



## Nuages de Points Concepts, outils et technologies

Paul Blottière, 10 Mai 2016

# Plan

---

- 1 - Présentation
- 2 - Mise en place de l'environnement
- 3 - LIDAR : Light Detection And Ranging
- 4 - LAS, LAZ et LAStools/libLAS (atelier étape 1)
- 5 - PostgreSQL, PostGIS et PDAL
- 6 - PGPointCloud et pgAdmin (atelier étape 2)
- 7 - QGIS (atelier étape 3)
- 8 - Conclusion



## Oslandia

Expertise SIG open source

Formations

Développement

Conseil

Support

## Technologies

QGIS, PostGIS (contributeurs majeurs)

PDAL, PGPointCloud

Applications métiers C++/Python

# Mise en place de l'environnement

```
git clone https://github.com/Oslandia/workshop-pointcloud
```

## Environnement

Slides : *supports.ods*

Ateliers : *README.md*, fichiers de données

Machine virtuelle Ubuntu (Virtualbox)

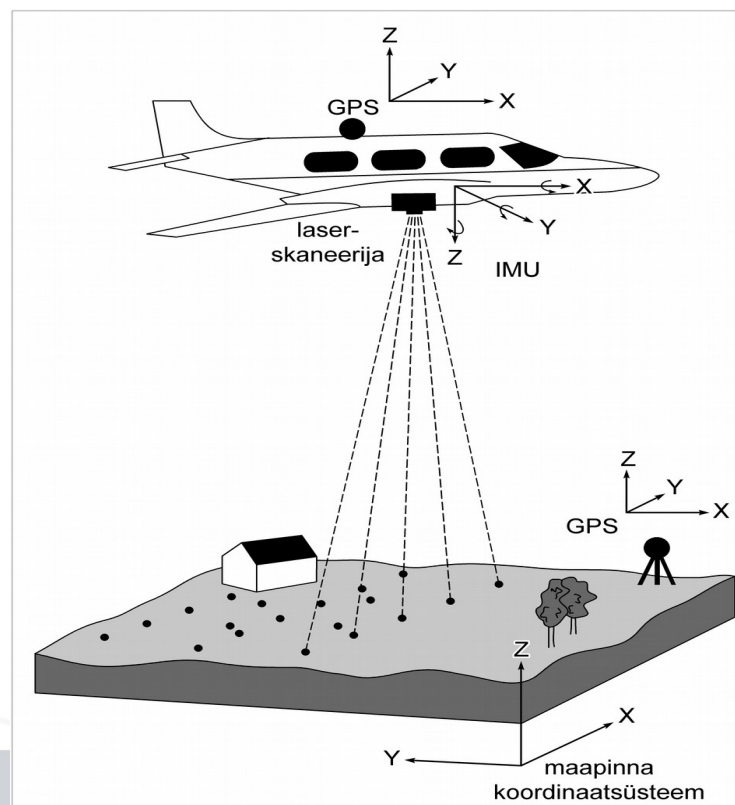


# LIDAR : Light Detection And Ranging (1)

Capteur au sol ou embarqué (avion, voiture, ...)

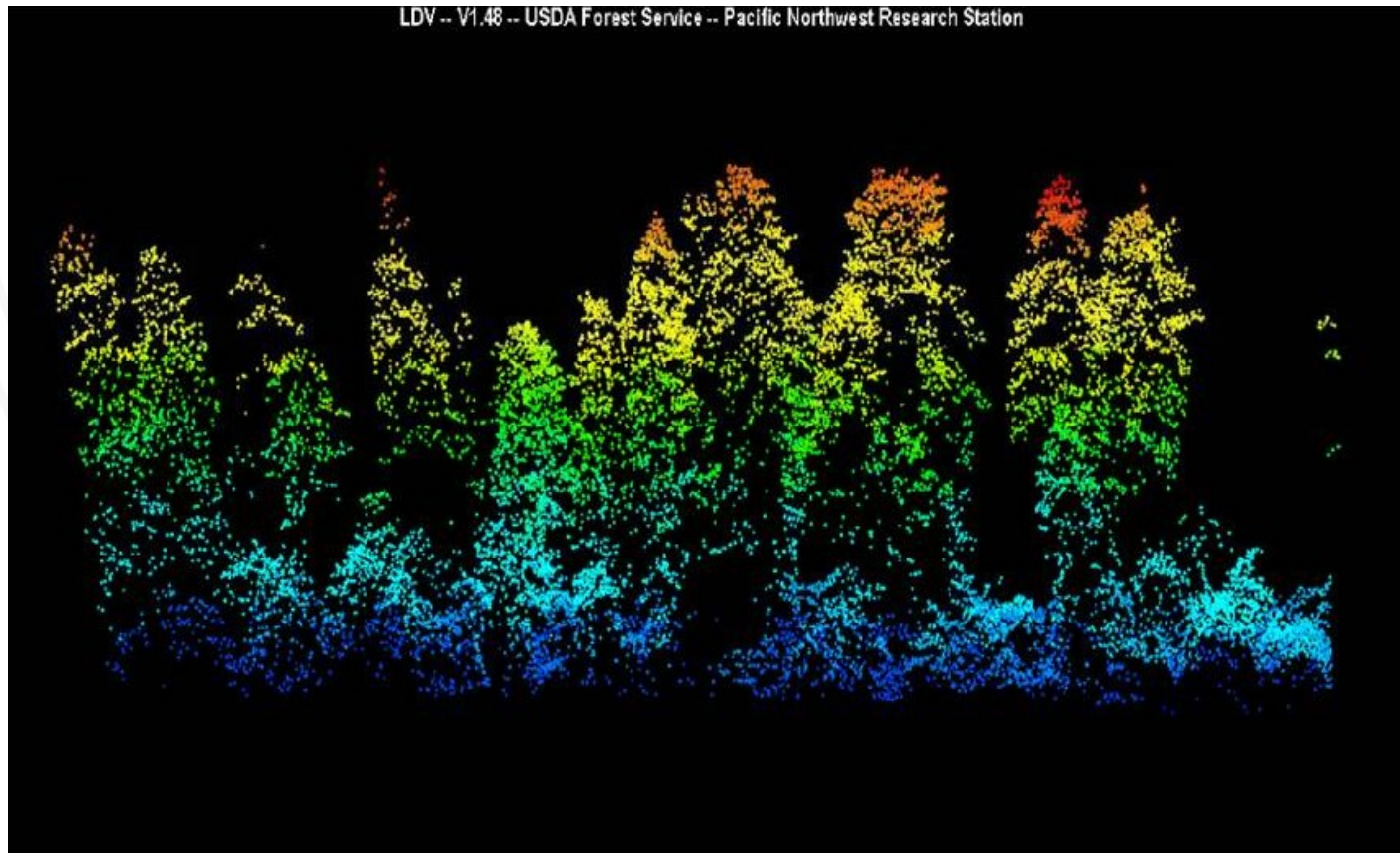
Plus de 100 000 impulsions par secondes

Impulsions réfléchies (végétation, bâtiments, sol, ...) et récupérées par le scanner



# LIDAR : Light Detection And Ranging (2)

Analyse de la forme d'onde réfléchiée : on garde un “retour” pour chaque pic d'intensité majeur (seuil)

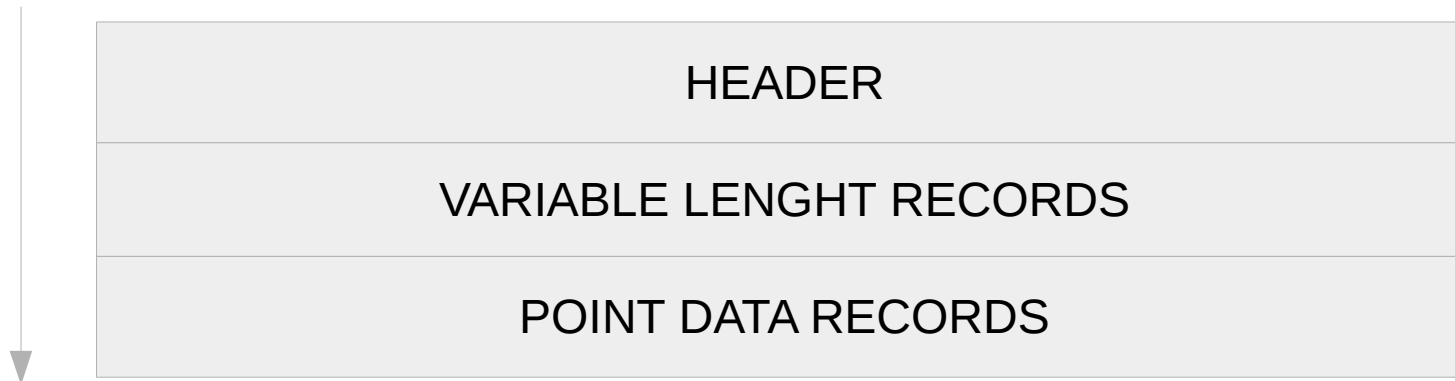


$$z = \frac{t * c}{2}$$

## Format ouvert d'échange des données

### Spécifications des différentes versions de LAS :

<http://www.asprs.org/committee-general/laser-las-file-format-exchange-activities.html>



### LAZ : compression de fichiers LAS

<https://www.cs.unc.edu/~isenburg/lastools/download/laszip.pdf>

## Outils de manipulation de fichiers LAS :

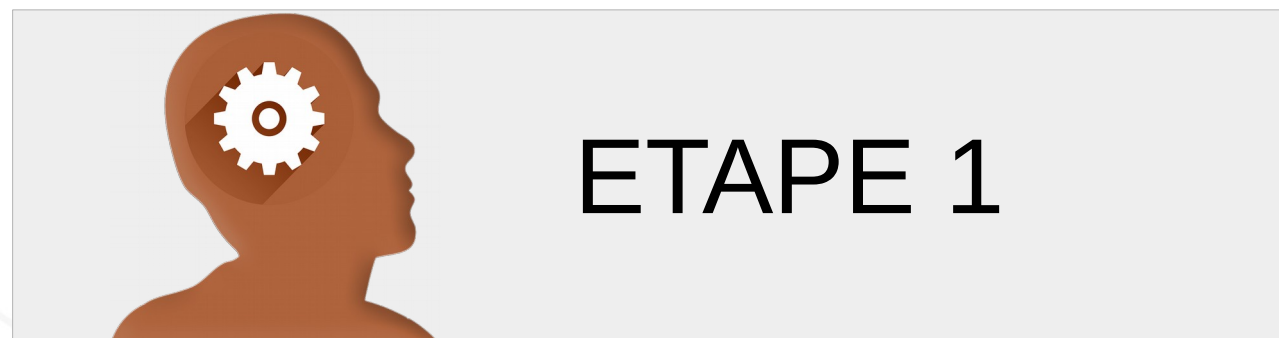
<https://github.com/LAStools>

<https://github.com/libLAS/libLAS>



## Laquelle ?

<http://www.liblas.org/lastools.html>





## Chaque point du nuage est spatialisé

volonté naturelle de stocker les points en base de données!

## Base de données :

PostgreSQL : SGBDRO, libre, possibilité d'ajouter de nouveaux type de données, des opérateurs, des fonctions, ...

PostGIS : support d'objets géographique à PostgreSQL



## Un nuage de points

peut contenir des milliards de points

... où chaque point peut posséder plus de 10 dimensions



Le stockage de chaque point individuellement dans une base de données est inenvisageable !

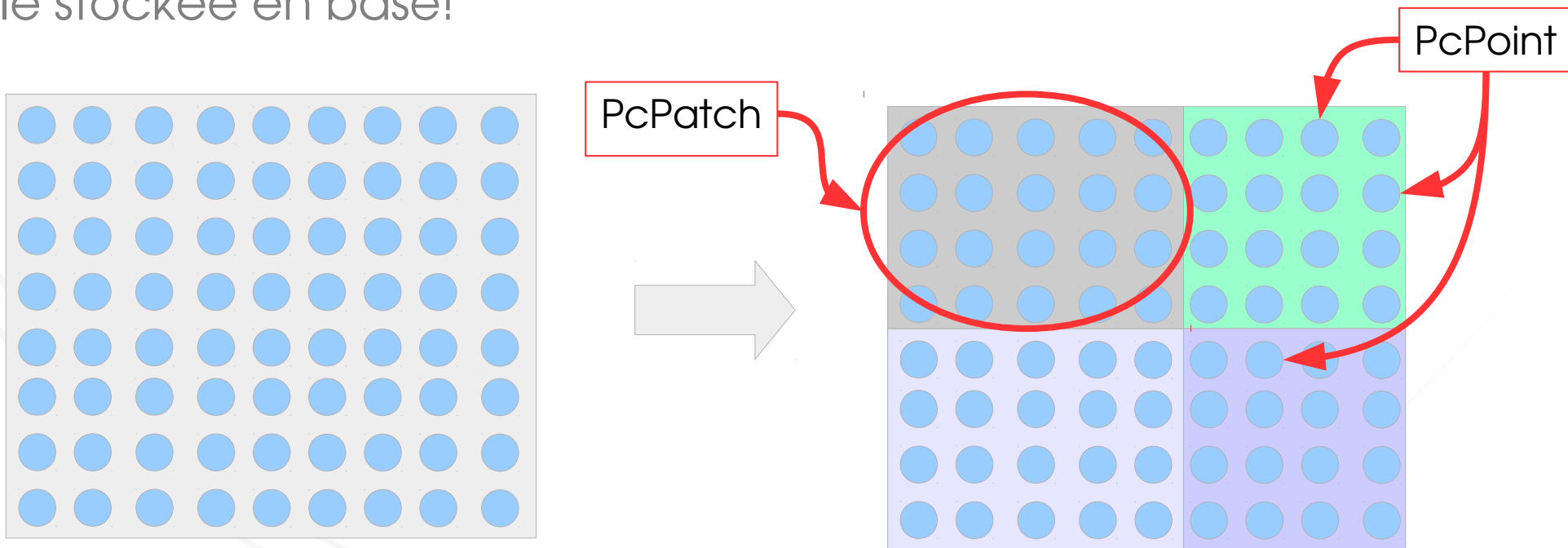
# PGPointcloud (2)

## PGPointcloud :

<https://github.com/pgpointcloud/pointcloud>

Extension PostgreSQL pour le stockage de nuage de points

Organise le stockage des points en patch pour réduire la taille de la table stockée en base!



# PGPointcloud (3)

## Schéma :

Gère la variabilité de format des points

Document XML

Stocké dans la table *pointcloud\_formats*

```
INSERT INTO pointcloud_formats (pcid, srid, schema) VALUES (1, 4326,
'<?xml version="1.0" encoding="UTF-8"?>
<pc:PointCloudSchema xmlns:pc="http://pointcloud.org/schemas/PC/1.1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <pc:dimension>
    <pc:position>1</pc:position>
    <pc:size>4</pc:size>
    <pc:description>X coordinate as a long integer. You must use the
                        scale and offset information of the header to
                        determine the double value.</pc:description>
    <pc:name>X</pc:name>
    <pc:interpretation>int32_t</pc:interpretation>
    <pc:scale>0.01</pc:scale>
  </pc:dimension>
  <pc:dimension>
    <pc:position>2</pc:position>
    <pc:size>4</pc:size>
    <pc:description>Y coordinate as a long integer. You must use the
                        scale and offset information of the header to
                        determine the double value.</pc:description>
    <pc:name>Y</pc:name>
    <pc:interpretation>int32_t</pc:interpretation>
    <pc:scale>0.01</pc:scale>
```

## Compression des patches :

Aucune, dimensionnelle ou GHT

Définie dans le schéma XML

```
<pc:metadata>  
  <Metadata name="compression">dimensional</Metadata>  
</pc:metadata>
```

## Compression dimensionnelle :

Adapté aux petits patches (faible variabilité)

Chaque dimension d'un PcPatch utilise son algorithme de compression dimensionnelle (RLE, ZLIB, SIGBITS)

```
int  
pc_dimstats_update(PCDIMSTATS *pds, const PCPATCH_DIMENSIONAL *pdl)
```



## Point Data Abstraction Library

Suite d'outils en ligne de commande

Permet de manipuler les nuages de points (lecture, filtrage, écriture, ...)

### Pipeline :

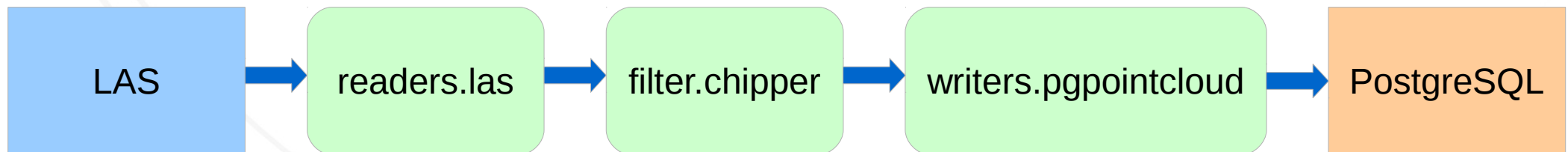
Suite d'opérations à réaliser pour construire une chaine de traitement

En JSON depuis la version 1.2 (avant en XML)



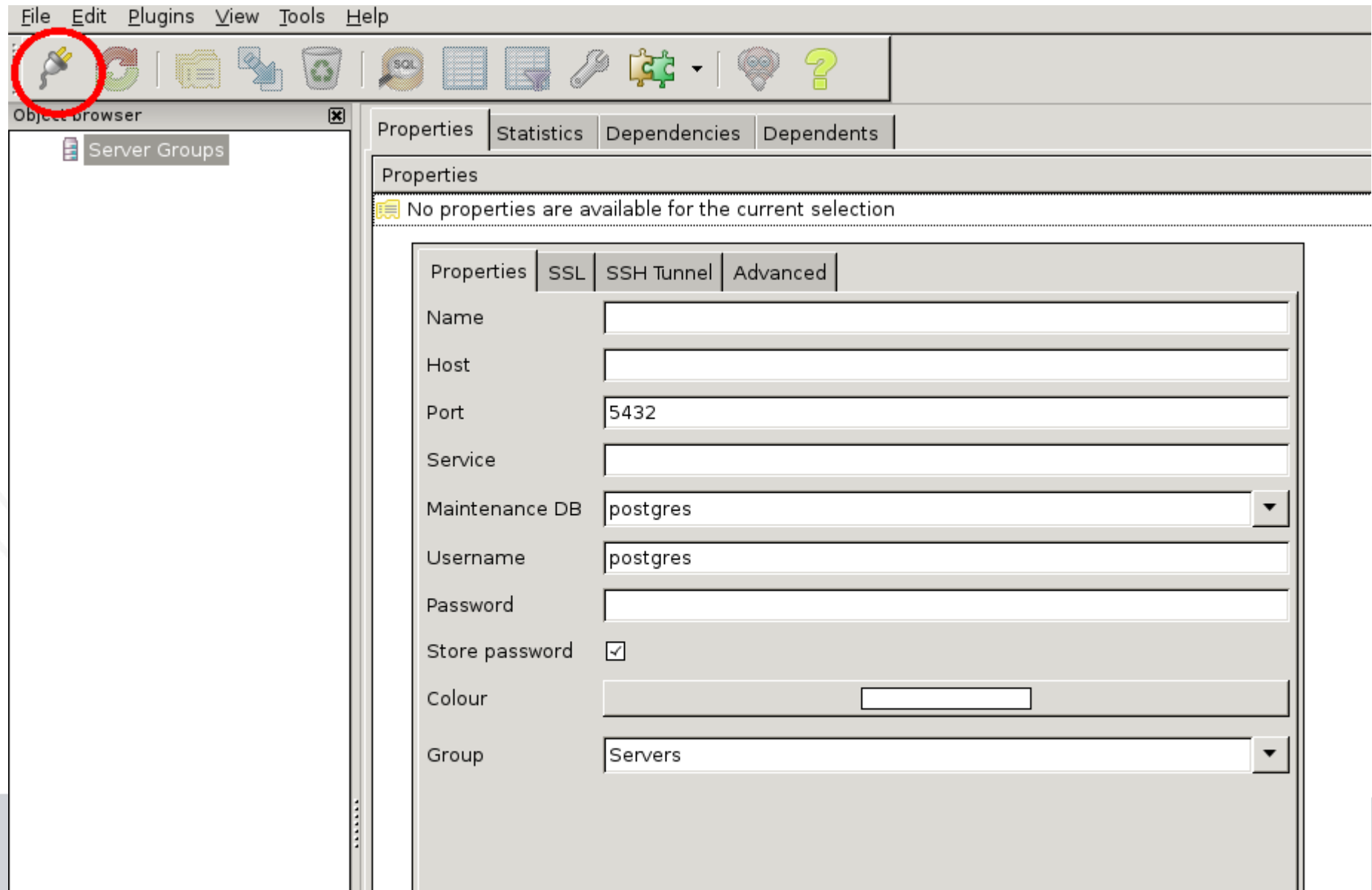
## Exemple :

```
{
  "pipeline": [
    "/home/vpi/data/auvergne/lidarverne/.opendata.craig.fr/opendata/lidar/agglos/2013_clermont-ferrand/clermont.las",
    {
      "type": "filters.chipper",
      "capacity": "1000"
    },
    {
      "type": "writers.pgpointcloud",
      "connection": "dbname='foss4g' user='postgres' port='5433'",
      "table": "lidar",
      "compression": "dimensional",
      "srid": "2154"
    }
  ]
}
```



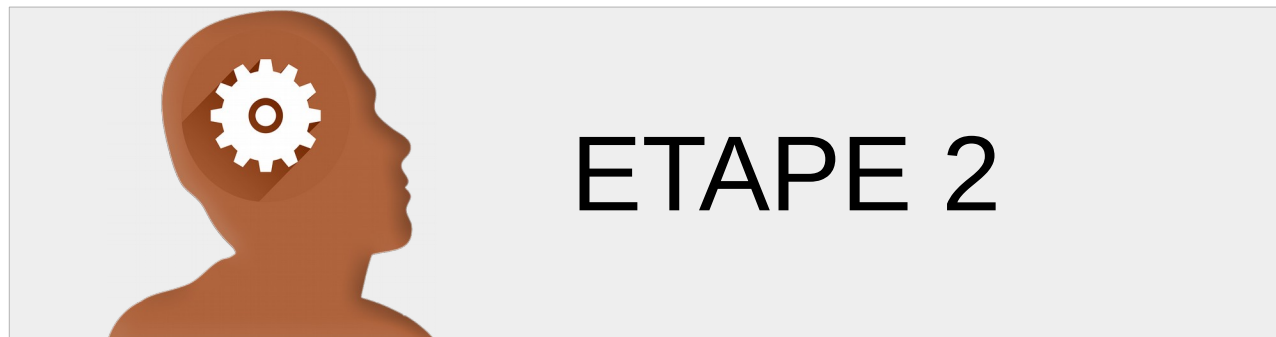
# PgAdmin (1)

## Outil d'administration graphique pour PostgreSQL





# PgAdmin (2)



## Logiciel SIG :

libre

multiplate-forme

C++, plugin Python



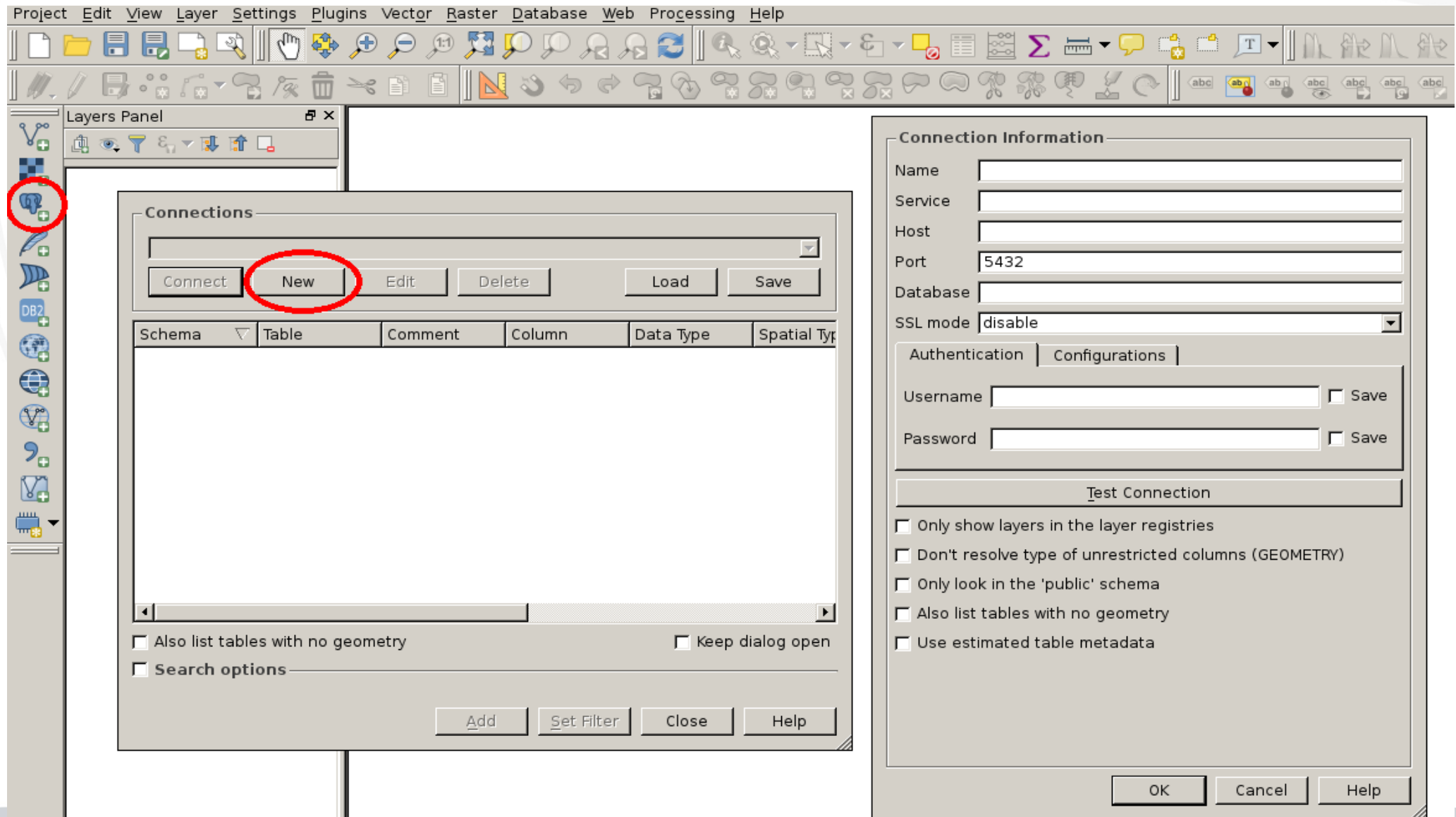
## Le lien avec les nuages de points?

Connexion aux base de données spatiales comme PostGIS!

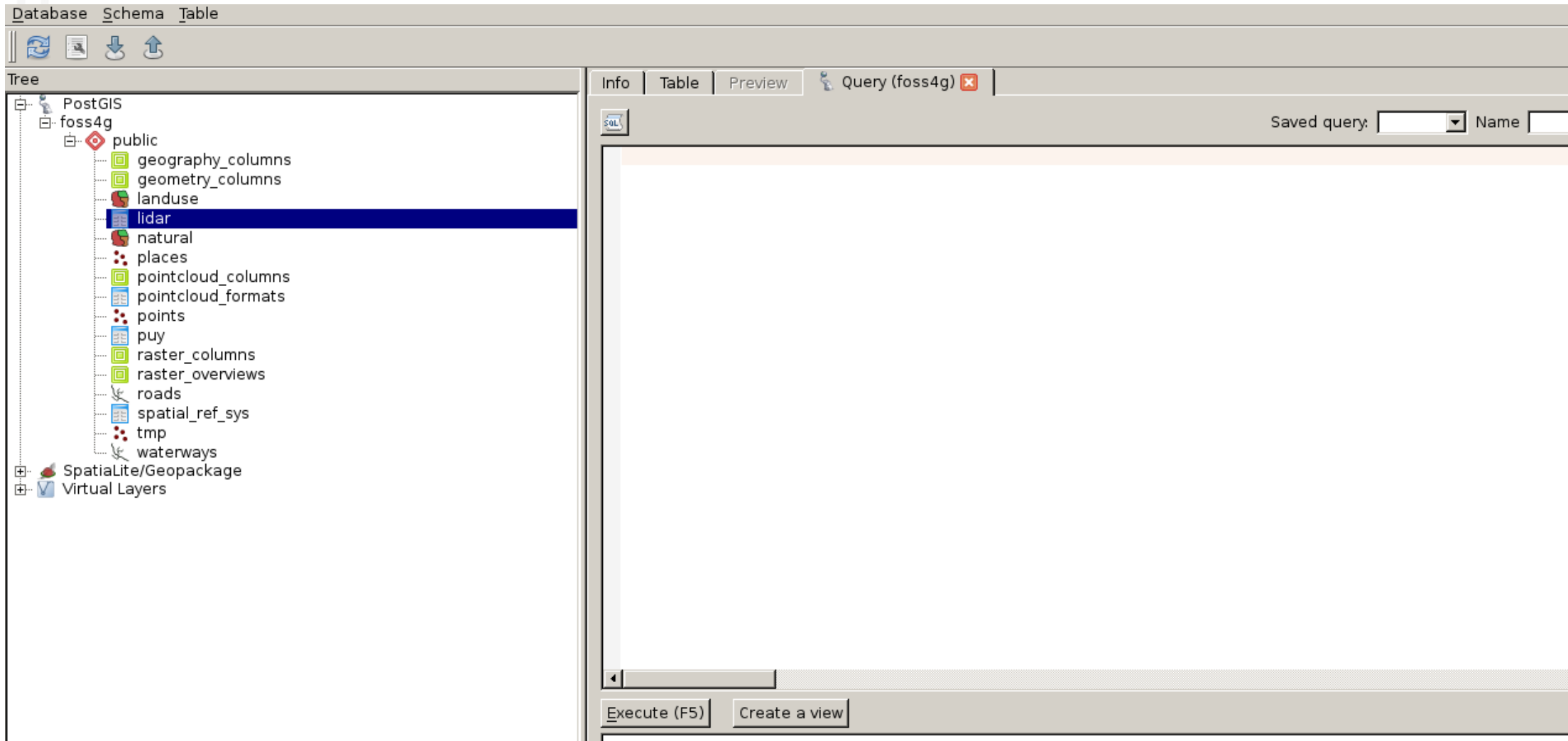
Supporte la notion de patch

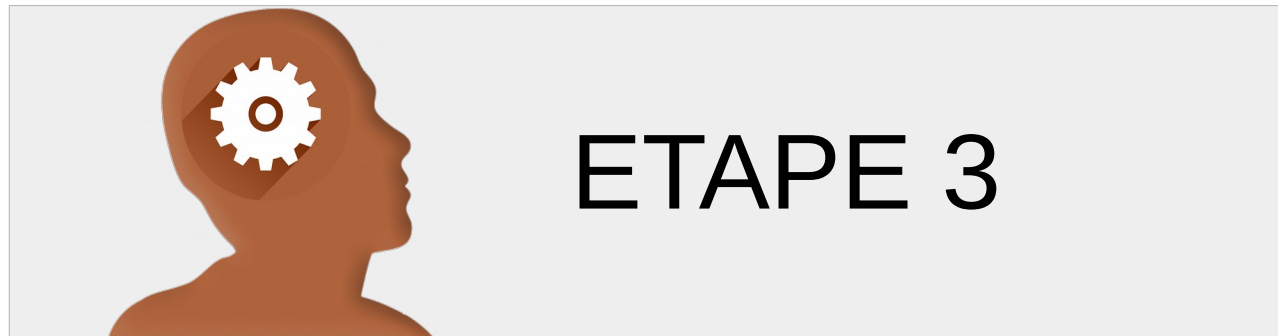
# QGIS (2)

## Add PostGIS layer :

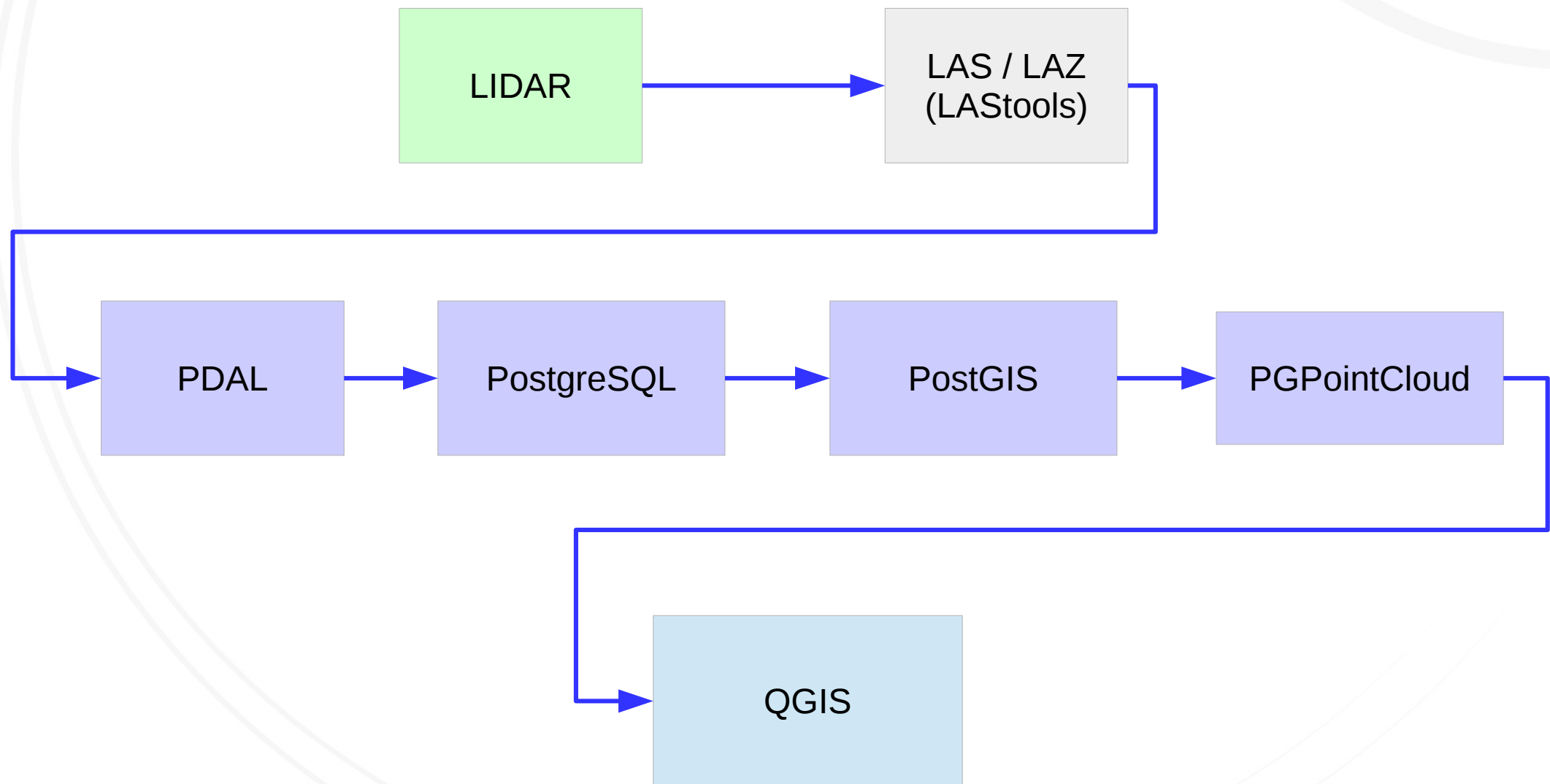


## Database Manager :





# Conclusion



# Questions

