**Postgres 12.2 version: Installation/Configuration on Redhat77**

<https://tecadmin.net/install-postgresql-11-on-centos/>

Adapted to version 12 from notes from this site which installs PostgreSQL version 11

<https://www3.ntu.edu.sg/home/ehchua/programming/sql/PostgreSQL_GetStarted.html>

**Step 1 – Configure Yum Repository**

Firstly you need to configure the PostgreSQL repository in your system. Use one of the below commands as per your operating system version.

rpm -Uvh https://yum.postgresql.org/12/redhat/rhel-7-x86\_64/pgdg-redhat-repo-latest.noarch.rpm

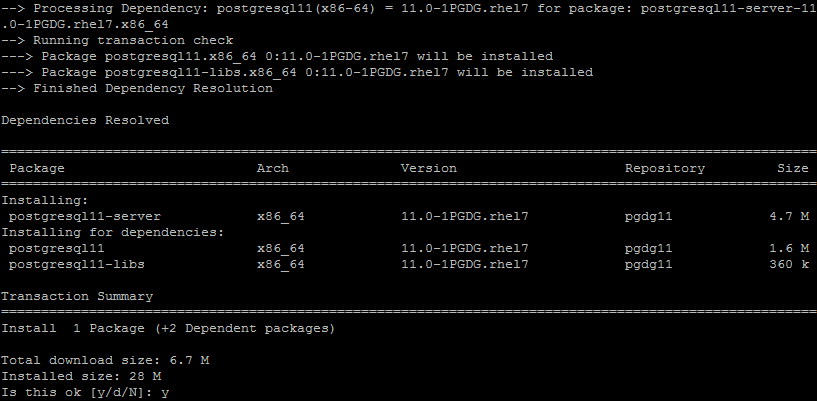
For more details visit [PostgreSQL repositories](https://yum.postgresql.org/repopackages.php#pg11) link page where you can get repository package rpm for various operating systems.

**Step 2 – Install PostgreSQL 11 on CentOS**

After enabling PostgreSQL yum repository in your system use following command to install **PostgreSQL 12** on your system with yum package manager.

yum install postgresql12-server

This will also install some additional required packages on your system. Enter y to confirm and complete the installation process.

[](https://tecadmin.net/wp-content/uploads/2018/10/postgresql11-centos-install.png)

**Step 3 – Initialize PGDATA**

After that, you need to initialize the PostgreSQL instance. In other words, this will create a data directory and other configuration files on your system. To initialize database use below command.

/usr/pgsql-12/bin/postgresql-12-setup initdb

[Initialize PostgreSQL 11 on CentOS](https://tecadmin.net/wp-content/uploads/2018/10/postgresql11-centos-initdb.png)

Above command will take some time to initialize PostgreSQL first time. **PGDATA** environment variable contains the path of data directory.

PostgreSQL 12 default data directory location is **/var/lib/pgsql/12/data**

In this directory 2 postgres configuration files are located:

**/var/lib/pgsql/12/data/pg\_hba.conf**

/var/lib/pgsql/12/data/**postgresql.conf**

# HowTo Safely Open a PostgreSQL Port for Remote Access?

This document describes how to open a PostgreSQL database port for remote access and includes security considerations for both Linux and Windows.

### Why do you Need to Open a Port

You need to open a remote port if you want to access data in the database from a client running on a different computer in the network. The same is true, if you want to use [pgAdmin](https://www.pgadmin.org/) for or similar tools for SQL development.

### Check for Port IP

Please check if PostgreSQL is listening on a public port:

Linux:

# netstat -nlp | grep 5432

tcp 0 0 127.0.0.1:5432 0.0.0.0:\* LISTEN 1272/postgres

tcp6 0 0 ::1:5432 :::\* LISTEN 1272/postgres

Windows:

C:\>netstat -a | grep 5432

TCP 127.0.0.1:5432 tarraco:0 LISTENING

TCP [::1]:5432 tarraco:0 LISTENING

The results above (including "127.0.0.1:5432") shows that PostgreSQL is listening only for connects originating from the local computer, so we will have to edit the "postgresql.conf" configuration file. A result including "0.0.0.0:5432" indicates that PostgreSQL is already listening for remote connections.

You can usually ignore the line with "::1", as it refers to the IP v6 protocol, which is rarely used.

### Edit postgresql.conf

This file is usually located in /var/lib/pgsql/data/ on Linux or C:\PostgreSQL\data\ on Windows or similar.

In this file we will edit the "listen\_address" and "port" parameters, so that they look like below:

#------------------------------------------------------------------------------

# CONNECTIONS AND AUTHENTICATION

#------------------------------------------------------------------------------

# - Connection Settings -

listen\_addresses = '0.0.0.0' # what IP address(es) to listen on;

# comma-separated list of addresses;

# defaults to 'localhost'; use '\*' for all

# (change requires restart)

port = 5432 # (change requires restart)

### Restart PostgreSQL

After that we need to restart PG to activate the changes.

On recent Linux system you have to enter as user root:

systemctl restart postgresql.service

On Windows you can use Control Panel -> Administrative Tools -> Services and restart the PostgreSQL service. For ]po[, the PostgreSQL service is called "]po[ PostgreSQL".

Repeating the "Check for Port IP" step above, you should now see that the port IP is "0.0.0.0", meaning that it will accept connections from any remote computer.

### Authentication Configuration using pg\_hba.conf

pg\_hba.conf is located in the same directly as postgresql.conf.

Please  add the following two lines at the end of the file:

host all all 0.0.0.0/0 md5

host all all ::/0 md5

This means that remote access is allowed using IP v4 and IP v6 to all databases and all users using the "md5" authentication protocol.

Please "restart postgresql" again.

### Open Linux Firewall Port

Does your PostgreSQL database run on a Linux server with the firewall enabled (like the [[https://sourceforge.net/projects/project-open/files/project-open/V5.0/|CentOS 7 ]project-open[ virtual appliance]])?

In this case you will have to poke a hole (as root):

# firewall-cmd --zone=public --add-port=5432/tcp --permanent

success

# firewall-cmd --reload

success

PostgreSQL 12 default data directory location is **/var/lib/pgsql/12/data**

In this directory 2 postgres configuration files are located:

**/var/lib/pgsql/12/data/pg\_hba.conf** ( controls client authentication. Hba stands for Host Based Authentication ). Each record specifies a

1. **connection type**,
2. **a client IP address range** (if relevant for the connection type),
3. **a database name**,
4. **a user name**, and
5. **the authentication method** to be used for connections matching these parameters.

The first record with a matching connection type, client address, requested database, and user name is used to perform authentication. There is no “fall-through” or “backup”: if one record is chosen and the authentication fails, subsequent records are not considered. If no record matches, access is denied.

To edit the pg\_hba.conf login as postgres. This is the example content of the pg\_hba.conf file:

[monicarhvm@rhel77 ~]$ vi /var/lib/pgsql/12/data/postgresql.conf

# TYPE DATABASE USER ADDRESS METHOD

# "local" is for Unix domain socket connections only

local all postgres md5

local all monicadba md5

local all all peer

# IPv4 local connections:

host all all 127.0.0.1/32 md5

# IPv6 local connections:

host all all ::1/128 md5

# Allow replication connections from localhost, by a user with the

# replication privilege.

local replication all peer

host replication all 127.0.0.1/32 ident

host replication all ::1/128 ident

**host all all 0.0.0.0/0 md5**

**host all all ::/0 md5**

**Setp 4 – Start PostgreSQL Server**

To start PostgreSQL service using the following command as per your operating systems. Also, enable PostgreSQL service to autostart on system boot.

**CentOS/RHEL – 7**

systemctl enable postgresql-12.service

systemctl start postgresql-12.service

**Step 5 – Verify PostgreSQL Installation**

After completing the above all steps. Your PostgreSQL 12 server is ready to use. Log in to postfix instance to verify the connection.

su - postgres -c "psql"

psql (12.2)

Type "help" for help.

postgres=#

You may create a password for user postgres for security purpose.

postgres=# \password postgres

#### 4.1  Default Superuser "postgres"

During installation, a "UNIX USER" (who cannot login to the system interactively) called postgres is created. You can verify by checking the entry in /etc/passwd and /etc/shadow:

$ **sudo less /etc/passwd | grep postgres**

postgres:x:120:130:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash

$ **sudo less /etc/shadow | grep postgres**

postgres:\*:16049:0:99999:7:::

// \* indicates that password is not enabled (this user cannot login interactively).

#### 4.2  Authentication Methods

PostgreSQL supports a number of authentication methods. The commonly-used are:

* ident, peer: identical (or fully match) to the OS account, with an optional name mapping defined in pg\_ident.conf file. ident is applicable to TCP/IP; while peer for "local" connection.
* md5: require md5-hashed password (most common).
* password: require clear-text password.
* trust: no password, as long as meeting the IP, user, and database criteria defined in the HBA.
* reject: reject login immediately.
* others, such as GSSAPI, SSPI, Kerberos, LDAP, RADIUS, Certificate, PAM.

There are two ways to login PostgreSQL:

1. By running the "psql" command as a UNIX user which is also configured as PostgreSQL user using so-called IDENT/PEER authentication, e.g., "sudo -u postgres psql".
2. Via TCP/IP connection using PostgreSQL's own managed username/password (using so-called MD5 authentication).

#### 4.3  Set a Password for User postgres

To set a password, login to PostgreSQL server using postgres via psql and issue command "\password username", as follows:

-- Login in to server via "psql" with user "postgres"

$ **sudo -u postgres psql**

......

-- Change password for current user "postgres"

postgres=# **\password postgres**

Enter new password: xxxx

Enter it again: xxxx

-- Display the user table

postgres=# **SELECT \* FROM pg\_user;**

usename | usesysid | usecreatedb | usesuper | usecatupd | userepl | passwd | valuntil | useconfig

----------+----------+-------------+----------+-----------+---------+----------+----------+-----------

postgres | 10 | t | t | t | t | \*\*\*\*\*\*\*\* | |

-- Quit

postgres=# **\q**

To test the password login, you need to change the the authentication method from "peer" to "md5" in pg\_hba.conf. Restart the server, and login via sudo -u postgres sql. The system will prompt you for the password.

#### 4.4  Add your UNIX user as PostgreSQL user

Using user "postgres" (**which is a UNIX user as well as PostgreSQL user**) to run "psql" involves switching user (via "sudo -u username" or "su - username"). You can simply the process by configuring your current UNIX userID as PostgreSQL user, as follows:

-- Switch to default superuser "postgres",

-- run utility "createuser" to create a superuser same name as current login.

-- "$USER" is an environment variable denoting the current login user.

$ **sudo -u postgres createuser --superuser $USER**

-- Create the default database which shall be the same as the username.

$ **sudo -u postgres createdb $USER**

-- Now, you can invoke "psql" from your user account.

$ **psql**

......

yourusername=# **SELECT \* FROM pg\_user;**

usename | usesysid | usecreatedb | usesuper | usecatupd | userepl | passwd | valuntil | useconfig

--------------+----------+-------------+----------+-----------+---------+----------+----------+-----------

yourusername | 16424 | t | t | t | t | \*\*\*\*\*\*\*\* | |

-- Perform database operations.

.......

#### 4.5  Create Group and User

The latest PostgreSQL treats both group and user as role. Some roles can login (i.e. user), some roles have member of other roles (i.e., group). You should use CREATE ROLE to create both users and groups (CREATE USER and CREATE GROUP are meant for compatibility).

-- Create a login user role

CREATE ROLE user1 LOGIN PASSWORD 'xxxx' CREATEDB VALID UNTIL 'infinity';

-- Create a login superuser role

CREATE ROLE user2 LOGIN PASSWORD 'xxxx' SUPERUSER VALID UNTIL '2019-12-31';

-- Create a group role

CREATE ROLE group1 INHERIT;

-- Add a user (or group) to this group

GRANT group1 TO user1;

#### 4.6  Backup and Restore

PostgreSQL provides two utilities for backup:

1. pg\_dump: backup a specific database.
2. pg\_dumpall: backup all databases and server globals. Need to be run by superuser.

For example,

-- Create a compressed backup for a database

pg\_dump -h localhost -p 5432 -U username -F c -b -v -f mydatabase.backup mydatabase

-- Create a plain-text backup for a database, including the CREATE DATABASE

pg\_dump -h localhost -p 5432 -U username -C -F p -b -v -f mydatabase.backup.sql mydatabase

To restore a database from backup generated by pg\_dump or pg\_dumpall:

1. Use "psql" to run the plain-text backup.
2. -- Run SQL script

$ psql -U username -f filename.sql

1. Use utility "pg\_restore" to restore compressed backup.