

Exercise # 4

Setting up your postgres database is an important element of the entire process. To do this it is imperative to understand the organization of the installation files on your system. Irrespective of the operating system, these files in focus will remain the same, and will be placed in similar directories.

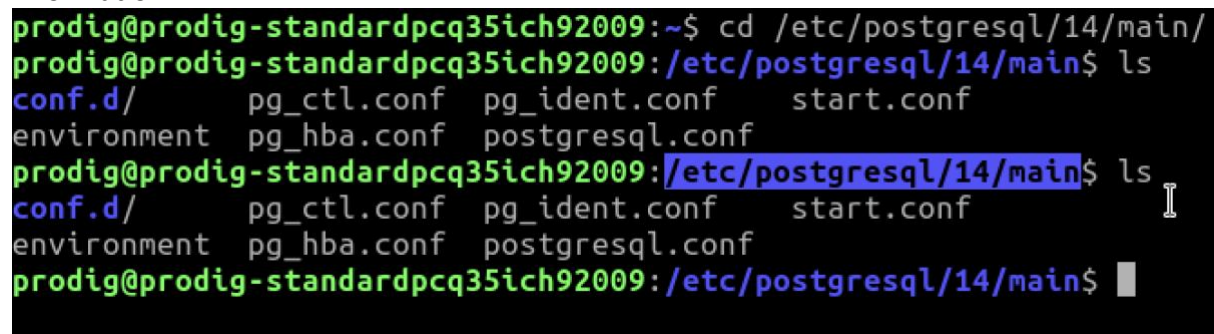
Type the following line of code in your command line terminal.

```
cd /etc/postgresql/14/main
```

Once, you are in the directory type the command.

```
ls
```

to print the content on the folder on your screen. Here you are going to see the following information.



```
prodig@prodig-standardpcq35ich92009:~$ cd /etc/postgresql/14/main/  
prodig@prodig-standardpcq35ich92009:/etc/postgresql/14/main$ ls  
conf.d/      pg_ctl.conf  pg_ident.conf  start.conf  
environment  pg_hba.conf  postgresql.conf  
prodig@prodig-standardpcq35ich92009:/etc/postgresql/14/main$ ls  
conf.d/      pg_ctl.conf  pg_ident.conf  start.conf  
environment  pg_hba.conf  postgresql.conf  
prodig@prodig-standardpcq35ich92009:/etc/postgresql/14/main$
```

If you notice, these files all have an extension .conf, suggesting these may be some kind of configuration files. These files are important part of our work on understanding how to manage our database, its access and on how to secure it in the long run. Among other things, which a db administrator learns with experience.

Let us open the following two file,

postgresql.conf and the
pg_hba.conf file

these files are the one's edited by most users to manage access to their databases, and to ensure that their database is secure. In some instances, one might have to reduce the security, to be able to learn to work around some important exercises, which we are going to do.

Let's explore the contents of these files, one by one. For that you will need to type the following command on your screen.

```
nano postgresql.conf
```

If you scroll down a little you will find the following screen. Here you can see that some text is in while. This is the text that is not commented and is a part of the active database protocol. Here two important things can be changed in case required by the user.

1. The access ports.
2. The maximum number of connections allowed to the database.

```
# CONNECTIONS AND AUTHENTICATION
#-----

# - Connection Settings -

#listen_addresses = 'localhost'          # what IP address(es) to listen on;
#                                          # comma-separated list of addresses;
#                                          # defaults to 'localhost'; use '*' for all
#                                          # (change requires restart)
port = 5432                              # (change requires restart)
max_connections = 100                    # (change requires restart)
#superuser_reserved_connections = 3      # (change requires restart)
unix_socket_directories = '/var/run/postgresql' # comma-separated list of directories
#                                          # (change requires restart)
#unix_socket_group = ''                  # (change requires restart)
#unix_socket_permissions = 0777         # begin with 0 to use octal notation
#                                          # (change requires restart)
#bonjour = off                           # advertise server via Bonjour
#                                          # (change requires restart)
#bonjour_name = ''                       # defaults to the computer name
#                                          # (change requires restart)

# - TCP settings -
# see "man tcp" for details

#tcp_keepalives_idle = 0                 # TCP_KEEPIRL, in seconds;
#                                          # 0 selects the system default
```

You will find many databases where the access ports have been modified, to ensure a certain higher grade of obscurity. If you want, you can change your default access port, however, we will as a common practice let it as, it is for now. We are not changing any other thing either. In case you plan to do so, you will be better off restarting your virtual machine. For the changes to take effect.

Can you scroll through the file and identify the lines that are not commented out? And add them to a text file. You might also be able to guess the purpose of most of these lines of text.

The other file is the `pg_hba.conf` file. This file contains, your host-based authentication information. And is an important component of how your database is accessed. And ensures, its security in part. To be able to access this file, you might need to use your superuser privileges.

```
sudo nano /etc/postgresql/14/main/pg_hba.conf
```

If you scroll down to the end of this file, you will find some uncommented set of lines, that defines how the database is accessed from within the system and by an external user, as well. These lines of codes are important, and will be required some modifications, as we progress during the exercise.

```
# DO NOT DISABLE!
# If you change this first entry you will need to make sure that the
# database superuser can access the database using some other method.
# Noninteractive access to all databases is required during automatic
# maintenance (custom daily cronjobs, replication, and similar tasks).
#
# Database administrative login by Unix domain socket
local  all             postgres           peer

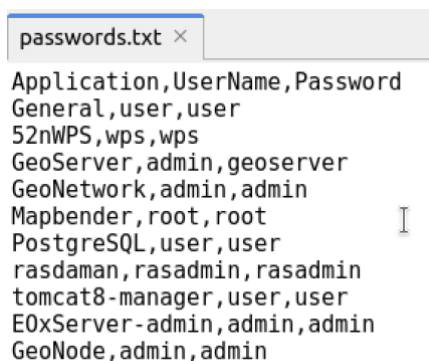
# TYPE  DATABASE      USER            ADDRESS           METHOD

# "local" is for Unix domain socket connections only
local  all             all              peer
# IPv4 local connections:
host   all          all             127.0.0.1/32      scram-sha-256
# IPv6 local connections:
host   all          all             ::1/128           scram-sha-256
# Allow replication connections from localhost, by a user with the
# replication privilege.
local  replication    all             peer
host   replication    all             127.0.0.1/32      scram-sha-256
host   replication    all             ::1/128           scram-sha-256
```

Methods, here are the key to access. Some provide better password security and reduce the chances of a password being sniffed such as scram-sha-256.

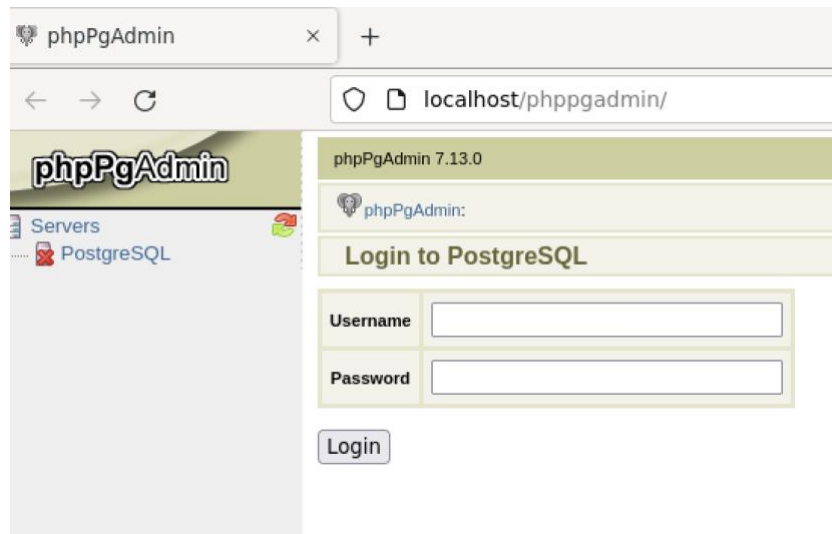
If you check in the programs and accessories section, you will also be able to find a phpPgAdmin application. It is not different from the usual pgAdmin application. However, if you install pgAdmin on your host computer, even that can be used to access this database on the virtual machine. We shall try that a little later down the line.

But before that you need to check the desktop of your virtual machine. It contains a textfile containing all passwords for each preinstalled software. For postgres the username and passwords can be found too.

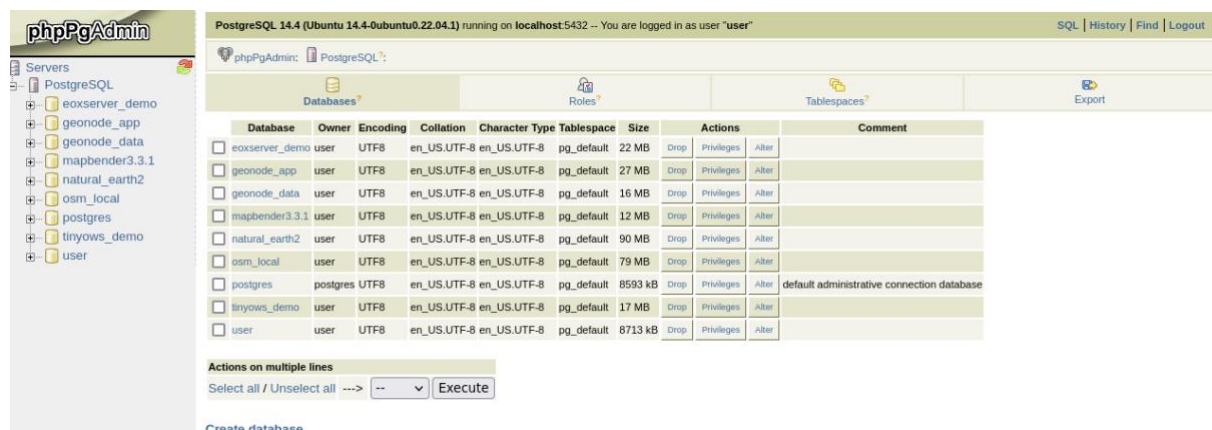


```
passwords.txt x
Application,UserName,Password
General,user,user
52nWPS,wps,wps
GeoServer,admin,geoserver
GeoNetwork,admin,admin
Mapbender,root,root
PostgreSQL,user,user
rasdaman,rasadmin,rasadmin
tomcat8-manager,user,user
EOxServer-admin,admin,admin
GeoNode,admin,admin
```

Your browser based phpPgAdmin is more of a php webpage, that helps you connect to your database. This is not always required, and you can directly access your database from the command line or an external pgAdmin tool too.



Type your username and passwords from the text file, and you will be able to connect to your database. Once, you are logged in the following screen will appear. Many of you might find it unsettling since, accessing a browser within a virtual machine is a time-consuming feat. However, the same can be eased out.



You can close your browser and go to your command line terminal and type the following line of code.

ifconfig

```
prodig@prodig-standardpcq35ich92009:~$ ifconfig
enp0s1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.64.3 netmask 255.255.255.0 broadcast 192.168.64.255
    inet6 fe80::ada9:ba28:b60:1548 prefixlen 64 scopeid 0x20<link>
    inet6 fdca:ceae:2fd1:b1e7:9aa9:1fd7:779f:5598 prefixlen 64 scopeid 0x0<global>
    inet6 fdca:ceae:2fd1:b1e7:a73a:9251:812d:bb8e prefixlen 64 scopeid 0x0<global>
    ether 0a:1d:89:9a:aa:e4 txqueuelen 1000 (Ethernet)
    RX packets 21479 bytes 22035370 (22.0 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 10119 bytes 1148563 (1.1 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Towards the top left you can see an IP address mentioned. This is an ip address that can be used to access your database, from an external browser or computer. However, for that you will need to change the postgresql.conf file too. Or else you will be getting the following message.

Forbidden

You don't have permission to access this resource.

Apache/2.4.52 (Ubuntu) Server at 192.168.64.3 Port 80

To fix this you will first have to edit the file.

```
sudo nano /etc/postgresql/14/main/postgresql.conf
```

And go to the line that says `#listen_addresses = 'localhost'`

```

#-----
# CONNECTIONS AND AUTHENTICATION
#-----

# - Connection Settings -

#listen_addresses = 'localhost'          # what IP address(es) to listen on;
#                                           # comma-separated list of addresses;
#                                           # defaults to 'localhost'; use '*' for all
#                                           # (change requires restart)
port = 5432                               # (change requires restart)
max_connections = 100                     # (change requires restart)
#superuser_reserved_connections = 3       # (change requires restart)
unix_socket_directories = '/var/run/postgresql' # comma-separated list of directories
#                                           # (change requires restart)
#unix_socket_group = ''                   # (change requires restart)
#unix_socket_permissions = 0777          # begin with 0 to use octal notation
#                                           # (change requires restart)
#bonjour = off                            # advertise server via Bonjour

```

And change it as follows,

```

#-----
# CONNECTIONS AND AUTHENTICATION
#-----
# - Connection Settings -

listen_addresses = '*'          # what IP address(es) to listen on;
                                # comma-separated list of addresses;
                                # defaults to 'localhost'; use '*' for all
                                # (change requires restart)
port = 5432                     # (change requires restart)
max_connections = 100           # (change requires restart)
#superuser_reserved_connections = 3 # (change requires restart)
unix_socket_directories = '/var/run/postgresql' # comma-separated list of directories
                                # (change requires restart)
#unix_socket_group = ''         # (change requires restart)
#unix_socket_permissions = 0777 # begin with 0 to use octal notation
                                # (change requires restart)
#bonjour = off                  # advertise server via Bonjour

```


Once, you have made the above, change the same database should be accessible using pgAdmin from your host computer. All that you will need to do is type the respective IP address of your host computer on the address bar of your pgAdmin in place of localhost.

You can add the respective information in your pgAdmin connection window and try and connect to the database. However, you might get an error message.

Register - Server

General **Connection** Parameters SSH Tunnel Advanced

Host name/address: 192.168.64.3

Port: 5432

Maintenance database: postgres

Username: user

Kerberos authentication? ☐

Password:

Save password? ☐

Role:

Unable to connect to server: connection failed: FATAL: no pg_hba.conf entry for host "192.168.64.1", user "user", database "postgres", SSL encryption connection to server at "192.168.64.3", port 5432 failed: FATAL: no pg_hba.conf entry for host "192.168.64.1", user "user", database "postgres", no encryption

Close Reset Save

Hence, it seems we need to fix yet an issue, with our pg_hba.conf file. For that open the pg_hba.conf file using the following command.

```
sudo nano /etc/postgresql/14/main/pg_hba.conf
```

and add the following line.

```
# TYPE DATABASE USER CIDR-ADDRESS METHOD
host all all 0.0.0.0/0 scram-sha-256
```

this time instead of restarting your computer, you may just start your postgresql services by using the following command.

```
sudo service postgresql restart
```

If you try to connect, now you will be greeted with a similar window as following. You might want to open the server (this was a name I assigned to my server) and list all the sample databases already present there.

