



GEO 7630

**Intégration et visualisation de données
géographiques**

Semaine 10 -

**Applications et infrastructures dorsales
Atelier conteneurs et webmapping**



Cours 10

Objectifs du cours

Webmapping et architecture informatique

- Principe d'architecture dorsale (backend)
- Atelier d'introduction aux conteneurs
- Atelier base de données spatiales (PostGIS)
- Revue des API géospatiales dorsales
- Atelier Mise en place de composantes backend pour le webmapping avancé

Laboratoire - Webmapping intermediaire

- Ajouter une couche de tuiles vectorielles
- Ajouter une couche WFS (pg_featureserv)
- Contrôle de carte
- Ajouter une icône
- Filtrer
- Évènement souris — Click, Hover
- Retour de propriétés
- Popup
- JumpTo



Cours 11

Objectifs du cours

Principes, Architectures et Services Géospatiaux pour le Développement d'Applications de Webmapping

- Développement d'application sur le web
- Principes et bases du webmapping
- Architecture d'une application web open source
- Les normes Open Geospatial Consortium (OGC)
- Notions de services web géospatiaux, WMTS, WFS, VTS ...
- Diffusion des données géospatiales avec GeoServer
- Notions de services web géospatiaux

Laboratoire webmapping avancé:

- Ajouter une icône
- Filtrer
- Évènement souris - Click, Hover
- Retour de propriétés
- Popup
- JumpTo



Cours 12

Objectifs du cours

Concepts de webmapping avancés

- Concepts d'utilisation des bibliothèques géospatiales open sources avancées backend et frontend
- Analyse spatiale dans le web
 - Distance
 - Buffer
 - Intersect / Dissolve ...
 - Grids
 - Interpolation ...
- Atelier analyse spatiale dans le web

Laboratoire - Analyse spatiale dans le web

- Changer la couleur des bâtiment en fonction de leur hauteur
- Créer une carte de chaleur
- Créer et styliser des clusters
- Bâtiments en 3D
- Expression avancée Paint et Layout
- TurfsJS



Cours 13

Objectifs du cours

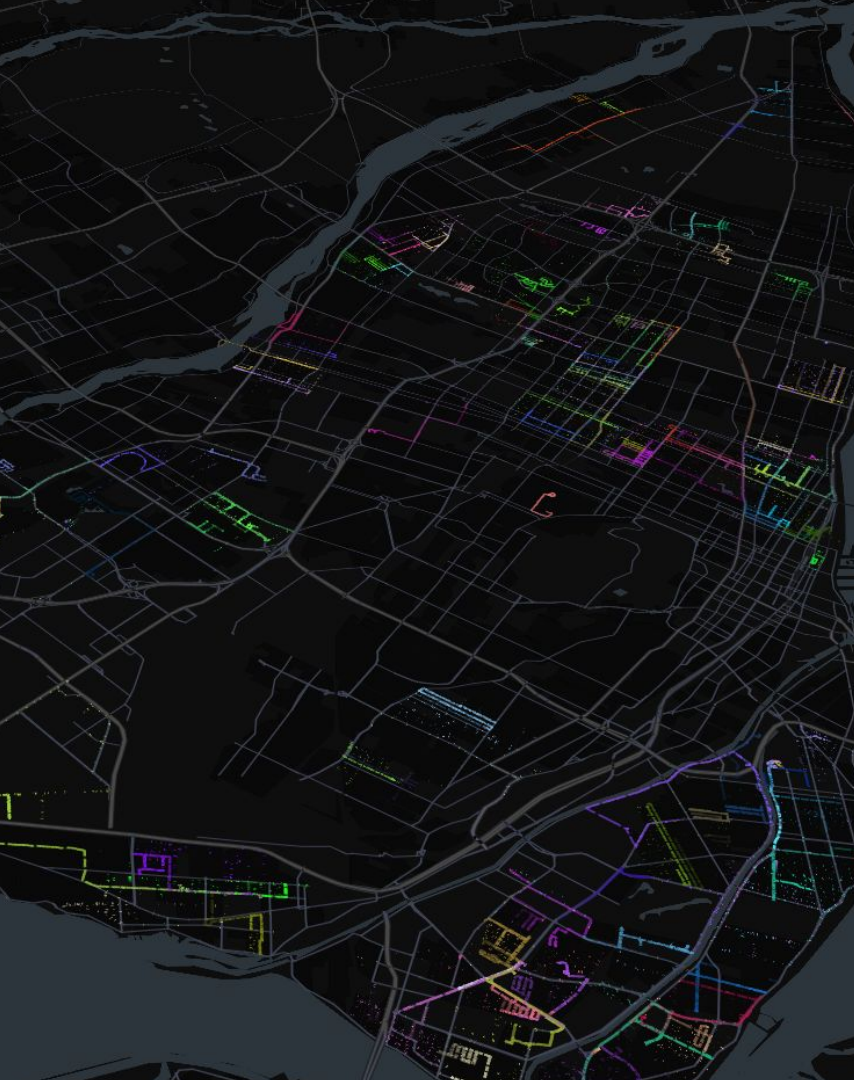
Principe de mise en production d'une application de webmapping

- Cycles de développement et bonnes pratiques
- Cycles de vie applicative
- Bonnes pratiques de diffusions
- Assurance qualité
- Publication et versionnement du code source
- Concepts de contribution open source
- Publication de l'application dans le web
- Préparation à l'examen final

Laboratoire -

3Dtiles + Itowns (lidar dans le web)

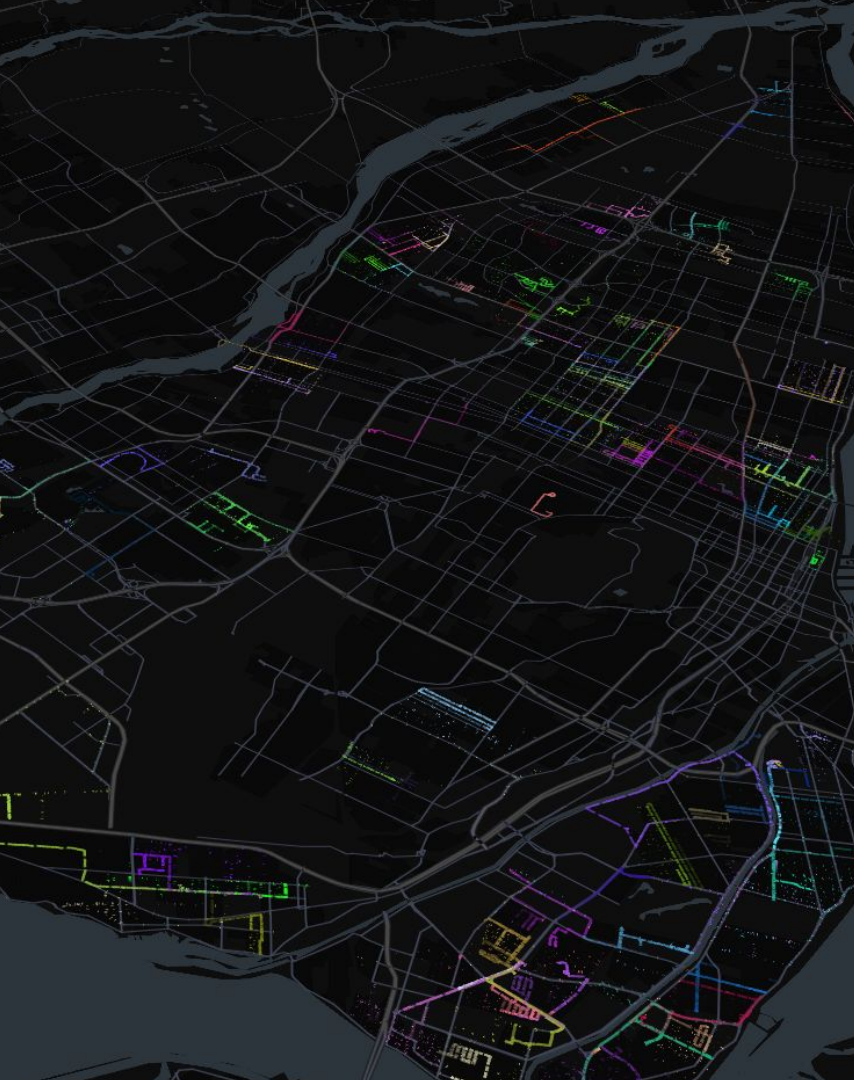
Mise en production d'une application de webmapping



Architecture d'une application web open source

Architecture d'une application web open source:

- L'architecture client-serveur
- La modélisation de données pour le webmapping
- Les bibliothèques de cartographie côté client
- La gestion des requêtes côté serveur
- Les outils de développement d'application web open source (Eclipse, NetBeans)



L'architecture client-serveur

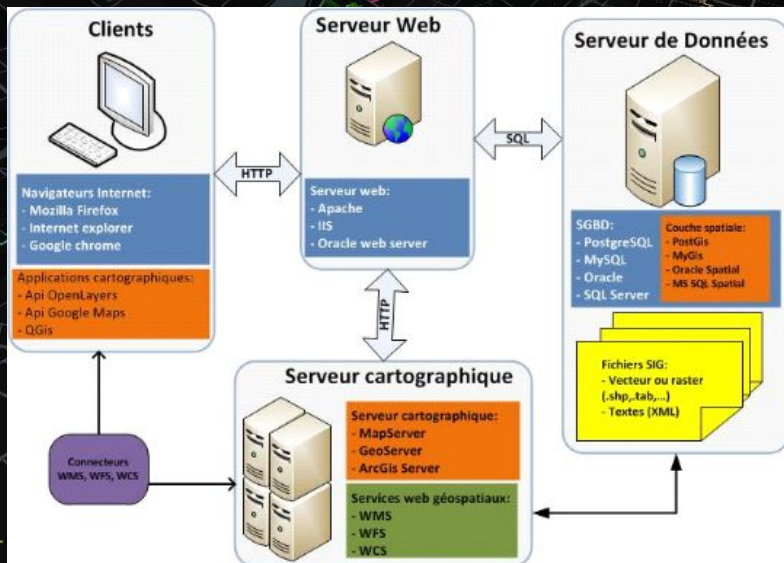
Introduction à l'architecture client-serveur

- Les composants d'une architecture client-serveur
- Les avantages et les inconvénients de l'architecture client-serveur
- Les protocoles de communication client-serveur (HTTP, TCP/IP, WebSocket)
- Les exemples d'application de l'architecture client-serveur

L'architecture client-serveur

Les composants d'une architecture client-serveur

- Le client
- Le serveur
- Le protocole
- La couche de données
- La couche de traitement
- La couche de sécurité



L'architecture client-serveur



Le client :

Interface utilisateur qui permet d'accéder aux services offerts par le serveur.

Le client peut être une application installée sur une machine, un navigateur web ou une application mobile.

L'architecture client-serveur

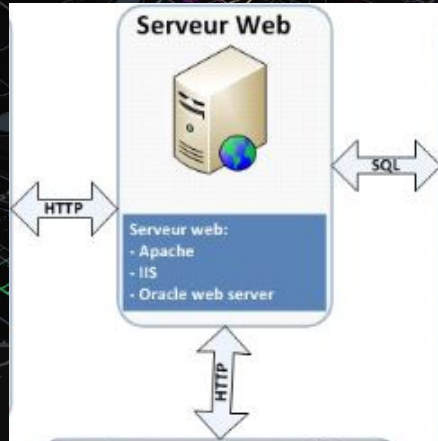


Le serveur :

Entité qui fournit des services aux clients.

Le serveur peut être une machine physique ou virtuelle, sur laquelle des logiciels et des services sont exécutés.

L'architecture client-serveur

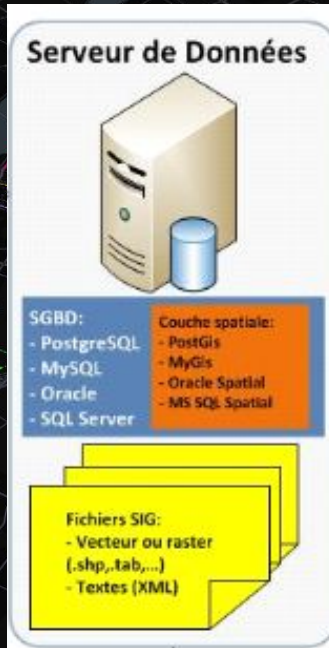


Le protocole :

Ensemble de règles et de formats qui permettent à un client de communiquer avec un serveur.

Les protocoles courants pour les architectures client-serveur sont HTTP, FTP, TCP/IP, WebSocket, etc.

L'architecture client-serveur



La couche de données :

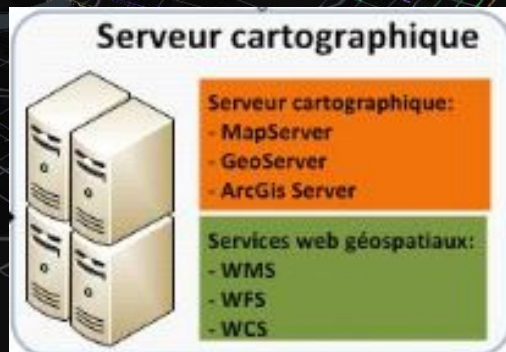
Couche qui gère l'accès et la manipulation des données côté serveur.

Cette couche peut inclure des bases de données, des services web, des API, etc.

L'architecture client-serveur

La couche de traitement :

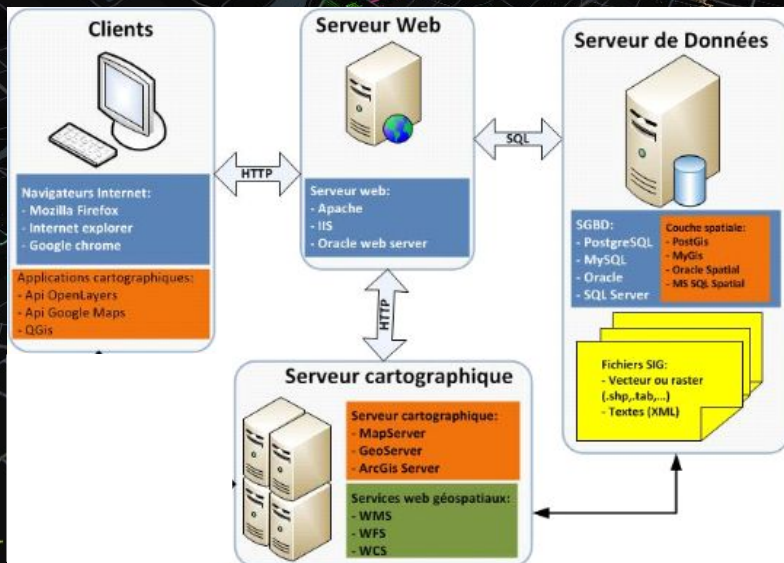
Couche qui gère les opérations de traitement côté client, par exemple la validation des données, les opérations de calcul, etc.



L'architecture client-serveur

La couche de sécurité :

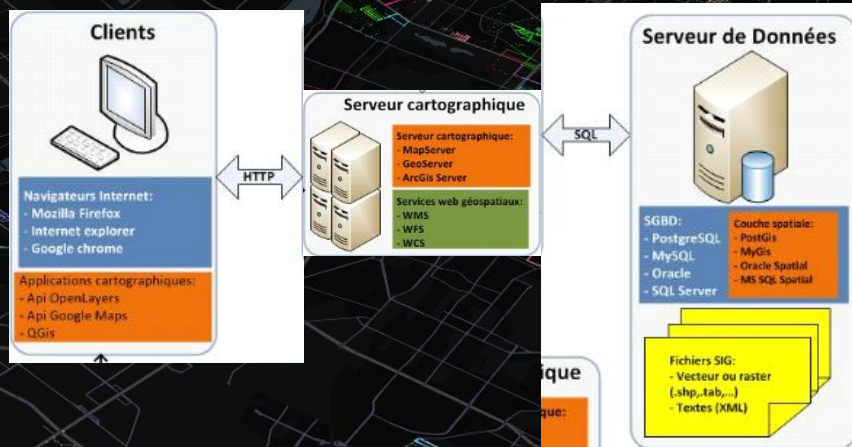
Couche qui garantit la sécurité des échanges entre le client et le serveur, en protégeant les données, les communications, les accès, etc.



L'architecture client-serveur

Les composants d'une architecture client-serveur

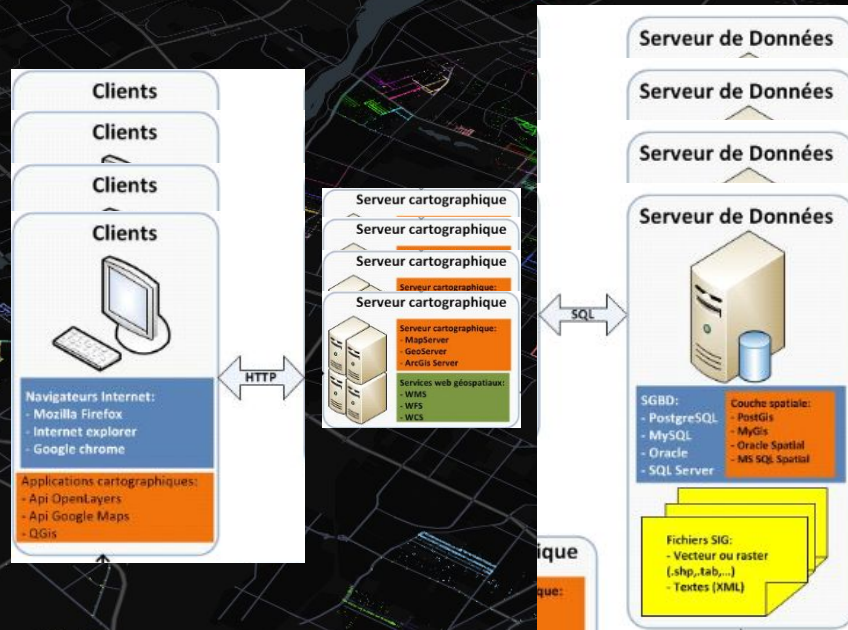
Dans le cadre du cours nous allons abstraire la couche **SERVEUR WEB** pour connecter un serveur cartographique multifonctions directement à nos bases de données



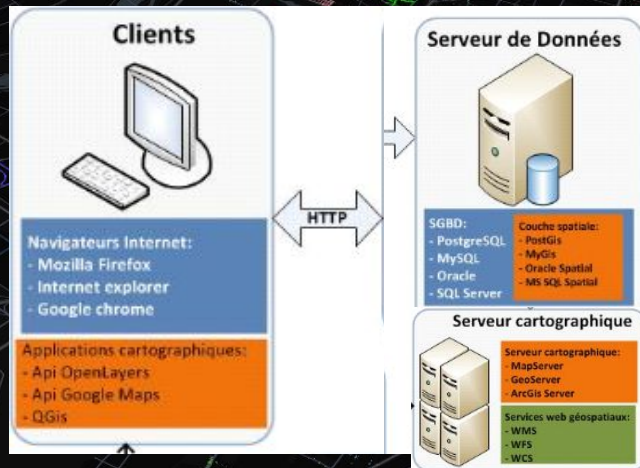
Les avantages et les inconvénients de l'architecture client-serveur

Avantages :

- La séparation
- La centralisation
- La communication
- La flexibilité et la scalabilité



Les avantages et les inconvénients de l'architecture client-serveur



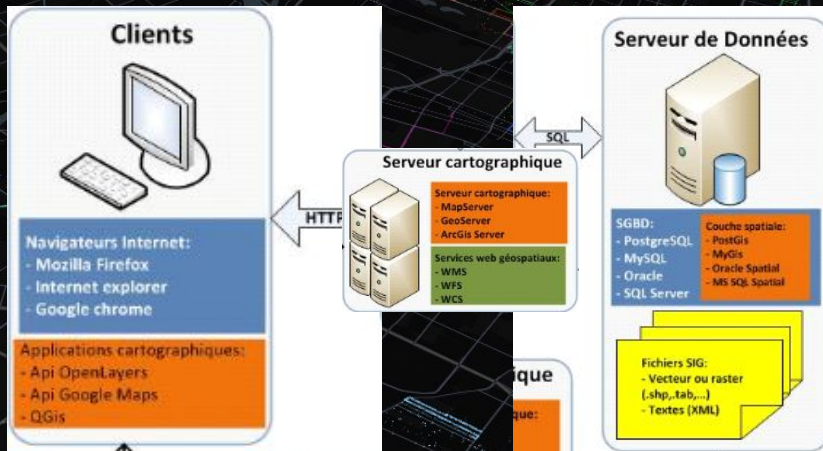
Avantages :

La séparation entre la logique métier et l'interface utilisateur facilite la maintenance, l'évolution et la réutilisation des composants.

Les avantages et les inconvénients de l'architecture client-serveur

Avantages :

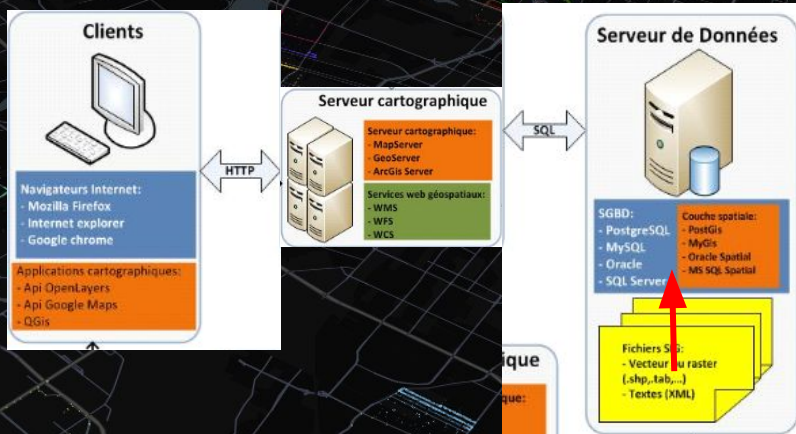
La séparation entre la logique métier et l'interface utilisateur facilite la maintenance, l'évolution et la réutilisation des composants.



Les avantages et les inconvénients de l'architecture client-serveur

Avantages :

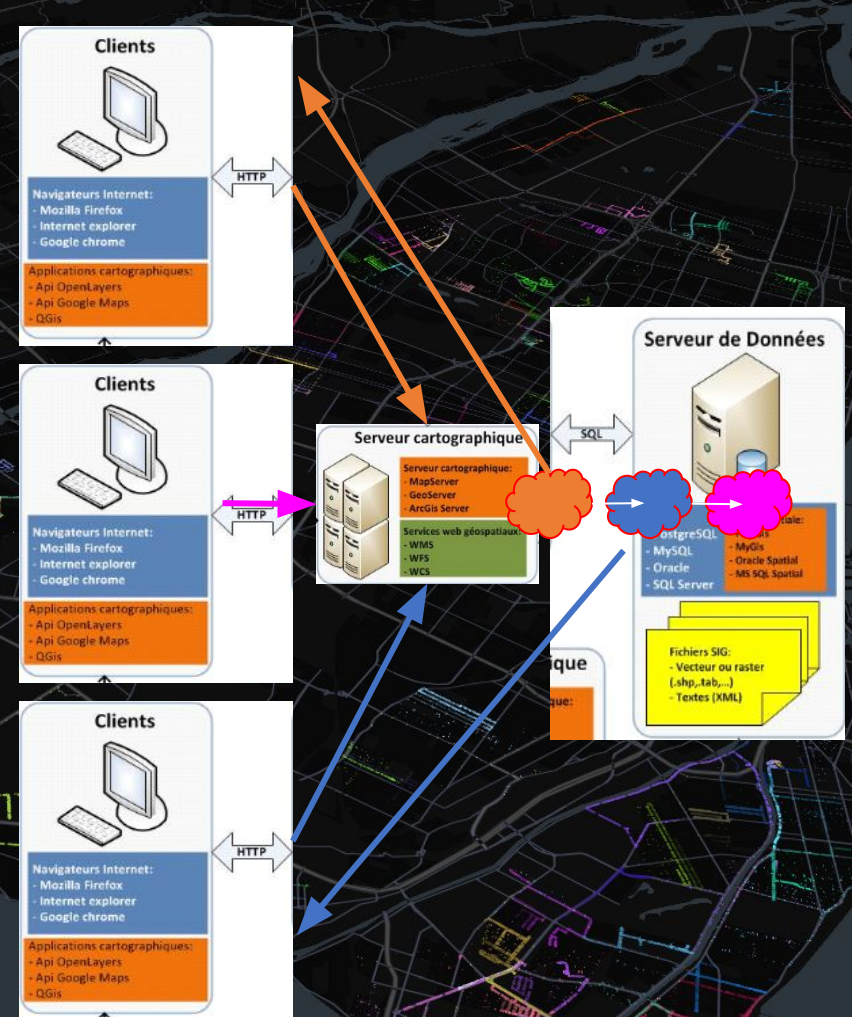
La centralisation des données et des services permet une meilleure gestion de la sécurité, de la disponibilité et de la performance.



Les avantages et les inconvénients de l'architecture client-serveur

Avantages :

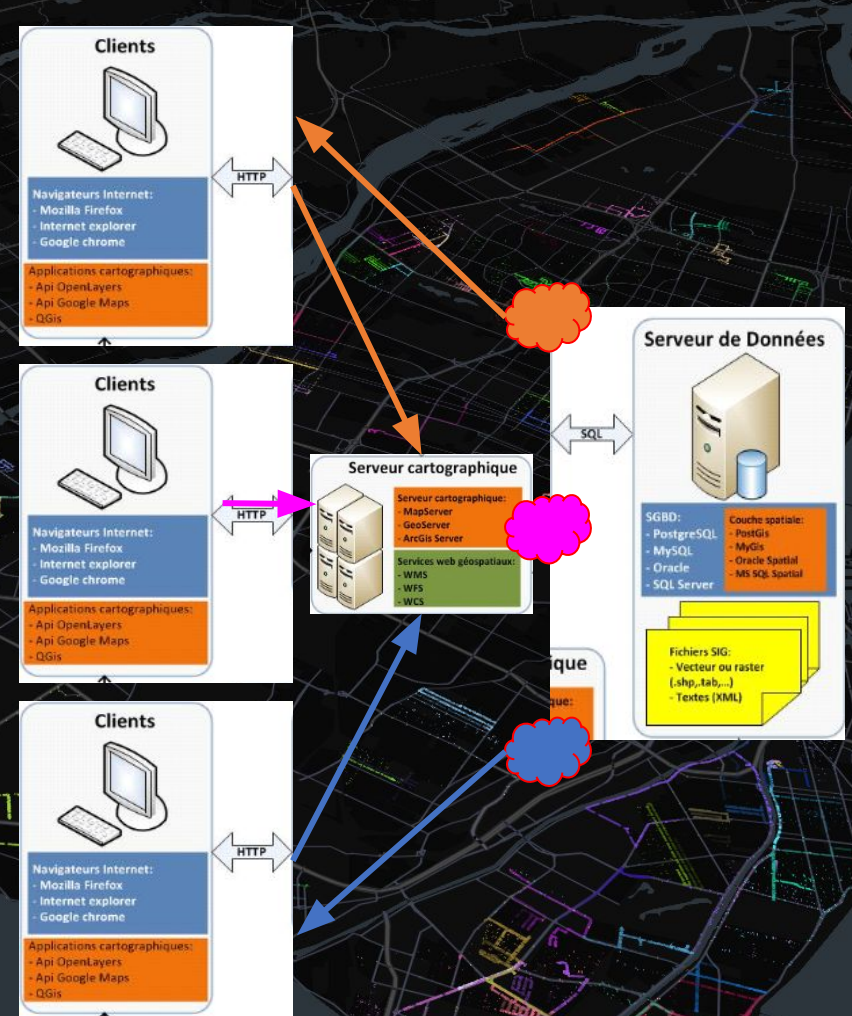
La communication entre le client et le serveur peut se faire de manière asynchrone, ce qui permet de rendre les applications plus réactives et plus performantes.



Les avantages et les inconvénients de l'architecture client-serveur

Avantages :

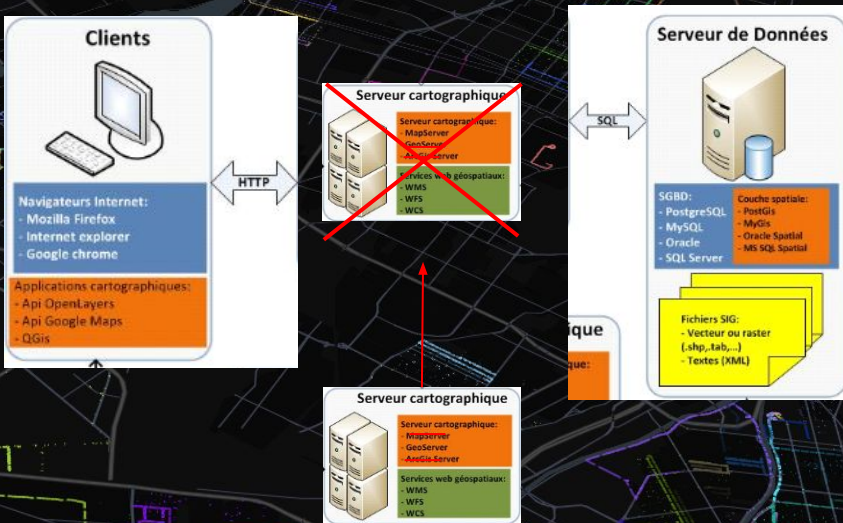
La communication entre le client et le serveur peut se faire de manière asynchrone, ce qui permet de rendre les applications plus réactives et plus performantes.



Les avantages et les inconvénients de l'architecture client-serveur

Avantages :

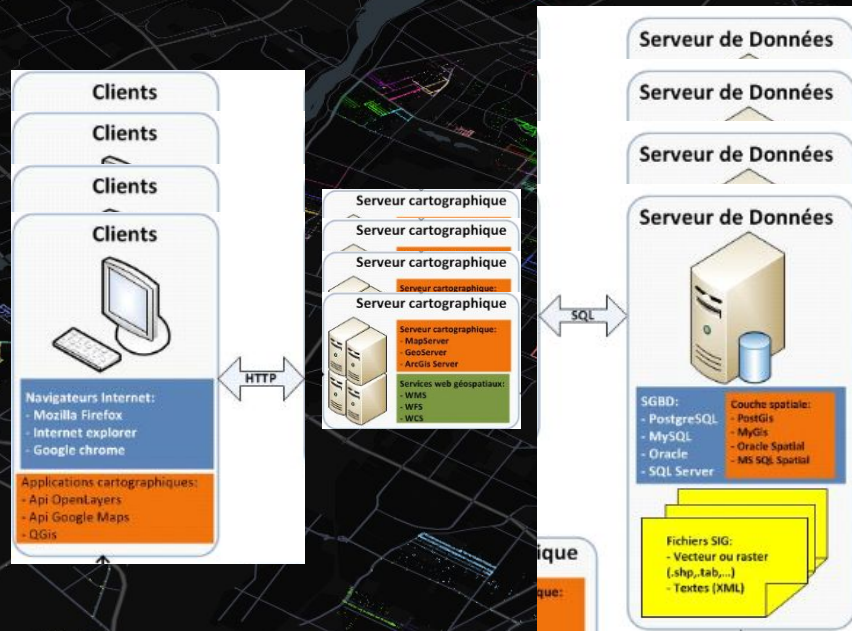
La flexibilité et la scalabilité de l'architecture client-serveur permettent de faire évoluer facilement l'application en ajoutant ou en supprimant des serveurs selon les besoins.



Les avantages et les inconvénients de l'architecture client-serveur

Avantages :

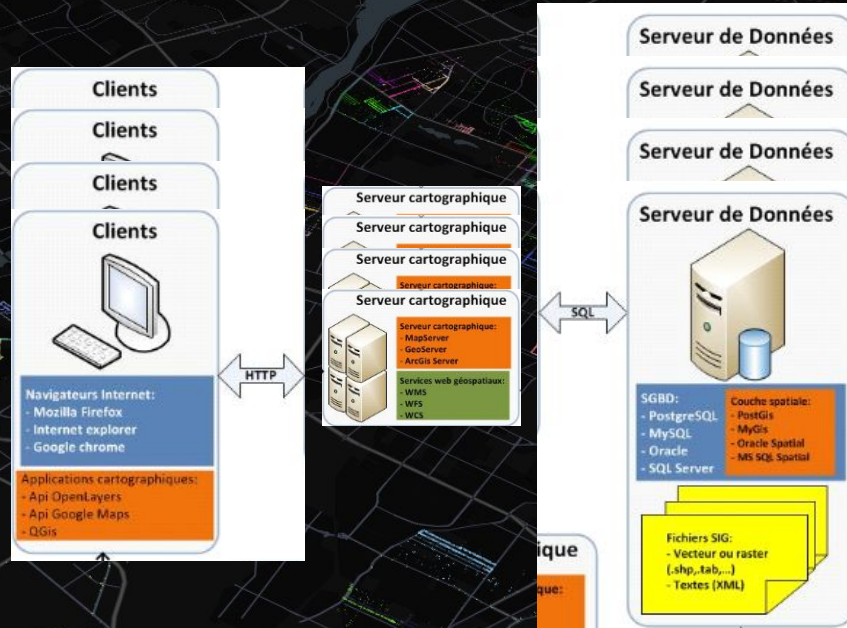
La flexibilité et la scalabilité de l'architecture client-serveur permettent de faire évoluer facilement l'application en ajoutant ou en supprimant des serveurs selon les besoins.

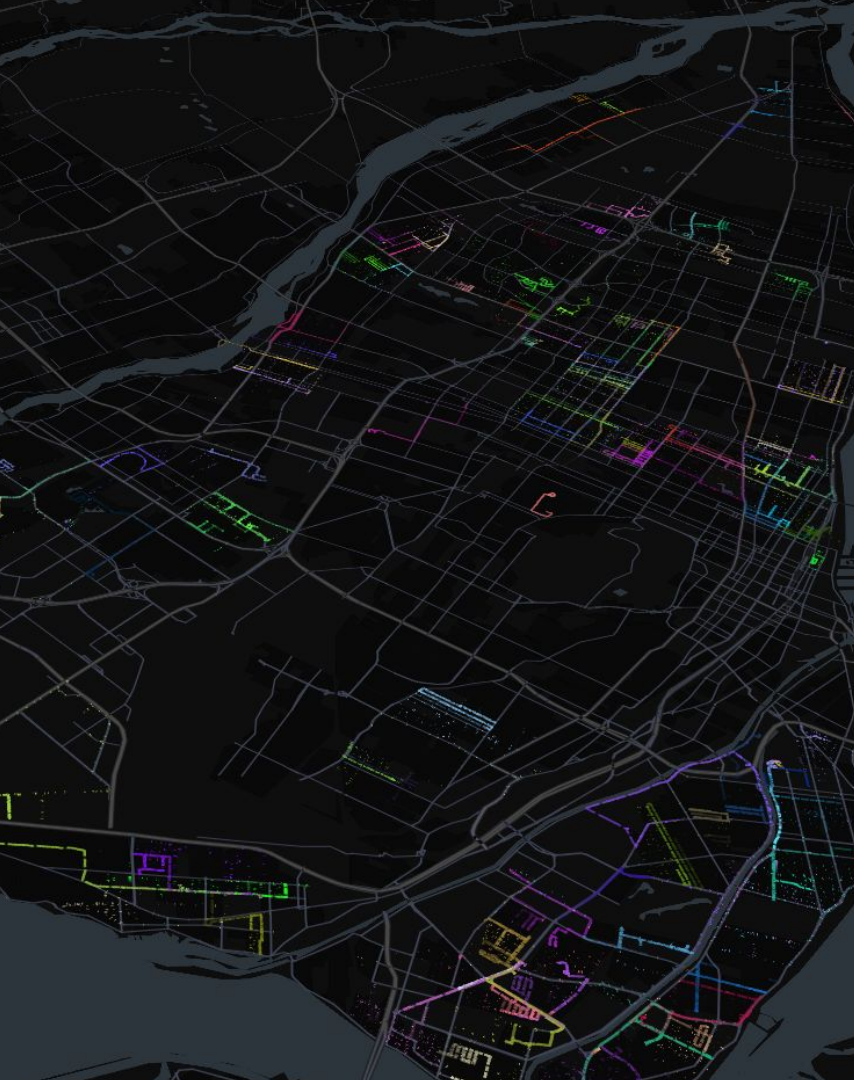


Les avantages et les inconvénients de l'architecture client-serveur

Inconvénients :

- La dépendance
- La complexité
- Les temps de latence
- Les problèmes de sécurité

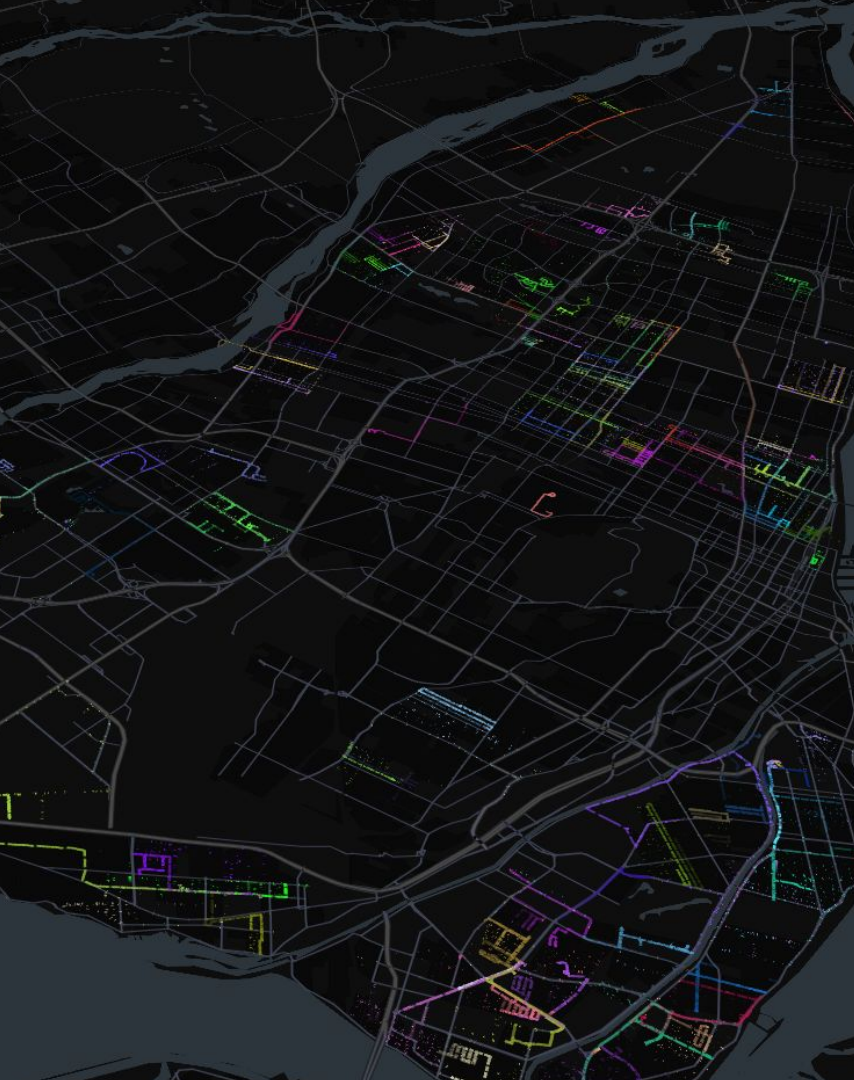




Les avantages et les inconvénients de l'architecture client-serveur

Inconvénients :

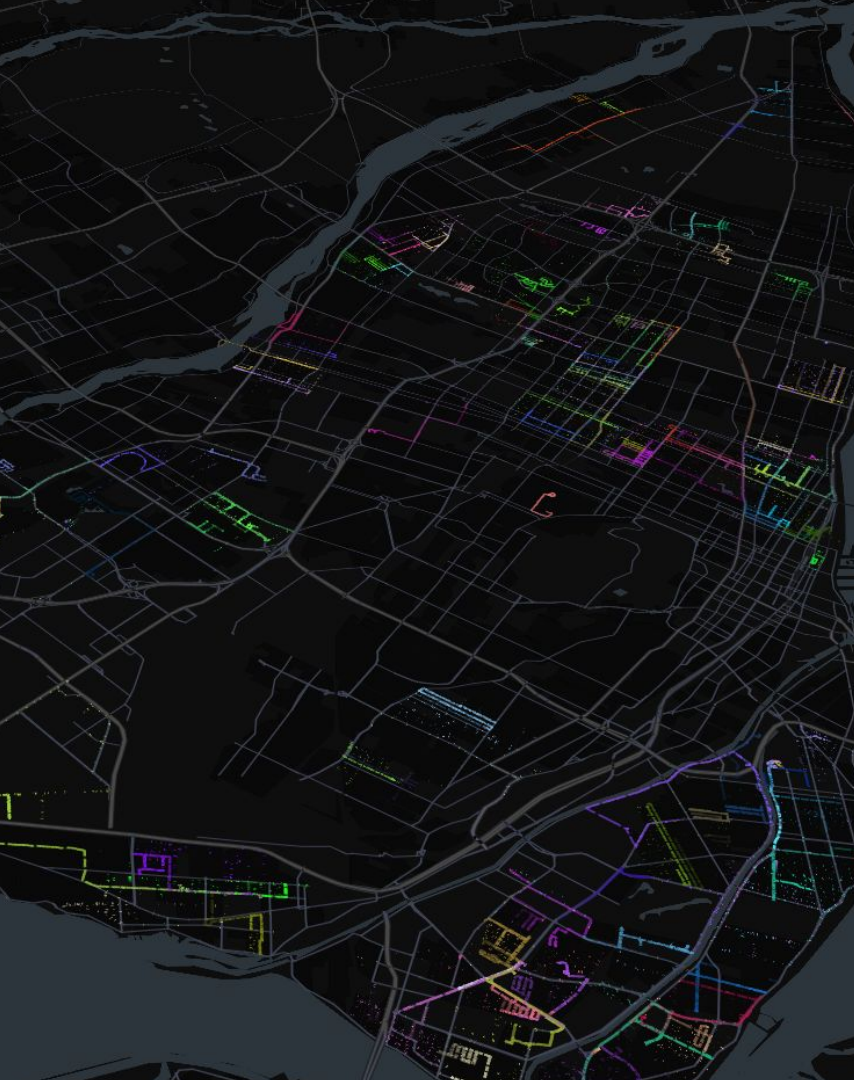
La dépendance entre le client et le serveur peut rendre l'application moins tolérante aux pannes, car la panne d'un composant peut entraîner l'indisponibilité de l'ensemble du système.



Les avantages et les inconvénients de l'architecture client-serveur

Inconvénients :

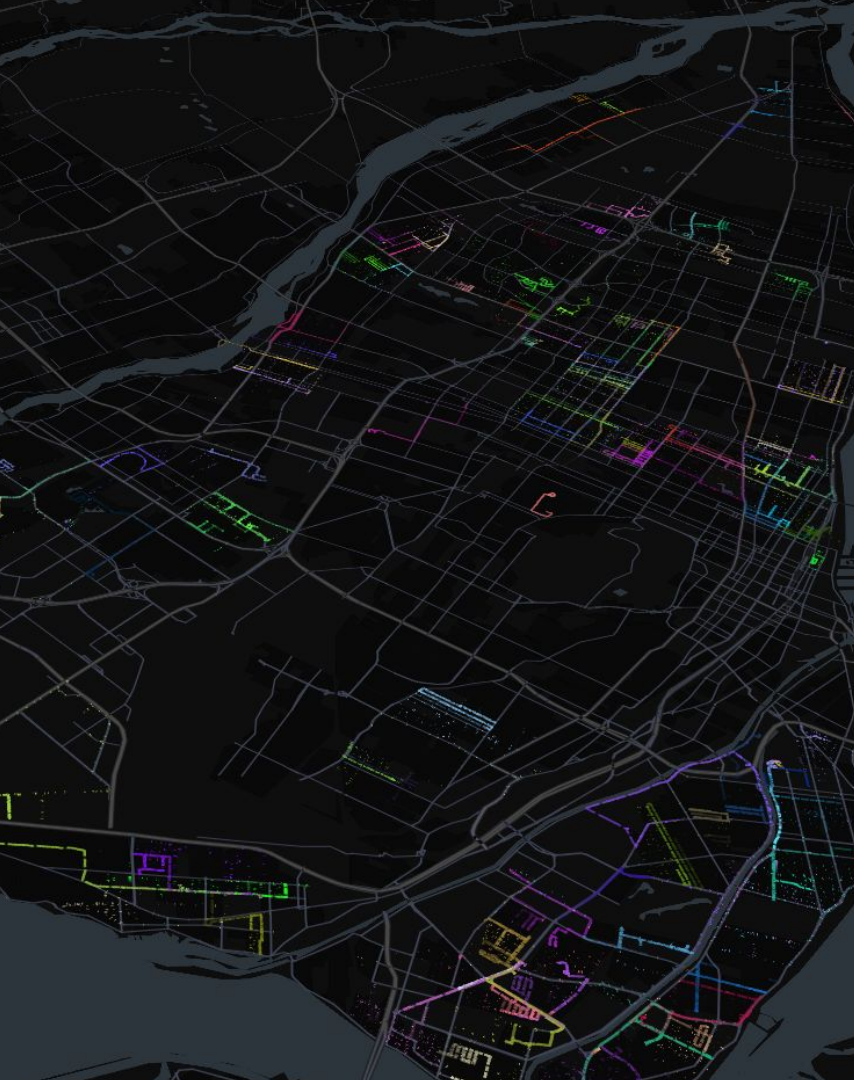
La complexité de l'architecture peut rendre la mise en place et la maintenance plus coûteuses.



Les avantages et les inconvénients de l'architecture client-serveur

Inconvénients :

Les temps de latence liés aux échanges entre le client et le serveur peuvent ralentir l'application, en particulier si le réseau est de mauvaise qualité.



Les avantages et les inconvénients de l'architecture client-serveur

Inconvénients :

Les problèmes de sécurité peuvent être plus complexes à gérer dans une architecture client-serveur, en particulier si les données sensibles sont stockées côté client.

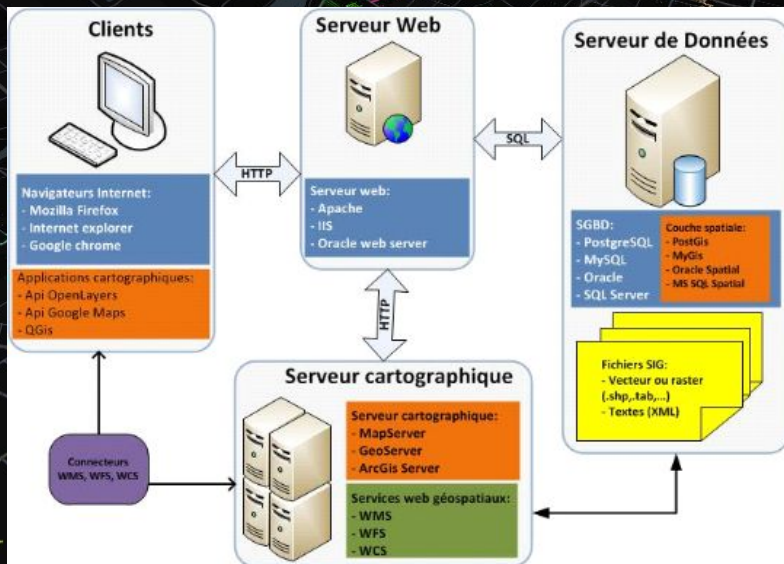
Principe d'architecture dorsale (backend)

Conteneurs et VMs

Les environnements de développement et de déploiement peuvent être complexes, avec de nombreuses dépendances et configurations requises pour faire fonctionner une application.

Anciennement il fallait un serveur ou une machine virtuelle déployée et administrée pour déployer des services dorsaux ou frontaux.

Aujourd'hui la tendance est à la **dématérialisation** !

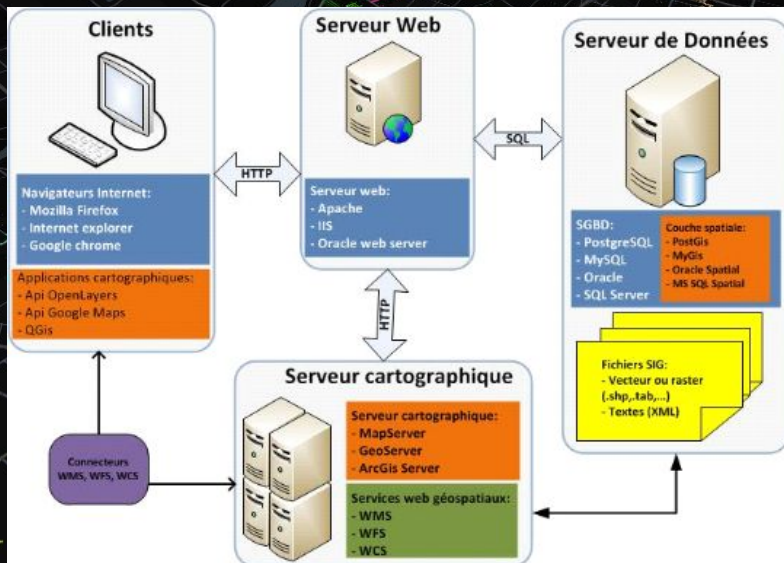


Principe d'architecture dorsale (backend)

Conteneurs et VMs

Les machines virtuelles permettent d'**isoler** complètement un environnement, ce qui évite les conflits de dépendances entre différentes applications et permet de travailler dans un environnement stable et **reproductible**.

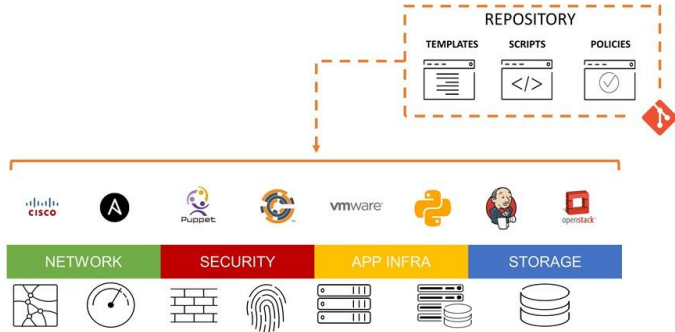
Imaginez un serveur où tout serait installé et une des composantes fait planter le serveur. Alors 100% des services seraient impactés.



Principe d'architecture dorsale (backend)

Conteneurs et VMs

INFRASTRUCTURE as CODE



Enfin, les conteneurs et les machines virtuelles sont des outils clés pour la gestion de l'infrastructure en tant que code (IaC), qui permet d'automatiser le déploiement et la gestion de l'infrastructure sous-jacente nécessaire pour exécuter les applications.

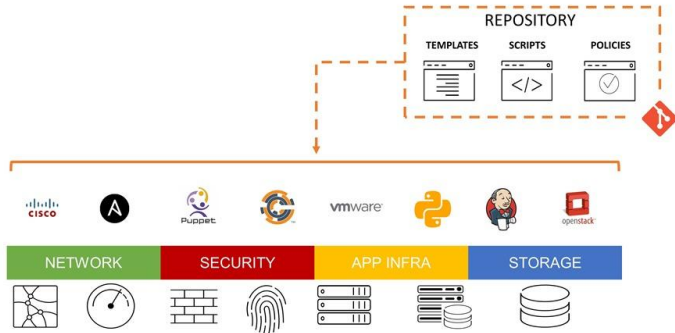
Infrastructure as Code (IaC)

[Infrastructure as code — Wikipédia](#)

Principe d'architecture dorsale (backend)

Conteneurs et VMs

INFRASTRUCTURE as CODE



Avantages :

- Réduction de coûts
- Réduction de risques
- Rapidité d'exécution
- Collaboration

Désavantages :

- Complexité
- Montée en compétence

[Infrastructure as code : l'essentiel en 8 minutes](#)

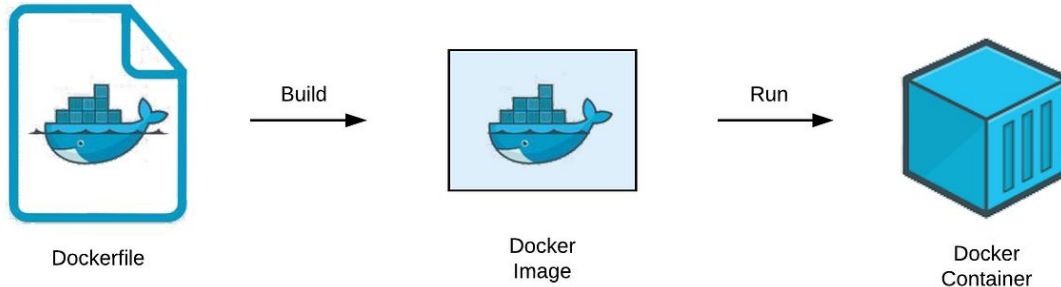
Rétro / Pause



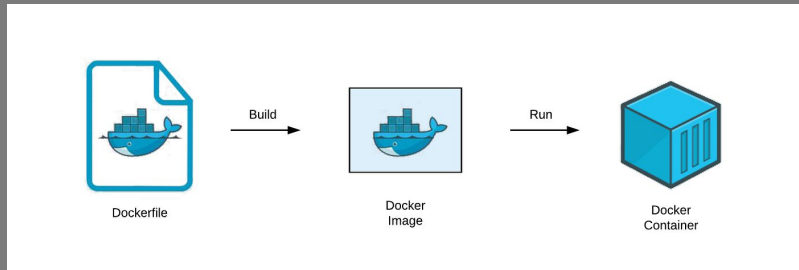
Principe d'architecture dorsale (backend)

Infrastructure as Code (IaC)

- Docker
- Docker-compose



Atelier d'introduction aux conteneurs



DOCKERFILE :

Fichier texte qui contient les instructions pour créer une image Docker personnalisée.

Il est utilisé pour automatiser la construction de l'image et la configuration de l'environnement de conteneurisation.

```
FROM postgres:latest

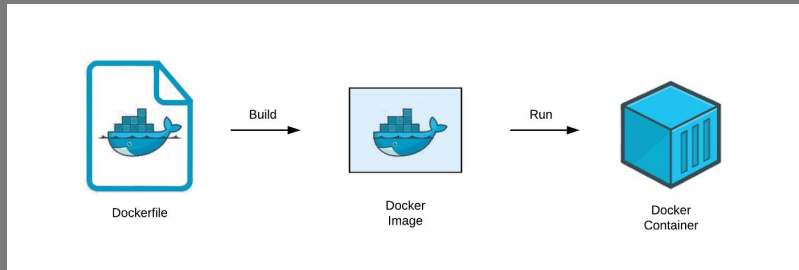
RUN apt-get update && apt-get install -y --no-install-recommends \
    postgresql-$PG_MAJOR-postgis-$POSTGIS_MAJOR \
    postgresql-$PG_MAJOR-postgis-$POSTGIS_MAJOR-scripts \
    && rm -rf /var/lib/apt/lists/*

RUN mkdir -p /docker-entrypoint-initdb.d

COPY ./initdb-postgis.sh /docker-entrypoint-initdb.d/postgis.sh
RUN chmod +x /docker-entrypoint-initdb.d/postgis.sh
```

[Dockerfile reference](#)

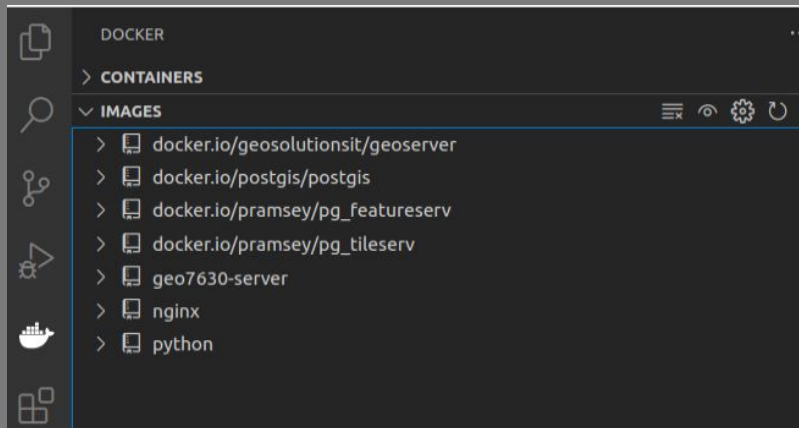
Atelier d'introduction aux conteneurs



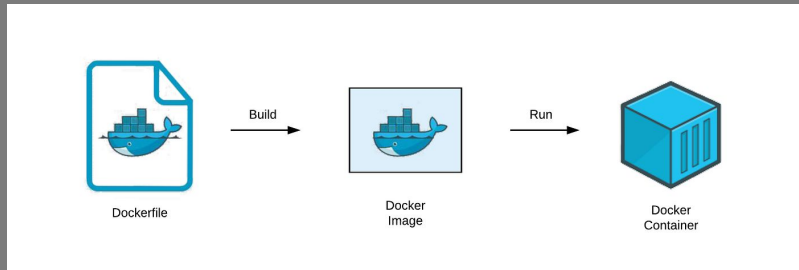
IMAGES :

Mini-systèmes d'exploitation stockés dans un fichier qui est construit spécifiquement en tenant compte de notre application.

Pensez-y comme un système d'exploitation personnalisé qui est stocké sur votre disque dur lorsque votre ordinateur est éteint.



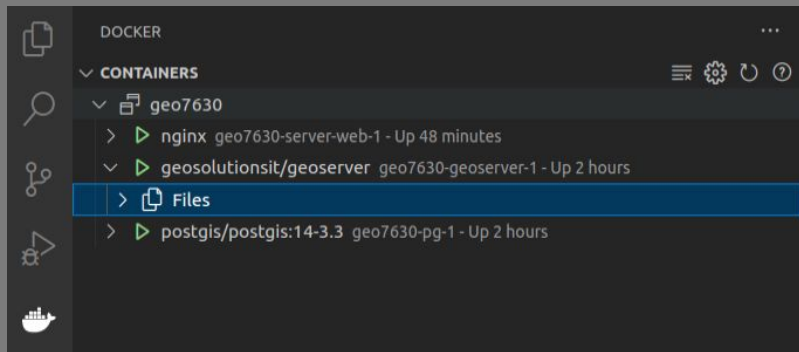
Atelier d'introduction aux conteneurs



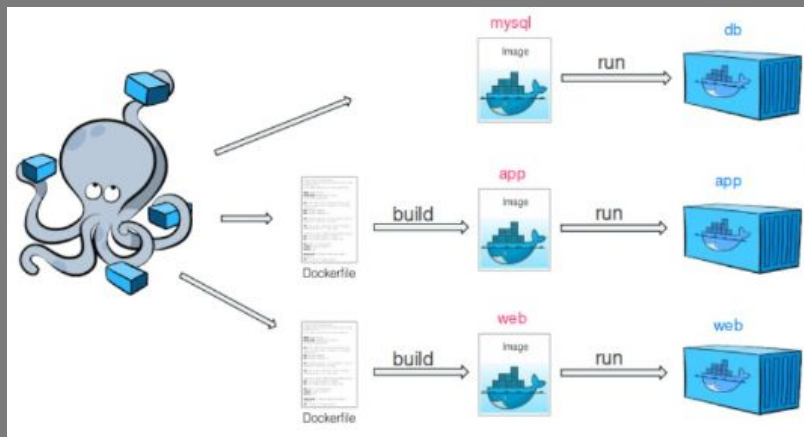
CONTAINERS :

Instances en cours d'exécution de votre image.

Imaginez un disque dur partagé ou un CD/DVD contenant un système d'exploitation pouvant fonctionner sur plusieurs machines. Les fichiers sur le disque sont appelés "image", et leur exécution sur une machine forme un "conteneur".



Atelier d'introduction aux conteneurs



Docker compose

Docker Compose est un outil qui permet de décrire, configurer et exécuter des applications multi-conteneurs.

Il utilise un fichier YAML pour définir les services de l'application et leurs dépendances, puis orchestre le déploiement et la gestion de ces conteneurs en un seul et même environnement.

[Docker Hub](#)

Atelier d'introduction aux conteneurs

```
docker-compose.yml
1  version: "3.4"
2  services:
3    pg:
4      image: "postgis/postgis:14-3.3"
5      ports:
6        - 8433:5432
7      environment:
8        - POSTGRES_PASSWORD=${SUPERUSERPASS}
9      volumes:
10       - ./create_postgres_stack.sql:/docker-entrypoint-initdb.d/create_postgres_stack.sql
11       - pgdata:/var/lib/postgresql/data
12     restart: always
13   geoserver:
14     image: geosolutionsit/geoserver
15     ports:
16       - 8080:8080
17     environment:
18       - INSTALL_EXTENSIONS=true
19       - STABLE_EXTENSIONS="PGRaster"
20     volumes:
21       - ./web.xml:/usr/local/tomcat/webapps/geoserver/WEB-INF/web.xml
22       - ./Laboratoires/Lab7/geoserverData:/var/geoserver/datadir
23     restart: always
24   server-web:
25     image: nginx
26     ports:
27       - 8000:80
28     volumes:
29       - ./usr/share/nginx/html
30 volumes:
31   pgdata:
32     name: installation_pgdata
```

Docker compose

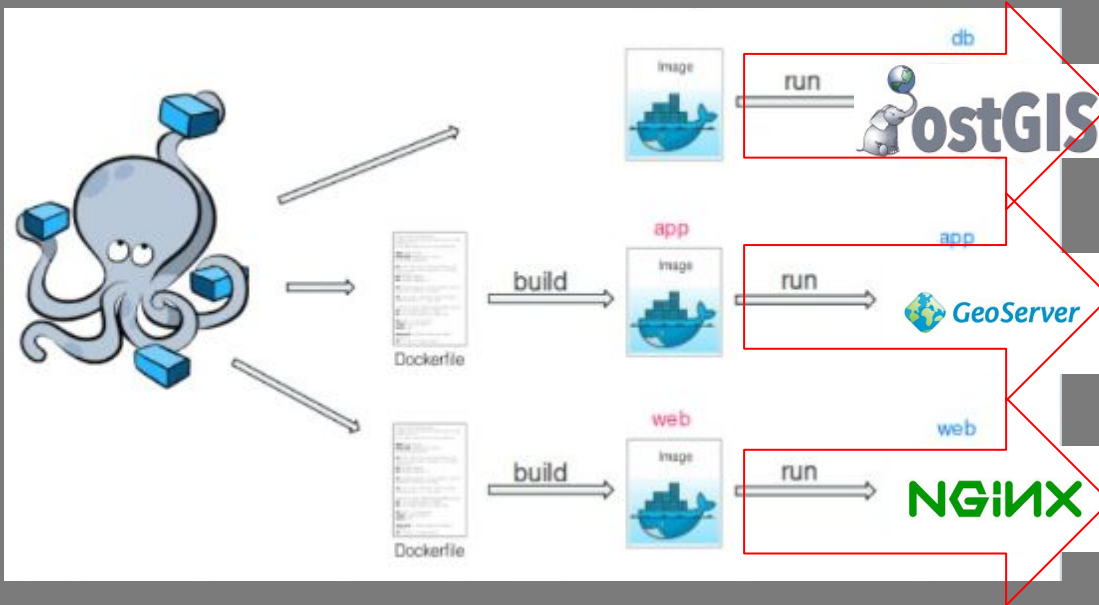
Docker Compose est un outil qui permet de décrire, configurer et exécuter des applications multi-conteneurs.

Il utilise un fichier YAML pour définir les services de l'application et leurs dépendances, puis orchestre le déploiement et la gestion de ces conteneurs en un seul et même environnement.

Atelier d'introduction aux conteneurs

```
docker-compose.yml

1 version: "3.4"
2 services:
3   pg:
4     image: "postgis/postgis:14-3.3"
5     ports:
6       - 8433:5432
7     environment:
8       - POSTGRES_PASSWORD=${SUPERUSERPASS}
9     volumes:
10      - ./create_postgres_stack.sql:/docker-entrypoint-initdb.d/create_postgres_stack.sql
11      - pgdata:/var/lib/postgresql/data
12     restart: always
13   geoserver:
14     image: geosolutionsit/geoserver
15     ports:
16       - 8080:8080
17     environment:
18       - INSTALL_EXTENSIONS=true
19       - STABLE_EXTENSIONS="PGRaster"
20     volumes:
21       - ./web.xml:/usr/local/tomcat/webapps/geoserver/WEB-INF/web.xml
22       - ./Laboratoires/Lab7/geoserverData:/var/geoserver/datadir
23     restart: always
24   server-web:
25     image: nginx
26     ports:
27       - 8000:80
28     volumes:
29       - ./:/usr/share/nginx/html
30   volumes:
31     pgdata:
32     name: installation_pgdata
```



Atelier d'introduction aux conteneurs

```
docker-compose.yml
1 version: "3.4"
2 services:
3   pg:
4     image: "postgis/postgis:14-3.3"
5     ports:
6       - 8433:5432
7     environment:
8       - POSTGRES_PASSWORD=${SUPERUSERPASS}
9     volumes:
10      - ./create_postgres_stack.sql:/docker-entrypoint-initdb.d/create_postgre
11      - pgdata:/var/lib/postgresql/data
12      restart: always
13
14   geoserver:
15     image: geosolutionsit/geoserver
16     ports:
17       - 8080:8080
18     environment:
19       - INSTALL_EXTENSIONS=true
20       - STABLE_EXTENSIONS="PGRaster"
21     volumes:
22       - ./web.xml:/usr/local/tomcat/webapps/geoserver/WEB-INF/web.xml
23       - ./Laboratoires/Lab7/geoserverData:/var/geoserver/datadir
24     restart: always
25
26   server-web:
27     image: nginx
28     ports:
29       - 8000:80
30     volumes:
31       - ./:/usr/share/nginx/html
32
33   volumes:
34     pgdata:
35       name: installation_pgdata
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

geo7630@GIS:~/geo7630\$ sudo docker compose up

DOCKER

CONTAINERS

- geo7630
 - nginx geo7630-server-web-1 - Up 48 minutes
 - geosolutionsit/geoserver geo7630-geoserver-1
 - Files
 - postgis/postgis:14-3.3 geo7630-pg-1 - Up 2 hou

Rétro / Pause





Revue des API géospatiales dorsales

osgeo/gdal

GDAL (Geospatial Data Abstraction Library) est une bibliothèque open source pour lire et écrire des données géospatiales dans différents formats.

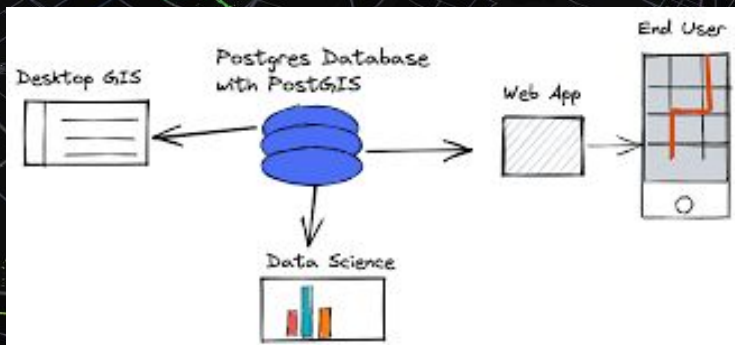
ogr2ogr est un outil de ligne de commande fourni avec GDAL, qui permet de convertir des données géospatiales entre différents formats, ainsi que d'effectuer des opérations de transformation, de projection et de filtrage des données.

[GDAL](#)



Crunchy Spatial

It's PostGIS, for the web



Revue des API géospatiales dorsales

PostGIS pour le Web (PostGIS FTW) est une famille croissante de micro-services spatiaux, dont `pg_featureserv` est l'un des composants.

Les applications centrées sur les bases de données ont naturellement une source centrale de coordination de l'état, la base de données, ce qui permet à des microservices indépendants de fournir un accès HTTP aux données avec une complexité minimale de middleware.

[PostgreSQL Blog | Crunchy Data](#)

[Paul Ramsey](#)

Paul Ramsey

Open source software
developer and information
technology professional.
Occasional blowhard.

[Home](#)

[About](#)

[Archive](#)

[Projects](#)

[Talks & Writing](#)

[GitHub](#)
[Twitter](#)
[Mastodon](#)
[LinkedIn](#)

Archive

[All](#) | [Technology](#) | [BC](#)

- 02 Feb 2023 » [When Proj Grid-Shifts Disappear](#)
postgis
- 22 Jan 2023 » [Mark Sondheim](#)
bc memoriam technology mentorship
- 23 Oct 2022 » [BC IT Outsourcing 2021/22](#)
esit ibm public accounts enterprise outsourcing bc it
- 17 Oct 2022 » [What you Need to Know](#)
hacking cloud technology
- 01 Oct 2022 » [Learning PostgreSQL Internals](#)
postgres hacking postgis
- 23 Jun 2022 » [Technology, Magic & PostgreSQL](#)
postgis
- 21 Jun 2022 » [Some More PostGIS Users](#)
postgis
- 20 Jun 2022 » [Some PostGIS Users](#)
postgis
- 13 Jun 2022 » [Who are the Biggest PostGIS Users?](#)
postgis
- 20 Apr 2022 » [Cloud Optimized Shape File](#)
shp cogs shapfile
- 01 Feb 2022 » [2022 Senate Mortality](#)
usa senate demographics actuarial
- 01 Jan 2022 » [Favourite Things](#)
music reading exercise
- 16 Dec 2021 » [PostGIS Nearest Neighbor Syntax](#)
postgis postgresql knn
- 28 Nov 2021 » [2018 BC Liberal Leadership Demographic Skew](#)
leadership liberal demographics party bc
- 25 Nov 2021 » [Open Source GIS Fights the Three-Horned Monster \[2002\]](#)
refractions open source gis history mapserver
- 17 Oct 2021 » [BC IT Outsourcing 2020/21](#)
esit ibm public accounts enterprise outsourcing bc it
- 01 Sep 2021 » [GeoMob Podcast - PostGIS](#)
postgis podcast open source
- 31 May 2021 » [PostGIS at 20, The Beginning](#)
refractions postgresql postgis history
- 06 May 2021 » [MapScaping Podcast - GDAL](#)
gdal podcast open source
- 04 May 2021 » [Spatial Indexes and Bad Queries](#)
postgis postgresql indexes
- 01 Apr 2021 » [Dumping a ByteA with psql](#)
postgis postgresql psql
- 16 Dec 2020 » [Waiting for PostGIS 3.1: GEOS 3.9](#)

Revue des API géospatiales dorsales

pg_tileserv fournit des tuiles MVT pour les clients interactifs et un rendu fluide

pg_featureserv fournit des services d'entités GeoJSON pour la lecture et l'écriture de données vectorielles et attributaires à partir de tables.

PostGIS pour le Web permet de mettre en place une architecture de services spatiaux de microservices sans état entourant un cluster de bases de données **PostgreSQL/PostGIS**, dans un environnement de conteneur standard, sur n'importe quelle plateforme cloud ou centre de données interne.

Revue des API géospatiales dorsales

CrunchyData/ **pg_tileserv**



A very thin PostGIS-only tile server in Go. Takes in HTTP tile requests, executes SQL, returns MVT tiles.

28 Contributors 25 Issues 639 Stars 121 Forks

CrunchyData/ **pg_featureserv**



Lightweight RESTful Geospatial Feature Server for PostGIS in Go

15 Contributors 33 Issues 358 Stars 71 Forks

pg_tileserv fournit des tuiles MVT pour les clients interactifs et un rendu fluide

pg_featureserv fournit des services d'entités GeoJSON pour la lecture et l'écriture de données vectorielles et attributaires à partir de tables.

PostGIS pour le Web permet de mettre en place une architecture de services spatiaux de microservices sans état entourant un cluster de bases de données PostgreSQL/PostGIS, dans un environnement de conteneur standard, sur n'importe quelle plateforme cloud ou centre de données interne.

[GitHub - CrunchyData/pg_tileserv](https://github.com/ CrunchyData/pg_tileserv)

[GitHub - CrunchyData/pg_featureserv](https://github.com/ CrunchyData/pg_featureserv)

Atelier Mise en place de composants backend pour le webmapping avancé



- configuration du docker compose
- ogr
- pg-tileserv
- interface pgtileserv
- pg-featureserv
- interface pgfeatureserv
- serveur local

Atelier docker backend

Configuration du docker compose

version: "3.4"

services:

mon Nom De Container :

image: le nom de l'image sur docker hub (ou autre docker registry)

ports:

- out:in

environment:

- une variable d'environnement au besoin

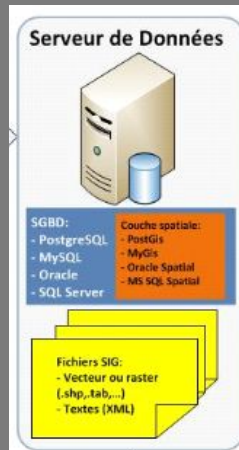
volumes:

- un dossier ou document que je veux copier dans le container: destination

volumes:

nom du volume persistant:

name: alias du volume



Atelier docker backend

Configuration du docker compose

version: "3.4"

services:

pg_lab8:

image: "postgis/postgis:14-3.3"

ports:

- 8434:5432

environment:

- POSTGRES_PASSWORD=password

volumes:

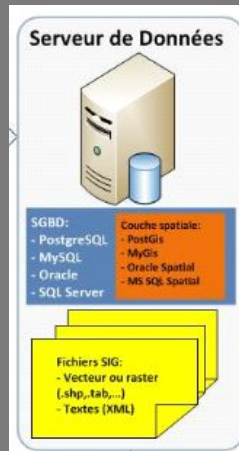
- ./create_postgres_stack.sql:/docker-entrypoint-initdb.d/create_postgres_stack.sql

- pgdata:/var/lib/postgresql/data

volumes:

pgdata:

name: installation_pgdata

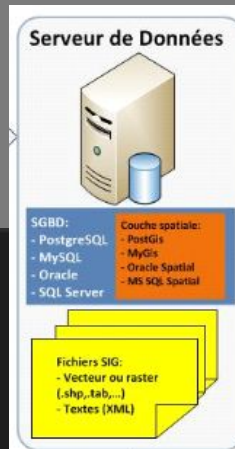


Atelier docker backend

Configuration du docker compose

Laboratoires > Lab8 >  docker-compose.yml

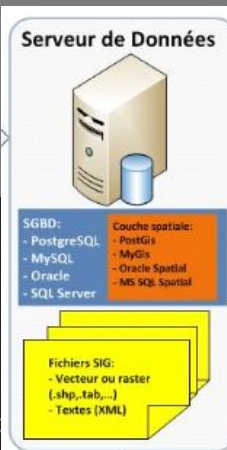
```
1  version: "3.4"
2  services:
3    pg_lab8:
4      image: "postgis/postgis:14-3.3"
5      ports:
6        - 8434:5432
7      environment:
8        - POSTGRES_PASSWORD=password
9      volumes:
10       - ./create_postgres_stack.sql:/docker-entrypoint-initdb.d/create_postgres_stack.sql
11       - pgdata:/var/lib/postgresql/data
12      restart: always
13  volumes:
14    pgdata:
15      name: installation_pgdata
16
```



Atelier docker backend

docker compose up

```
lab8-pg_lab8-1 | 2023-03-26 19:14:57.453 UTC [1] LOG: listening on IPv6 address "::", port 5432
lab8-pg_lab8-1 | 2023-03-26 19:14:57.456 UTC [1] LOG: listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
lab8-pg_lab8-1 | 2023-03-26 19:14:57.462 UTC [27] LOG: database system was shut down at 2023-03-26 19:14:14 UTC
lab8-pg_lab8-1 | 2023-03-26 19:14:57.467 UTC [1] LOG: database system is ready to accept connections
lab8-pg_tileserv-1 | time="2023-03-26T19:14:57Z" level=info msg="pg_tileserv latest"
lab8-pg_tileserv-1 | time="2023-03-26T19:14:57Z" level=info msg="Run with --help parameter for commandline options"
lab8-pg_tileserv-1 | time="2023-03-26T19:14:57Z" level=info msg="Using database connection info from environment variable DATABASE_URL"
lab8-pg_tileserv-1 | time="2023-03-26T19:14:57Z" level=info msg="Serving HTTP at http://0.0.0.0:7800/"
lab8-pg_tileserv-1 | time="2023-03-26T19:14:57Z" level=info msg="Serving HTTPS at http://0.0.0.0:7801/"
lab8-pg_tileserv-1 | time="2023-03-26T19:14:57Z" level=info msg="Using CoordinateSystem.SRID 3857 with bounds [-2.00375083427892e+07, -2.00375083427892e+07, 2.00375083427892e+07, 2.00375083427892e+07]"
lab8-pg_tileserv-1 | time="2023-03-26T19:14:57Z" level=info msg="Connected as 'admin_geo' to 'geo7630' @ 'pg_lab8'"
lab8-pg_featureserv-1 | time="2023-03-26T19:14:57Z" level=info msg="---- pg_featureserv - Version 1.3 -----\n"
lab8-pg_featureserv-1 | time="2023-03-26T19:14:57Z" level=info msg="Using config file: "
lab8-pg_featureserv-1 | time="2023-03-26T19:14:57Z" level=info msg="Using database connection info from environment variable DATABASE_URL"
lab8-pg_featureserv-1 | time="2023-03-26T19:14:57Z" level=info msg="Connected as admin_geo to geo7630 @ pg_lab8"
lab8-pg_featureserv-1 | time="2023-03-26T19:14:57Z" level=info msg="Serving HTTP at http://0.0.0.0:9000"
lab8-pg_featureserv-1 | time="2023-03-26T19:14:57Z" level=info msg="CORS Allowed Origins: *\n"
lab8-pg_featureserv-1 | time="2023-03-26T19:14:57Z" level=info msg="==== Service: pg-featureserv ====\n"
lab8-pg_tileserv-1 | time="2023-03-26T19:14:58Z" level=info msg="Serving HTTP at 0.0.0.0:7800"
```



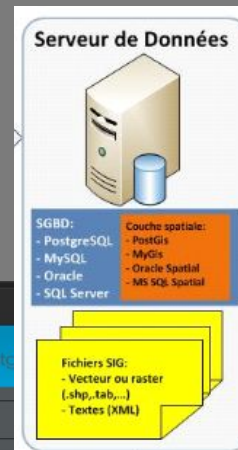
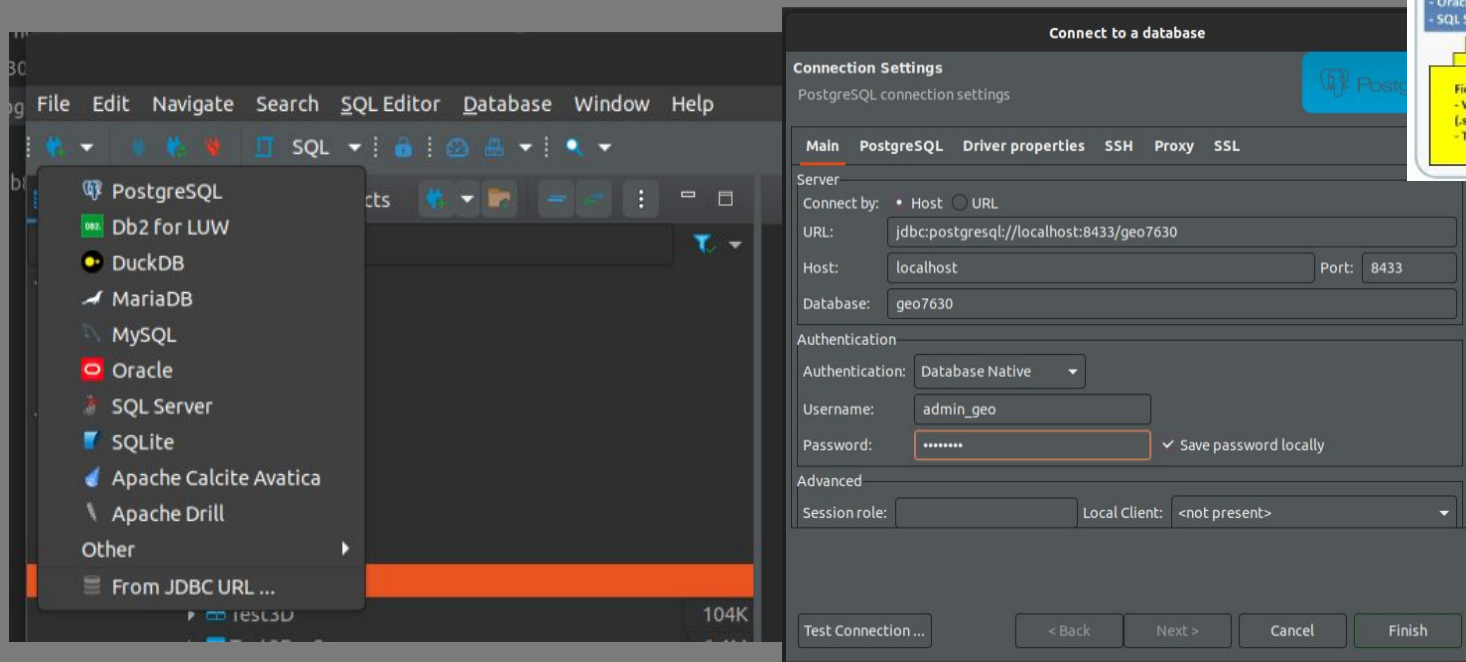
docker compose up -d

```
geo7630@GIS:~/geo7630/Laboratoires/Lab8$ docker compose up -d
[+] Running 2/2
  :: Volume "pg_lab8" Created
  :: Container lab8-pg_lab8-1 Started
```

Atelier docker backend

Configuration du docker compose

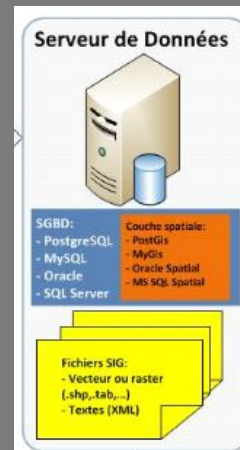
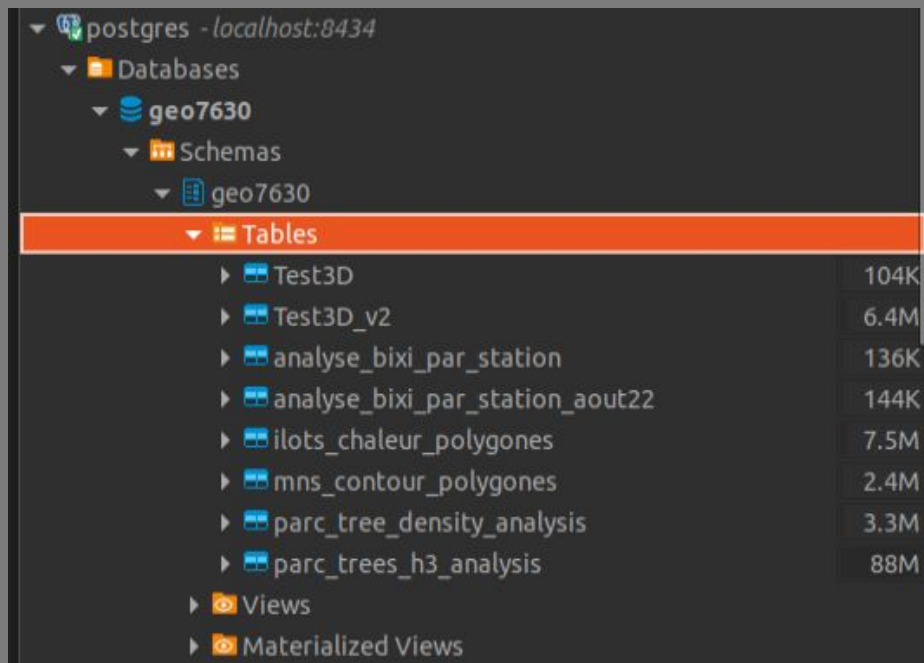
Création de la bd dans DBeaver



Atelier docker backend

Configuration du docker compose

Création de la bd dans DBeaver



Rétro / Pause



Atelier docker backend

Création workflow ETL avec un container

```
Laboratoires > Lab8 > docker-compose.yml
1  version: "3.4"
2  services:|
3      # pg_lab8:
4      #   image: "postgis/postgis:14-3.3"
5      #   ports:
6      #     - 8434:5432
7      #   environment:
8      #     - POSTGRES_PASSWORD=password
9      #   volumes:
10     #     - ./create_postgres_stack.sql:/docker-entrypoint-initdb.d/create_postgres_stack.sql
11     #     - pgdata:/var/lib/postgresql/data
12     #   restart: always
13   ogr:
14     image: ghcr.io/osgeo/gdal:alpine-small-latest
15     volumes:
16     - ./data:/data
```

ogr:

image: *ghcr.io/osgeo/gdal:alpine-small-latest*

volumes:

- ./data:/data

Atelier docker backend

Création workflow ETL avec un container

```
docker compose run --rm ogr ogr2ogr -f "FORMAT DE TRANSFORMATION" output input
```

```
docker compose run --rm ogr ogr2ogr -f "ESRI Shapefile" \  
/data/a_garage.shp \  
/data/garage.geojson
```

```
• geo7630@GIS:~/geo7630/Laboratoires/Lab8$ docker compose run --rm ogr ogr2ogr -f PostgreSQL -lco GEOMETRY_NAME=geom -lco FID=gid -lco SPATIAL_\  
nlt PROMOTE_TO_MULTI -nln geo7630.garage_lab8 -t_srs EPSG:3857 -overwrite PG:"dbname='geo7630' host='172.19.0.1' port='8434' user='admin_g\  
'password'" ./data/garage.geojson  
Warning 1: Several features with id = 46 have been found. Altering it to be unique. This warning will not be emitted anymore for this layer  
○ geo7630@GIS:~/geo7630/Laboratoires/Lab8$
```


Atelier docker backend

Création workflow ETL avec un container

▼	Lab8	●
▼	data	●
≡	a_garage.dbf	U
≡	a_garage.prj	U
≡	a_garage.shp	U
≡	a_garage.shx	U
{ }	garage.geojson	A
🔥	docker-compose.yml	M
ⓘ	README.md	U

Atelier docker backend

Import de données dans la BD avec un container ETL

```
docker compose run --rm ogr ogr2ogr -f PostgreSQL \  
PG:"dbname='geo7630' \  
host='172.19.0.1' \  
port='8433' \  
user='admin_geo' \  
password='password'" \  
./data/garage.geojson
```

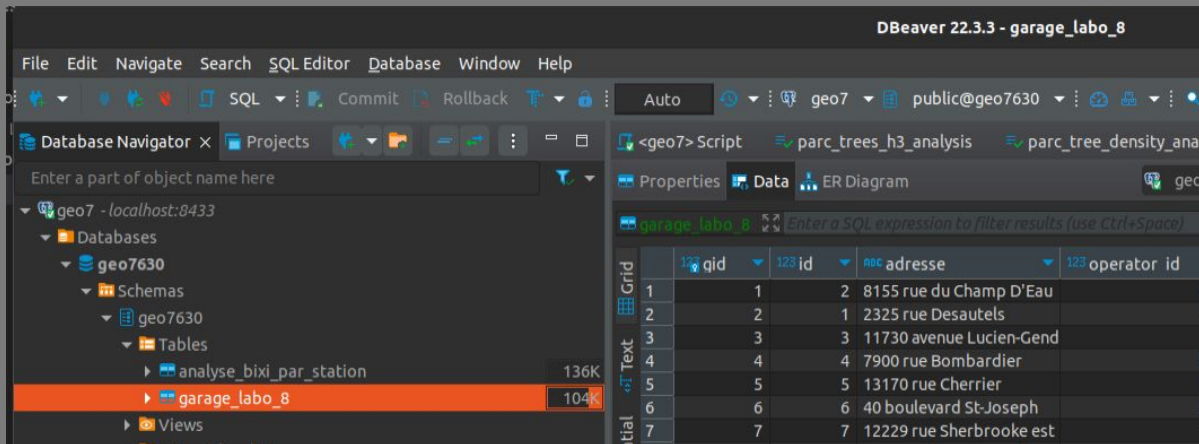
Atelier docker backend

Import de données dans la BD avec un container ETL

```
docker compose run --rm ogr ogr2ogr -f PostgreSQL \  
-lco GEOMETRY_NAME=geom \  
-lco FID=gid \  
-lco SPATIAL_INDEX=GIST \  
-nlt PROMOTE_TO_MULTI \  
-nln geo7630.garage_labo_8 \  
-t_srs EPSG:3857 \  
-overwrite \  
PG:"dbname='geo7630' host='172.19.0.1' port='8434' user='admin_geo' password='password'" \  
./data/garage.geojson
```

Atelier docker backend

Import de données dans la BD avec un container ETL



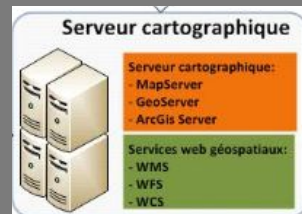
The screenshot shows the DBeaver 22.3.3 interface. The title bar indicates the connection is 'garage_labo_8'. The menu bar includes File, Edit, Navigate, Search, SQL Editor, Database, Window, and Help. The toolbar contains icons for file operations, SQL execution, and database management. The Database Navigator on the left shows a tree structure: geo7 - localhost:8433 > Databases > geo7630 > Schemas > geo7630 > Tables. The 'garage_labo_8' table is selected and highlighted in orange, with a size of 104K. The main panel displays the table data in a grid view. The table has columns: gid, id, adresse, and operator_id. The data is as follows:

	gid	id	adresse	operator_id
1	1	2	8155 rue du Champ D'Eau	
2	2	1	2325 rue Desautels	
3	3	3	11730 avenue Lucien-Gend	
4	4	4	7900 rue Bombardier	
5	5	5	13170 rue Cherrier	
6	6	6	40 boulevard St-Joseph	
7	7	7	12229 rue Sherbrooke est	

Rétro / Pause

Atelier docker backend

Création d'un microservice de diffusion de tuiles vectorielles



```
Laboratoires > Lab8 > docker-compose.yml
1  version: "3.4"
2  services:
3    # pg_lab8:
4    #   image: "postgis/postgis:14-3.3"
5    #   ports:
6    #     - 8434:5432
7    #   environment:
8    #     - POSTGRES_PASSWORD=password
9    #   volumes:
10   #     - ./create_postgres_stack.sql:/docker-entrypoint-initdb.d/create_postgres_stack.sql
11   #     - pgdata:/var/lib/postgresql/data
12   #   restart: always
13   # ogr:
14   #   image: ghcr.io/osgeo/gdal:alpine-small-latest
15   #   volumes:
16   #     - ./data:/data
17   pg_tileserv:
18     image: "pramsey/pg_tileserv:latest"
19     ports:
20       - 8801:7800
21     environment:
22       - DATABASE_URL=postgresql://admin_geo:password@pg_lab8/geo7630
23     volumes:
24       - ./pg_tileserv.toml:/etc/pg_tileserv.toml
25     depends_on:
26       - "pg_lab8"
27     restart: always
28 volumes:
29   pgdata:
30     name: installation_pgdata
31
```

pg_tileserv:

image: "pramsey/pg_tileserv:latest"

ports:

- 8801:7800

environment:

- DATABASE_URL=postgresql://admin_geo:password@pg_lab8/geo7630

volumes:

- ./pg_tileserv.toml:/etc/pg_tileserv.toml

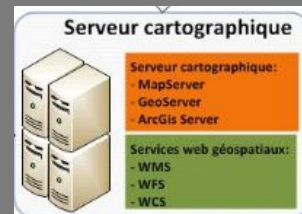
depends_on:

- "pg_lab8"

restart: always

Atelier docker backend

Création d'un microservice de diffusion de tuiles vectorielles



pg_tileserv

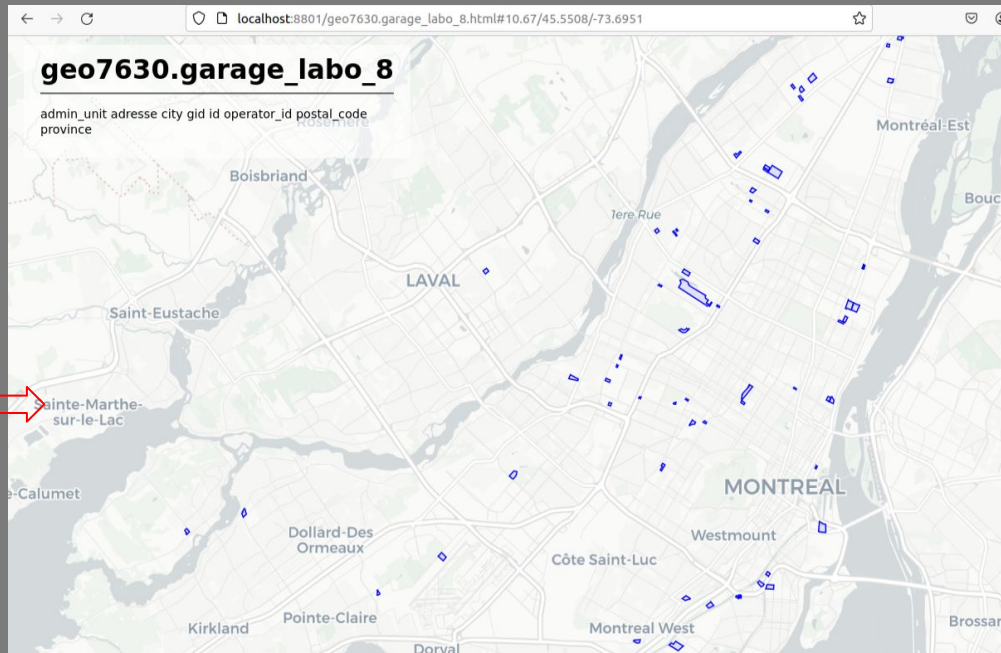
Service Metadata

- [index.json](#) for layer list

Table Layers

- geo7630.Test3D ([preview](#) | [json](#))
- geo7630.Test3D_v2 ([preview](#) | [json](#))
- geo7630.analyse_bixi_par_station ([preview](#) | [json](#))
- geo7630.analyse_bixi_par_station_aout22 ([preview](#) | [json](#))
- geo7630.garage_lab0_8 ([preview](#) | [json](#))
- geo7630.ilots_chaleur_polygones ([preview](#) | [json](#))
- geo7630.mns_contour_polygones ([preview](#) | [json](#))
- geo7630.parc_tree_density_analysis ([preview](#) | [json](#))
- geo7630.parc_trees_h3_analysis ([preview](#) | [json](#))
- geo7630.tset ([preview](#) | [json](#))
- public.parc_tree_density_analysis ([preview](#) | [json](#))
- public.parc_trees_h3_analysis ([preview](#) | [json](#))
- public.test_lidar ([preview](#) | [json](#))

Function Layers



Rétro / Pause



Atelier docker backend

Création d'un microservice de diffusion OGC-WFS

```
Laboratoires > Lab8 > docker-compose.yml
1  version: "3.4"
2  services:
3    # pg_lab8:
4    #   image: "postgis/postgis:14-3.3"
5    #   ports:
6    #     - 8434:5432
7    #   environment:
8    #     - POSTGRES_PASSWORD=password
9    #   volumes:
10   #     - ./create_postgres_stack.sql:/docker-entrypoint-initdb.d/create_postgres_stack.sql
11   #     - pgdata:/var/lib/postgresql/data
12   #     restart: always
13   # ogr:
14   #   image: ghcr.io/osgeo/gdal:alpine-small-latest
15   #   volumes:
16   #     - ./data:/data
17   # pg_tileserv:
18   #   image: "pramsey/pg_tileserv:latest"
19   #   ports:
20   #     - 8801:7800
21   #   environment:
22   #     - DATABASE_URL=postgresql://admin_geo:password@pg_lab8/geo7630
23   #   volumes:
24   #     - ./pg_tileserv.toml:/etc/pg_tileserv.toml
25   #   depends_on:
26   #     - "pg_lab8"
27   #   restart: always
28   pg_featureserv:
29     image: "pramsey/pg_featureserv:latest"
30     ports:
31       - 9000:9000
32     environment:
33       - DATABASE_URL=postgresql://admin_geo:password@pg_lab8/geo7630
34     depends_on:
35       - "pg_lab8"
36     restart: always
37   volumes:
38     pgdata:
39       name: installation_pgdata
40       external: true
41
```

pg_featureserv:

image: "pramsey/pg_featureserv:latest"

ports:

- 9000:9000

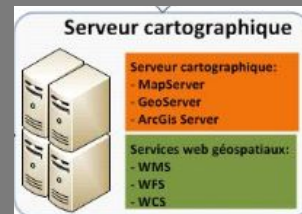
environment:

- DATABASE_URL=postgresql://admin_geo:password@pg_lab8/geo7630

depends_on:

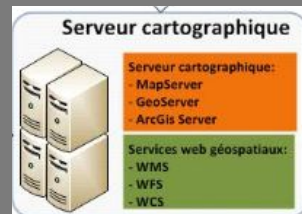
- "pg_lab8"

restart: always



Atelier docker backend

Création d'un microservice de diffusion OGC-WFS



localhost:9000/collections.html

pg-featureserv pg-featureserv - Version 1.3

Home / Collections JSON

Feature Collections

geo7630.Test3D JSON View
Data for table geo7630.Test3D

geo7630.Test3D_v2 JSON View
Data for table geo7630.Test3D_v2

geo7630.analyse_bixi_par_station JSON View
Data for table geo7630.analyse_bixi_par_station

geo7630.analyse_bixi_par_station_aout22 JSON View
Data for table geo7630.analyse_bixi_par_station_aout22

geo7630.garage_labo_8 JSON View
Data for table geo7630.garage_labo_8

geo7630.ilots_chaleur_polygones JSON View
Data for table geo7630.ilots_chaleur_polygones

geo7630.mns_contour_polygones JSON View
Data for table geo7630.mns_contour_polygones

geo7630.parc_tree_density_analysis JSON View
Data for table geo7630.parc_tree_density_analysis

geo7630.parc_trees_h3_analysis JSON View
Data for table geo7630.parc_trees_h3_analysis

geo7630.tset JSON View
Data for table geo7630.tset

public.parc_tree_density_analysis JSON View
Data for table public.parc_tree_density_analysis

public.parc_trees_h3_analysis JSON View
Data for table public.parc_trees_h3_analysis

public.test_lidar JSON View
Data for table public.test_lidar

pg-featureserv pg-featureserv - Version 1.3

Home / Collections / geo7630.garage_labo_8 / Items JSON

geo7630.garage_labo_8
Feature count: 65

Requery

Parameters

Limit:

BBox: ☐

Rétro / Pause

Atelier docker backend

Création d'un serveur web

```
Laboratoires > Lab8 > docker-compose.yml
19   ports:
20     - 8801:7800
21   environment:
22     - DATABASE_URL=postgresql://admin_geo:password@pg_lab8/geo7630
23   depends_on:
24     - "pg_lab8"
25   restart: always
26   pg_featureserv:
27     image: "pramsey/pg_featureserv:latest"
28     ports:
29       - 9000:9000
30     environment:
31       - DATABASE_URL=postgresql://admin_geo:password@pg_lab8/geo7630
32     depends_on:
33       - "pg_lab8"
34     restart: always
35   server-web:
36     image: nginx
37     ports:
38       - 8000:80
39     volumes:
40       - ./usr/share/nginx/html
41   volumes:
42     pgdata:
43       name: installation_pgdata
44       external: true
45
```

server-web:

image: nginx

ports:

- 8000:80

volumes:

- ./usr/share/nginx/html



Atelier docker backend

Création d'un serveur web

<http://localhost:8000/app/>

