

How to install PostgreSQL 11 on CentOS 7?

Tuesday, October 30, 2018 15:21

By

[angelomarquez](#)

-

October 26, 2018

[0](#)

472

PostgreSQL is one of the best relational database managers in the world. Its main virtue is to be a reliable rock. Today, I will show you how to install PostgreSQL 11 on CentOS 7.

In this site we have [talked before](#) about PostgreSQL, it is difficult not to do it when you have in front of a database manager as advanced and reliable as this one. Recently, version 11 of this fantastic software [was released](#).

In this new version, it highlights the speed improvements in the processing of very large databases. Something sysadmin will surely appreciate.

On the other hand, PostgreSQL 11 adds the ability to partition data by a hash key. This translates into improved data robustness and security.

Install PostgreSQL 11 on CentOS 7

PostgreSQL 11 is a novelty. It is the first big release since October 2017. Many sysadmins will look for a way to update or install it from 0. The latter is the goal of this article.

1. Upgrade the system

First, you need to update the system. This in order to get the latest security patches available for the system. With this, you gain stability and robustness.

```
1 :~$ su
2 :~# yum update
```



```
angelo@osradar:/home/angelo 169x44
[root@osradar angelo]# yum update
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
epel/x86_64/metalink | 3.5 kB 00:00:00
 * base: mirror.uce.edu.ec
 * epel: mirror.globo.com
 * extras: mirror.uce.edu.ec
 * updates: mirror.uce.edu.ec
Tuleap | 2.9 kB 00:00:00
base | 3.6 kB 00:00:00
ce_10.0 | 1.2 kB 00:00:00
centos-sclo-rh | 3.0 kB 00:00:00
centos-sclo-sclo | 2.9 kB 00:00:00
docker-ce-stable | 2.9 kB 00:00:00
```

1.- Upgrading CentOS 7

Now you can continue the installation.

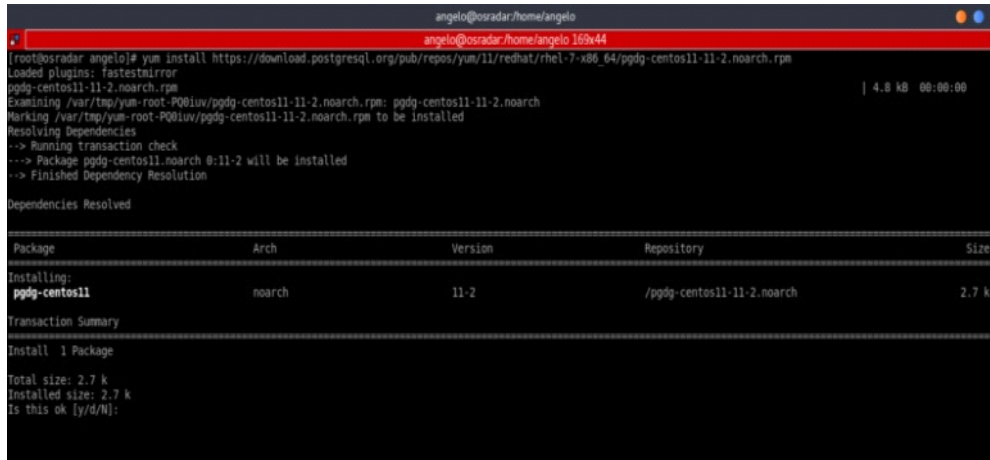
2. Adding the PostgreSQL repository

Actually, PostgreSQL is available for many operating systems such as Debian, Ubuntu, or CentOS 7, but the version that comes in the official repositories of each distribution is usually obsolete and outdated.

In short, you need to resort to an external repository to perform the installation. Fortunately, PostgreSQL has one and it is very easy.

First, run this command as root user:

```
1 :~# yum install https://download.postgresql.org/pub/repos/yum/11/redhat/rhel-7-x86_64/pgdg-centos11-11-2.noarch.rpm
```



```
angelo@osradar:/home/angelo
[root@osradar angelo]# yum install https://download.postgresql.org/pub/repos/yum/11/redhat/rhel-7-x86_64/pgdg-centos11-11-2.noarch.rpm
Loaded plugins: fastestmirror
pgdg-centos11-11-2.noarch.rpm                                | 4.8 kB  00:00:00
Examining /var/tmp/yum-root-P08iuv/pgdg-centos11-11-2.noarch
Marking /var/tmp/yum-root-P08iuv/pgdg-centos11-11-2.noarch.rpm to be installed
Resolving Dependencies
--> Running transaction check
--> Package pgdg-centos11.noarch 0:11-2 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====================================================================================================================================
 Package                               Arch                               Version                               Repository                               Size
=====================================================================================================================================
Installing:
pgdg-centos11                          noarch                            11-2                                 /pgdg-centos11-11-2.noarch              2.7 k

Transaction Summary
-----
Install 1 Package

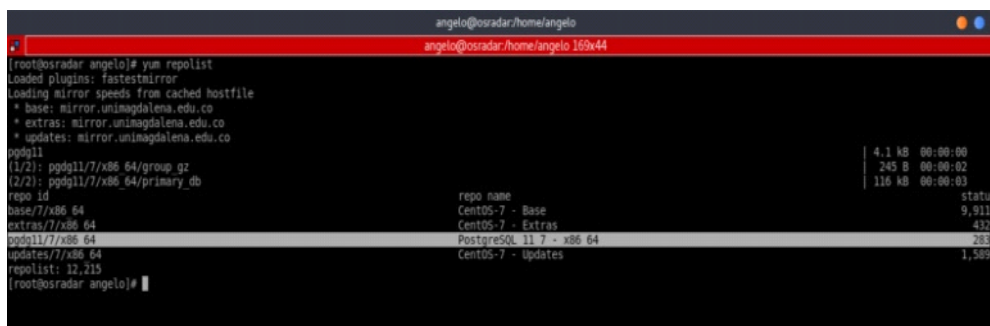
Total size: 2.7 k
Installed size: 2.7 k
Is this ok [y/d/N]:
```

2.- Adding the repository

After that, the repository has been added.

You can also verify that.

```
1 :~# yum repolist
```



```
angelo@osradar:/home/angelo
[root@osradar angelo]# yum repolist
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: mirror.unimagdalena.edu.co
 * extras: mirror.unimagdalena.edu.co
 * updates: mirror.unimagdalena.edu.co
pgdg11
(1/2): pgdg11/7/x86_64/group.gz
(2/2): pgdg11/7/x86_64/primary_db
repo id                                repo name                                status
base/7/x86_64                          CentOS-7 - Base                          9,911
extras/7/x86_64                         CentOS-7 - Extras                        432
pgdg11/7/x86_64                         PostgreSQL 11.7 - x86_64                 378
updates/7/x86_64                        CentOS-7 - Updates                       1,589
repolist: 12,215
[root@osradar angelo]#
```

3.- The repositories added

As you can see, the repository is added

3. Install PostgreSQL 11

Thanks to the repository, now, you can install the new version of PostgreSQL in an easy way. Just run this command:

```
1 :~# yum install postgresql11 postgresql11-server
```

```
angelo@osradar:/home/angelo
angelo@osradar:/home/angelo 159v44

[root@osradar angelo]# yum install postgresql11 postgresql11-server
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: mirror.unimagdalena.edu.co
 * extras: mirror.unimagdalena.edu.co
 * updates: mirror.unimagdalena.edu.co
Resolving Dependencies
--> Running transaction check
--> Package postgresql11.x86_64 0:11.0-1PGDG.rhel7 will be installed
--> Processing Dependency: postgresql11-libs(x86-64) = 11.0-1PGDG.rhel7 for package: postgresql11-11.0-1PGDG.rhel7.x86_64
--> Processing Dependency: libicu for package: postgresql11-11.0-1PGDG.rhel7.x86_64
--> Processing Dependency: libpq.so.5()(64bit) for package: postgresql11-11.0-1PGDG.rhel7.x86_64
--> Package postgresql11-server.x86_64 0:11.0-1PGDG.rhel7 will be installed
--> Running transaction check
--> Package libicu.x86_64 0:50.1.2-15.el7 will be installed
--> Package postgresql11-libs.x86_64 0:11.0-1PGDG.rhel7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

Package Arch Version Repository Size
--
Installing:
 postgresql11 x86_64 11.0-1PGDG.rhel7 pgdg11 1.6 M
 postgresql11-server x86_64 11.0-1PGDG.rhel7 pgdg11 4.7 M
Installing for dependencies:
 libicu x86_64 50.1.2-15.el7 base 6.9 M
 postgresql11-libs x86_64 11.0-1PGDG.rhel7 pgdg11 360 k

Transaction Summary
--
Install 2 Packages (+2 Dependent packages)
Total download size: 14 M
Installed size: 53 M
Is this ok [y/d/N]:
```

4.- Install PostgreSQL 11

After that, initialize the database:

```
1 :~# /usr/pgsql-11/bin/postgresql-11-setup initdb
```

Next, start and enable PostgreSQL service:

```
1 :~# systemctl enable postgresql-11
2 :~# systemctl start postgresql-11
```

```
angelo@osradar:/home/angelo
angelo@osradar:/home/angelo 169x44
[root@osradar angelo]# systemctl enable postgresql-11
Created symlink from /etc/systemd/system/multi-user.target.wants/postgresql-11.service to /usr/lib/systemd/system/postgresql-11.service.
[root@osradar angelo]# systemctl start postgresql-11
[root@osradar angelo]#
```

5.- Starting the Postgresql 11 service

This process must be done because of the security policies of CentOS. CentOS 7 installs the package but does not start the PostgreSQL service automatically.

Finally, check the service status.

```
1 :~# systemctl status postgresql-11
```

```
angelo@osradar/home/angelo
angelo@osradar/home/angelo 159x44

[root@osradar angelo]# systemctl status postgresql-11
● postgresql-11.service - PostgreSQL 11 database server
   Loaded: loaded (/usr/lib/systemd/system/postgresql-11.service; enabled; vendor preset: disabled)
   Active: active (running) since Thu 2018-10-25 14:26:05 EDT; 1min 23s ago
     Docs: https://www.postgresql.org/docs/11/static/
   Process: 1306 ExecStartPre=/usr/pgsql-11/bin/postgresql-11-check-db-dir ${PGDATA} (code=exited, status=0/SUCCESS)
  Main PID: 1311 (postmaster)
    CGroup: /system.slice/postgresql-11.service
            └─1311 /usr/pgsql-11/bin/postmaster -D /var/lib/pgsql/11/data/
               └─1313 postgres: logger
                  └─1315 postgres: checkpointer
                     └─1316 postgres: background writer
                        └─1317 postgres: walwriter
                           └─1318 postgres: autovacuum launcher
                              └─1319 postgres: stats collector
                                 └─1320 postgres: logical replication launcher

Oct 25 14:26:05 osradar systemd[1]: Starting PostgreSQL 11 database server...
Oct 25 14:26:05 osradar postmaster[1311]: 2018-10-25 14:26:05.432 EDT [1311] LOG: listening on IPv6 address "::1", port 5432
Oct 25 14:26:05 osradar postmaster[1311]: 2018-10-25 14:26:05.432 EDT [1311] LOG: listening on IPv4 address "127.0.0.1", port 5432
Oct 25 14:26:05 osradar postmaster[1311]: 2018-10-25 14:26:05.449 EDT [1311] LOG: listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
Oct 25 14:26:05 osradar postmaster[1311]: 2018-10-25 14:26:05.463 EDT [1311] LOG: listening on Unix socket "/tmp/.s.PGSQL.5432"
Oct 25 14:26:05 osradar postmaster[1311]: 2018-10-25 14:26:05.467 EDT [1311] LOG: redirecting log output to logging collector process
Oct 25 14:26:05 osradar systemd[1]: 2018-10-25 14:26:05.467 EDT [1311] HINT: Future log output will appear in directory "log".
Oct 25 14:26:05 osradar systemd[1]: Started PostgreSQL 11 database server.
[root@osradar angelo]#
```

6.- Checking the status of the service

The installation has been successful

4. Using PostgreSQL 11

Before using PostgreSQL You must install sudo. You also have to add your current user to the sudoers file:

```
1 :~# yum install sudo
```

Then, open the file `/etc/sudoers` and edit it. See the image for the guide. In “angelo” write your user.

```
1 :~# nano /etc/sudoers
```

```
angelo@osradar:/home/angelo
angelo@osradar:/home/angelo 169x44
GNU nano 2.3.1 File: /etc/sudoers

## Next comes the main part: which users can run what software on
## which machines (the sudoers file can be shared between multiple
## systems).
## Syntax:
##
##      user    MACHINE=COMMANDS
##
## The COMMANDS section may have other options added to it.
##
## Allow root to run any commands anywhere
root    ALL=(ALL)    ALL
angelo  ALL=(ALL)    ALL
## Allows members of the 'sys' group to run networking, software,
## service management apps and more.
# %sys ALL = NETWORKING, SOFTWARE, SERVICES, STORAGE, DELEGATING, PROCESSES, LOCATE, DRIVERS

## Allows people in group wheel to run all commands
%wheel  ALL=(ALL)    ALL

## Same thing without a password
# %wheel    ALL=(ALL)    NOPASSWD: ALL

## Allows members of the users group to mount and unmount the
## cdrom as root
# %users    ALL=/sbin/mount /mnt/cdrom, /sbin/umount /mnt/cdrom

## Allows members of the users group to shutdown this system
# %users    localhost=/sbin/shutdown -h now

## Read drop-in files from /etc/sudoers.d (the # here does not mean a comment)
#includedir /etc/sudoers.d
```

7.- Editing the sudoers file

Then, close session and access as postgres user.

```
1  :~# exit
2  :~$ sudo -i -u postgres
3  :~$ psql
```

```
angelo@osradar:~
angelo@osradar:~ 169x44
[angelo@osradar ~]$ sudo -i -u postgres
[sudo] password for angelo:
-bash-4.2$ psql
psql (11.0)
Type "help" for help.

postgres=#
```

8.- PostgreSQL log in

Now, you can write some commands. For example, list all database:

```
1  postgres=# \l
```

```
angelo@osradar:~  
[angelo@osradar ~]$ sudo -i -u postgres  
[sudo] password for angelo:  
-bash-4.2$ psql  
psql (11.0)  
Type "help" for help.  
  
postgres=# \l  
  
              List of databases  
  Name      | Owner   | Encoding | Collate | Ctype   | Access privileges  
-----+-----+-----+-----+-----+-----  
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  
(3 rows)  
  
postgres=#
```

9.- List all databases

Conclusion

PostgreSQL is a great database manager. Each time it improves more and more and aims to be the leading open source option in this area. This new version adds performance improvements and installing it in CentOS 7 is not a big complication compared to its benefits.

Please share this article on your social networks.

From <<https://www.osradar.com/how-to-install-postgresql-11-centos-7/>>

Change a Password for PostgreSQL on Linux via Command Line

PostgreSQL supports many client authentication methods, but in this case we're only going to concern ourselves with two: **password** and **md5**.

Note: The default authentication method for PostgreSQL is **ident**. If you'd like to [change the PostgreSQL authentication method from ident to md5, then visit the linked tutorial!](#)

Pre-Flight Check

- These instructions are intended specifically for changing a password in PostgreSQL.
- I'll be working from a Liquid Web Core Managed CentOS 7 server, and I'll be logged in as root.
- PostgreSQL is installed per our tutorial on: [How to Install and Connect to PostgreSQL on CentOS 7](#).

Step #1: Switch to the PostgreSQL User: postgres

If you're working from a default PostgreSQL installation, then PostgreSQL will be configured with the user **postgres**. Since we're logged in as **root**, and we're assuming that root doesn't have a user for PostgreSQL, switch to the default PostgreSQL user: **postgres**:

```
su - postgres
```

... then attempt a connection to PostgreSQL:

```
psql
```

... enter your password at the prompt:

```
Password:
```

... the correct, valid response will be similar to:

```
psql (9.3.9)
```

```
Type "help" for help.
```

```
postgres=#
```

Step #2: Add/Change the Password for the PostgreSQL User: postgres

Use the following command to change the password for your current user, which is now **postgres**:

```
\password
```

Enter your new password, and then enter it again to confirm it:

```
Enter new password:
```

```
Enter it again:
```

Now quit the PostgreSQL interface:

`\q`

Bonus Information!

You can do all of step one in exactly one command:

`su -c "psql" - postgres`

Be Sociable, Share!

From <https://www.liquidweb.com/kb/change-a-password-for-postgresql-on-linux-via-command-line/>

Configure PostgreSQL to allow remote connection

January 23, 2016 - Neeraj Singh

By default PostgreSQL is configured to be bound to "localhost".

```
$ netstat -nl
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp      0      0 0.0.0.0:443             0.0.0.0:*               LISTEN
tcp      0      0 127.0.0.1:11211          0.0.0.0:*               LISTEN
tcp      0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
tcp      0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp      0      0 127.0.0.1:5432           0.0.0.0:*               LISTEN
tcp      0      0 127.0.0.1:3737           0.0.0.0:*               LISTEN
tcp6     0      0 :::22                   :::*                     LISTEN
```

As we can see above port 5432 is bound to 127.0.0.1. It means any attempt to connect to the postgresql server from outside the machine will be refused. We can try hitting the port 5432 by using telnet.

```
$ telnet 107.170.11.79 5432
Trying 107.170.11.79...
telnet: connect to address 107.170.11.79: Connection refused
telnet: Unable to connect to remote host
```

Configuring postgresql.conf

In order to fix this issue we need to find postgresql.conf. In different systems it is located at different place. I usually search for it.

```
$ find \ -name "postgresql.conf"
/var/lib/pgsql/9.4/data/postgresql.conf
```

Open postgresql.conf file and replace line

`listen_addresses = 'localhost'`

with

`listen_addresses = '*'`

Now restart postgresql server.

```
$ netstat -nl
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp      0      0 127.0.0.1:11211          0.0.0.0:*               LISTEN
tcp      0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
tcp      0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp      0      0 0.0.0.0:5432            0.0.0.0:*               LISTEN
tcp      0      0 127.0.0.1:25            0.0.0.0:*               LISTEN
tcp      0      0 127.0.0.1:2812          0.0.0.0:*               LISTEN
tcp6     0      0 ::1:11211               :::*                     LISTEN
tcp6     0      0 :::22                   :::*                     LISTEN
```



```
tcp6    0    0 :::5432          :::*             LISTEN
tcp6    0    0 :::1:25         :::*             LISTEN
```

Here we can see that “Local Address” for port 5432 has changed to 0.0.0.0.

Configuring pg_hba.conf

Let’s try to connect to remote postgresql server using “psql”.

```
$ psql -h 107.170.158.89 -U postgres
psql: could not connect to server: Connection refused
        Is the server running on host "107.170.158.89" and accepting
        TCP/IP connections on port 5432?
```

In order to fix it, open pg_hba.conf and add following entry at the very end.

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

The second entry is for IPv6 network.

Do not get confused by “md5” option mentioned above. All it means is that a password needs to be provided. If you want client to allow connection without providing any password then change “md5” to “trust” and that will allow connection unconditionally.

Restart postgresql server.

```
$ psql -h 107.170.158.89 -U postgres
Password for user postgres:
psql (9.4.1, server 9.4.5)
Type "help" for help.
postgres=# \l
```

You should be able to see list of databases.

Now we are able to connect to postgresql server remotely.

Please note that in the real world you should be using extra layer of security by using “iptables”.

From <<https://blog.bigbinary.com/2016/01/23/configure-postgresql-to-allow-remote-connection.html>>