



A framework for user centred privacy and security in the cloud

Dataset “ades_geofla_centroid”

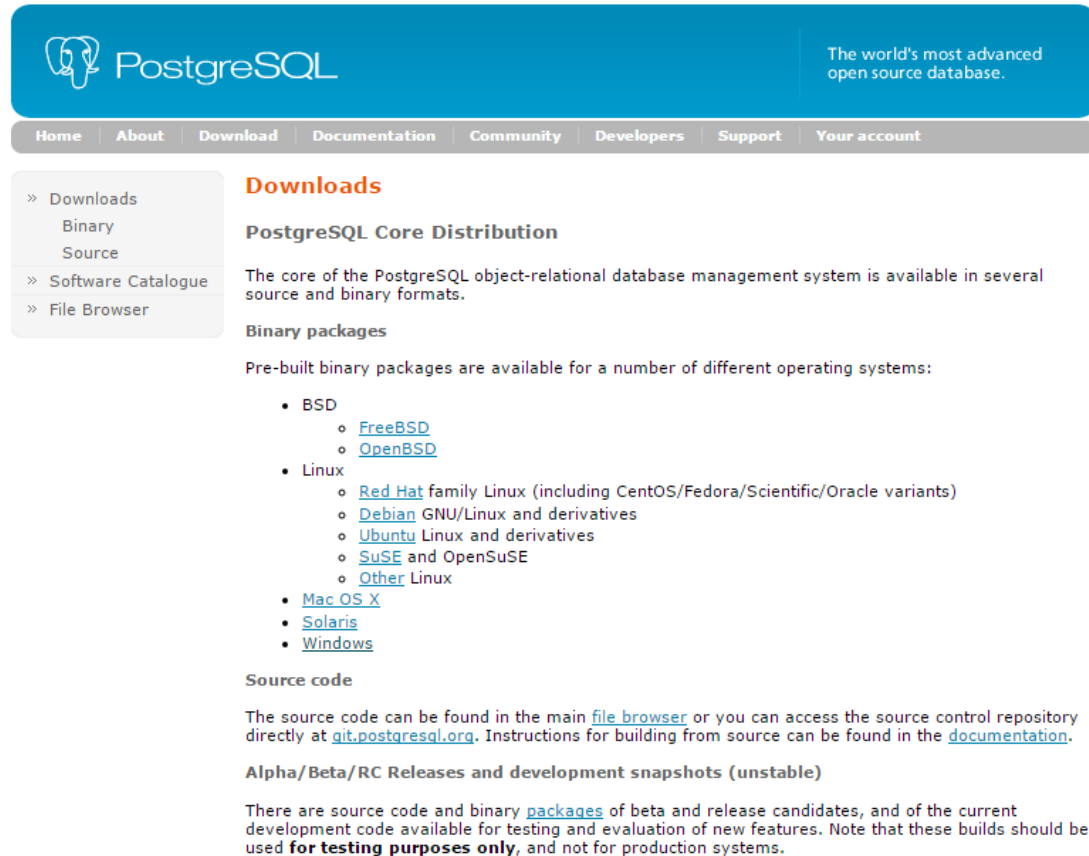
Author(s)	Manel Ruíz, Jesús A. Manjón
Document ID	CLARUS-DatasetAdesGeoflaCentroid-v1.0
Version	1.0
Date of Issue	01/02/2016
Document Distribution	Public
Abstract	<p>In this document, installation of necessary tools for importation and use of dataset “ades_geofla_centroid” file are shown, covering from the installation of PostgreSQL Database Management System to the PostGIS plugin and the Shapefile importing procedure.</p> <p>This dataset corresponds to the descriptions and (fake) coordinates of qualimeters in the Provence-Alpes-Cote-d'Azur region. The true coordinates have been modified with the centroid of the city wherein each qualimeter is located.</p> <p>Coordinate Reference system is EPSG:4326</p>

Table of Contents

1	POSTGRESQL DBMS INSTALLATION	3
2	POSTGIS INSTALLATION	6
3	CREATION OF POSTGIS ENABLED DATABASES.....	10
4	IMPORTING A SHAPEFILE TO DATABASE.....	12

1 PostgreSQL DBMS installation

We will download the right PostgreSQL installer for the OS from the official site (<http://www.postgresql.org/download/>).



The screenshot shows the PostgreSQL website's 'Downloads' page. The header features the PostgreSQL logo and the tagline 'The world's most advanced open source database.' Below the header is a navigation bar with links: Home, About, Download, Documentation, Community, Developers, Support, and Your account. The main content area is titled 'Downloads' and 'PostgreSQL Core Distribution'. It states that the core of the PostgreSQL object-relational database management system is available in several source and binary formats. Under 'Binary packages', it lists pre-built binary packages for various operating systems: BSD (FreeBSD, OpenBSD), Linux (Red Hat family Linux, Debian GNU/Linux, Ubuntu Linux, SuSE and OpenSUSE, Other Linux), Mac OS X, Solaris, and Windows. There is also a section for 'Source code' and 'Alpha/Beta/RC Releases and development snapshots (unstable)'.

PostgreSQL

The world's most advanced open source database.

Home About Download Documentation Community Developers Support Your account

» Downloads
Binary
Source
» Software Catalogue
» File Browser

Downloads

PostgreSQL Core Distribution

The core of the PostgreSQL object-relational database management system is available in several source and binary formats.

Binary packages

Pre-built binary packages are available for a number of different operating systems:

- BSD
 - [FreeBSD](#)
 - [OpenBSD](#)
- Linux
 - [Red Hat](#) family Linux (including CentOS/Fedora/Scientific/Oracle variants)
 - [Debian](#) GNU/Linux and derivatives
 - [Ubuntu](#) Linux and derivatives
 - [SuSE](#) and [OpenSUSE](#)
 - [Other](#) Linux
- [Mac OS X](#)
- [Solaris](#)
- [Windows](#)

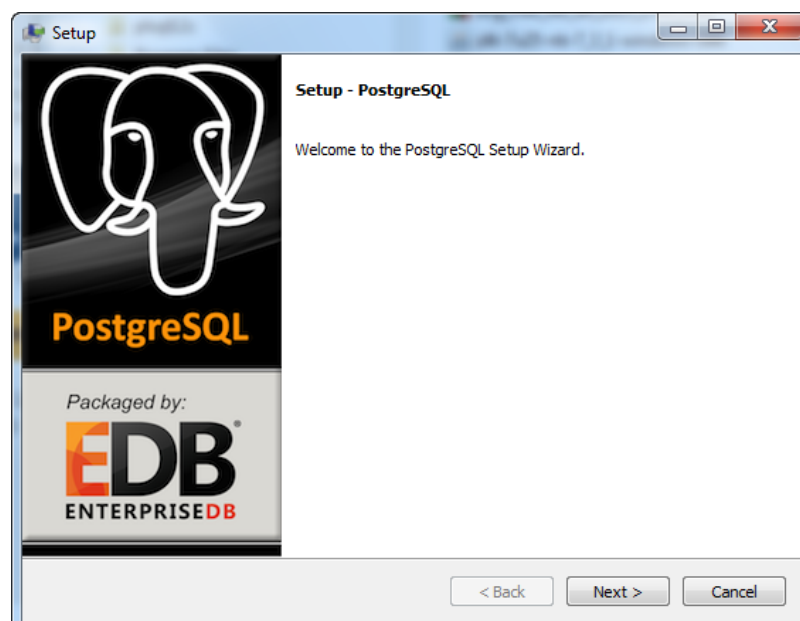
Source code

The source code can be found in the main [file browser](#) or you can access the source control repository directly at git.postgresql.org. Instructions for building from source can be found in the [documentation](#).

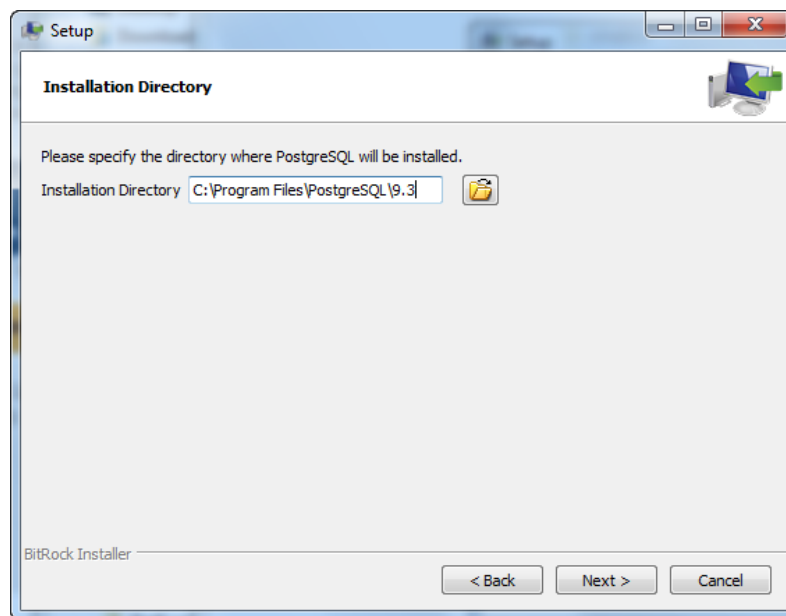
Alpha/Beta/RC Releases and development snapshots (unstable)

There are source code and binary [packages](#) of beta and release candidates, and of the current development code available for testing and evaluation of new features. Note that these builds should be used **for testing purposes only**, and not for production systems.

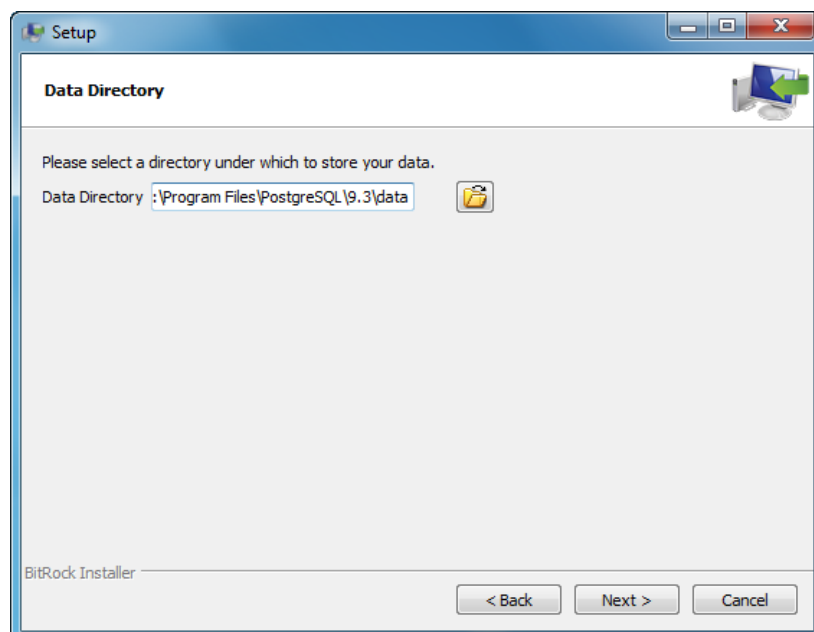
We will next install and configure the software.



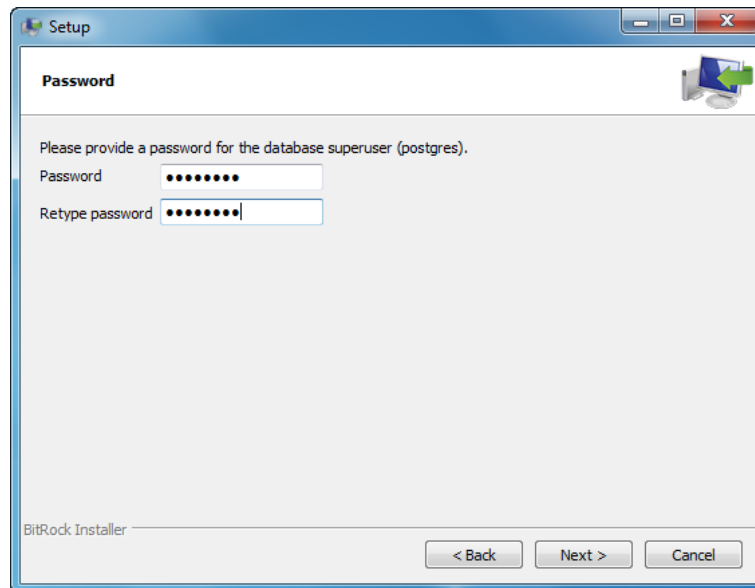
We will then choose a directory for PostgreSQL installation,



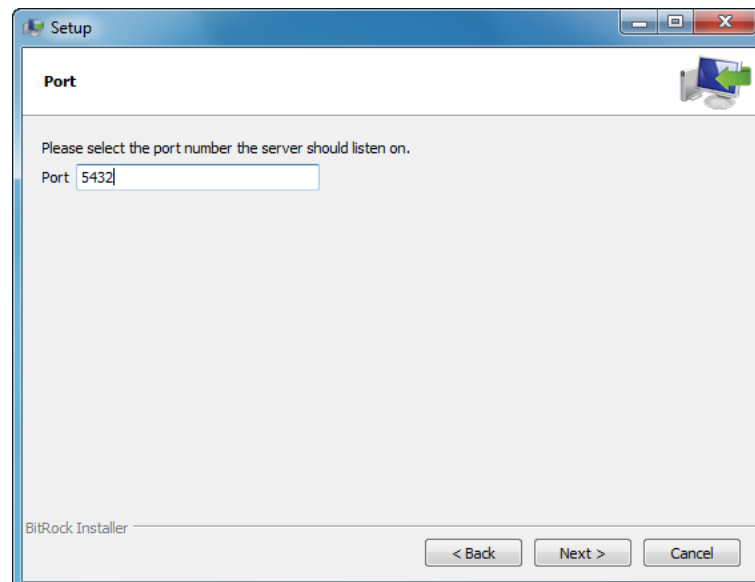
and a directory that will store all the data (including program configurations, database files and configurations, etc.).



The installer will now ask for a password. This will be the password for *postgres* user (root)



We can also specify the port number in which the server will listen. We can use the default port if it's not being used by any other service on the computer we are carrying out the installation.

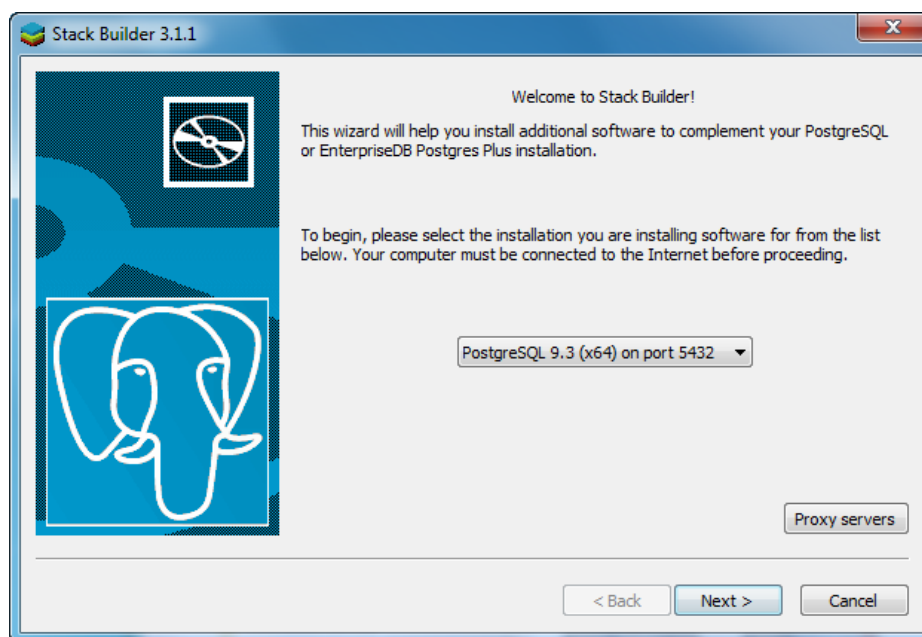


Once the installation is completed, we will be asked to launch a software named “Stack Builder”. This software presents a GUI which allows to install plugins, drivers and new features for our PostgreSQL installation without ease.

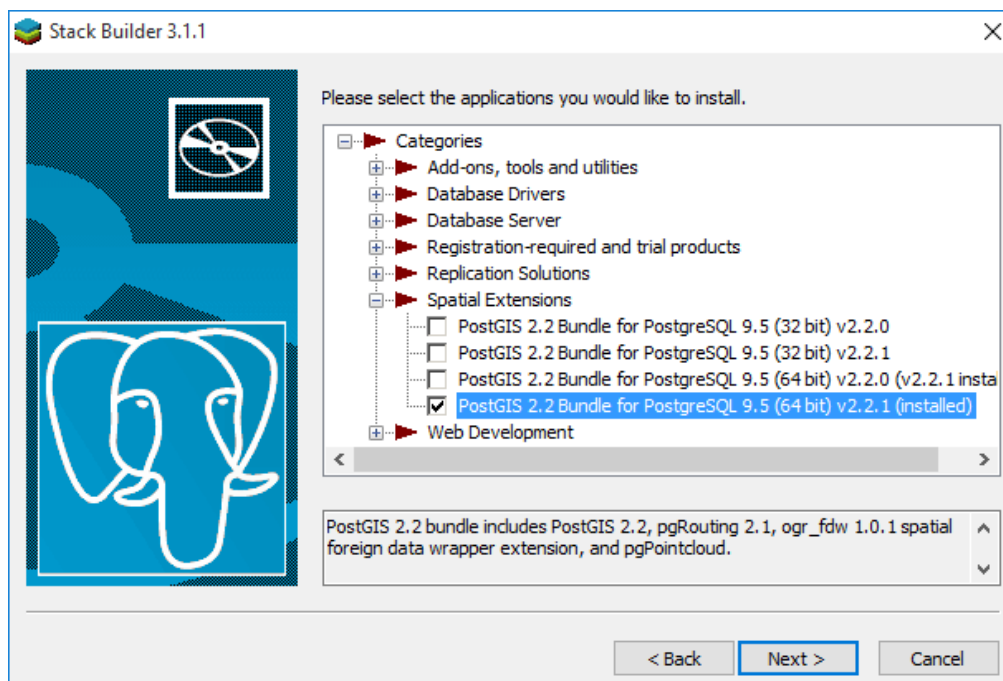
2 PostGIS installation

In our case, we will use Stack Builder, but PostGIS can also be manually downloaded and installed from the official website (<http://postgis.net/>), which has all the necessary files.

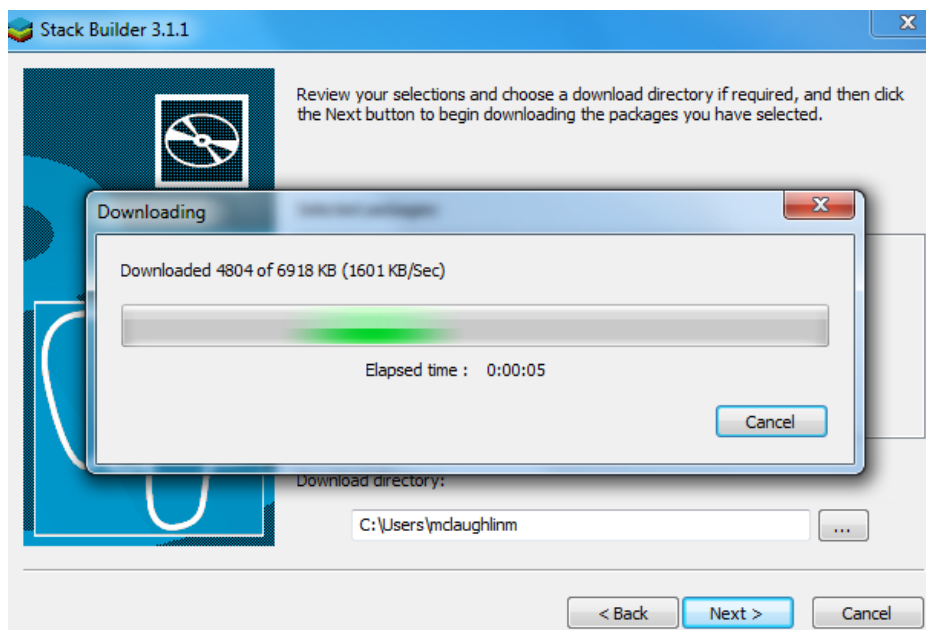
When Stack Builder is executed, it will look for local valid PostgreSQL installations to install on it. It's worth noting that Stack Builder can also install on remote servers.



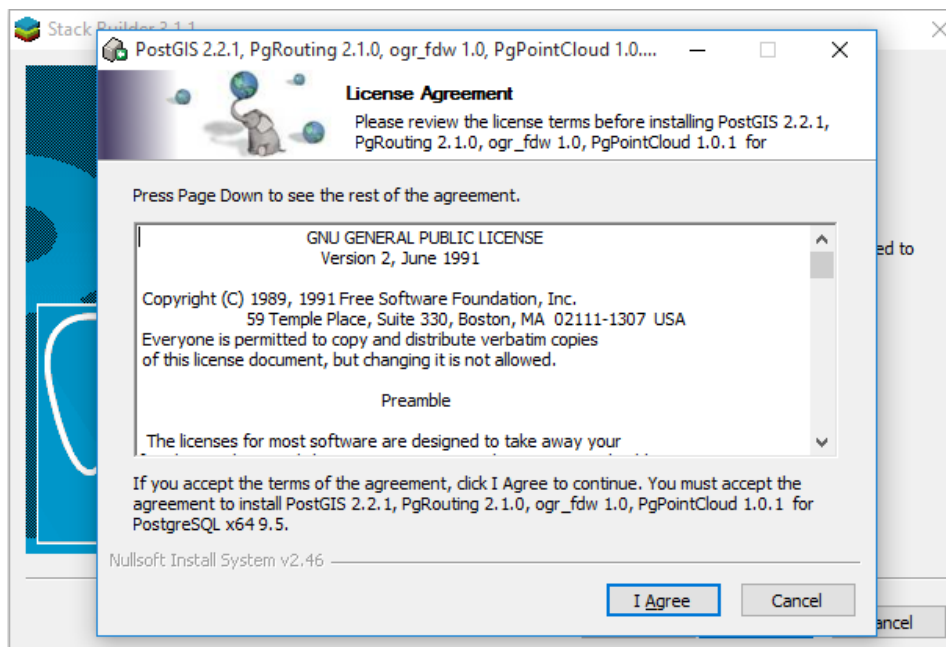
In this screen, extensions can be selected in order to install them. We can find PostGIS inside the “Spatial Extensions” category. We will choose the right version for the server (in our case, PostGIS 2.2.1, 64bit).

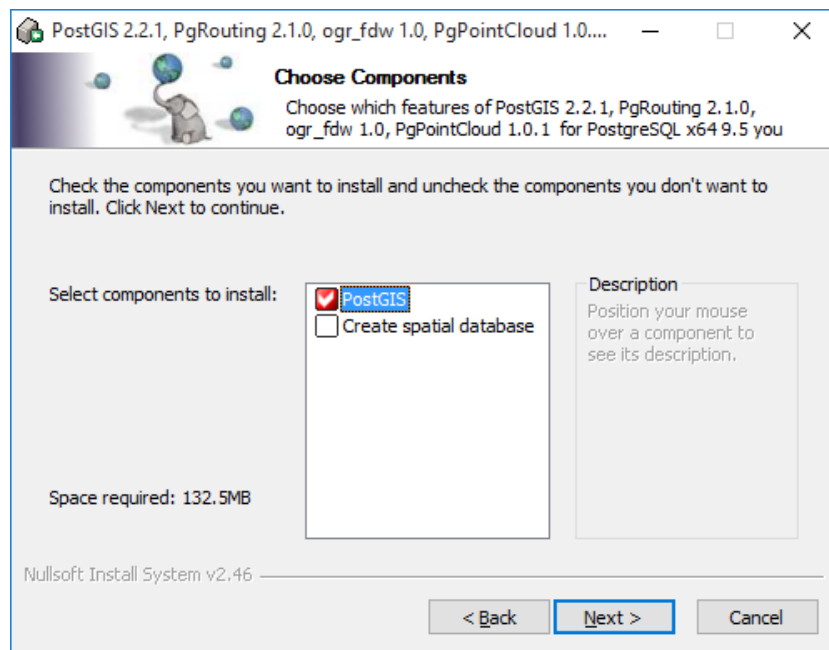


Once Next button is pressed, Stack Builder will start to download all checked extensions.

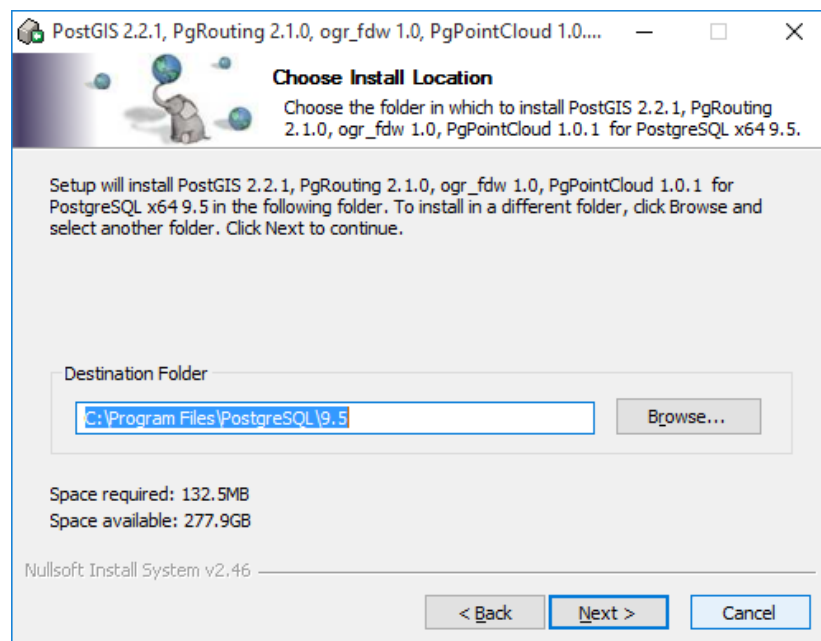


After the required downloads end, Stack Builder will launch the selected installers. From now on, the steps described in this section will be exactly the same as though we downloaded the official installer from PostGIS website.



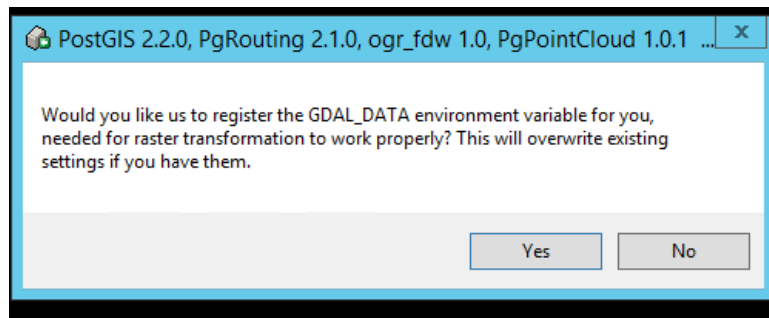


In addition to installing PostGIS, an option for creating a spatial database appears. After that, we will be asked for the PostgreSQL installation directory, and the installation will start right away.

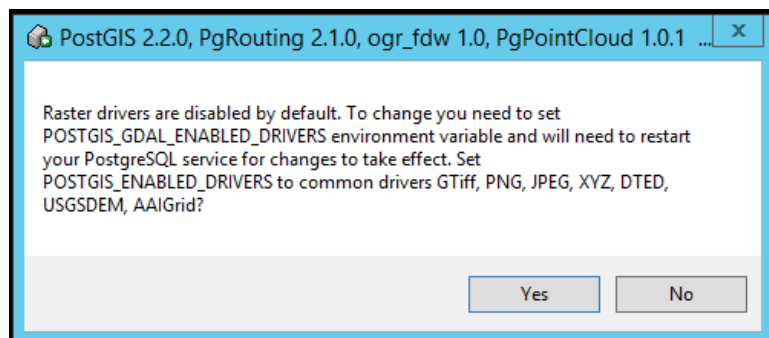


When the installation is over, we will be presented with 3 dialog boxes which enable some configurations of PostGIS plugin.

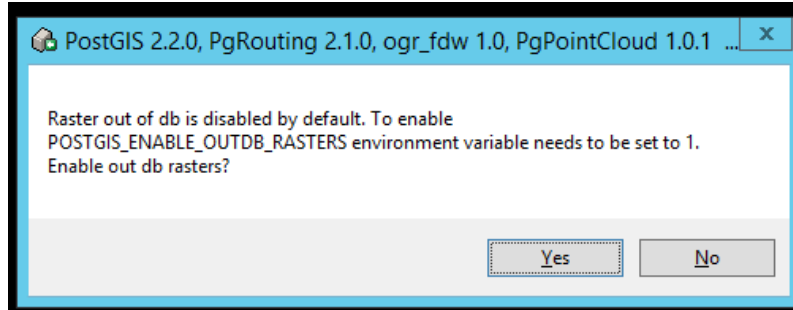
The first one will ask to register an environment variable (GDAL_DATA) for us. This variable is needed for converting the vector information into images. It is recommended to let it register the variable in the first install.



The next dialog can enable the raster drivers. These drivers are, according to the documentation, safe (meaning that they don't access external web services for rastering the images, and refers to functions such as `st_aspng()`)



The last dialog will enable the out of database rastering. We will activate it if needed.



These functionalities (and some additional drivers not shown in the second dialog) can be enabled at any time if needed.

3 Creation of PostGIS enabled databases

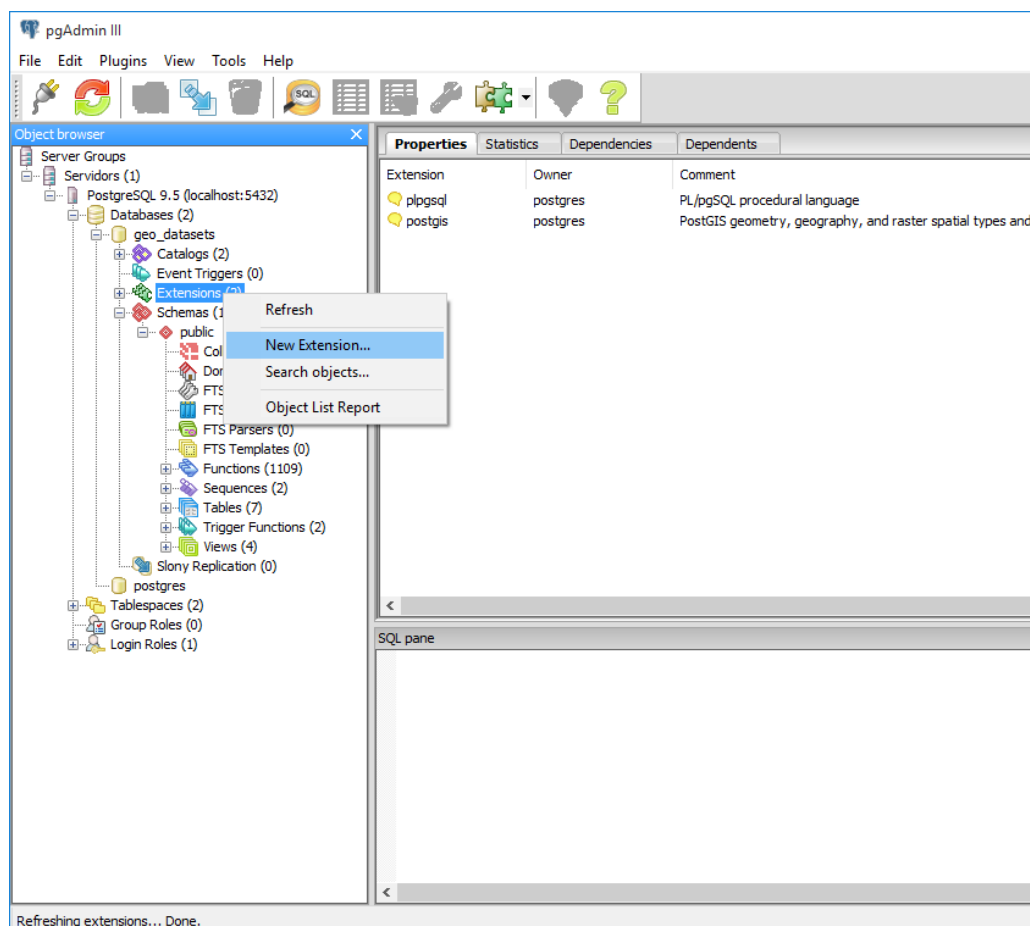
The fastest and easiest way to create a PostGIS enabled database is by using SQL queries. An example is presented as follows:

```
-- Create a new database named "geo_datasets"
CREATE DATABASE geo_datasets;
-- Connect to our newly created database
\connect geo_datasets
-- Enabling PostGIS plugin
CREATE EXTENSION postgis;
```

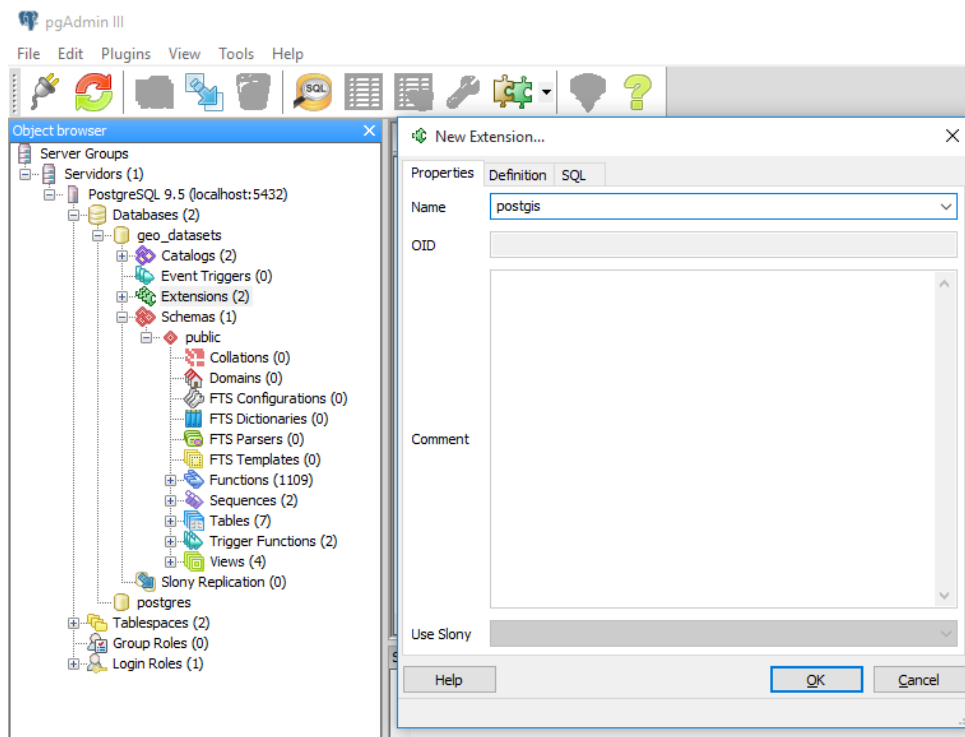
With this, the database that was just created can already use functions and data type from PostGIS.

But in our case, where we import data from a shapefile, as in the case of a new database creation, we will need to enable the PostGIS plugin (per database) in order to be capable of using "geom" data type, and the functions that process that type of data.

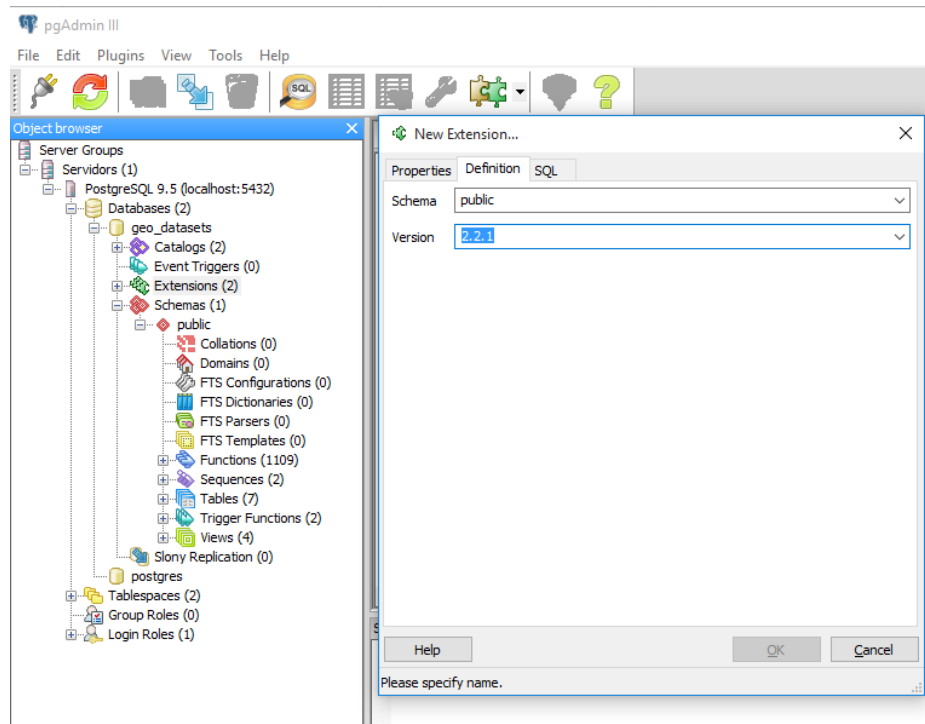
We can also add the plugin using pgAdmin, PostgreSQL's GUI administration tool, right clicking on the extensions section of our desired database, which will bring up this contextual menu:



On “name” field, we will write or select the extension we want to enable from the dropdown (postgis).



Inside the definition tab, we will configure the extension to be used on our desired schema (the schema where the geolocation data tables will be stored), and the plugin version to use.



By doing this, the database will be now ready to receive the imported data from the shapefiles.

4 Importing a Shapefile to database

The next step will be to import the datasets to a table. PostGIS comes with a GUI tool for importing Shapefiles to a PostGIS enabled database. We also have the console application `shp2pgsql`, which is used with this syntax:

```
shp2pgsql -s SRID SHAPEFILE.shp SCHEMA.TABLE | psql -h HOST -d DATABASE -U USER
```

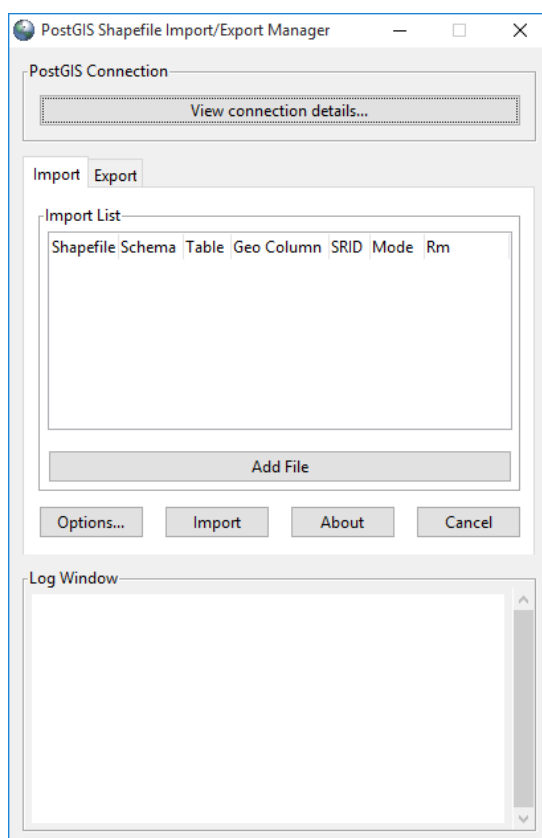
`-s SRID` (Coordinate system definition, for example: 4326, which corresponds to WGS84, used by GPS positioning)

`SHAPEFILE.shp` (File or file route to the shapefile we want to import the data from)

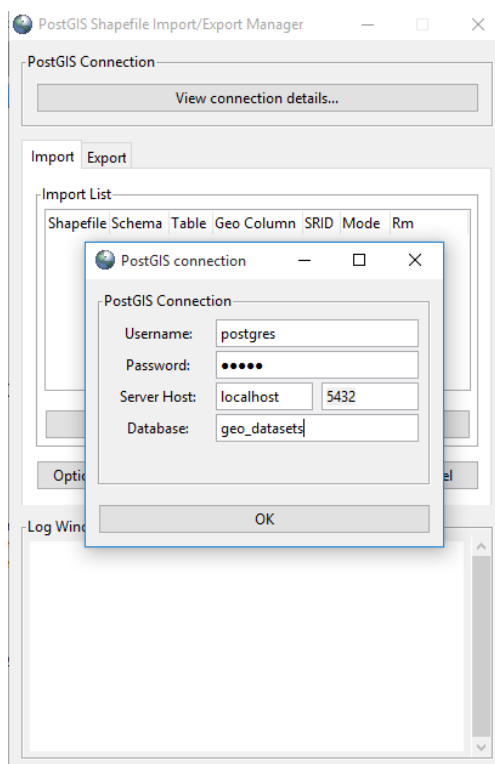
`SCHEMA.TABLE` (Schema and table where we will import data. The table might not exist, and in that case, it will be automatically created when importing)

The piped `psql` command will connect to the specified database. It is also possible to redirect the `shp2pgsql` command output to a `.sql` file and import it to a database, and it will have exactly the same data.

We can also use the import GUI (PostGIS Shapefile Import/Export Manager), which will show this screen on start:



First of all, we will set the database connection parameters. Clicking on *View connection details* will lead us to the connection configuration screen, which looks like this:

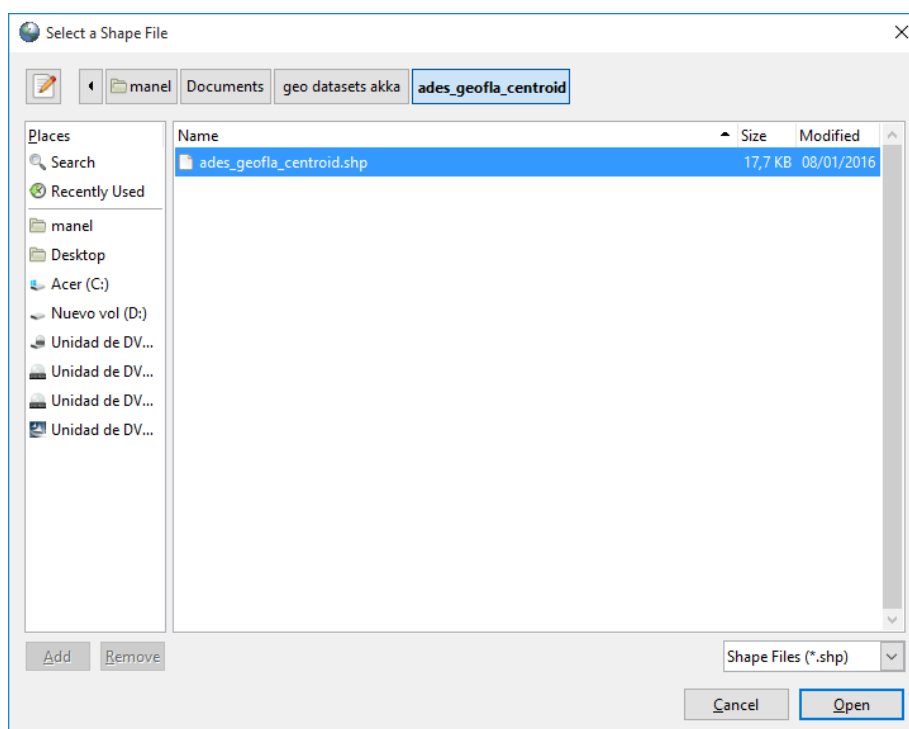


We will fill these parameters with our server information.

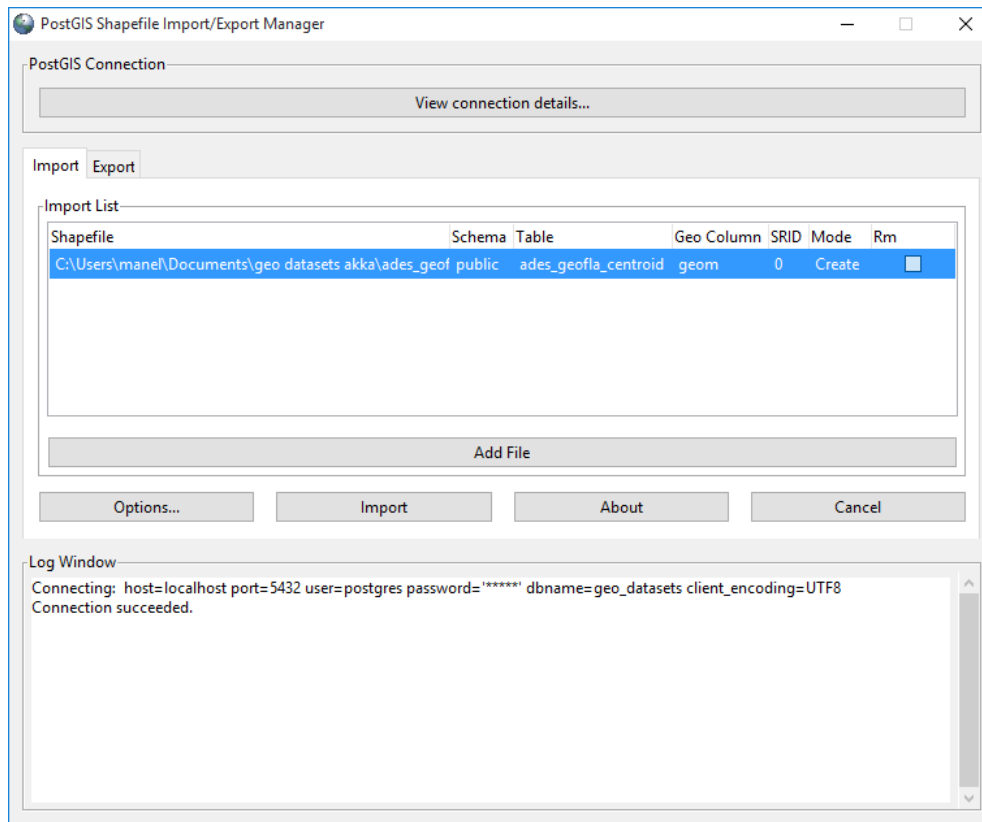
After clicking the OK button, we will see the following message on the log window:

Connecting: host=localhost port=5432 user=postgres password='*****' dbname=geo_datasets client_encoding=UTF8
Connection succeeded.

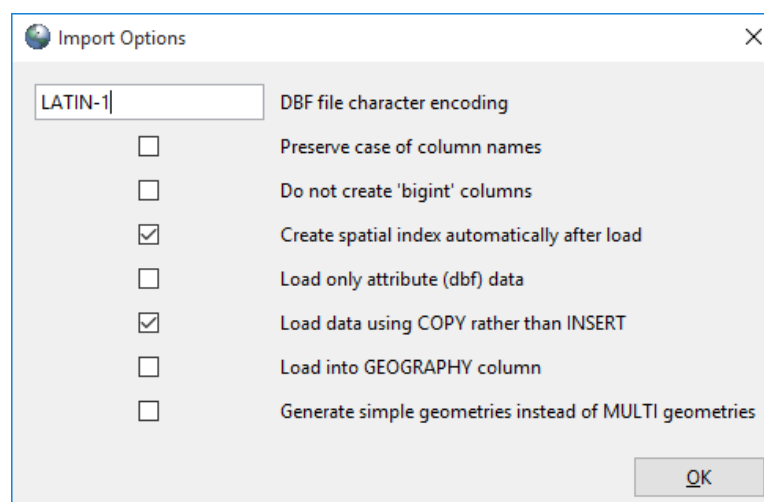
This message confirms that we are connected to the server and database. So, we will proceed to import the shapefile. By pressing the Add File button, a file selection dialog will be displayed.



Once an importing file has been selected, a new line will be added to the Import tab. We can specify the SRID (which is a parameter that specifies the spatial reference system), or change any of the other parameters shown.



For this dataset, we will need to change the encoding settings, because there seems to be some problems when using UTF-8 on a windows database, so we will use LATIN-1 instead.



Once we check every configuration, we can click on Import button. The log window will show the message: “Shapefile import completed” when the process is finished correctly. The table has been created now, and the dataset has been successfully imported.