# POSTGERES DB

PostgreSQL is an free open-source database system that supports both relational (SQL) and non-relational (JSON) queries.

1. provide a structured way to store, organize, and access information.
2. features like reliable transactions and concurrency without read locks.

## Step 1 — Installing PostgreSQL

```
sudo apt update

sudo apt install postgresql postgresql-contrib

sudo systemctl start postgresql.service
```

## Step 2 — Using PostgreSQL Roles and Databases

By default, Postgres uses a concept called “roles” to handle authentication and authorization.

These are, in some ways, similar to regular Unix-style accounts, but Postgres does not distinguish between users and groups and instead prefers the more flexible term “role”.

Upon installation, Postgres is set up to use peer authentication, meaning that it associates Postgres roles with a matching Unix/Linux system account. If a role exists within Postgres, a Unix/Linux username with the same name is able to sign in as that role.

The installation procedure created a user account called postgres that is associated with the default Postgres role.

In order to use Postgres, you can log into that account.

There are a few ways to utilize this account to access Postgres.

### Switching Over to the postgres Account

Switch over to the postgres account on your server by typing:

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

You can now access the PostgreSQL prompt immediately by typing:

```
$ psql
```

Exit out of the PostgreSQL prompt by typing

```
postgres=# \q
```

this will bring you back to the postgres Linux command prompt.

Accessing a Postgres Prompt Without Switching Accounts
---
You can also run the command you'd like with the postgres account directly with sudo.

For instance, in the last example, you were instructed to get to the Postgres prompt by first switching to the postgres user and then running psql to open the Postgres prompt. You could do this in one step by running the single command psql as the postgres user with sudo, like this:

```
$ sudo -u postgres psql
```

This will log you directly into Postgres without the intermediary bash shell in between.
Again, you can exit the interactive Postgres session by typing:

```
postgres=# \q
```

Step 3 — Creating a New Role
==========

Many use cases require more than one Postgres role. Read on to learn how to configure these.

Currently, you just have the postgres role configured within the database. You can create new roles from the command line with the createrole command. The --interactive flag will prompt you for the name of the new role and also ask whether it should have superuser permissions.

If you are logged in as the postgres account,
---
you can create a new user by typing:

```
postgres@server:~$ createuser --interactive
```

If you prefer to use sudo for each command
---
without switching from your normal account, type:

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

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

```
$ man createuser
```

Your installation of Postgres now has a new user, but you have not yet added any databases. The next section describes this process.

## Step 4 — Creating a New Database

Another assumption that the Postgres authentication system makes by default is that for any role used to log in, that role will have a database with the same name which it can access.

This means that if the user you created in the last section is called sammy, that role will attempt to connect to a database which is also called “sammy” by default. You can create the appropriate database with the createdb command.

If you are logged in as the postgres account, you would type something like:

```
postgres@server:~$ createdb sammy
```

If, instead, you prefer to use sudo for each command without switching from your normal account, you would type:

```
$ sudo -u postgres createdb sammy
```

This flexibility provides multiple paths for creating databases as needed.

## Step 5 — Opening a Postgres Prompt with the New Role

To log in with peer authentication, you'll need a Linux user with the same name as your Postgres role and database.

If you don't have a matching Linux user available, you can create one with the adduser command. You will have to do this from your non-root account with sudo privileges (meaning, not logged in as the postgres user):

```
$ sudo adduser sammy
```

Once this new account is available, you can either switch over and connect to the database by typing:

```
$ sudo -i -u sammy
$ psql
```

Or, you can do this inline:

```
$ sudo -u sammy psql
```

This command will log you in automatically, assuming that all of the components have been properly configured.

If you want your user to connect to a different database, you can do so by specifying the database like this:

```
$ psql -d postgres
```

Once logged in, you can get check your current connection information by typing:

```
sammy=# \conninfo
```

Output
```
You are connected to database "sammy" as user "sammy" via
socket in
"/var/run/postgresql" at port "5432".
```

This is useful if you are connecting to non-default databases or with non-default users.