



POSTGRESQL INSTALLATION GUIDE

Remote KTH-access	4
Connecting with Windows	4
Connecting with Mac/Linux:	5
Retrieve the password for your database account.	5
Accessing the database	6
Collaborating with your partner on the same Database	7
Populating the database with sample data	8
Moving files local → KTH-remote	8
A couple of things to note here:	9
Populating the KTH-remote database with sample datasets	10
Local installation	12
Using Brew	12
Starting PostgreSQL	14
Using Windows	15
Installing the installer	15
Starting the installation	16
Post installation settings	20
Adding psql to the PATH	21
Using the Windows installation	23
Stopping the db server	23
USEFUL COMMANDS WHILE EXPLORING THE DATABASE	25

POSTGRESQL INSTALLATION GUIDE

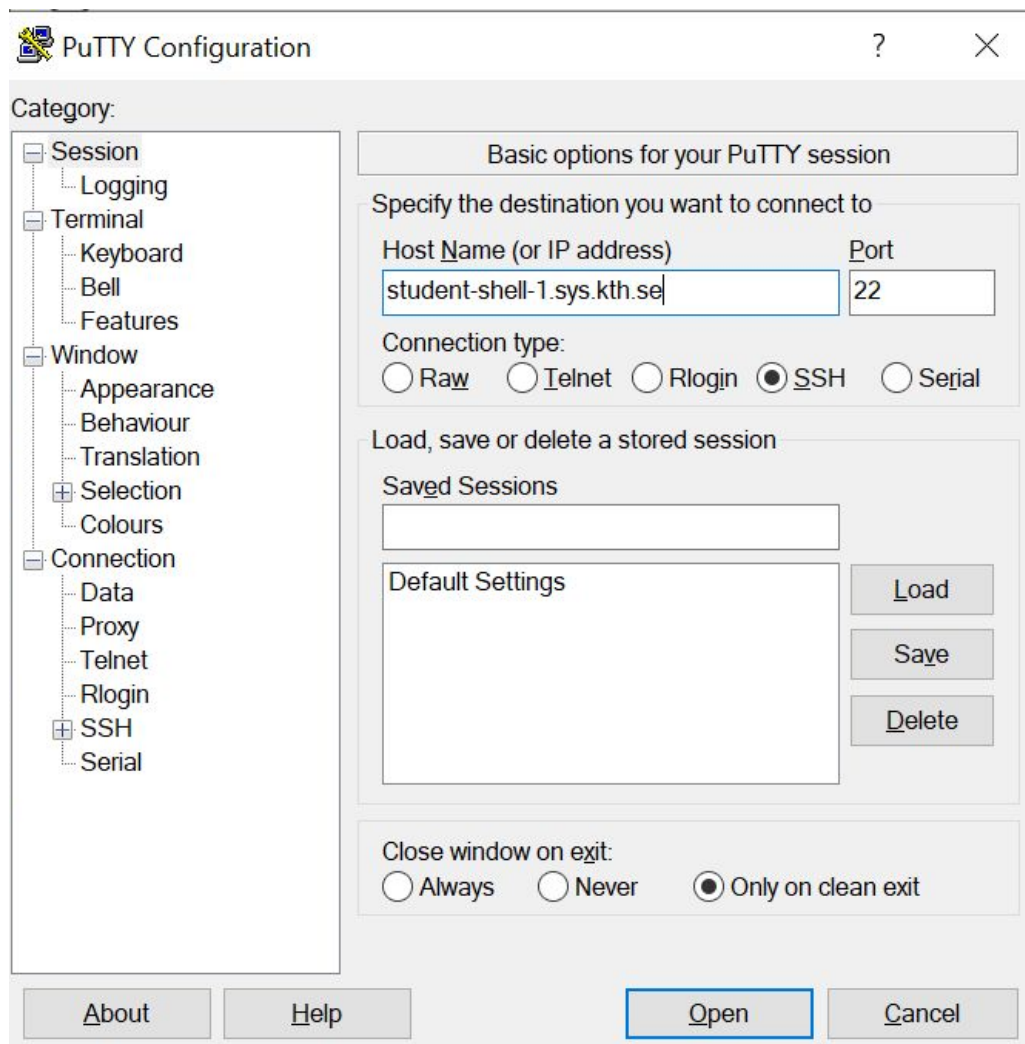
Remote KTH-access

This is the recommended method to access the KTH remote server.

Connecting with Windows

Download and install the latest version of Putty. Please follow the following instructions for installing and configuring [Putty](#).

Start Putty and add the Host IP-address of the KTH server:
student-shell-1.sys.kth.se, see the below figure for reference.



Then press “Open” and you should be sent to a terminal in your home folder of the KTH-server.



POSTGRESQL INSTALLATION GUIDE

Connecting with Mac/Linux:

We will be accessing KTHs server with SSH. The only thing you have to do is to login with your kthid and kth-password (i.e the first part of your kth-email).

```
$ ssh <Insert kthid here>@student-shell-1.sys.kth.se
```

You might get a warning that KTHs authenticity cannot be verified. In that case you just accept the connection regardless by writing **yes**.

You might need to write your kth-password in order to connect. Please note that no chars will be shown when you write your password! Not even stars (*)!

If everything is successful you should be in a new terminal environment!

Retrieve the password for your database account.

In order to be able to access the database environment, you will need a password. In order to retrieve the password you first navigate to the following directory:

```
$ cd Private/
```

When you are in the directory **Private/**, the password is in a file called **.psqlpw-nestor2.csc.kth.se**. Make sure that this file exists in this directory:

```
$ ls -lah
```

If there is no file named **.psqlpw-nestor2.csc.kth.se**, you do not have a database account and need to go to IT support and have them set you up with one.

If that file exists, you can print the contents of the file to retrieve the password. Please write the following command:

```
$ cat .psqlpw-nestor2.csc.kth.se
```



POSTGRESQL INSTALLATION GUIDE

```
@share-01: ~/Private
login as:
@share-01.csc.kth.se's password:
Last login: Mon Jul 22 19:37:30 2019 from
@share-01:~$ cd Private
@share-01:~/Private$ cat .psqlpw-nestor2.csc.kth.se
```

Accessing the database

If everything is working at this stage, you should be able to start the database with the following command:

```
$ psql -h nestor2.csc.kth.se
```

If that command fails please restart your terminal and redo the KTH-ssh (or Putty) connection again.

If the command is successful you will be prompted to write the password that was in the file **.psqlpw-nestor2.csc.kth.se**. Again, remember that when you write the password you will not get any chars not even stars (*)!

```
@student-shell-1:~$ psql -h nestor2.csc.kth.se
Password:
psql (10.12 (Ubuntu 10.12-0ubuntu0.18.04.1), server 9.5.21)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256, compression: off)
Type "help" for help.

=>
```



POSTGRESQL INSTALLATION GUIDE

Collaborating with your partner on the same Database

With your lab partner, you have to decide on who should be hosting the database. The partner that does not host the database will have to connect to the hosts' database every time they want to access it using the following commands:

```
$ psql -h nestor2.csc.kth.se

Password: <contents of .psqlpw-nestor2.csc.kth.se>

mydb=>\c <insert the host user name here>
```

```
psql (9.5.18)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256, compression: off)
Type "help" for help.

=> \c Username
```

Both you (the host) and your partner will be able to create tables in the database. But your partner will not have automatic access to your created tables. In order to give your partner read and write privileges on a table perform the following Query when connected to the database:

```
dd1368=> GRANT ALL PRIVILEGES ON TABLE <insert table name here>
        TO <insert partner username here>;
```

(Remember the semicolon (;) on the end!)

That is it! You have now established a connection to the databases located on KTH's servers! When you want to stop the connection you simply write:

```
mydb=> \q                                # Exit psql with a psql-command

mydb=> Ctrl+D                             # or with the shortcut
```

Then you can exit the ssh-connection by writing:

```
$ exit                                    # Exit the ssh-connection

$ Ctrl+D                                  # or with the shortcut
```

(Otherwise, you can always just close the terminal window)

POSTGRESQL INSTALLATION GUIDE

Moving files local → KTH-remote

Sometimes you will have to populate your database with some sample data. For instance a [mondial-sample dataset](#) or maybe just your own **cool_dataset.sql**. However, how do you populate the KTH-remote database servers with your dataset?

One of the easiest ways of moving files is with [scp](#), you can also find out more about this in the [KTH-scp](#) guide. This command makes secure file copies between computers. The syntax might be a bit confusing, so we will cover the details here.

Starting off, say that you have a file called **cool_dataset.sql** in your **Downloads/** directory:

```
PS C:\Users\... \Downloads> ls

Directory: C:\Users\... \Downloads

Mode                LastWriteTime         Length Name
----                -
-a-----         24/08/2020   14:17           11 cool_dataset.sql
```

Then say that you have some directory on the KTH-remote called **~/Documents/datasets**:

```
@student-shell-1:~/Documents$ ls -lah

total 12K
drwx-----  5 ... default  2.0K Aug 24 14:11 .
drwx----- 39 ... default  4.0K Aug 24 11:56 ..
drwxr-xr-x  2 ... kth-student 2.0K Aug 24 14:11 datasets
drwxr-xr-x  2 ... kth-student 2.0K May 14 2018 MATLAB
drwx-----  5 ... kth-student 2.0K May 14 2019 progp-master
```

In order to move the **cool_dataset.sql** to the above KTH-remote directory, we issue the following command:

```
$ scp ~/Downloads/cool_dataset.sql
<kth-id>@student-shell-1.sys.kth.se:~/Documents/datasets
```



POSTGRESQL INSTALLATION GUIDE

A couple of things to note here:

- “~” indicates your home directory. You could just, and might actually need to, expand this reference explicitly. So what you can do is change “~” with the result of calling:

```
$ cd ~/ && pwd                # for Unix
$ cd ~/; pwd                  # for Windows
```

- Change `**kth-id**` to your personal kth-id.
- You will be prompted to write your KTH password.

If successful you will see a progress bar and status of the upload/copy:

```
PS C:\Users\ > scp 'C:\Users\ \Downloads\cool_dataset.sql'
i@student-shell-1.sys.kth.se:~/Documents/datasets/
i@student-shell-1.sys.kth.se's password:
cool_dataset.sql                                100%  11
2.2KB/s  00:00
```

That is it! Then you can `ssh` or Putty into the KTH-remote servers and find the files at the destination:

```
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 4.15.0-112-generic x86_64)
Last login: Mon Aug 24 14:30:58 2020 from 85.230.83.208
@student-shell-1:~$ ls -lah Documents/datasets/
total 5.0K
drwxr-xr-x 2          kth-student 2.0K Aug 24 14:32 .
drwx----- 5          default    2.0K Aug 24 14:11 ..
-rw-r--r-- 1          kth-student  11 Aug 24 14:32 cool_dataset.sql
@student-shell-1:~$
```



POSTGRESQL INSTALLATION GUIDE

Populating the KTH-remote database with sample datasets

Now that you have moved your `cool_dataset.sql` to the KTH-remote servers, we can start populating the database! First thing, locate the file that you want to populate, in our case it is the following file, `~/Documents/datasets/cool_dataset.sql`, with the content:

```
CREATE TABLE account(  
    accountNumber INT NOT NULL PRIMARY KEY,  
    name VARCHAR(255) NOT NULL UNIQUE,  
    password VARCHAR(255) NOT NULL,  
    address VARCHAR(255) NOT NULL,  
    joined DATE NOT NULL,  
    expires DATE NOT NULL,  
    paid DATE,  
    bill INT NOT NULL  
);
```

The most straight-forward approach is to pass the flag “`-f <filename>`” to the `psql` command like so:

```
$ psql -h nestor2.csc.kth.se -f <filename>
```

This will read the passed file as commands to be invoked on the database, which means that the file needs to be syntactically correct. In my case invoke the following:

```
@student-shell-1:~$ psql -h nestor2.csc.kth.se -f ~/Documents/datasets/cool_dataset.sql  
Password:  
CREATE TABLE
```

(Where the password is the contents of **Private/.psqlpw-nestor2.csc.kth.se**)

We can check that the table actually got created by passing `\dt` when logged in to the db:

```
psql (10.12 (Ubuntu 10.12-0ubuntu0.18.04.1), server 9.5.21)  
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256, compression: off)  
Type "help" for help.  
  
=> \dt  
List of relations  
Schema | Name   | Type  | Owner  
-----+-----+-----+-----  
public | account | table |  
(1 row)
```




POSTGRESQL INSTALLATION GUIDE

Another way of populating the data is to log into the database and then call `\i <filename>`. It is the same idea as previously (you provide a syntactically correct .sql-file), but now you are already logged in to the database.



POSTGRESQL INSTALLATION GUIDE

Local installation

The following instructions cover a local installation of a psql-server. Please note that there could be issues getting things running or requiring some installation-debugging. **Therefore we recommend that you at least make sure that the “KTH-remote access” above works, so that you have a backup option. Using a local installation will not influence the grading of your laborations. Additionally, lab assistants will only be able to provide limited support on issues with a local installation and prioritize helping students with query related issues.** Because, the most important thing is that you complete the LABS, not which installation you have used!

If you want to use a local Database instance, you can install and configure PostgreSQL for yourselves. There are many different methods for installing PostgreSQL on the internet, but in an effort to navigate the installation jungle we will go through one installation method for OSX and Windows.

Hot tip: If you are keen on keeping your computer fresh and clean, you can use a virtual machine for the installation. However, keep in mind that using a virtual machine comes with a range of installation and configuration that might take up valuable LAB-time. **So please ensure that the KTH servers work so that you have a backup option!**

Using Brew

Homebrew is an open-source software package management system.

Install [Homebrew](#). Please follow the instructions of the Homebrew installation on that webpage. Then make sure that you have brew installed:

```
$ brew --version
```

POSTGRESQL INSTALLATION GUIDE

```
simon@simon-VirtualBox: ~  
simon@simon-VirtualBox:~$ brew --version  
Homebrew 2.3.0  
Homebrew/linuxbrew-core (git revision a4e505; last commit 2020-06-01)  
simon@simon-VirtualBox:~$
```

(Installation-Gotcha: If you are having troubles like “brew: command not found”, it is probably related to your PATH environment-variables.)

Install PostgreSQL

```
$ brew install postgresql
```

Make sure that you have PostgreSQL installed

```
$ psql --version
```

POSTGRESQL INSTALLATION GUIDE

```
simon@simon-VirtualBox: ~  
export PKG_CONFIG_PATH="/home/linuxbrew/.linuxbrew/opt/python@3.8/lib/pkgconfi  
g"  
  
==> libxslt  
To allow the nokogiri gem to link against this libxslt run:  
gem install nokogiri -- --with-xslt-dir=/home/linuxbrew/.linuxbrew/opt/libxslt  
==> perl  
By default non-brewed cpan modules are installed to the Cellar. If you wish  
for your modules to persist across updates we recommend using `local::lib`.  
  
You can set that up like this:  
PERL_MM_OPT="INSTALL_BASE=$HOME/perl5" cpan local::lib  
echo 'eval "$(perl -I$HOME/perl5/lib/perl5 -Mlocal::lib=$HOME/perl5)"' >> /hom  
e/simon/.bash_profile  
==> postgresql  
To migrate existing data from a previous major version of PostgreSQL run:  
brew postgresql-upgrade-database  
  
Warning: postgresql provides a launchd plist which can only be used on macOS!  
You can manually execute the service instead with:  
pg_ctl -D /home/linuxbrew/.linuxbrew/var/postgres start  
simon@simon-VirtualBox:~$ psql --version  
psql (PostgreSQL) 12.3  
simon@simon-VirtualBox:~$
```

Starting PostgreSQL

There are two options for starting a PostgreSQL instance with Brew. The first option will start a Daemon (a service that runs in the background) automatically every time you boot the computer. The second option will manually start a Daemon, that will terminate when you shutdown the computer.

Automatic start

```
$ brew services start postgresql
```

Automatic stop

```
$ brew services stop postgresql
```

Manual start (assuming default config)

```
$ pg_ctl -D /usr/local/var/postgres start
```

Manual stop

```
$ pg_ctl -D /usr/local/var/postgres stop
```

Now you need to make sure that you can access the PostgreSQL CLI.

POSTGRESQL INSTALLATION GUIDE

If everything was successful your terminal environment should have changed to be in psql. You can test it by writing:

```
db1368#=> \?
```

You are now up and running!

Using Windows

The installation for Windows will be assuming that you are running on a Windows 10 64-bit system.¹

Installing the installer

Begin by visiting the Postgres website, we go to the downloads page and select the correct system (Windows): <https://www.postgresql.org/download/windows/>. Press the link “Download the Installer”, you will be linked to another site. At this site you should be seeing the following:



PostgreSQL Database Download

Version	Linux x86-64	Linux x86-32	Mac OS X	Windows x86-64	Windows x86-32
12.3	N/A	N/A	Download	Download	N/A
11.8	N/A	N/A	Download	Download	N/A
10.13	Download	Download	Download	Download	Download
9.6.18	Download	Download	Download	Download	Download
9.5.22	Download	Download	Download	Download	Download
9.4.26 (Not Supported)	Download	Download	Download	Download	Download
9.3.25 (Not Supported)	Download	Download	Download	Download	Download

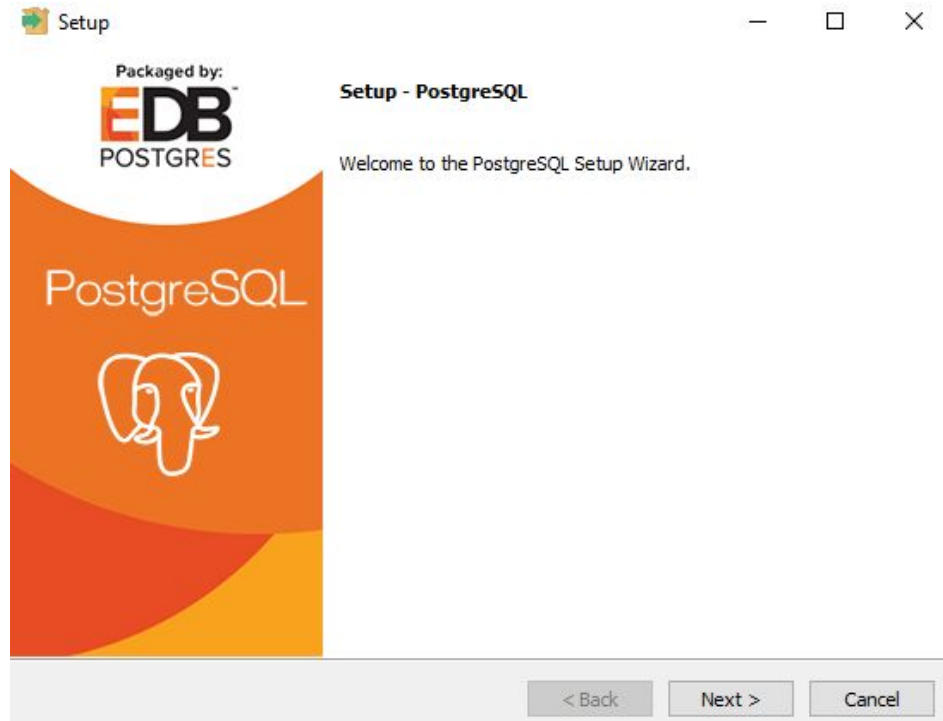
We press the button on row “12.3”, and column “Windows x86-64”. This will download an installer.

¹ If this is not the case, there might be a need to choose other binaries when downloading and installing.

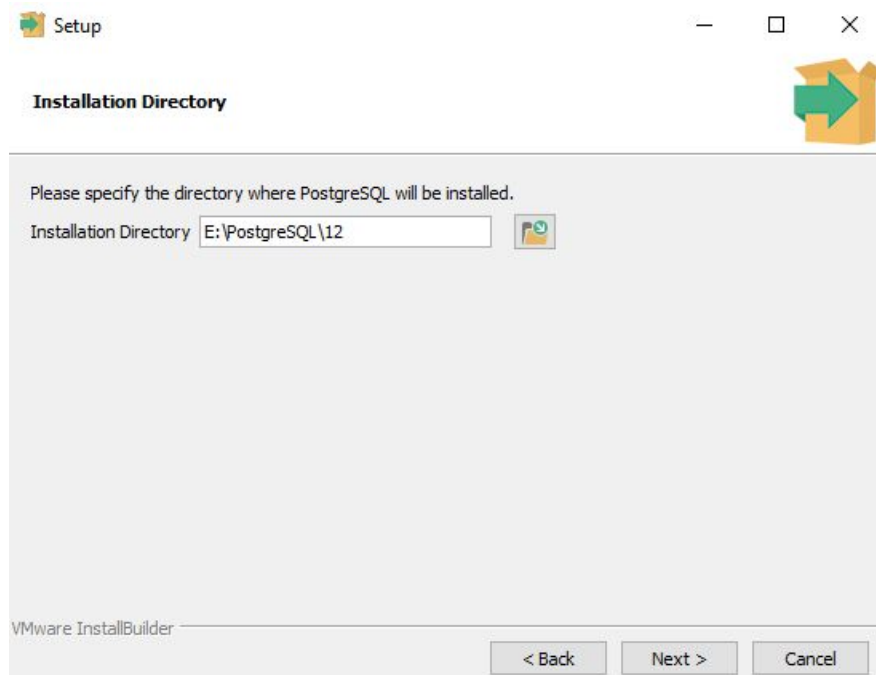
POSTGRESQL INSTALLATION GUIDE

Starting the installation

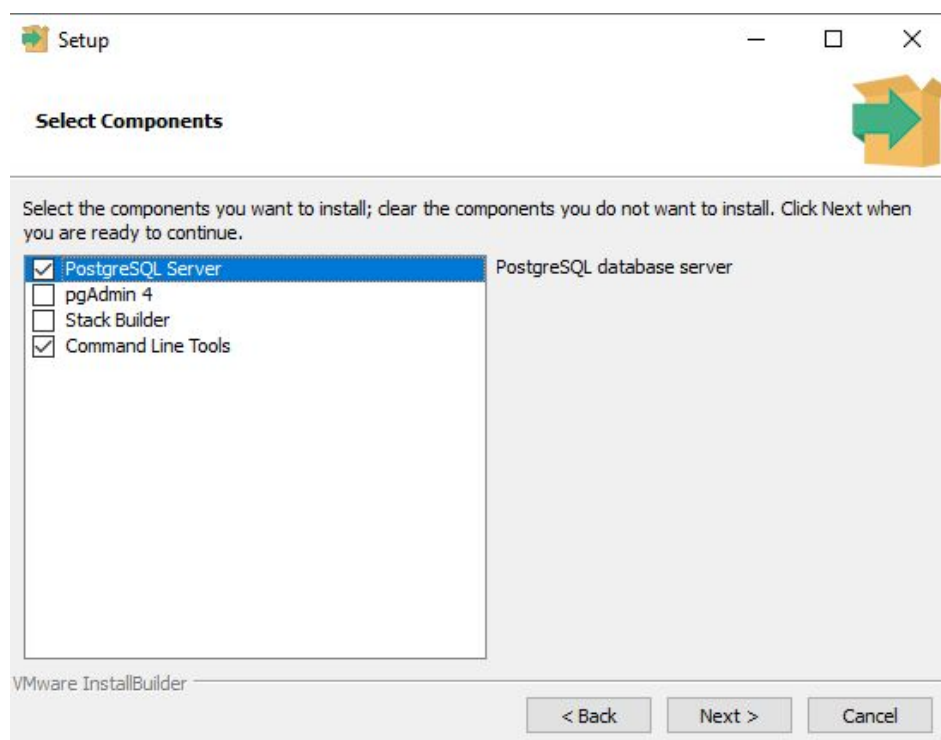
The installer is rather linear and explains what will be installed to your system.



POSTGRESQL INSTALLATION GUIDE

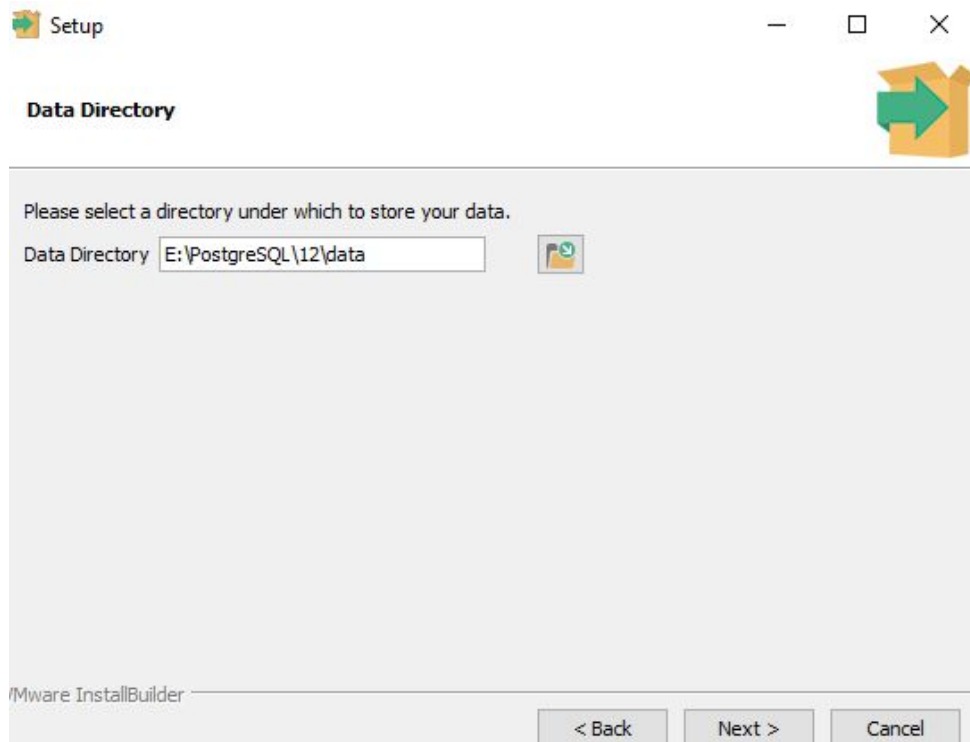


Feel free to change the directory where the source files will be installed to. You can leave the location if you feel like it, but try to remember where it will be installed (it will be important later!). For simplicity I will call this directory: **\$PSQL_HOME=E:\PostgreSQL\12**

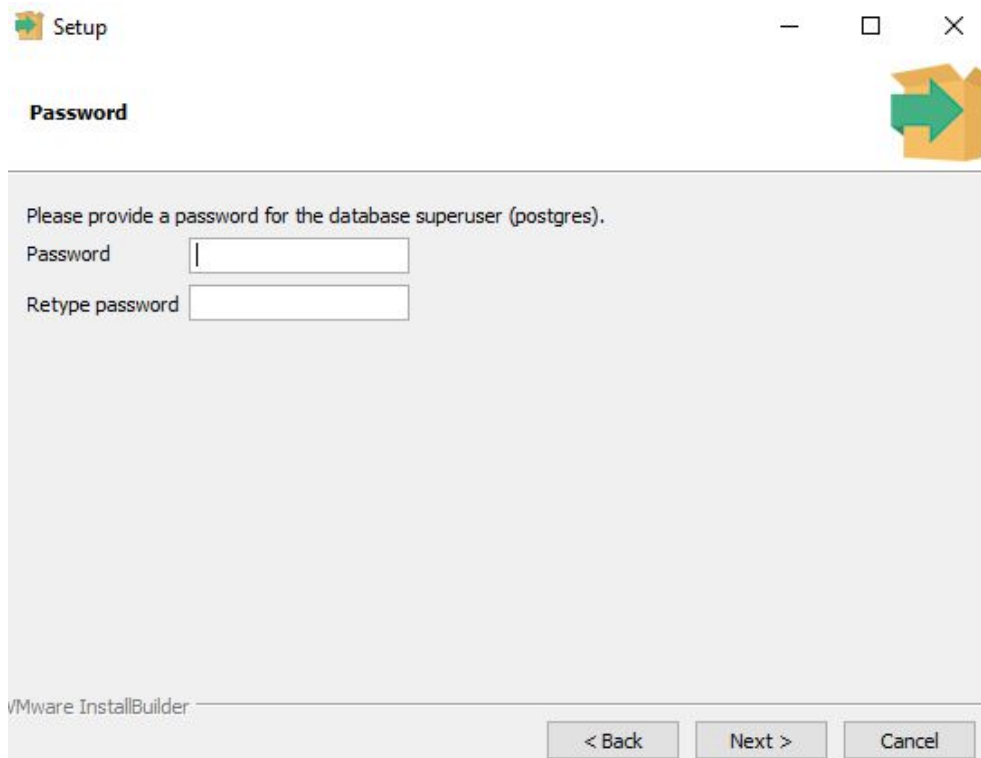


I have chosen to skip some of the components because they are not necessary for the Labs in this course. The two components that I have chosen **need to be selected**.

POSTGRESQL INSTALLATION GUIDE

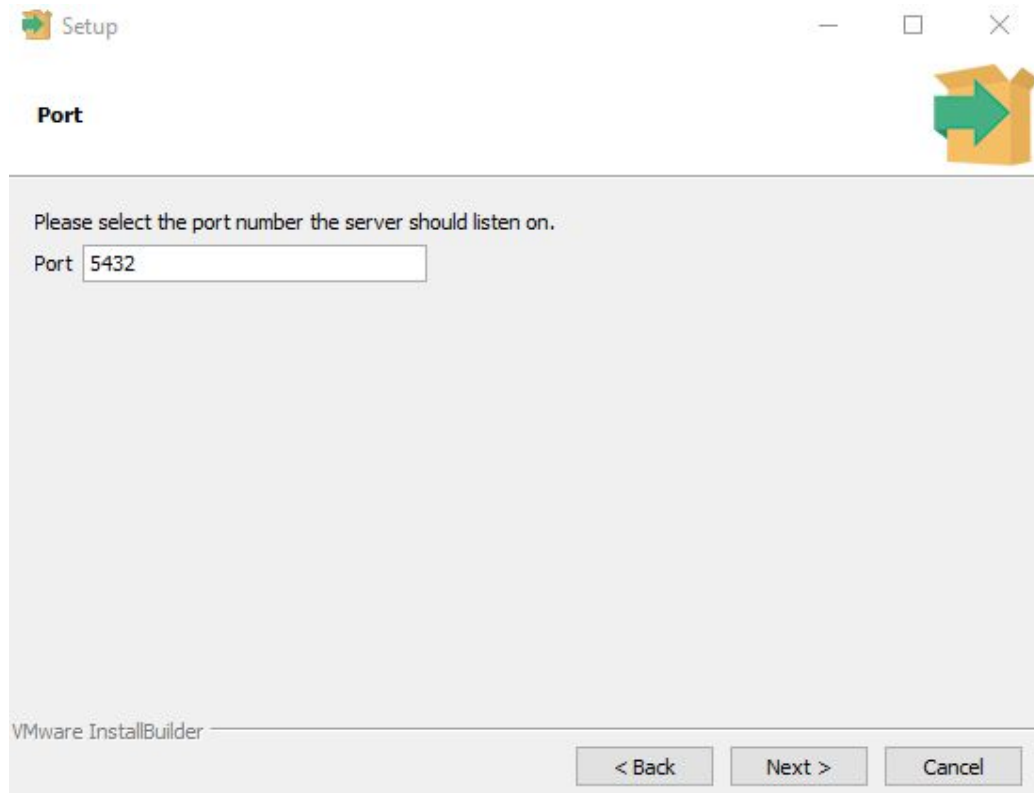


I leave this to be the defaulted directory (it should be a subdirectory of the \$PSQL_HOME).

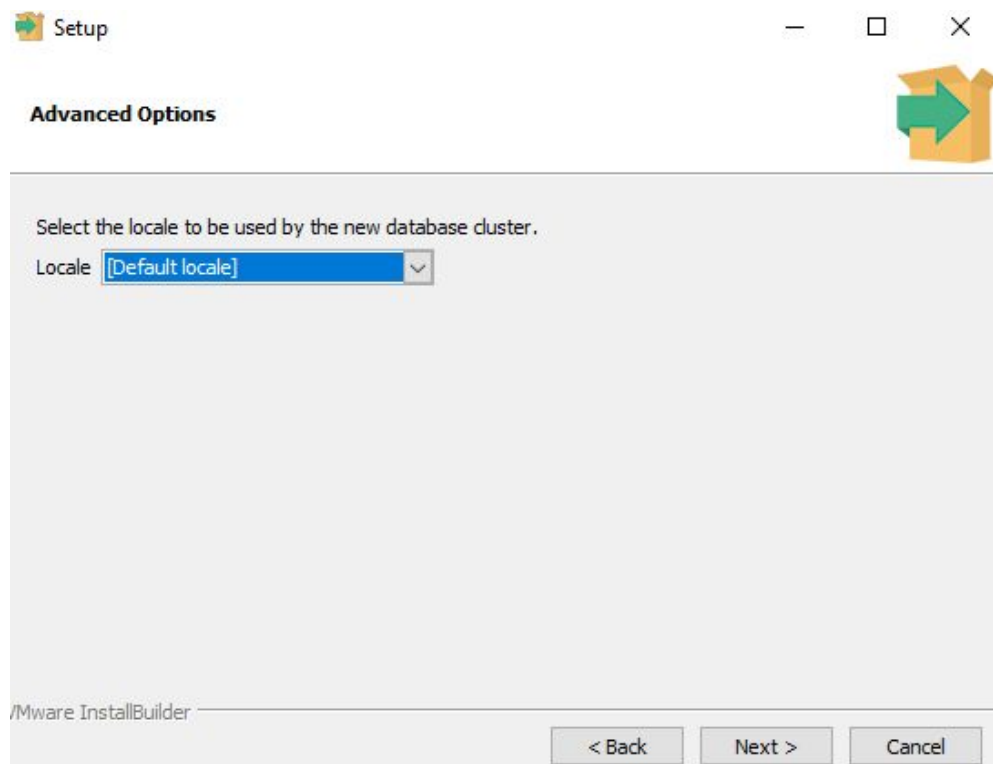


This next step is to choose a password for the root (admin) user of the whole database system.
This can be whatever you want but make sure that you do not forget this password!

POSTGRESQL INSTALLATION GUIDE

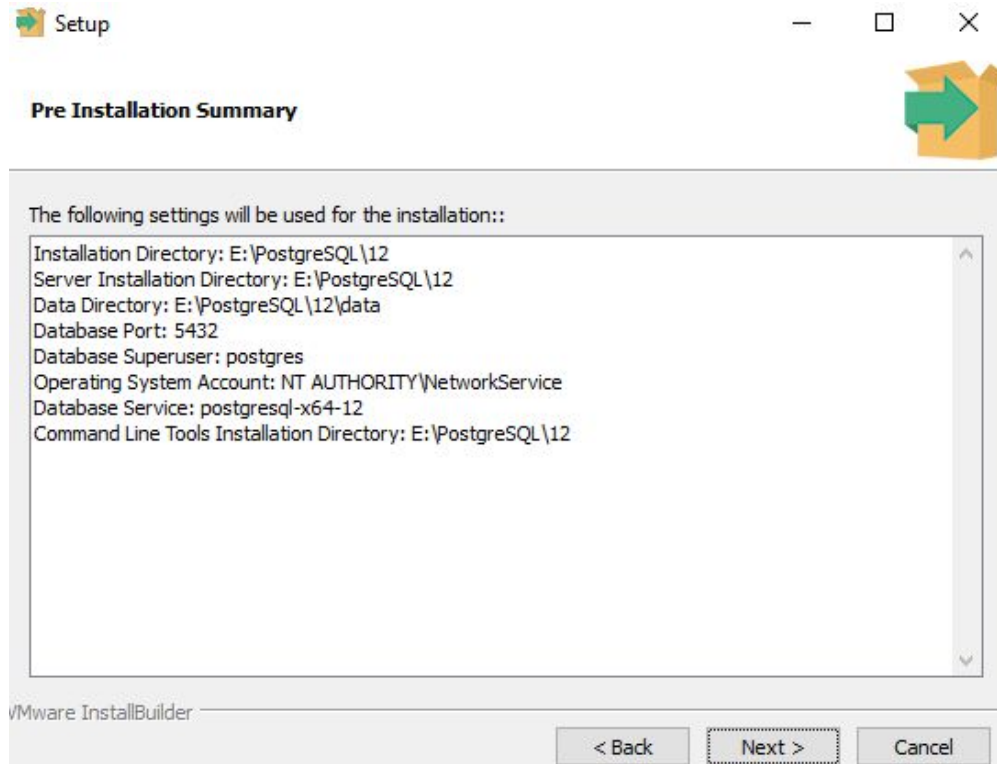


Next you are to choose the port of the psql server. The default port will do just fine!



This is the locale settings for your database. The default locale will do just fine!

POSTGRESQL INSTALLATION GUIDE

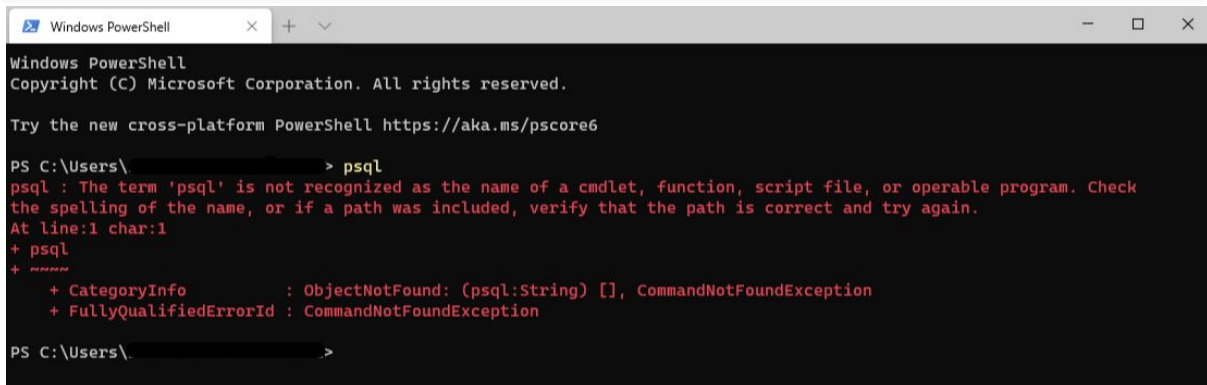


The last step gives you an overview of your installation, then press next and the installation will start!

Post installation settings

After the installation has finished there are still some things we need to do. First of all check if the psql command has been added to the PATH. This can be done e.g. by opening Powershell and typing:

```
> psql
```

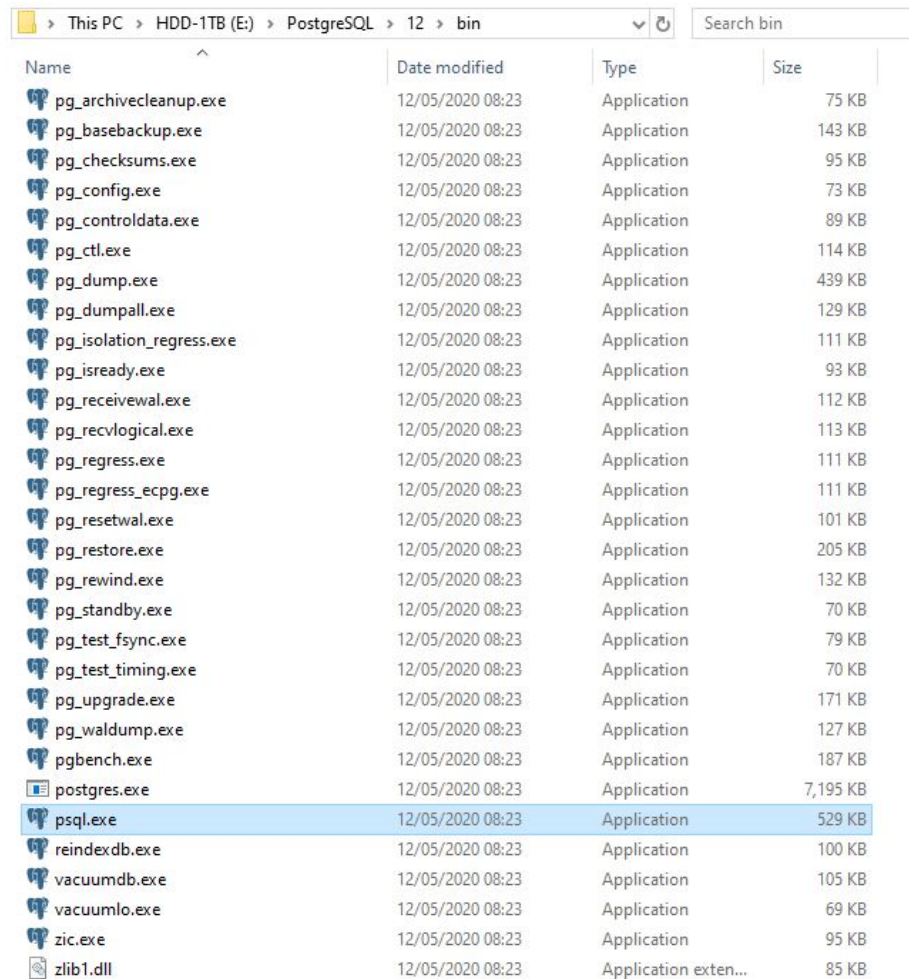


If the command has not been added to the PATH, the following error (in red) should appear. If the command has been added to the PATH, you can skip the following few steps.

POSTGRESQL INSTALLATION GUIDE

Adding psql to the PATH

In order to add the command to the PATH we need to find where the command is. The command should be located in **\$PSQL_HOME\bin** (in my case it will be **E:\PostgreSQL\12\bin**), make sure that there is an executable called **psql.exe**:

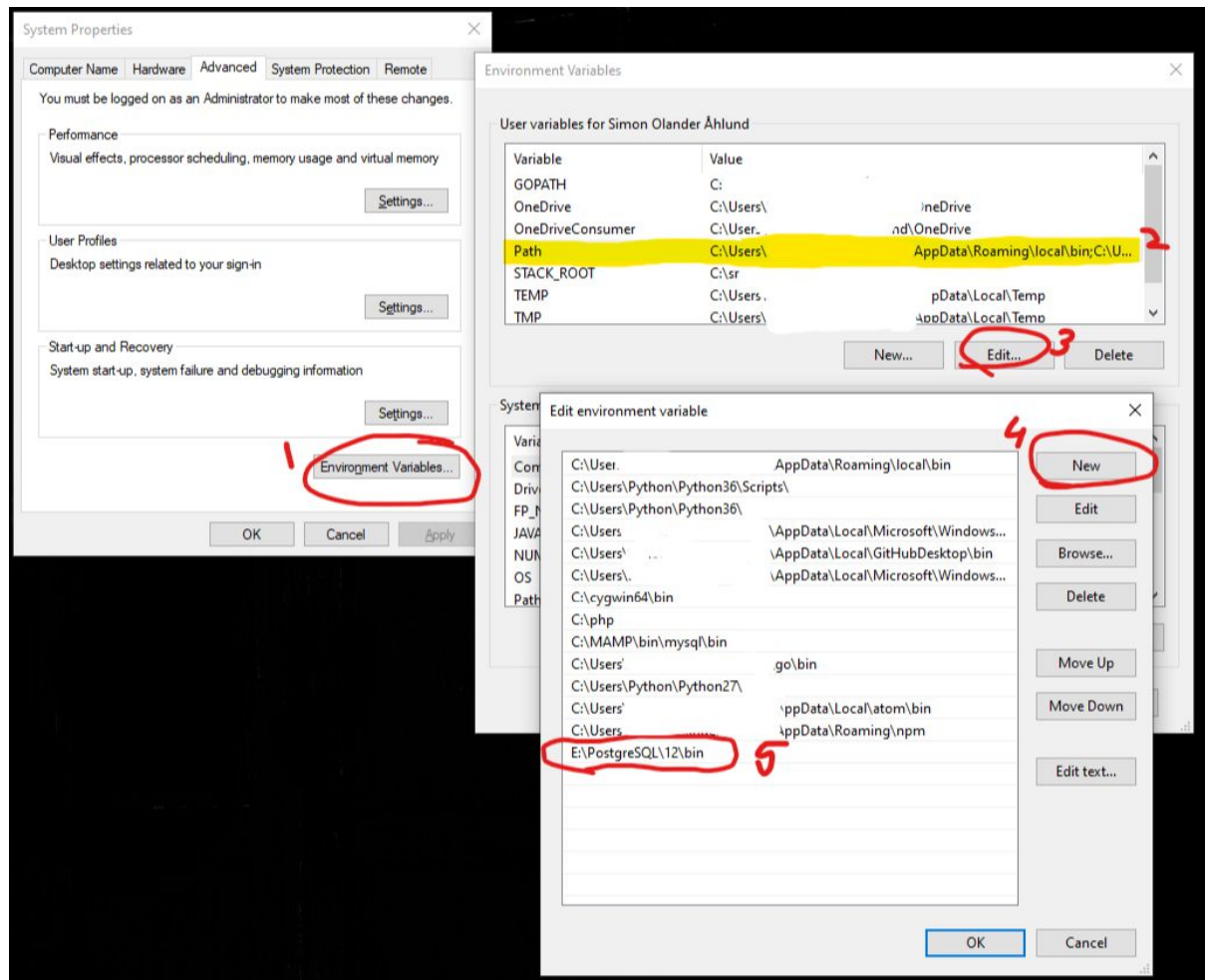


Name	Date modified	Type	Size
pg_archivecleanup.exe	12/05/2020 08:23	Application	75 KB
pg_basebackup.exe	12/05/2020 08:23	Application	143 KB
pg_checksums.exe	12/05/2020 08:23	Application	95 KB
pg_config.exe	12/05/2020 08:23	Application	73 KB
pg_controldata.exe	12/05/2020 08:23	Application	89 KB
pg_ctl.exe	12/05/2020 08:23	Application	114 KB
pg_dump.exe	12/05/2020 08:23	Application	439 KB
pg_dumpall.exe	12/05/2020 08:23	Application	129 KB
pg_isolation_regress.exe	12/05/2020 08:23	Application	111 KB
pg_isready.exe	12/05/2020 08:23	Application	93 KB
pg_receivewal.exe	12/05/2020 08:23	Application	112 KB
pg_recvlogical.exe	12/05/2020 08:23	Application	113 KB
pg_regress.exe	12/05/2020 08:23	Application	111 KB
pg_regress_ecpg.exe	12/05/2020 08:23	Application	111 KB
pg_resetwal.exe	12/05/2020 08:23	Application	101 KB
pg_restore.exe	12/05/2020 08:23	Application	205 KB
pg_rewind.exe	12/05/2020 08:23	Application	132 KB
pg_standby.exe	12/05/2020 08:23	Application	70 KB
pg_test_fsync.exe	12/05/2020 08:23	Application	79 KB
pg_test_timing.exe	12/05/2020 08:23	Application	70 KB
pg_upgrade.exe	12/05/2020 08:23	Application	171 KB
pg_waldump.exe	12/05/2020 08:23	Application	127 KB
pgbench.exe	12/05/2020 08:23	Application	187 KB
postgres.exe	12/05/2020 08:23	Application	7,195 KB
psql.exe	12/05/2020 08:23	Application	529 KB
reindexdb.exe	12/05/2020 08:23	Application	100 KB
vacuumdb.exe	12/05/2020 08:23	Application	105 KB
vacuumlo.exe	12/05/2020 08:23	Application	69 KB
zic.exe	12/05/2020 08:23	Application	95 KB
zlib1.dll	12/05/2020 08:23	Application exten...	85 KB

When we have found this executable, we take note of the directory (in my case it is **\$PSQL_HOME\bin**). With this information we can open the “Environment variable settings” manager in windows²:

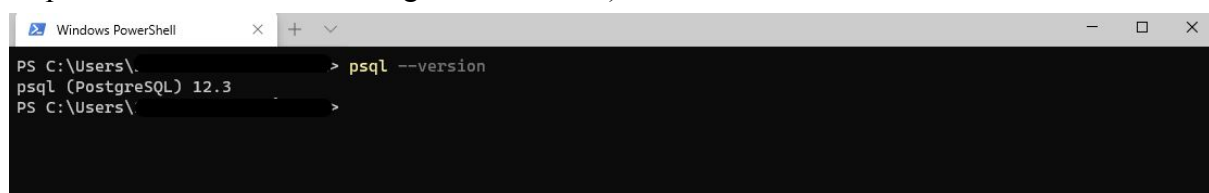
² In swedish it should be something like “Ändra systemets miljövariabler”.

POSTGRESQL INSTALLATION GUIDE



1. Press "Environment Variables..."
2. Choose the "Path" row in the top window.
3. Press "Edit"
4. Press "New"
5. Write the path to the directory where psql.exe were located in (in my case it is **\$PSQL_HOME\bin**).
6. Press ok, ok, ok.

Now we should be able to see the command in e.g. Powershell (you might need to close and reopen Powershell for the change to take effect):



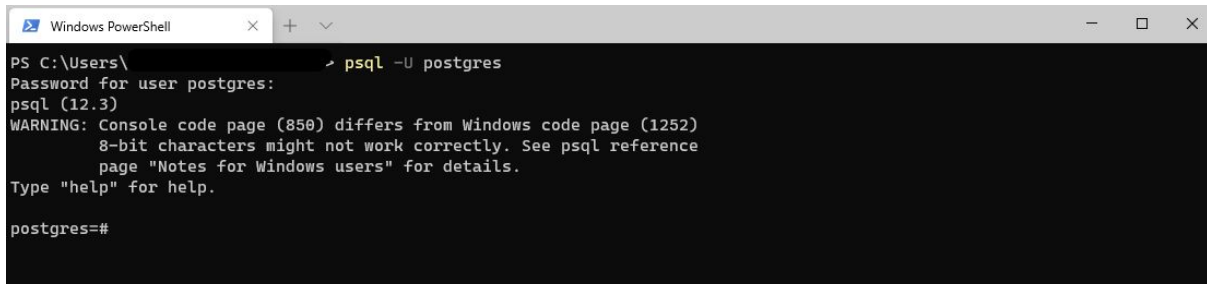
POSTGRESQL INSTALLATION GUIDE

Using the Windows installation

Now you are ready to get started. To log into the database you can write the following:

```
> psql -U postgres
```

Then write the password that you set previously



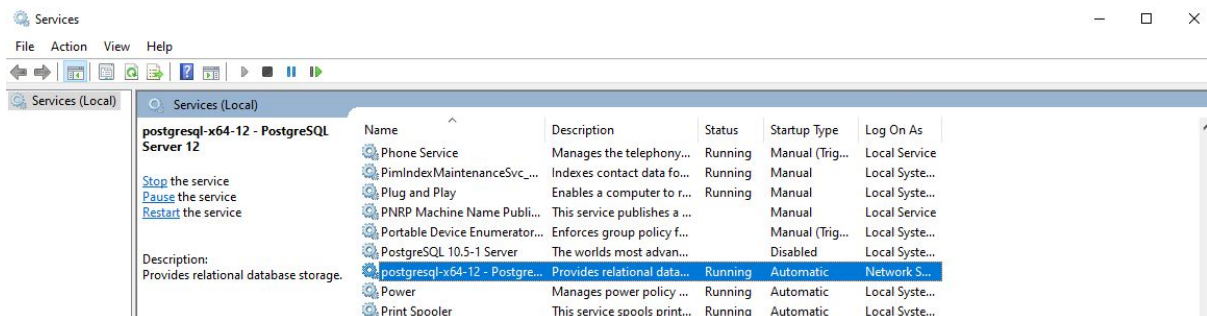
```
Windows PowerShell
PS C:\Users\ > psql -U postgres
Password for user postgres:
psql (12.3)
WARNING: Console code page (850) differs from Windows code page (1252)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
Type "help" for help.

postgres=#
```

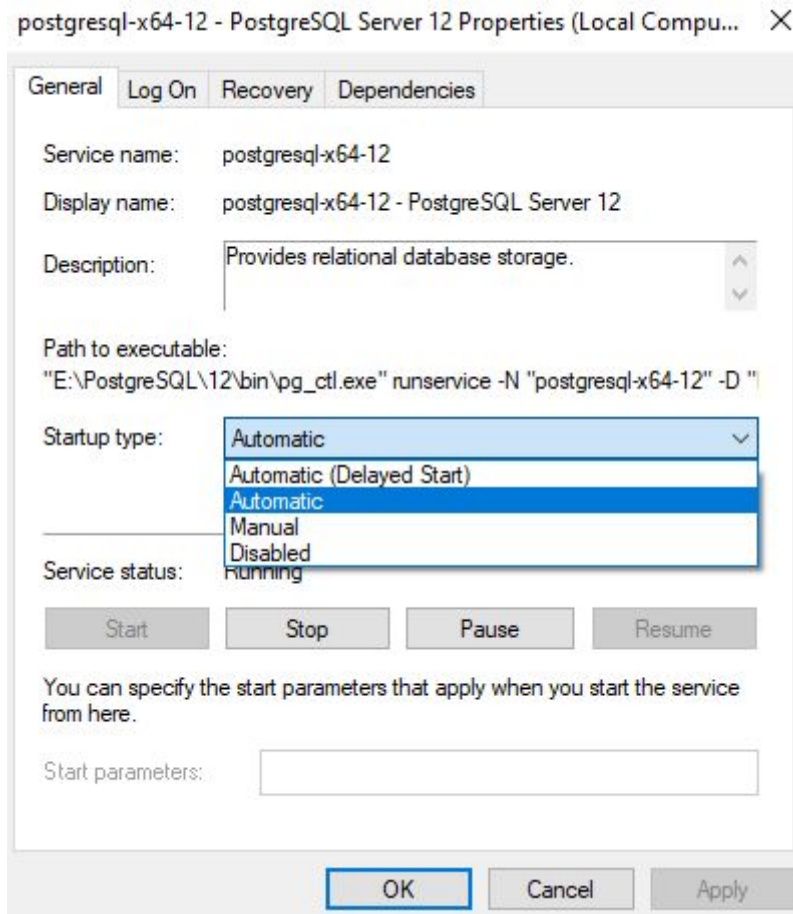
Now you are ready to start SQLing!

Stopping the db server

On a Windows machine the PostgreSQL server will be up and running by default. If you want to control when the server is up and not you can open “Services” in Windows and locate the postgres service:



POSTGRESQL INSTALLATION GUIDE



Here you can choose what startup behaviour the server will have and whether you want to force start/stop the server. If you have disabled or stopped the server the following command should give following error message:

```
> psql
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\...> psql
psql: error: could not connect to server: could not connect to server: Connection refused (0x0000274D/10061)
        Is the server running on host "localhost" (::1) and accepting
        TCP/IP connections on port 5432?
could not connect to server: Connection refused (0x0000274D/10061)
        Is the server running on host "localhost" (127.0.0.1) and accepting
        TCP/IP connections on port 5432?
PS C:\Users\...>
```



POSTGRESQL INSTALLATION GUIDE

USEFUL COMMANDS WHILE EXPLORING THE DATABASE

\? - Shows all commands available

\q - Quit psql.

\dt - Shows all current tables in your database and their owners.

\dp - Shows all current tables in your database and the access privileges.

\timing - Shows query execution time while toggled on.

\x - Show extended view, good for big tables

\i <filename> - Reads and executes the contents of a specific file.