**PostgreSQL**PostgreSQL is a common database server. While not quite as used as MariaDB (because Wordpress doesn’t natively support it), it is still used by quite a few applications and companies to manage information.

**Installation**The installation process for PostgreSQL will vary with each OS.  
  
Debian/Ubuntu:  
sudo apt install postgresql postgresql-contrib  
  
RHEL:  
sudo yum install postgresql-server postgresql-contrib  
  
Gentoo:  
sudo emerge dev-db/postgresql  
  
Arch:  
sudo pacman -S postgresql  
  
The postgresql-contrib package that RHEL systems and Debian based systems can install has a few extra editions to build upon postgresql. To my knowledge, there is no mysql\_secure\_installation type thing to use with postgresql, at least not an official one.  
  
After installing the package for postgresql on your system, everything for it becomes essentially the same no matter what distro you are using, aside from Gentoo possibly not starting the service for it automatically.

**Making a Database**Now that we have PostgreSQL installed, we can connect to it and get to work on creating and managing our database. Since this guide is being written from a WRCCDC point of view, I will make that the theme of our database. To get into our database, we first must log into the postgres user that was made during the install, and then run the command to enter the psql shell:  
  
sudo su postgres (Using sudo before su allows you to log into a different account using your password instead of theirs)  
psql  
  
To view all the currently existing databases we use the list command:  
\l  
  
PostgreSQL breaks from the norm when it comes to a lot of their non-editing commands, using a \ and a letter instead of a longer command, such as show databases. Now then, let’s create our database:  
create database wrccdc;  
  
Rather than a traditional use database command, postgres has us disconnect from the server and then reconnect into a specific database. We can do this by doing:  
\c wrccdc  
  
Now that we are within our database, we can create a table. Let’s make one for members of the team and have it contain their first name, last name, their role, and the date they joined:  
create table members (first\_name VARCHAR(15), last\_name VARCHAR(15), role VARCHAR(10), join\_date date);  
  
As I said earlier, non-editing commands in PostrgeSQL are generally \ and a letter. To view all the tables, we can use:  
\d  
  
We should populate our empty table. To do this, we can use the insert command:  
insert into members (first\_name, last\_name, role, join\_date) values (‘Bailey’, ‘Kasin’, ‘Databases’, to\_date(‘2016-09-21’, ‘YYYY-MM-DD’));  
  
Adding each row to the table manually is an extremely inefficient way of handling data, especially if there is going to be quite a few rows and you already have the data in a text file or CSV file. In situations such as those, it is possible to have a shell script handle inputting the information for you. But that is going to come up in a later chapter. For now, add a few more rows to the table and a row or two to the table to get used to the command. Have one of those additions be:  
insert into members (first\_name, last\_name, role, join\_date) values (‘Rob, ‘Hancock’, ‘Linux’, to\_date(‘2017-09-21’, ‘YYYY-MM-DD’));  
  
While that entry was correct close to the start of the season, now that Rob is specializing more towards Web App administration, we should edit his entry in the database to be reflect this in his role. We can do this using the update command:  
update members set role=’Web’ where first\_name=’Rob’;  
  
Finally, while we currently have nothing that we need to delete from our database, let’s say that we do. Maybe someone from the Windows team quit and their name in Bob. To remove them we could use the delete command, like so:  
delete from members where role=’Windows’ and first\_name=’Bob’;

**Other Things to Look Into**Some other to things that you may want to research from here are:

* Encrypting a database (Good to do in real world, not worth it in CCDC)
* Accessing data using a script (Very useful for managing a database once it grows past a few rows and tables)
* Using fancy front ends for PostgreSQL such as phpPyAdmin