

# Lecture 02 - Installing PostgreSQL

---

Install PostgreSQL using Virtual System - <https://youtu.be/A0-GrSAubEc>

From Amazon S3 - for download (same as youtube videos)

Install PostgreSQL using Virtual System

## Install PostgreSQL

---

### Linux

I am using Ubuntu 20.04 for this. We are starting out with Linux because most installs of PostgreSQL are on Linux. Because of the “working model” memory management in Windows a relational database server runs better on a non-Windows system. Usually this is Linux. FreeBSD/OpenBSD also work (but use a different install process). MacOS (Os X) is also good - but is rarely seen as a production database environment.

You can run PostgreSQL in a Docker Container - but I don't recommend it in a production context. Docker is only suitable for use with a database in a development environment. (That means that you can use Docker for this class and it will work - but don't plan on deploying a production database on top of docker - the performance is horrible)

If you are using a different version of Linux then the install process will be different.

We are going to use PostgreSQL as our primary database in the class. You will need a copy of it running where you can access the entire database. This could be on a virtual machine on your box or on Digital Ocean (\$5.00 a month) or Linode (\$5 to \$20 a month) or on a free micro system on AWS if you qualify for one or on a \$15-20 AWS EC3 small system.

If you choose a virtual or remote system my recommendation would be to install Ubuntu on it.

### Step 1

After you have gotten your Linux system up and running you should refresh the local package index.

```
$ sudo apt update
```

Now install the PostgreSQL software:

```
$ sudo apt install postgresql postgresql-contrib
```

`sudo` will prompt you for your login password. You will need to have an account that is configured to be able to do `sudo`. You should not do this from the root account. Generally speaking I set up a system, generate public/private keys for SSH login, create a user account, set it to be capable of `sudo`, verify that I can login from my local account to the user account using the public/private keys, then turn off login on the root account. Double check that I can login without getting a password prompt from my local system. Then I start installs like this.

## Step 2 - Configure basic "roles"

Security in most databases, PostgreSQL, Oracle, MySQL are all based on 'roles' - these are groups of users that have similar properties for how they access the database. This is similar to Unix/Linux groups.

When you installed PostgreSQL it created a Linux user. This is a non-root user that "owns" all of the data and files in the database. This Linux user, called `postgres`, is associated with the 'postgres' role in the database.

Normally you can't just login to the 'postgres' user. You have to access it via the `root` account.

```
$ sudo -i -u postgres
```

Now you can access the database using `psql`

```
$ psql
```

You can now use the database. Note that you are in a special privileged account in the database and you can break things from this account.

To exit out of the PostgreSQL prompt, type:

```
\q
```

This will bring you back to the Linux shell prompt. To get back to your login account you can send the shell and end of file (EOF) by entering a Control-D or

```
$ exit 0
```

## Step 3 — Creating a New Role

When you are logged in as the postgres Linux account, you can create a new role by typing:

```
$ createuser --interactive
```

This runs a number of commands in the database to create a user. It is best to match the username to your login Linux user.

A run of this for my user, `pschlump` looks like:

```
Enter name of role to add: pschlump
Shall the new role be a superuser? (y/n) y
```

At this point you should be able to use `psql` to access the database from your user.

## Step 4 — Creating Additional Databases

In the PostgreSQL and MySQL/MariaDB world a 'database' is storage and a set of tables that works together. In the Oracle world the 'database' is an installed system. For us a 'database' is what we connect to so that we can see our tables.

In the interactive system each of you is using a 'database' in a single instance of PostgreSQL.

When PostgreSQL created a user with `createuser` it created a database with the same name that is owned by that login user. For a user to login it has to have some database to connect to.

This implies that when I create a PostgreSQL user `pschlump` it will have a `pschlump` database. A user can connect to other databases that it owns.

From the postgres Linux account:

```
createdb studentdb
```

Will create a database.

Usually by this point I want to create a database without access to the `postgres` Linux login. This means that I do it via `psql`.

```
$ psql
pschlump=# create database newuser;
```

```
pschlump=# create user newuser with encrypted password 'mypass';
pschlump=# grant all privileges on database newuser to newuser;
```

This creates a new database user and associates it with a new database.

If you want `psql` to connect to a different user from your account:

```
psql -d newuser
newuser=# \conninfo
newuser=# \q
```

The output should look like:

```
You are connected to database "newuser" as user "newuser" via socket in "/var/run/postg
```

## Windows install

---

You can install PostgreSQL directly on Windows 10 if you are running an x86 based 64 bit processor. I recommend the following YouTube video on how to do this. <https://www.youtube.com/watch?v=w32xHj2nMSc>

Given the choice I will setup a virtual machine running Linux and install PostgreSQL in the virtual environment.

## MacOS (OS X)

---

PostgreSQL runs and works nicely on Mac OS. There is a package version of PostgreSQL that you can install from a .dmg and it puts a cute little Icon up on the top of the screen - that has start/stop for the database. After I install it I want to access it via the command line (Using iTerm 2.x). I always have to go and find where it installed it so that I can add that to my path.

```
$ find /Applications -name psql
```

When it finds the database then in your `.bashrc` (old) or now `.zshrc` file add that path to the `PATH` variable.

Login to the `postgres` login user using

```
$ sudo -i -u postgres  
$ createuser --interactive
```

Then follow the Linux instructions to create a user. You may encounter difficulties if your username has a blank in it.

Copyright © University of Wyoming 2021.