



Home



My Network



Jobs



Messaging



Notifications



Me



For Business



Installing PostgreSQL: RPM and Source Code Methods



Syed Asad hussain
PostgreSQL | Redis DBA



December 2, 2024

Installing PostgreSQL on Red Hat: RPM and Source Code Methods.

In this section, we will explore two methods for installing PostgreSQL on Red Hat-based systems: using RPM packages and compiling from source code. Understanding these methods will give you flexibility in setting up PostgreSQL according to your needs and preferences.

Method 1: Installing PostgreSQL Using RPM

Step 1: Enable the PostgreSQL Repository

Before installing PostgreSQL, you need to enable the PostgreSQL Global Development Group (PGDG) repository. This repository contains the latest versions of PostgreSQL.

```
sudo dnf install -y  
https://download.postgresql.org/pub/repos/yum/reporp  
ms/EL-8-x86_64/pgdg-redhat-repo-latest.noarch.rpm
```

Explanation: This command downloads and installs the PGDG repository RPM package, allowing you to access PostgreSQL packages.

Step 2: Install PostgreSQL

Now that the repository is enabled, you can install PostgreSQL. For example, to install PostgreSQL 16:

```
sudo dnf install -y postgresql16-server
```

Explanation: This command installs the PostgreSQL server package. The -y flag automatically answers "yes" to any prompts during installation.

Step 3: Initialize the Database

After installation, you need to initialize the database:

```
sudo /usr/pgsql-16/bin/postgresql16-setup initdb
```

Explanation: This command initializes the PostgreSQL database cluster, setting up the necessary files and directories.

Step 4: Start and Enable PostgreSQL Service

Finally, start the PostgreSQL service and enable it to start on boot:

```
1. sudo systemctl enable postgresql-16
2. sudo systemctl start postgresql-16
```

Explanation: The first command starts the PostgreSQL service, while the second command ensures that PostgreSQL starts automatically when the system boots.

Method 2: Installing PostgreSQL from Source Code

Installing PostgreSQL from source code allows you to customize the installation. This method is applicable for Red Hat

Step 1: Install Required Dependencies

Before downloading the source code, install the necessary development tools and libraries:

```
1. sudo dnf install gcc* -y
2. sudo dnf install -y readline-devel zlib-devel
   bison flex
3. sudo yum install -y postgresql16-devel
   postgresql16-contrib
4. sudo yum install -y libicu libicu-devel
```

Explanation: These commands install the essential development tools and libraries required to compile PostgreSQL from source.

Step 2: Download PostgreSQL Source Code

Next, download the latest PostgreSQL source code from the official website. You can use wget for this:

```
wget
https://ftp.postgresql.org/pub/source/v16.5/postgres
ql
16.5.tar.gz
```

Explanation: This command downloads the specified version of PostgreSQL in a compressed tarball format.

Step 3: Extract the Source Code

After downloading, extract the tarball:

```
1. tar -xzf postgresql-16.5.tar.gz
2. cd postgresql-16.5
```

Explanation: The tar command extracts the contents of the tarball, and cd changes the directory to the extracted folder.

Step 4: Configure and Compile PostgreSQL

Now, configure the build options and compile the source code:

```
1. ./configure --prefix=/usr/bin/pgsql
2. make
```

Explanation: The ./configure command prepares the build environment, and make compiles the source code into executable binaries.

Step 5: Install PostgreSQL

Finally, install PostgreSQL:

```
sudo make install
```

Explanation: This command installs the compiled binaries and libraries to the specified prefix directory.

Step 6: Setting Up and Configuring the PostgreSQL Data Directory

In this step, we will create a directory to store the PostgreSQL data, ensure that the correct user owns the directory, and set the appropriate permissions for PostgreSQL to access and use the data directory. This process is crucial for ensuring that the PostgreSQL server can read and write to its data storage area.

```
1. mkdir pgdata
2. chown -R postgres:postgres data/
3. chmod -R 0700 data/
```

Explanation: mkdir data: Creates a new directory where PostgreSQL will store its database files.

chown -R postgres:postgres data/: Ensures that the postgres user and group own the directory and its contents, allowing PostgreSQL to function correctly.

chmod -R 0700 data/: Grants **full access to postgres user** for the data directory (not recommended for production; consider using chmod 0700 for better security in production).

Step 7: Initialize and Start PostgreSQL Server

In this step, we will initialize the PostgreSQL database system, set up its necessary files, and then start the PostgreSQL server using the correct data directory. This is crucial for getting PostgreSQL up and running on your system.

```
1. su - postgres
2. /pgdata/bin/initdb -D /pgdata/data
3. /pgdata/bin/pg_ctl -D /pgdata/data -l logfile
start
4. psql -U postgres
```

Explanation: The `pg_ctl` command is used to start, stop, or restart the PostgreSQL server. In this case, we use it to start the server.

- `-D /pgdata/data` specifies the location of the database data directory, which we set earlier with `initdb`.
- `-l logfile` specifies the log file where PostgreSQL will write its output and any errors.
- `start` tells `pg_ctl` to start the PostgreSQL server using the specified data directory and log file.

After this command is executed successfully, PostgreSQL will be running, and you will be able to connect to the database using `psql` or any other PostgreSQL client.

Installing PostgreSQL on Ubuntu: RPM and Source Code Methods

In this section, we will explore two methods for installing PostgreSQL on Ubuntu: using the official APT repository and compiling from source code. This will help you understand how to set up PostgreSQL in a straightforward manner.

Method 1: Installing PostgreSQL Using APT on Ubuntu

The easiest way to install PostgreSQL on Ubuntu is by using the APT package manager. This method ensures that you get the latest stable version along with all necessary dependencies.

Step 1: Update the Package List

Before installing PostgreSQL, it's a good practice to update your package list to ensure you have the latest information about available packages.

```
syed@syed-VirtualBox:~$ sudo apt update
Reading package lists... Done
```

Explanation: This command updates the local package index with the latest changes made in the repositories.

Step 2: Install PostgreSQL

Now, you can install PostgreSQL along with the necessary client packages:

```
syed@syed-VirtualBox:~$ sudo apt install -y
postgresql postgresql-contrib
```

```

Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
postgresql-contrib is already the newest version
(17+267.pgdg22.04+1).
The following packages will be upgraded:
  postgresql
1 upgraded, 0 newly installed, 0 to remove and 122
not upgraded.
Need to get 69.9 kB of archives.
After this operation, 1,024 B of additional disk
space will be used.
Get:1 http://apt.postgresql.org/pub/repos/apt jammy-
pgdg/main amd64 postgresql all 17+267.pgdg22.04+1
[69.9 kB]
Fetched 69.9 kB in 1s (116 kB/s)
(Reading database ... 242825 files and directories
currently installed.)
Preparing to unpack
.../postgresql_17+267.pgdg22.04+1_all.deb ...
Unpacking postgresql (17+267.pgdg22.04+1) over
(16+257.pgdg22.04+1) ...
Setting up postgresql (17+267.pgdg22.04+1) ...

```

Explanation: This command installs the PostgreSQL server and additional utilities (postgresql-contrib) that provide extra features and functionalities.

Step 3: Start and Enable PostgreSQL Service

After installation, you need to start the PostgreSQL service and enable it to start automatically on boot:

```

sudo systemctl start postgresql
sudo systemctl enable postgresql

```

Explanation: The first command starts the PostgreSQL service, while the second command ensures that PostgreSQL starts automatically when the system boots.

Step 4: Verify the Installation

To verify that PostgreSQL is running, you can check the status of the service:

```

syed@syed-VirtualBox:~$ sudo systemctl status
postgresql
● postgresql.service - PostgreSQL RDBMS
   Loaded: loaded
   (/lib/systemd/system/postgresql.service; enabled;
   vendor pr>
   Active: active (exited) since Sun 2024-12-01
14:23:30 IST; 2h 48min ago
   Main PID: 1816 (code=exited, status=0/SUCCESS)
   CPU: 4ms

Dec 01 14:23:30 syed-VirtualBox systemd[1]: Starting
PostgreSQL RDBMS...

```

```
Dec 01 14:23:30 syed-VirtualBox systemd[1]: Finished  
PostgreSQL RDBMS.
```

Explanation: This command displays the current status of the PostgreSQL service, confirming whether it is active and running.

Method 2: Installing PostgreSQL from Source Code

If you want more control over the installation or need a specific version of PostgreSQL, you can compile it from source. This method is slightly more complex but allows for customization.

Step 1: Install Required Dependencies

Before downloading the source code, you need to install the necessary development tools and libraries:

```
sudo apt install -y build-essential libreadline-dev  
zlib1g-dev flex bison
```

Explanation: This command installs essential development tools and libraries required to compile PostgreSQL from source.

Step 2: Download PostgreSQL Source Code

Next, download the latest PostgreSQL source code from the official website. You can use `wget` for this:

```
wget  
https://ftp.postgresql.org/pub/source/v16.5/postgres  
ql-16.5tar.gz
```

Explanation: This command downloads the specified version of PostgreSQL in a compressed tarball format.

Step 3: Extract the Source Code

After downloading, extract the tarball:

```
1. tar -xzf postgresql-16.5.tar.gz  
2. cd postgresql-16.5
```

Explanation: The `tar` command extracts the contents of the tarball, and `cd` changes the directory to the extracted folder.

Step 4: Configure and Compile PostgreSQL

Now, configure the build options and compile the source code:

```
1. ./configure  
2. make
```

Explanation: The `./configure` command prepares the build environment, specifying where to install PostgreSQL. The `make` command compiles the source code into executable binaries.

Step 5: Install PostgreSQL

Finally, install PostgreSQL:

```
make install
```

Explanation: This command installs the compiled binaries and libraries to the specified prefix directory.

Step 6: Initialize the Database

After installation, you need to initialize the database:

```
/usr/local/pgsql/bin/initdb -D /usr/local/pgsql/data
```

Explanation: This command initializes the PostgreSQL database cluster, setting up the necessary files and directories.

Step 7: Start PostgreSQL

To start the PostgreSQL server, use the following command:

```
/usr/local/pgsql/bin/pg_ctl -D /usr/local/pgsql/data  
start
```

Explanation: This command starts the PostgreSQL server using the specified data directory.

Creating a Database and Table

Once you have PostgreSQL installed and running, you can start creating databases and tables. Here's a quick guide to get you started:

Step 1: Access the PostgreSQL Command Line

To interact with your PostgreSQL server, you can use the `psql` command-line interface. Switch to the `postgres` user (or the user you created) and access the PostgreSQL prompt:

```
syed@syed-VirtualBox:~$ su - postgres  
Password:  
postgres@syed-VirtualBox:~$ psql  
psql (17.2 (Ubuntu 17.2-1.pgdg22.04+1), server 16.6  
(Ubuntu 16.6-1.pgdg22.04+1))  
Type "help" for help.  
  
postgres=#
```

Step 2: Create a Database

To create a new database, use the following command:

```
postgres=# CREATE DATABASE my_database;  
CREATE DATABASE
```

Explanation: This command creates a new database named my_database.

Step 3: Connect to the Database

To start working with the newly created database, connect to it:

```
postgres=# \c my_database  
psql (17.2 (Ubuntu 17.2-1.pgdg22.04+1), server 16.6  
(Ubuntu 16.6-1.pgdg22.04+1))  
You are now connected to database "my_database" as  
user "postgres".
```

Explanation: The \c command connects you to the specified database.

Step 4: Create a Table

Now, let's create a simple table. For example, to create a table named users with columns for id, name, and email, use the following command:

```
my_database=# CREATE TABLE users (  
    id SERIAL PRIMARY KEY,  
    name VARCHAR(100),  
    email VARCHAR(100) UNIQUE NOT NULL  
);  
CREATE TABLE
```

Explanation:

- id SERIAL PRIMARY KEY: This creates an auto-incrementing primary key.
- name VARCHAR(100): This creates a column for the user's name with a maximum length of 100 characters.
- email VARCHAR(100) UNIQUE NOT NULL: This creates a column for the user's email, ensuring that it is unique and cannot be null.

Step 5: Insert Data into the Table

You can insert data into the users table using the following command:

```
my_database=# INSERT INTO users (name, email) VALUES  
( 'John Doe', 'john.doe@example.com' );  
INSERT 0 1
```

Explanation: This command inserts a new user into the users table.

Step 6: Query the Table

To retrieve data from the table, you can use a simple SELECT statement:


```
my_database=# SELECT * FROM users;
 id |   name   |      email
-----+-----+-----
  1 | John Doe | john.doe@example.com
(1 row)
```

Explanation: This command retrieves all records from the users table.

Basic Commands to Know

Here are some basic PostgreSQL commands that beginners should be familiar with:

- **List Databases:** \l - Lists all databases.
- **List Tables:** \dt - Lists all tables in the current database.
- **Describe Table:** \d table_name - Shows the structure of a specified table.
- **Exit psql:** \q - Exits the PostgreSQL command line.

Conclusion

In this article, we covered two methods for installing PostgreSQL: using RPM packages and compiling from source on both Red Hat and Ubuntu. Each method has its advantages, and you can choose the one that best fits your needs.


We also walked through the process of creating a database and a table, inserting data, and querying that data. These foundational skills are essential for anyone looking to work with PostgreSQL or any relational database management system.



In the next article, we will delve into the crucial topic of **Backup and Recovery** in PostgreSQL. We will explore various strategies and tools to ensure your data is safe and can be restored in case of failure. Stay tuned!

Report this article


Comments

 57 · 4 comments · 2 reposts

 Like Comment Share

Add a comment...  

Most recent ▾

 **Syed Jaffer Hussain** • 2nd (edited) 3mo ...

Talks, learn #blockchain #opensource #AI #kafka #technology #...
Neatly explained. Great job. Keeps making the knowledge #SecureMax
#Postgres #Database Follow to never miss an update.

Like | Reply

 **Britto Martin** • 2nd 3mo ...

Senior DBA at Arab National Bank, Saudi Arabia
Easy to understand . very useful. Asad hussain
PostgreSQL | Redis DBA


Like | Reply

 **Shaik Kashif** • 3rd+ 1mo ...




Student at Chaitanya Bharathi Institute of Technology
Its very easy to understand .Great job ☐




Like | Reply



More articles for you

 **Ahsan Iqbal** • 2nd 3mo ...

Oracle Database | RAC | Data Guard | MySQL Technologies Expert | Postgr...
pfu
1

 PostgreSQL
Getting Started With PostgreSQL
Code Purple Academy
  12


Postgres for Everything
Timescale Newsletter
  25 · 1 comment · 3 reposts


PostgreSQL Upgrade: From Fashionably
Late to Cutting Edge! ☐ ☐
PiPulse: Micro Marvel Digest
 1

○ ○ ○ ○ ○