# Assignment 04 Install Docker and PostgreSQL

---

You will be installing Docker and PostgreSQL on your own computers.

**PostgreSQL** is one of many RDBMS packages available today. It is free and open-source.

**Docker** runs server programs in a sandboxed environment and provides better security, reliable installation, and ease of administration. It is also free and open-source.

Both applications will be used throughout the semester.

---

- Note that you may complete this assignment by performing the installations on your home / dorm desktop system or your laptop.
- For future in-class activites, you are *encouraged* but not *required* to bring a laptop with these applications already installed. If it is not possible, you must use the PCSE server **docker.pcs.cnu.edu** during any such classroom activies. Therefore, you should *email me* immediately if you are going to need an account on this system.

---

**Docker** is available at docker.com. The installation process is easy to follow, and there are no choices to make. Docker should be running on your computer at the end of this installation process.


- You must be running MacOS, Linux, or Windows 10 to install Docker. Docker will install on Windows 8 or 7, but requires additional components such as Docker Toolkit. You may try to use Docker on older operating systems, but I cannot give you tech support.

---

**PostreSQL** must be installed using docker. Open up your favorite command terminal and run each of these commands in order.

```
docker ps
```

This command will show you a list of all running processes controlled by docker. It should show nothing at this stage of the assignment.

```
docker ps -a --no-trunc
```

Like before, but with more details.

```
docker images
```

This command will show you all installed images. An image is like a server package — it comes with all of the commands, libraries, and configuration files needed for a program. Again, it should be empty at the moment, but you should come back later and try these commands after we install **PostgreSQL**.

Docker has commands to install new images from a central repository. Docker will install applications automatically when you use the `run` command if the application is not already installed on your system. Try this:

```
docker run --name my_postgres -e POSTGRES_PASSWORD=guessme -d postgres
```

Note that, on a shared server, you should replace "my" with your name.

Docker runs server processes in isolated environments called *containers*, which work somwhat like a virtual machine. **my_postgres** in the command above is the name of the container running the postgres process, an id that is is used by other *docker* commands. The **-e** flag passes environment variables to the process, and the **-d** flag indicates that the process is to run in the background after executing the command.

The final parameter, **postgres**, is the name of the image to load from the docker repository.

Try `docker ps` again to make sure it is running. Then make a note of the server password and we can continue.

The first client we will try will be psql, which can be run on the command line. We need docker to run psql because the postgres server is running in a docker- isolated container.

```
docker run -it --rm --link my_postgres:postgres postgres psql -h postgres -U postgres
```

The **i** and **t** flags are responsible for creating a terminal with input and output for the command. The **rm** flag will remove the container once the process ends.

The **link** flag is very important, as it allows this process to have communication with the otherwise isolated postgres process. Link's target is in two part syntax, container : alias. *container* is the postgres container we created earlier, while *alias* is what that container will be called in this new command. The alias can be used as a hostname in our new commands.

Aliasing a container as a hostname is very powerful. You could actually *ping* this alias from within your new container. This is part of the magic of Docker! When docker links together two containers, it creates within the new client a host alias — an IP address and hostname — that is actually mapped to the linked container. This means that you can use any application to talk to the linked container using that special hostname.

The two arguments **postgres psql** go together. **postgres** is the image,  while **psql** is the command *within* the image to run (instead of the default postgres server). We're using the same image to run a client that we used to run the server, just different commands.

The rest of the arguments are actually arguments for that **psql** command, which specify to use *postgres* as both the hostname and username. You will need to supply the password after the prompt.

Take a look at https://www.postgresql.org/docs/9.5/static/app-psql.html for details on how it works. Try some SQL commands, like `select 1+1;` and `select NULL is NULL;`. Don't forget the semicolons.

Try this command :

```
select * from pg_tables;
```

## Take a screenshot of this completed command and save it somewhere.

**Note: you will need to quit with Control-D, an unprintable character meaning STOP.**

We need a better interface with postgres than psql, so we will run a background server program that connects to the postgres server as a client, and provides a php/javascript web site as a server that we can connect to. This program is called pgadmin4.

```
docker run --name my_pgadmin4 --link my_postgres:postgres -p 5050:5050 -d
bernardelli/pgadmin4
```

We have seen many of these command options before. The image name bernardelli/pgadmin4 is simply a namespace/source format that picks a specific version of an image uploaded by a specific user or project. Popular products go by one name, like *postgres*.

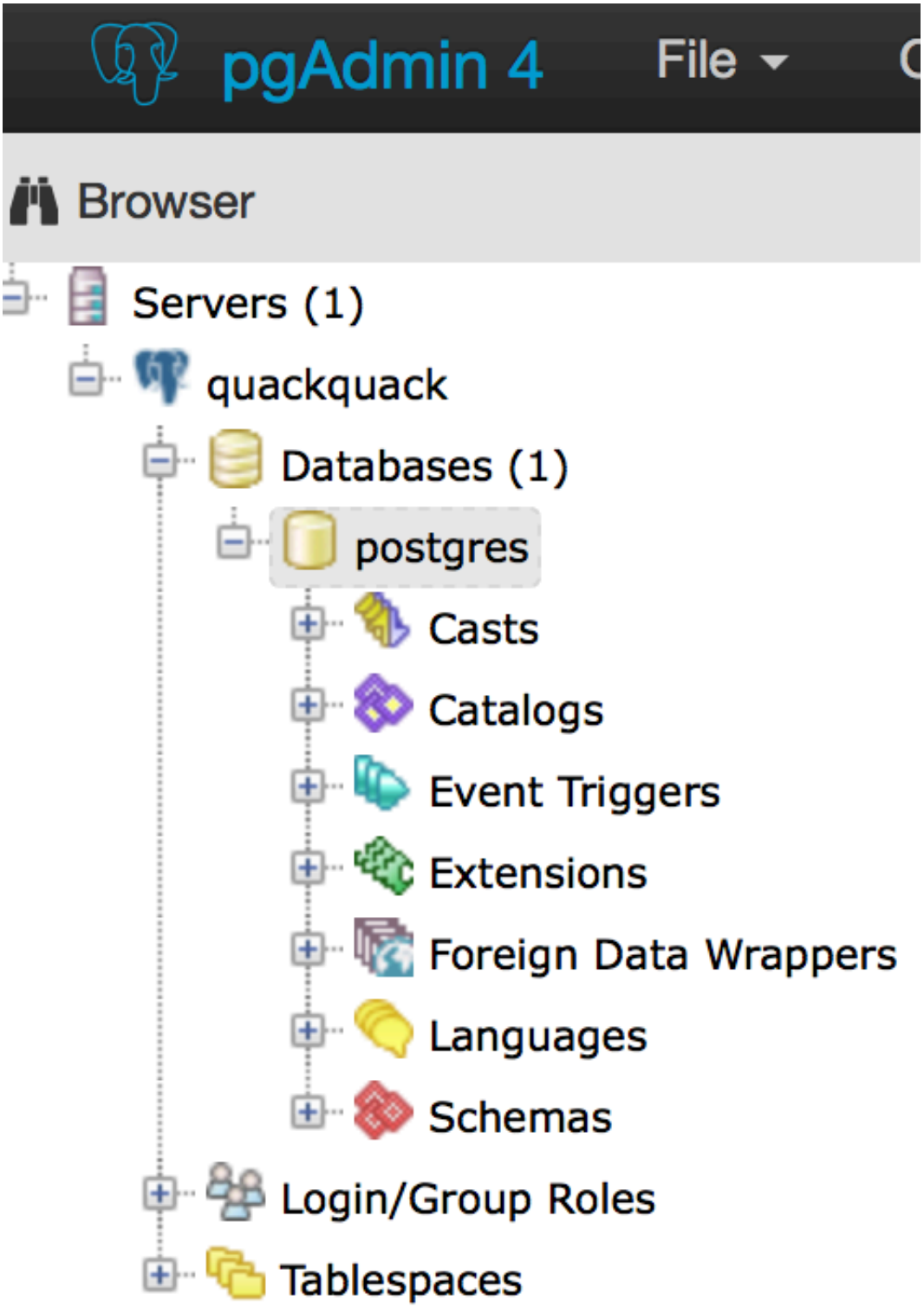
The **-p** option forwards network traffic from the container to the host (your computer). The format of the option is hostport:containerport. Container port 5050 is used by the pgAdmin4 server to publish itself, and by forwarding it on to 5050 on our host, we gain the ability to connect to it through a browser.

Open up your favorite web browser and connect to localhost:5050.


Once pgAdmin4 loads, right-click **Servers**, select **Create** -> **Server**. One the tabbed wizards, name your server whatever you like then click the *connection* tab. Here you want to fill in :

- **host** : postgres
- port : 5432
- database : postgres
- username : postgres
- password : guessme (created when running server above)
- save password : yes

Hit save and then refresh your browser.




Open the server objects, right-click the database and start the Query Tool. Run `select * from pg_tables;` as before, and call it a night.

### Take a screenshot of this completed command and save it somewhere.

Finally, you can use `docker stop` and `docker start containername` to stop and start your processes. If it fails, you can always use `docker rm containername` and run them again.