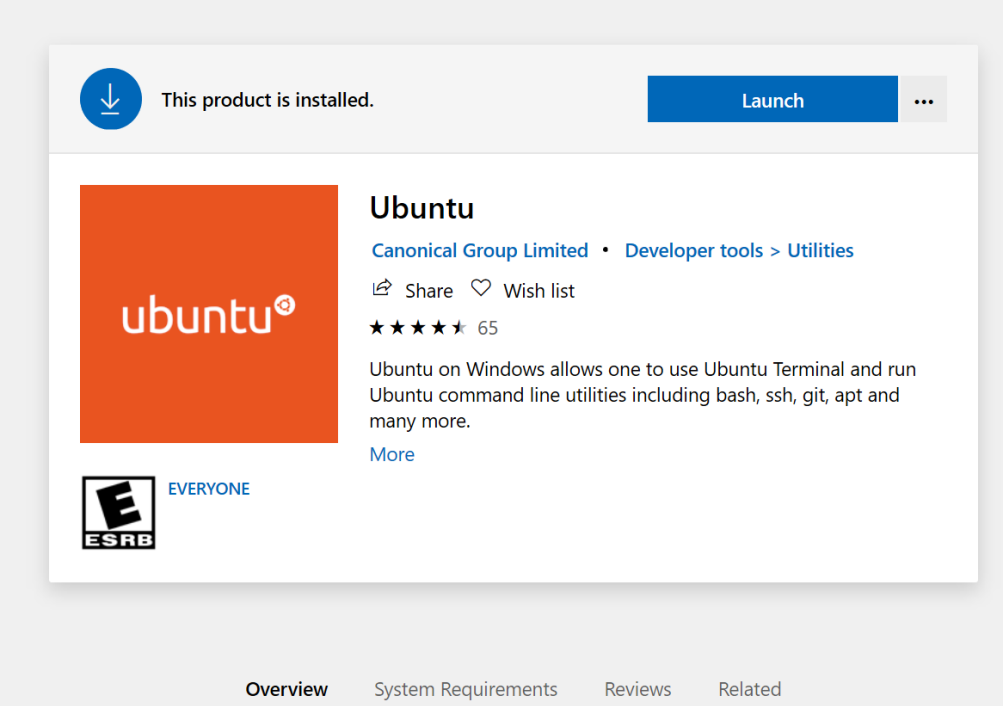
Note: If you are using Linux/Ubuntu on Virtual Machine then you can skip straight to Step 2.

Note: Do not copy paste commands mentioned below. Type each command out as there are some issues when you paste the command copied from this document directly.

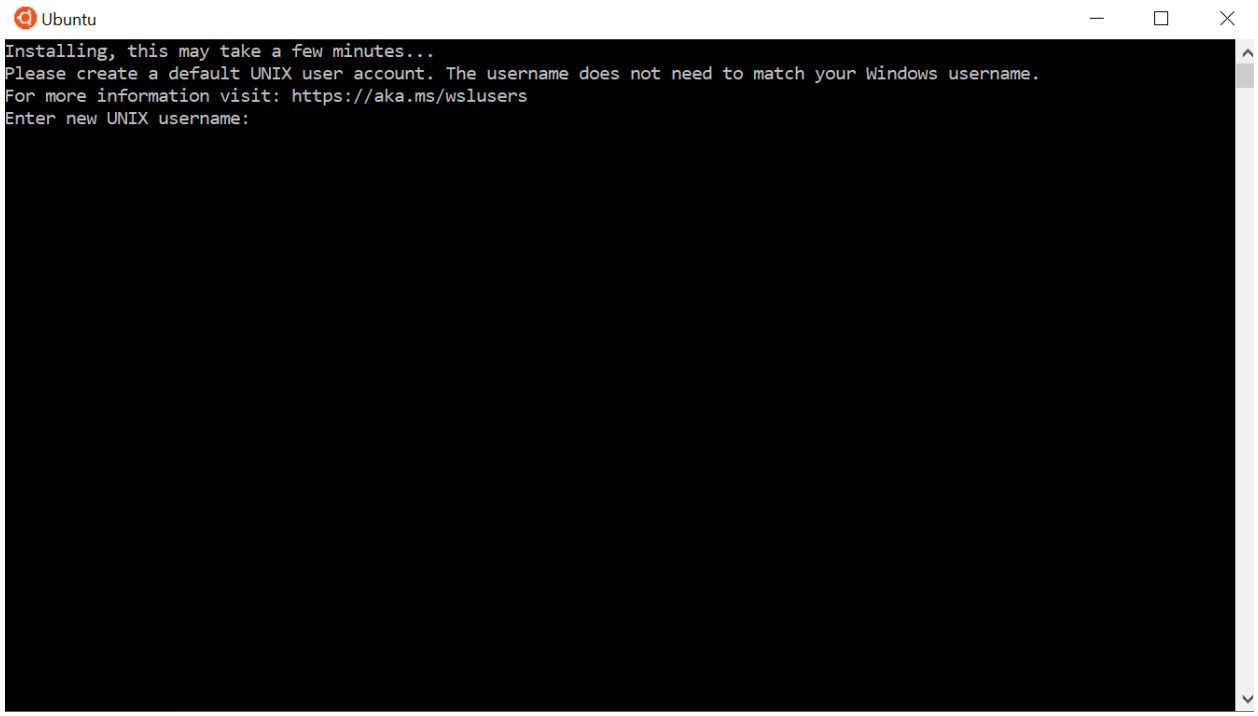
Step 1 (Only tested for Windows 10 users):

Install the Ubuntu App from Microsoft App store

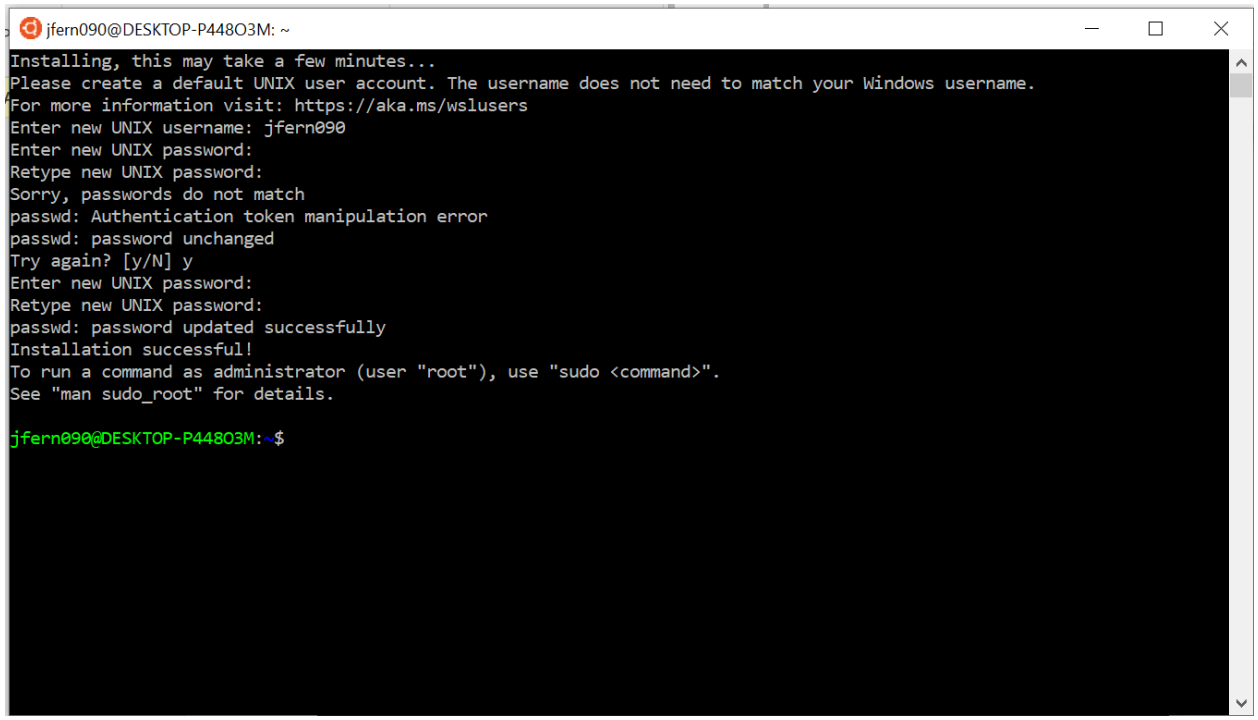


Launch the application and you should see this screen:

Note: Application can be launched from the Start up menu as well. It also takes a few mins to provide a prompt to request for the super user details.



Once you've entered your details you will be able to see a prompt with your username on it. In this case it is my uOttawa id:

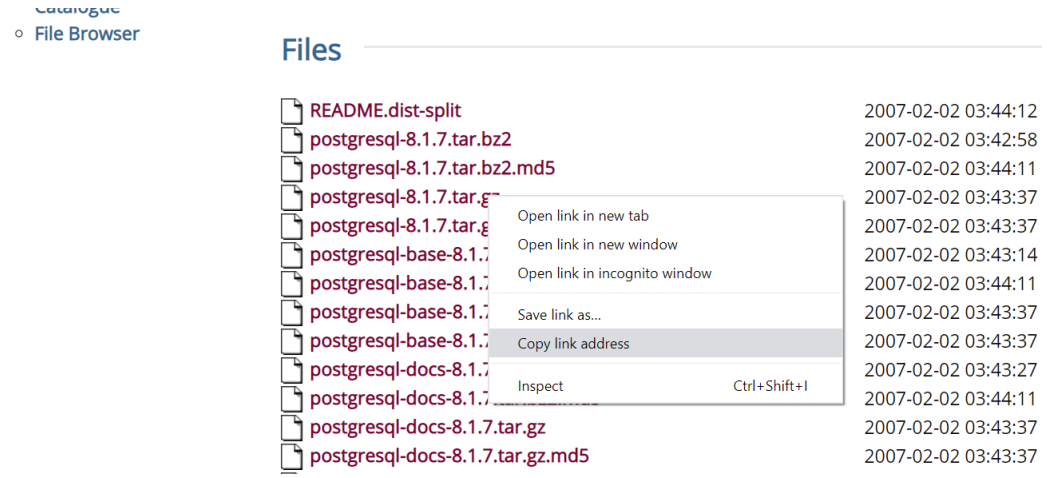


Step 2. Download and extract postgresql source files (From here on these steps are for all users)

Simply go to the FTP server link : https://www.postgresql.org/ftp/source/ on your favourite browser.

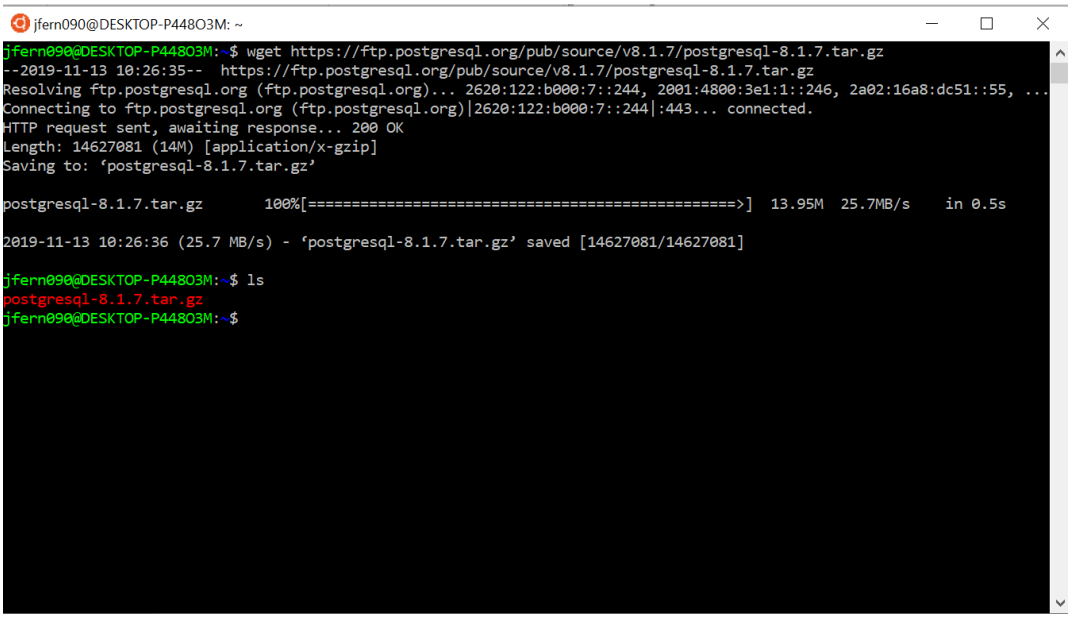
Scroll down to the version you would like to install. For this document I've used v8.1.7. You could also use v8.1.4.

Now right click on the postgresql-8.1.7.tar.gz file in the list and copy the link address:



Go back to the Ubuntu app and type the below command and link

wget https://ftp.postgresql.org/pub/source/v8.1.7/postgresql-8.1.7.tar.gz



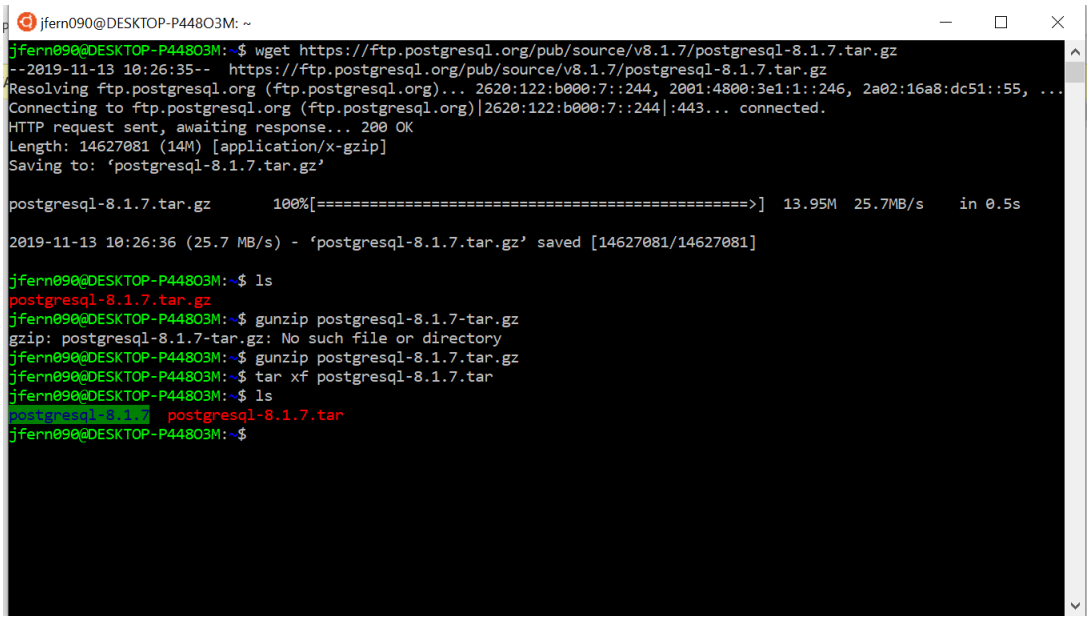
This step will download the file and you will be able to see it in your /home/<user name>/ folder.

Next run these two commands to extract the source files:

gunzip postgresql-8.1.7.tar.gz

tar xf postgresql-8.1.7.tar

This will create a postgres-8.1.7 folder in the same destination:



Step 3: Install requisite libraries

Here's the list of commands for libraries that we found were not available in this Ubuntu application. If you are using Ubuntu/Linux on a VM you probably have these libraries already none the less, it's better to check before the next step.

sudo apt-get update

sudo apt-get install make

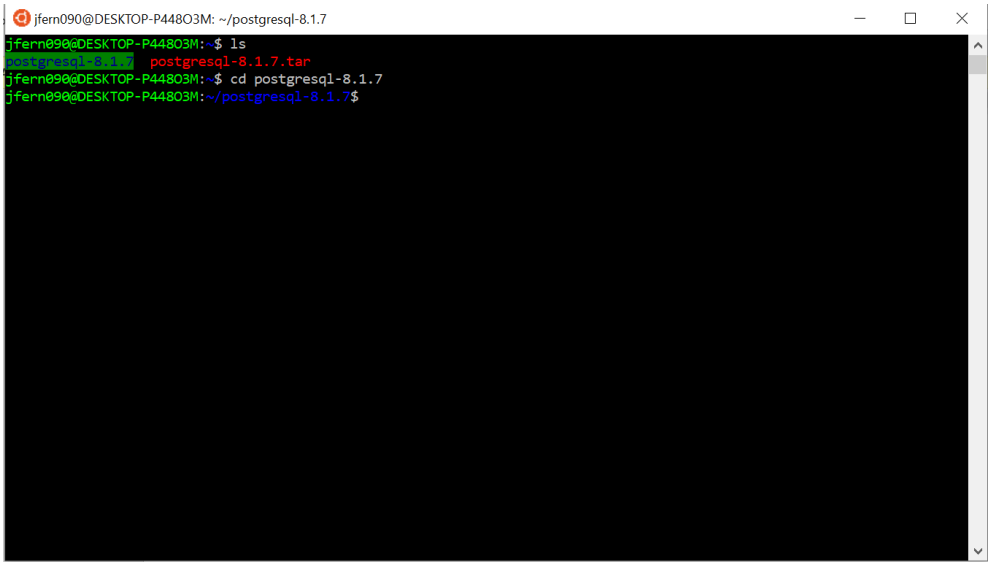
sudo apt-get install libreadline7 libreadline-dev

sudo apt-get install zlib1g zlib1g-dev

sudo apt-get install clang

Note: We have to use clang to compile this version of Postgres. An older version of GCC would work but the current GCC versions will throw some errors during compilation.

Step 4: Configure this version of postgresql



Use this command to start configureation:

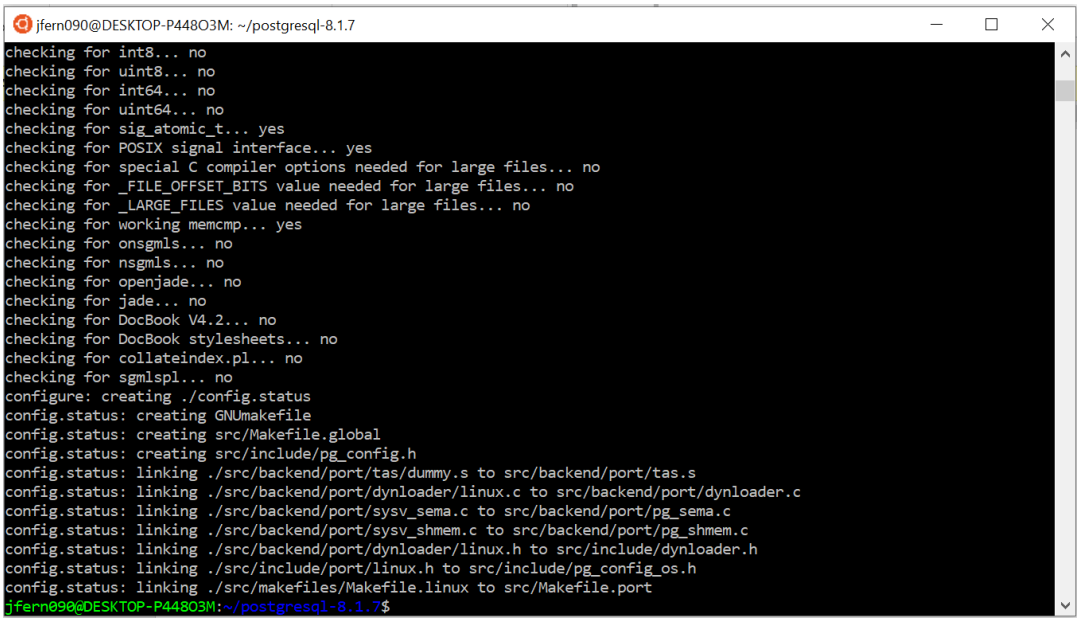
sudo ./configure CC=”clang” CFLAGS=”-O1” --enable-debug -enable-depend

if the above fails, try the option below

sudo ./configure CC=/usr/bin/clang CFLAGS="-O1"

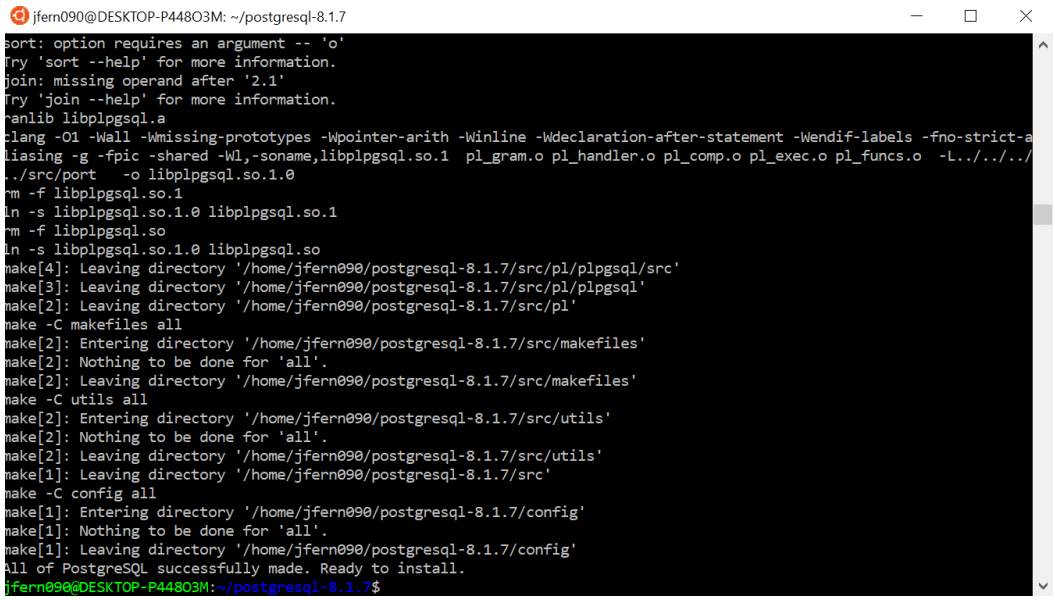
*Note that it’s a capital O and not a Zero

This will throw an error in case we missed any other requisite libraries. In case everything goes well we will see this screen.



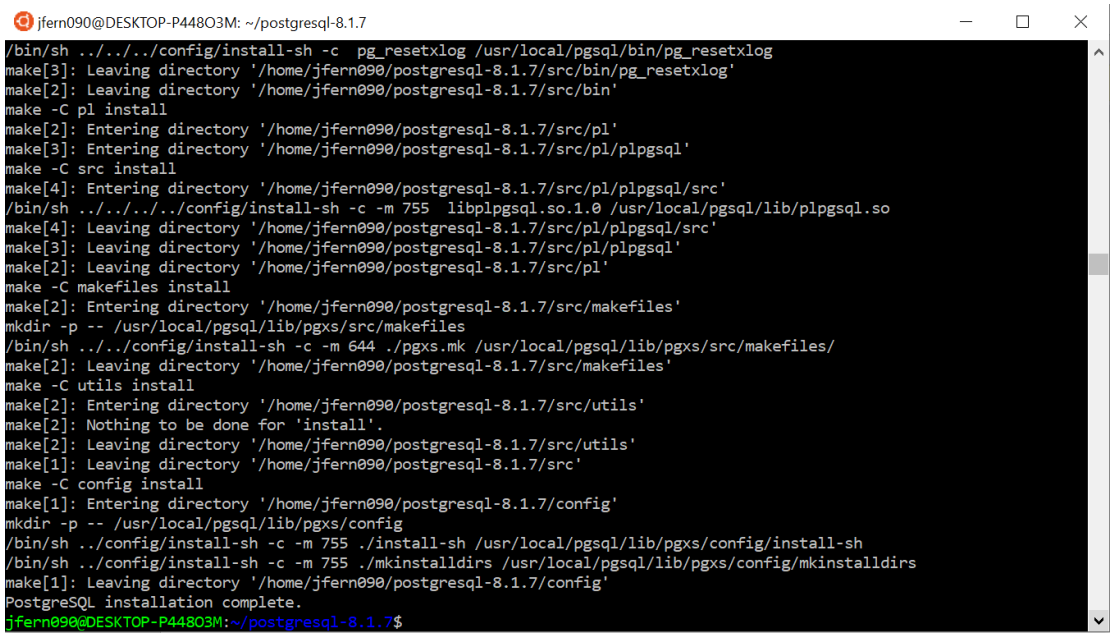
Next, type this command:

sudo make



Great! Now it's time to install it with this command

sudo make install



Great all done with this step.

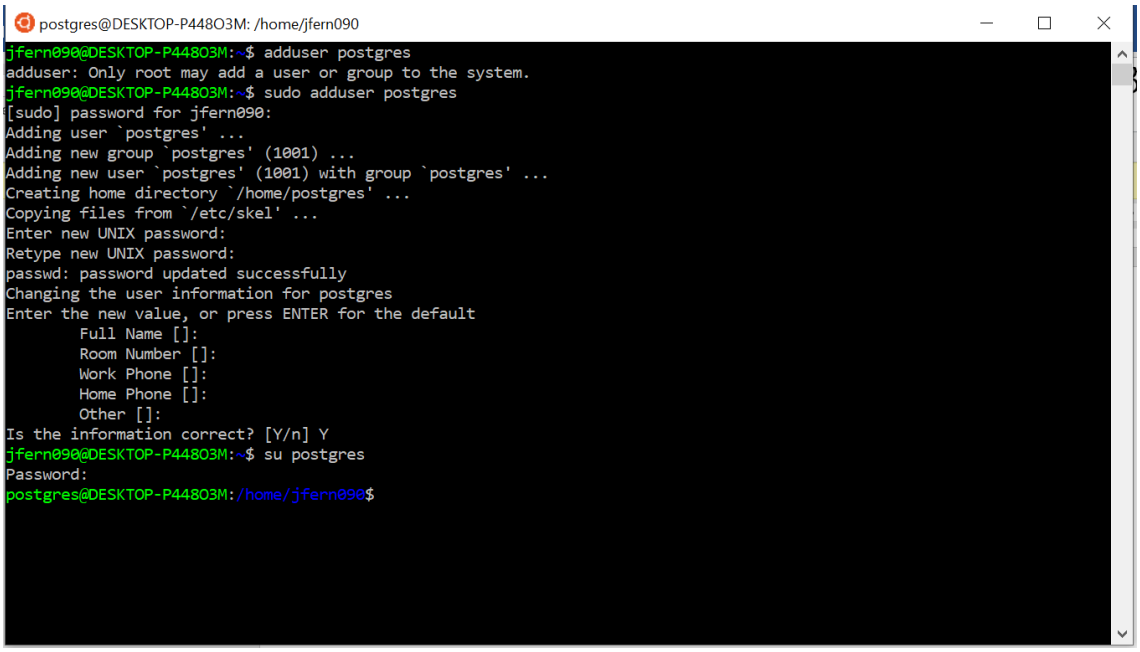
Step 5: Create postgres user and assign rights to files

Create a folder data in this path where all the files can be located:

sudo mkdir /usr/local/pgsql/data

sudo adduser postgres

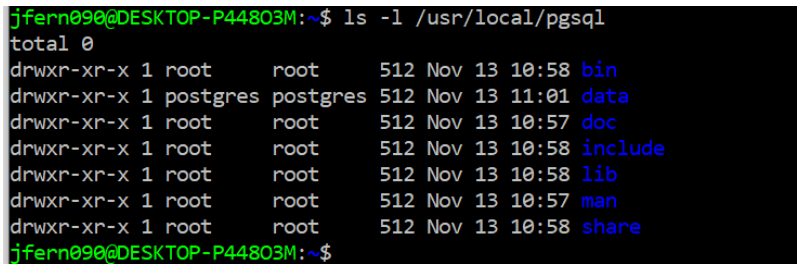
su postgres



By this step you will have created a new user. Now time to assign access to the new user for the data folder

Chown postgres /usr/local/pgsql/data/

Confirm that postgres user has access to the folder:



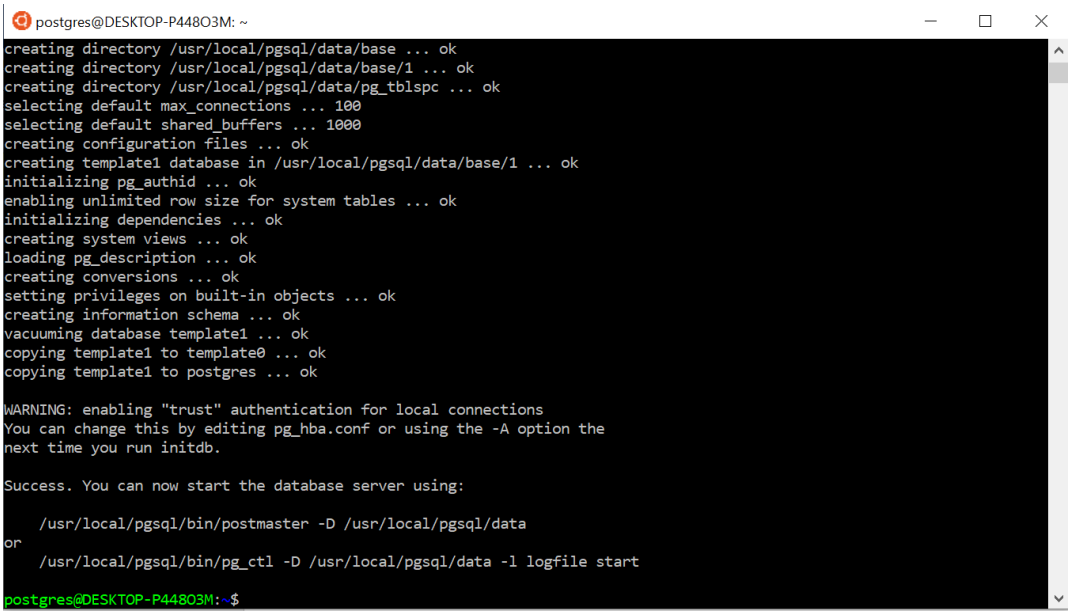
Step 6: Initialize DB and start the postgres server

Switch the user:

su postgres

Run this command to initialize the data folder with the necessary files

/usr/local/pgsql/bin/initdb -D /usr/local/pgsql/data/



Once you see the **Success** message you can then start the postgres server

Use this command to start the server:

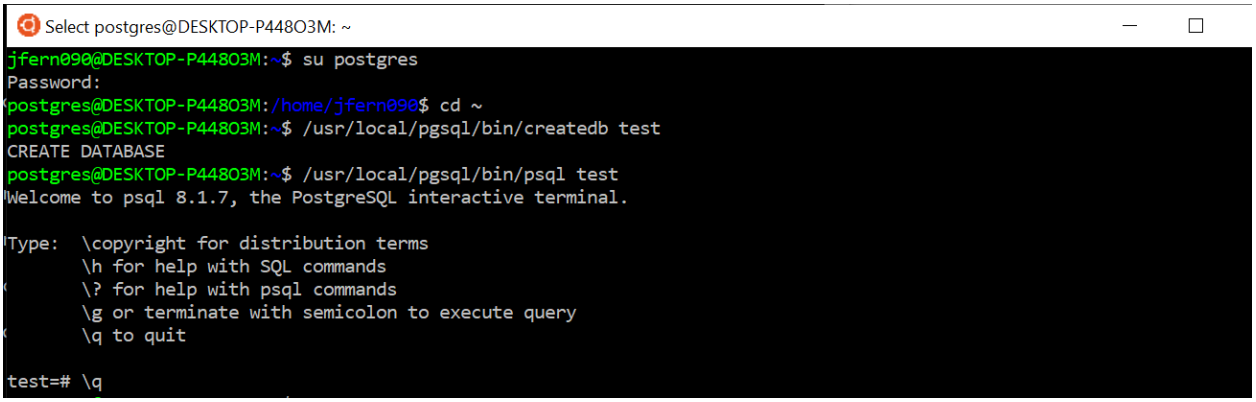
/usr/local/pgsql/bin/pg_ctl start -D /usr/local/pgsql/data

```
postgres@DESKTOP-P448O3M:~$ /usr/local/pgsql/bin/pg_ctl  start -D /usr/local/pgsql/data
postmaster starting
postgres@DESKTOP-P448O3M:~$ LOG:  database system was shut down at 2019-11-13 12:20:52 STD
LOG:  checkpoint record is at 0/38FF90
LOG:  redo record is at 0/38FF90; undo record is at 0/0; shutdown TRUE
LOG:  next transaction ID: 565; next OID: 10794
LOG:  next MultiXactId: 1; next MultiXactOffset: 0
LOG:  database system is ready
LOG:  transaction ID wrap limit is 2147484146, limited by database "postgres"
```

Great! Now open another terminal / Ubuntu instance and run this command to first create a database and then access that database on the currently running postgres server.

/usr/local/pgsq/bin/create test

/usr/local/pgsq/bin/psql test



Great! Now you are able to access the server. Stop the server with this command

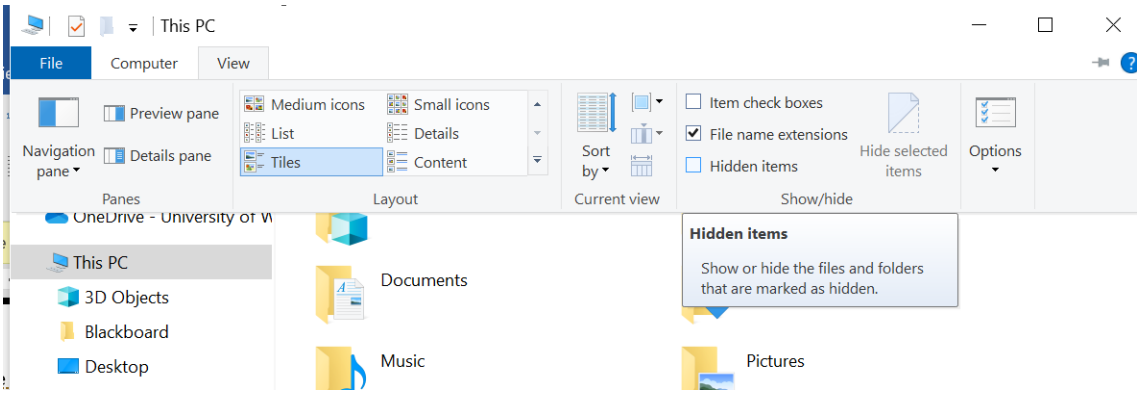
/usr/local/pgsql/bin/pg_ctl stop -mimmediate -D /usr/local/pgsql/data

Step 7: Add files from Windows Explorer (Only for users who are using this approach for the project. VM users can normally add their files normally)

I have added a schema.sql file to the Project folder. Use that to initialize the test database with the new tables. When adding files I've notices Ubuntu terminal needs to be restarted so please do so before you do this step and start it again after you've added the file.

Here's how you can view your Ubuntu files from Windows Explorer.

First, display all Hidden files. Select/Check the Hidden items option here



Follow this link on the explorer:

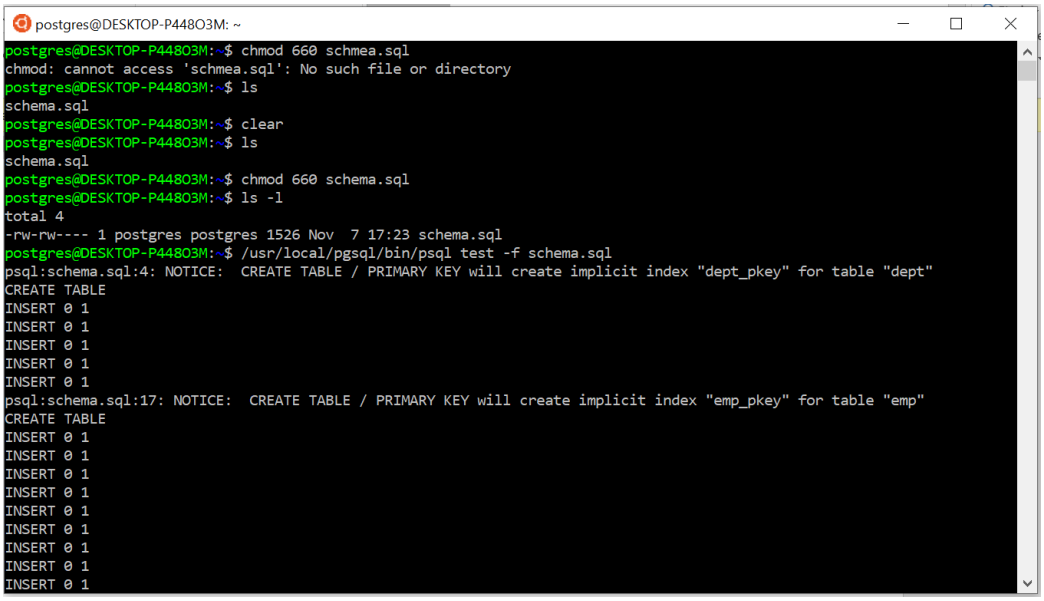
C:\Users**<Windows user>**\AppData\Local\Packages\CanonicalGroupLimited.UbuntuonWindows_79rhkp1fndgsc\LocalState\rootfs\home\postgres

You will notice within the rootfs file you can see all the related Ubuntu files. Here you can edit your C Files here. (which will be done a little later in this document).

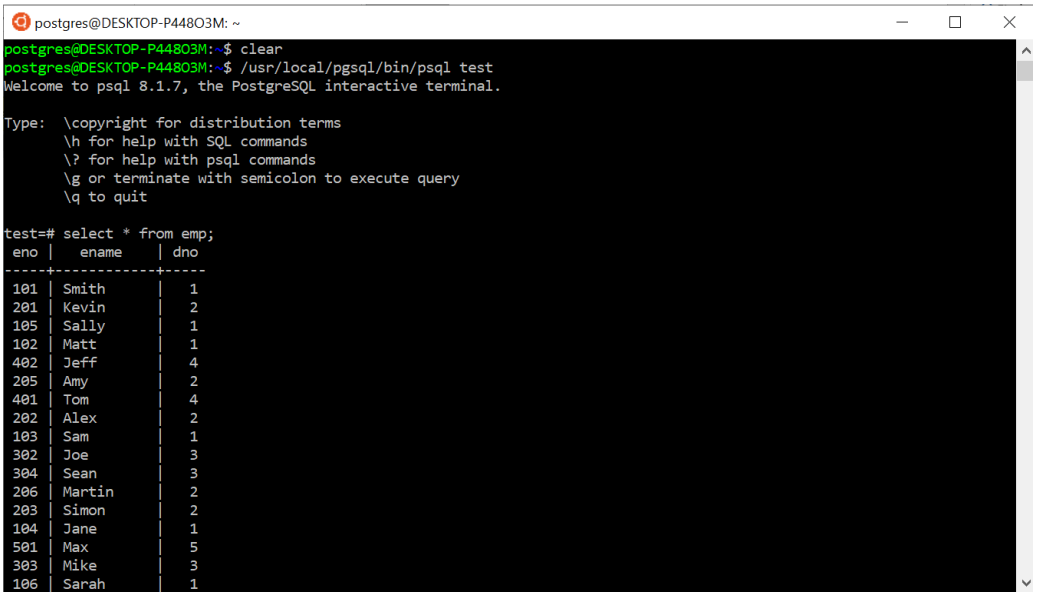
Add the schema file here to this location. Start the Ubuntu terminal and postgres server and run this command to first reset the access rights to the schema file and then the next command to create all the table and rows in the test data base.

chmod 660 schema.sql

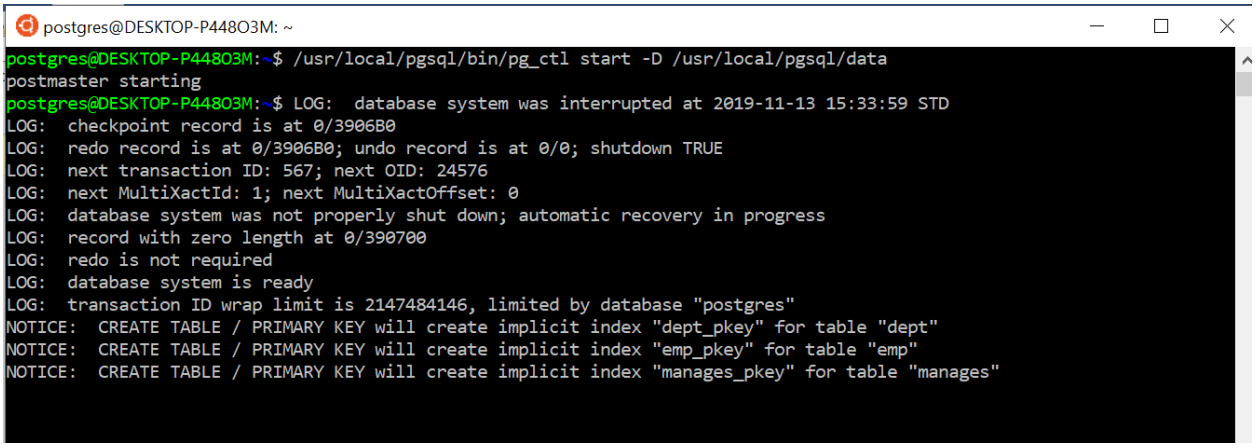
/usr/local/pgsql/psql test -f schema.sql



Test the existence of each table. Here's a snapshot for that.



You will notice on the other terminal where you started your server will have these updates as well.



Step 8: Change some C code and see the output.

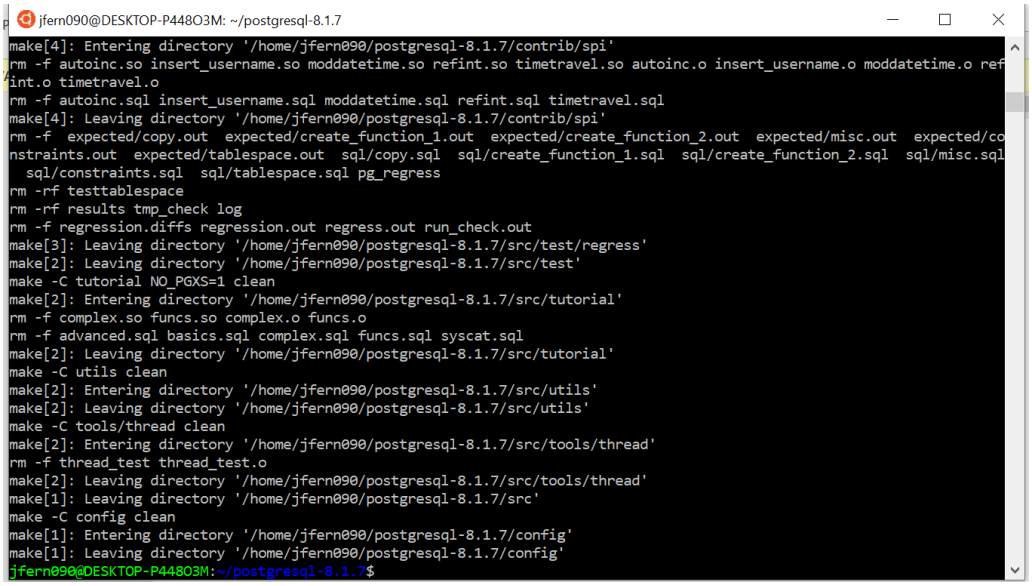
This is the final step where we will update the postgres C code to just print out at what time it excuted the symmetric hash join algorithm. This step assumes you have done some work on nodeHash.c and nodeHashjoin.c files in order to implement the symmetric hash join algorithm.

1. Change postgresql.conf settings by turning off these values. The postgres.conf file can be found in the /usr/local/pgsql/data folder

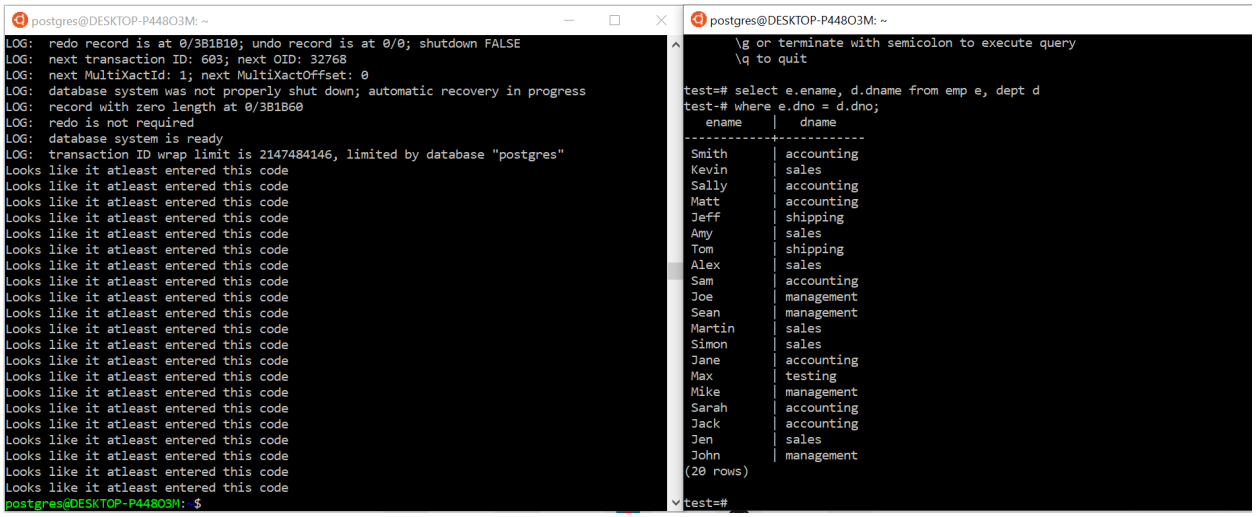
   enable_mergejoin = off
   enable_nestloop = off

2. On editing postgresql.conf the access to the file will change so make sure to run this command *chmod 660 /usr/local/pgsql/data/postgresql.conf*

3. After you've added your let's say print statement in one of these file it's time to clean the postgresql files and re compile the codes. Run the *sudo make clean* command in the postgresql-8.1.7 folder.



4. Run the *sudo make* and *sudo make install commands* again here so that the files get recompiled.
5. Start the postgres server now and run a simple join query. You should see your print statement as an output on the terminal where the server's log details are currently displayed.
   In the image below, the terminal on the right is where I ran the query and the log details where displayed on the terminal on the left which I used to start the postgres server.



The end