run the following commands to create a user **postgreuser** and allow them access to the database we created above:

```
# CREATE USER postgreuser WITH ENCRYPTED PASSWORD 'MyPassword';
CREATE ROLE

# GRANT ALL PRIVILEGES ON DATABASE mydb to postgreuser;
GRANT
```

Run the following to connect to the database we created above:

```
# \c mydb
You are now connected to database "mydb" as user "postgres".
mydb=#
```

Step 4: Changing PostgreSQL Service Port

Check the default port the PostgreSQL is listening on:

```
$ sudo netstat -nltp
[sudo] password for frank:
```

```
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
PID/Program name
tcp        0      0 0.0.0.0:22              0.0.0.0:*
LISTEN      13141/sshd: /usr/bi
tcp        0      0 127.0.0.1:5432          0.0.0.0:*
LISTEN      14450/postgres
tcp6       0      0 :::22                   :::*
LISTEN      13141/sshd: /usr/bi
tcp6       0      0 ::1:5432                :::*
LISTEN      14450/postgres
```

From the above output is running on port **5432**.

We can change PostgreSQL default service port which is **5432** by editing the `postgresql.conf` file located in ***/var/lib/postgres/data/*** directory.

```
sudo vim /var/lib/postgres/data/postgresql.conf
```

In the above file search for **#port = 5432** and change to **5433**:

```
# line 63 uncomment and change it to 5433
port = 5433                             # (change requires restart)
```

Now, restart PostgreSQL for the changes to take effect:

```
sudo systemctl restart postgresql
```

We can now confirm if PostgreSQL is listening on port **5433**:

```
$ sudo netstat -nltp
Active Internet connections (only servers)
```

```
Proto Recv-Q Send-Q Local Address           Foreign Address         State
PID/Program name
tcp        0      0 0.0.0.0:22              0.0.0.0:*
LISTEN      13141/sshd: /usr/bi
tcp        0      0 127.0.0.1:5433          0.0.0.0:*
LISTEN      20246/postgres
tcp6       0      0 :::22                   :::*
LISTEN      13141/sshd: /usr/bi
tcp6       0      0 ::1:5433                :::*
LISTEN      20246/postgres
```

PostgreSQL is now listening on port **5433**.

Step 5: Enabling Remote Database connections

Set Listen address to your server IP address or "*" for all interfaces in the file `/var/lib/postgres/data/postgresql.conf` to allow remote database connection.

```
$ sudo vim /var/lib/postgres/data/postgresql.conf
# line 59
listen_addresses = '192.168.15.65'
```

Also set PostgreSQL to accept remote connections as shown below:

```
$ sudo vim /var/lib/postgres/data/pg_hba.conf

# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     trust
# IPv4 local connections:
host    all             all             192.168.15.0/24            trust
```

```
Also for replication
local   replication    all                                    trust
host    replication    all             192.168.15.0/24           trust
```

Restart the database service after saving the changes:

```
sudo systemctl restart postgresql
```

Test connection with the **psql** command while providing **username** and optionally **databasename**.

```
$ psql -U <dbuser> -h <serverip> -p 5432 <dbname>
```

Step 6: Changing PostgreSQL Default Data Directory

By default, PostgreSQL stores its data in the `/var/lib/postgres` directory. As seen below, PostgreSQL may be configured to store data in a specified path.

Before doing so, we need to stop ***postgresql.service*** first:

```
sudo systemctl stop postgresql.service
```

Create a directory to store PostgreSQL data:

```
sudo mkdir -p /data/postgres
```

Set the ownership of the directory to `postgres` as below:

```
sudo chown postgres /data/postgres
sudo chmod 700 /data/postgres
```

Now, move PostgreSQL data from the old folder to new one:

```
sudo mv /var/lib/postgres/data /data/postgres
```

Make a symlink from PostgreSQL old folder to new one:

```
sudo ln -s /data/postgres /var/lib/postgres/data
```

Now start the `postgresql.service` :

```
sudo systemctl start postgresql.service
```

Confirm if it’s active and running:

```
$ sudo systemctl status postgresql
[sudo] password for frank:
● postgresql.service - PostgreSQL database server
     Loaded: loaded (/usr/lib/systemd/system/postgresql.service; enabled;
vendor preset: disabled)
     Active: active (running) since Wed 2021-09-01 14:13:32 EAT; 2h 30min
ago
    Process: 20244 ExecStartPre=/usr/bin/postgresql-check-db-dir
${PGROOT}/data (code=exited, status=0/SUCCESS)
   Main PID: 20246 (postgres)
      Tasks: 7 (limit: 3108)
     Memory: 23.1M
        CPU: 4.928s
     CGroup: /system.slice/postgresql.service
             ├─20246 /usr/bin/postgres -D /var/lib/postgres/data
             ├─20248 postgres: checkpointer
             ├─20249 postgres: background writer
             ├─20250 postgres: walwriter
             ├─20251 postgres: autovacuum launcher
             ├─20252 postgres: stats collector
             └─20253 postgres: logical replication launcher

Sep 01 14:13:31 frank-bett systemd[1]: Starting PostgreSQL database
server...
Sep 01 14:13:31 frank-bett postgres[20246]: 2021-09-01 14:13:31.976 EAT
[20246] LOG:  starting PostgreSQL 13.3 on x86_64-pc-linux-gnu, compiled
by gcc (GCC) 11.1.0>
Sep 01 14:13:31 frank-bett postgres[20246]: 2021-09-01 14:13:31.999 EAT
[20246] LOG:  listening on IPv6 address "::1", port 5433
Sep 01 14:13:31 frank-bett postgres[20246]: 2021-09-01 14:13:31.999 EAT
[20246] LOG:  listening on IPv4 address "127.0.0.1", port 5433
Sep 01 14:13:32 frank-bett postgres[20246]: 2021-09-01 14:13:32.155 EAT
[20246] LOG:  listening on Unix socket "/run/postgresql/.s.PGSQL.5433"
Sep 01 14:13:32 frank-bett postgres[20247]: 2021-09-01 14:13:32.513 EAT
[20247] LOG:  database system was shut down at 2021-09-01 14:13:30 EAT
Sep 01 14:13:32 frank-bett postgres[20246]: 2021-09-01 14:13:32.749 EAT
[20246] LOG:  database system is ready to accept connections
Sep 01 14:13:32 frank-bett systemd[1]: Started PostgreSQL database
server.
```

Conclusion

We have come to an end of our tutorial on how to install PostgreSQL 13 on Arch | Manjaro | Garuda Linux. We hope you found this information educative, we are bringing you more new and useful tutorial. Stay tuned.

Latest guides on our site:

- Install OpenResty Web Platform on Ubuntu & Debian
- Install Icinga 2 & Icinga Web 2 on Rocky Linux 8
- Install and Configure VNC Server on Rocky Linux 8
- How To Install MongoDB 5 on Amazon Linux 2

Your support is our everlasting motivation, that cup of coffee is what keeps us going!

As we continue to grow, we would wish to reach and impact more people who visit and take advantage of the guides we have on our blog. This is a big task for us and we are so far extremely grateful for the kind people who have shown amazing support for our work over the time we have been online.

Thank You for your support as we work to give you the best of guides and articles. Click below to buy us a coffee.

Frankline Bett