Formation CQP Bloc 2 - U5 - S18

Introduction à la plateforme Open Source Kalisio

https://kalisio.com

Notre identité

- **KALISIO** est une SAS créée le 14 septembre 2017 dans l'Aude, Occitanie

- Les membres fondateurs de **KALISIO** disposent d'une **forte expérience technologique** et **fonctionnelle** dans le développement d'**applications géospatiales** pour des marchés aussi variés que l'aéronautique, l'espace, la défense, la sécurité…

- Aujourd'hui nous sommes désireux de mettre à profit nos compétences pour créer et promouvoir des solutions reposant sur des technologies géospatiales de demain. Nous:
  - Développons, sponsorisons et utilisons des solutions libres (**Open Source**, **Open Data**).
  - Accompagnons nos clients et partenaires dans un esprit d'**innovation ouverte.**
  - Participons à la **numérisation** des territoires et luttons contre la fracture numérique.

Notre offre

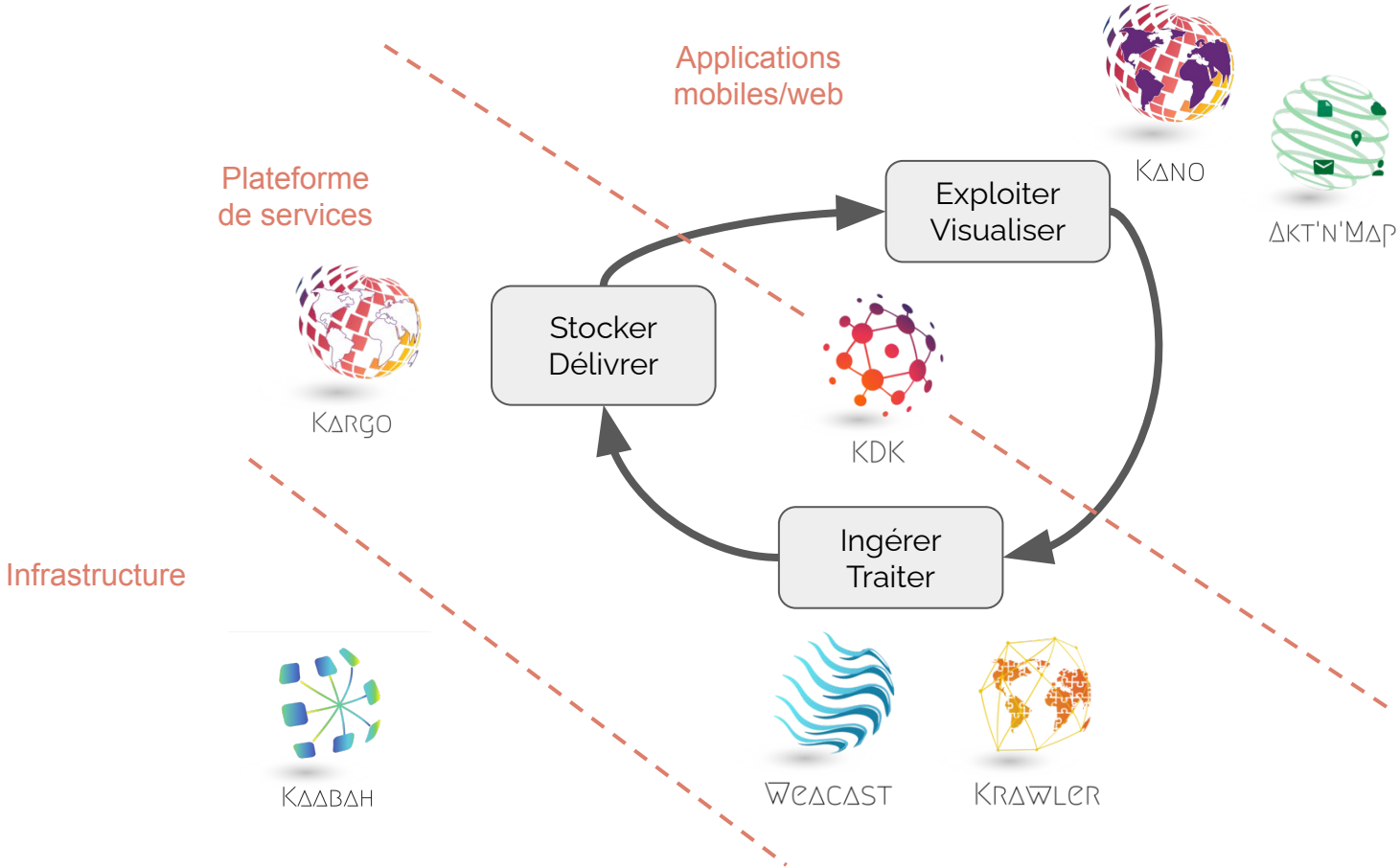
Exploiter la donnée géospatiale de bout en bout pour aider les organisations à maîtriser leur environnement en temps-réel

Superviser des assets géolocalisés (biens & personnes)
Visualiser les données pour prendre des décisions
Exploiter les données dans des processus métier

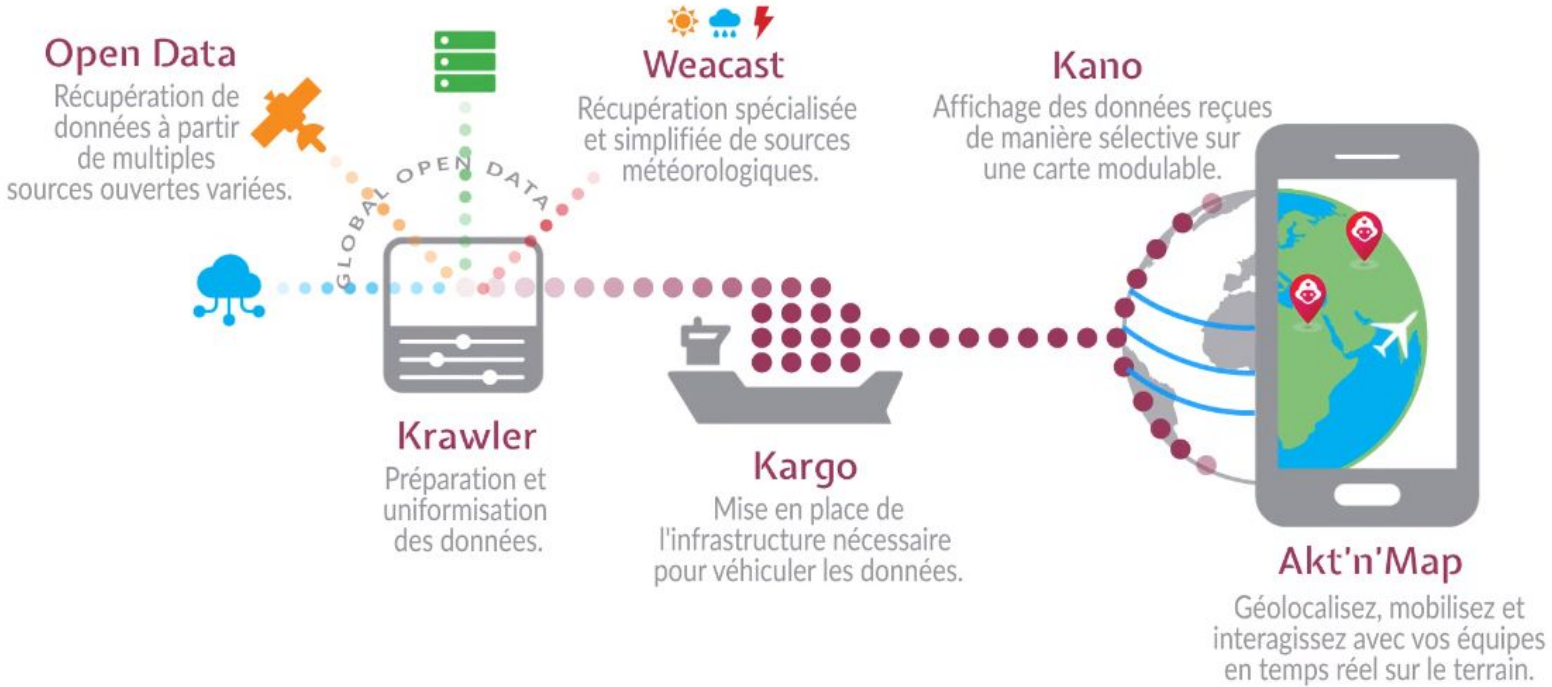
Rendre les données exploitables aux logiciels métier
Rendre la donnée accessible dans des logiciels tiers
Ingérer, traiter et stocker les données de façon adaptée aux usages

Assurer une haute disponibilité des infrastructures
Superviser et maintenir des infrastructures
Concevoir et provisionner des infrastructures

Notre écosystème de solutions



Notre écosystème de solutions



Nos produits - Akt'n'Map

Akt'n'Map

DISPONIBLE SUR Google play

DISPONIBLE SUR App Store

Géolocalisez, mobilisez et communiquez instantanément avec vos équipes sur le terrain.

Akt'n'Map est une application opérée par Kalisio et paramétrables pour vos besoins.

Géolocalisez vos équipes à tout moment, et notifiez-les en temps réel pour les mobiliser au plus vite sur un secteur d'intervention.

Communiquez en temps réel pour permettre d'établir un diagnostic et un suivi précis de la situation sur le terrain.

- Géolocalisez vos équipes en temps réel
- Notifiez et partagez des informations
- Communiquez en temps réel
- Centralisez l'information
- Structurez votre organisation
- Facilitez la collaboration entre équipes

Kalisio

Nos produits - Kano

Kano

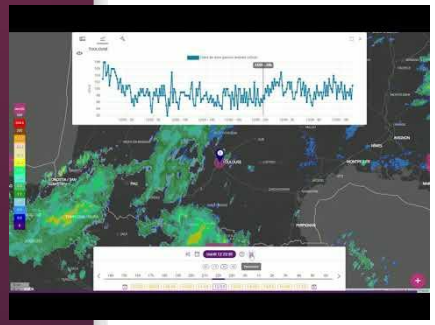
Kano est un puissant module de visualisation de données, supportant de multiples couches et fonds cartographiques 2D et 3D.

Il prend en charge une grande variété de données géolocalisées, mesurées et météorologiques : sondes de mesure, éléments météorologiques (température, précipitations, vent, courants marins...), véhicules, personnes, etc.

Tirez enfin le meilleur parti de vos différents jeux de données et confrontez-les dans un contexte spécifique grâce à cet outil modulable et adaptable à tout environnement.

Kano fait partie intégrante d'Akt'n'Map.

- Fonds de cartes personnalisés
- Vue cartographique multi-couches
- Affichage topographique et 3D
- API d'intégration
- Données géolocalisées, métriques, météorologiques
- Données en temps réel, archivées et prévisionnelles
- Sources et formats multiples
- Support des standards OGC
- Catalogue de données



Nos autres briques technologiques

KDK

Kit de développement d'applications Web et Mobile sur lequel sont basées les solutions **Kano** et **Ak'n'Map**

+

Krawler

Outil d'extraction , transformation et chargement (ETL) de données géospatiales orienté pour le développement de services web.

+

Weacast

Weacast (pour "Weather forecast") est spécialisé dans le traitement et la restitution de données de prévisions météorologiques (GFS, AROME, ARPEGE)

+

Kargo

Solution de déploiement de plateformes de services géospatiaux. Principalement pensée pour le cloud (AWS, OVH, Scaleway)

+

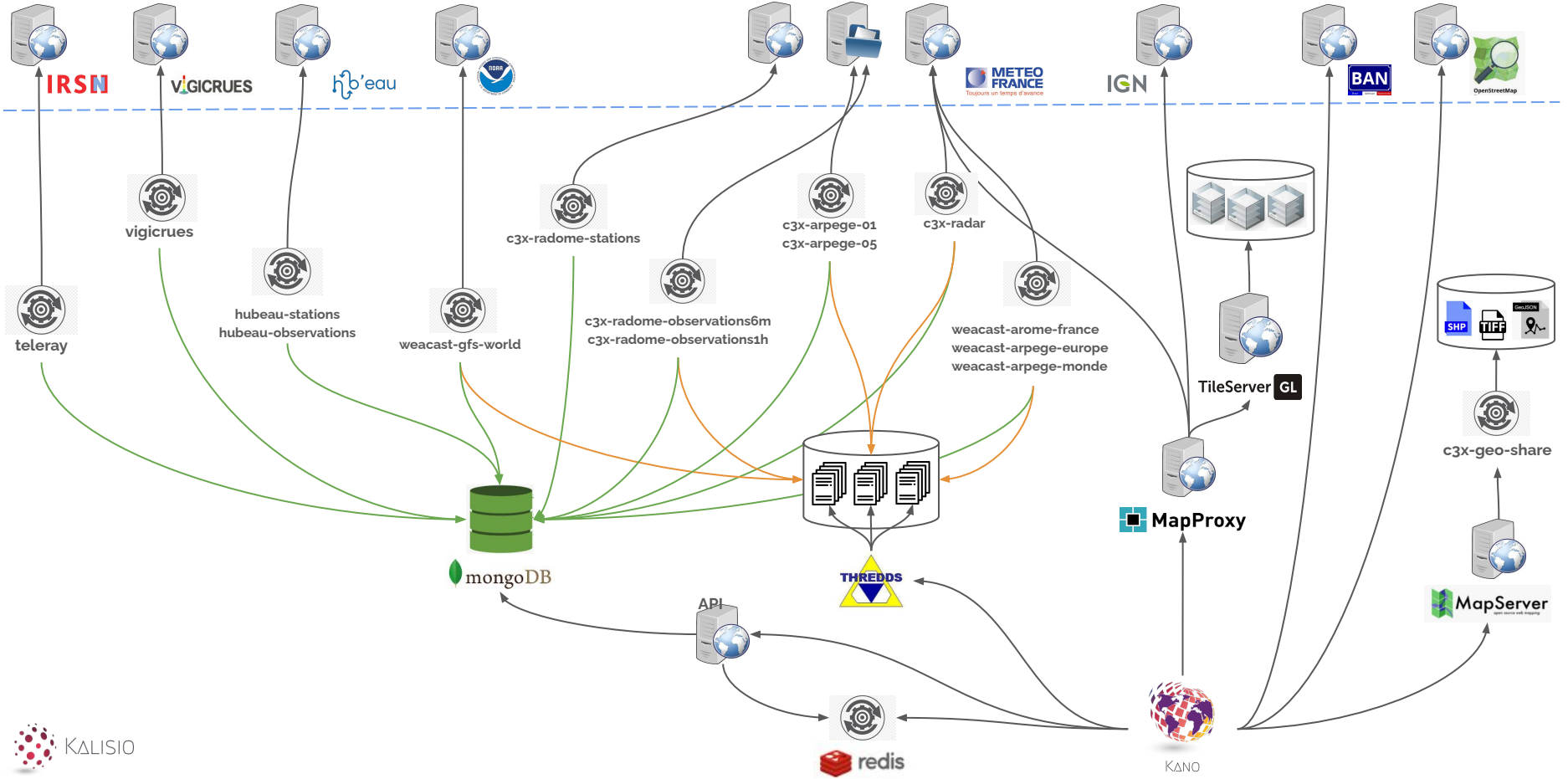
Quelques cas d'usage

- Pour le compte d'**AIRBUS** et sur la base de nos solutions nous réalisons la plateforme Geographical Information system for Flight Test (GIFT). Cette plateforme de services est notamment utilisée par les applications X-Wind (développée par KALISIO), Fomax, Airline1, Attol, Kallisté….

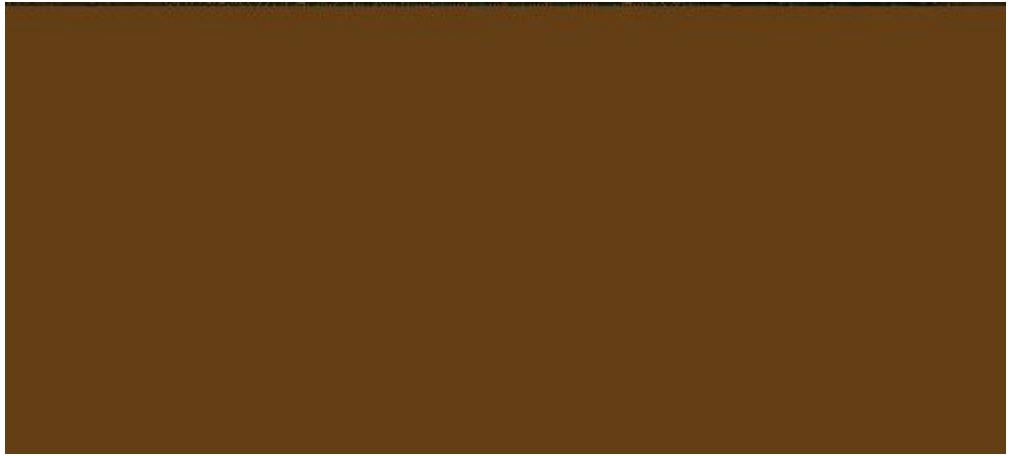
- Pour le compte de l'**IRSN**, nous réalisons la plateforme C3X Planet offrant des services cartographiques et météorologiques exploités par différentes applications métier.

- La chaine de commandement du **SDIS47** et du **SDIS32** exploite l'application Akt'n'Map pour notifier et planifier les interventions à venir

Sous le capot...c'est complexe



Démonstration des solutions

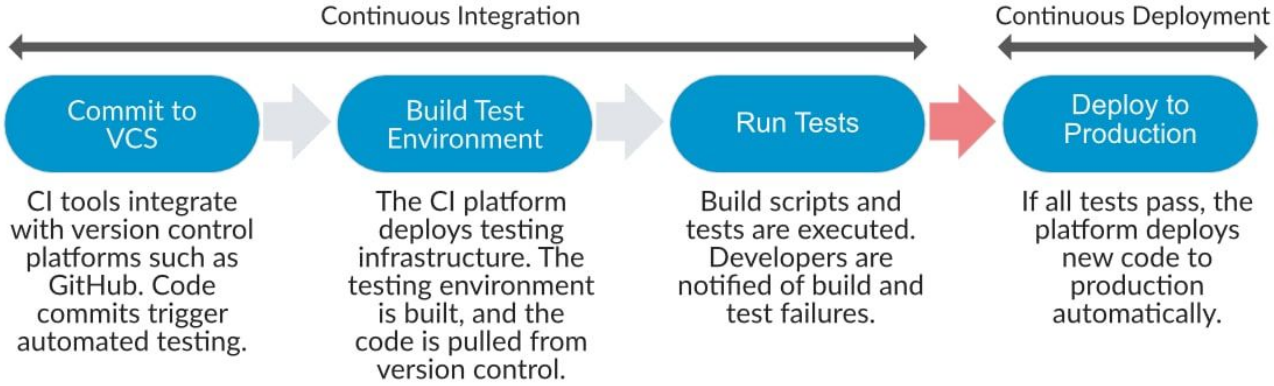


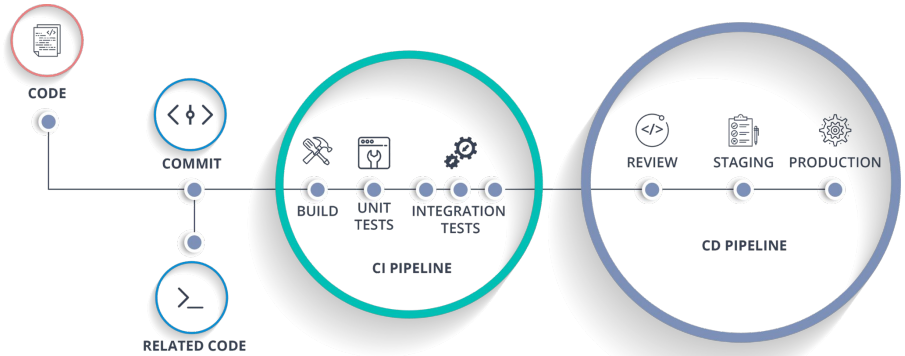
Kalisio

Méthodologie de Développement

Avec quoi peut-on construire tout cela ?

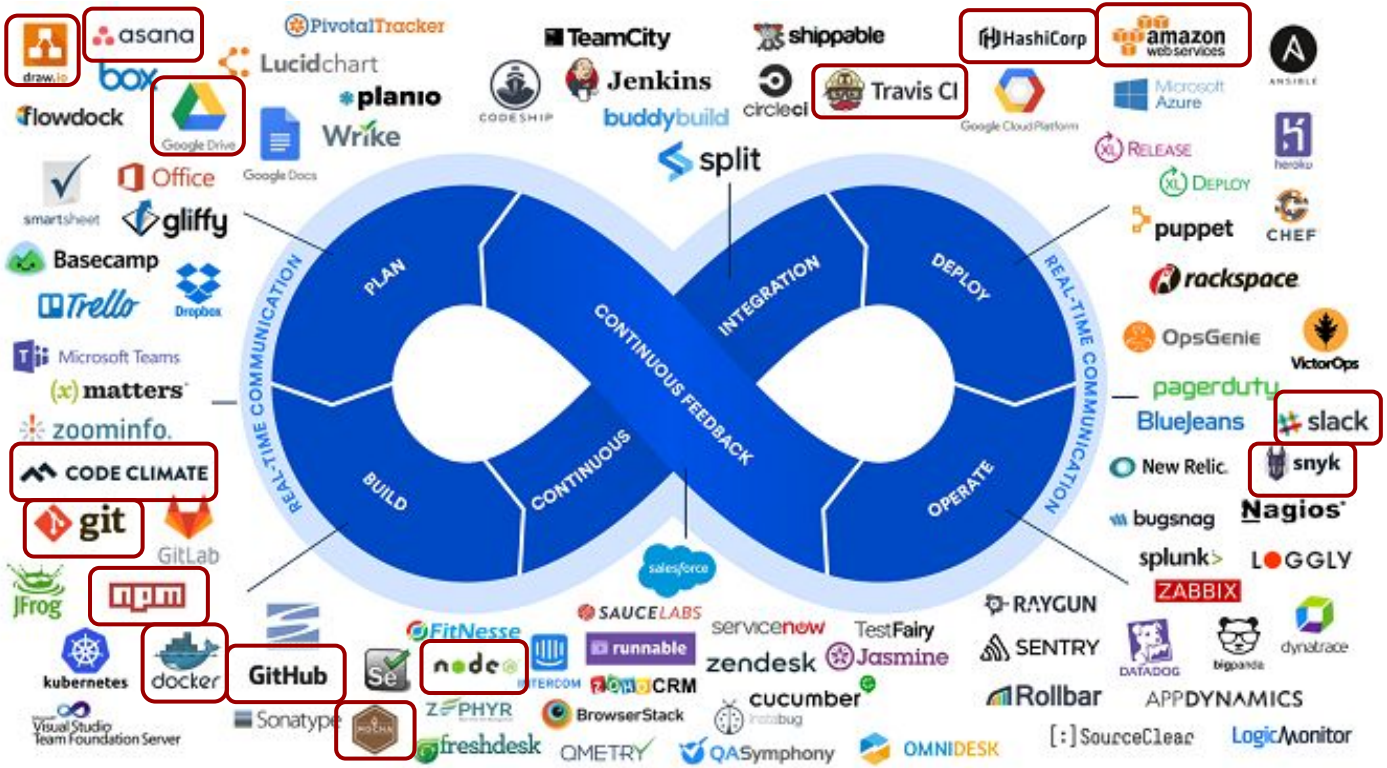
1. Des méthodes de développement





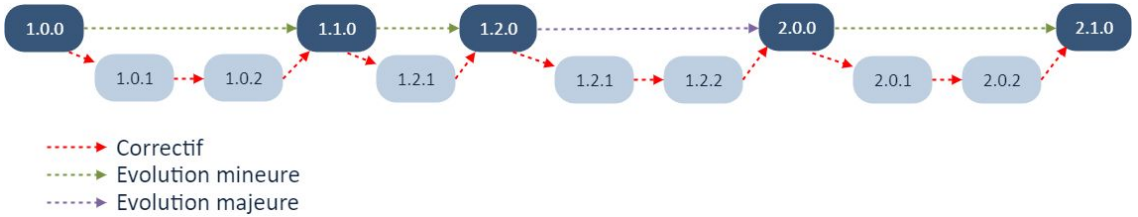
Méthodologies de développement: les outils

2. Des outils pour les mettre en place

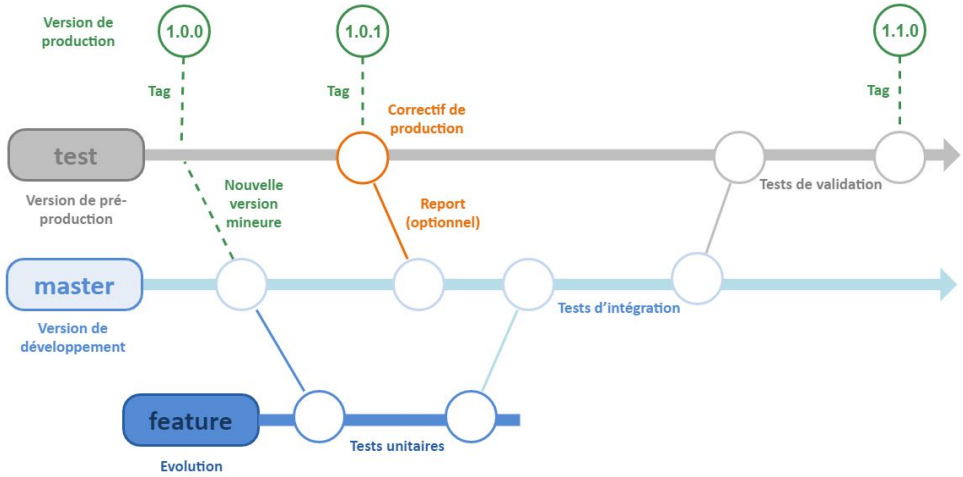


Méthodologies de développement: gestions des versions

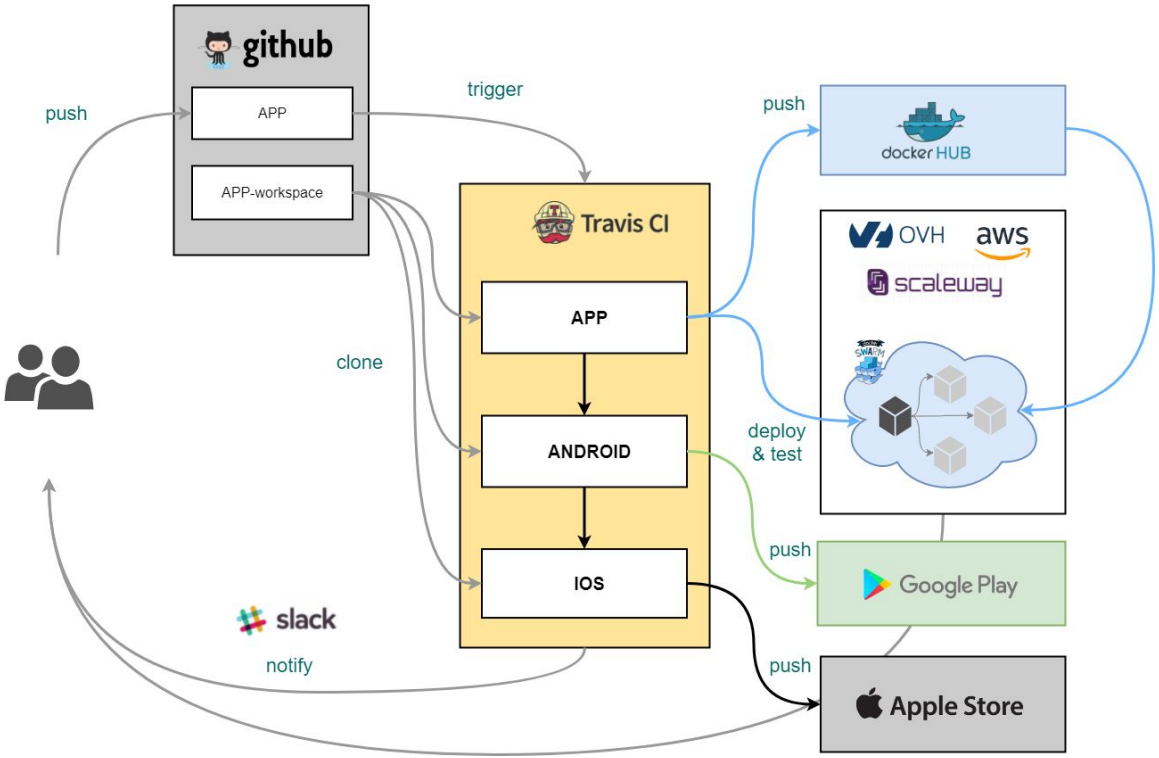
1. [Semantic versioning (semver)]



2. [Gitflow]



Méthodologies de développement: déploiement continu



Plus de détails [ici]

Exercices

➜ Vérifier le bon fonctionnement de l'environnement de développement

- NodeJS - Environnement d'exécution de services web.
- Yarn - Gestionnaire de paquets pour NodeJS.
- MongoDB - Base de données.
- Git - Contrôle de version.
- Docker - Gestionnaire de conteneurs.

a. Récupérer le code de la dernière version de **krawler**

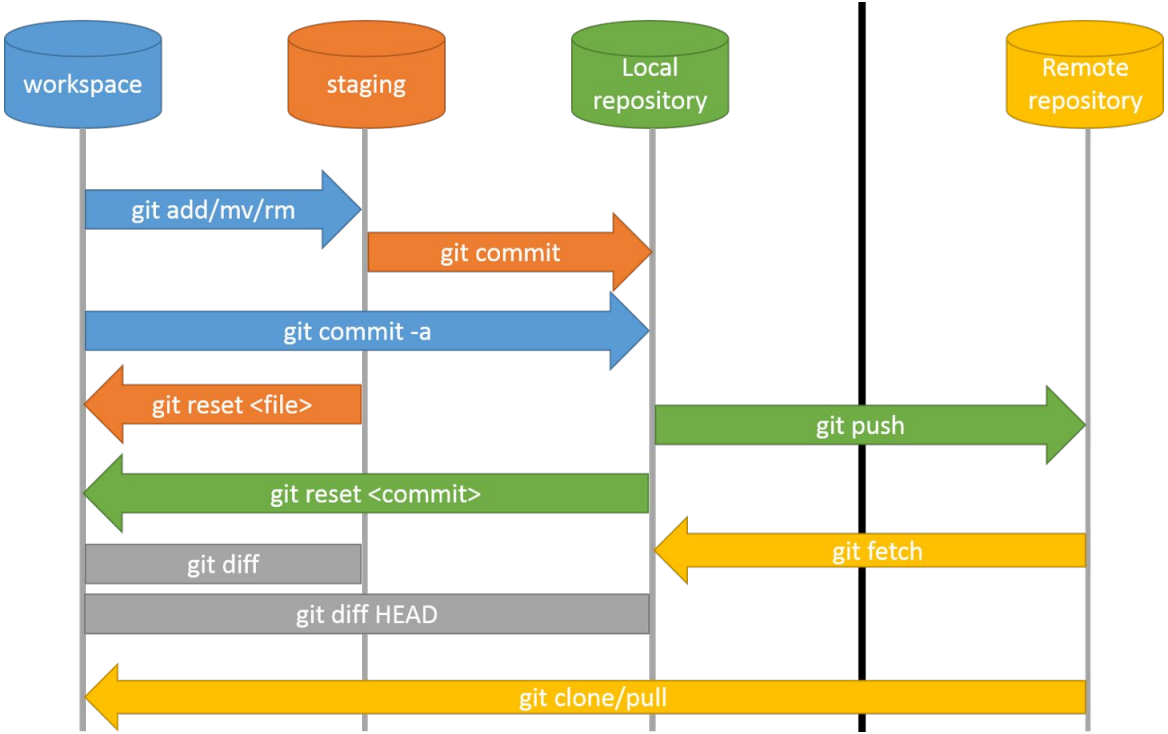
b. Procéder à l'installation du module

c. Lancer l'exemple **csv2db** MongoDB via NodeJS

d. Lancer l'exemple via un **container Docker**

e. **Requêter** les données produites dans MongoDB pour trouver les villes les plus peuplées

Exercices: Git



http://ndpsoftware.com/git-cheatsheet.html

https://learngitbranching.js.org/

Exercices: Yarn

A TINY CHEATSHEET ON

yarn is an alternative package manager to **npm.** It has several advantages over npm like package caching and parallel package installation

yarn init
Create a new **yarn** project/package. Add the **-y** flag to initialize with default configuration

yarn add [package]
Install a **[package]**.
Add **-D** flag to install as devDependency
Add **global** option to install globally

yarn remove [package]
Remove a **[package]**
Removes *devDependency* as well

yarn upgrade [package]
Update a **[package]**
To update the whole project dependencies don't specify **[package]**

yarn list
List all the packages
Specify depth with **--depth** flag

yarn [script]
Run the script called **[script]** as mentioned in *package.json*

yarn install
Install all the packages mentioned in *package.json* of the current project. Running just **yarn** does the same thing

yarn global <command>[package]
Apply **add, upgrade, remove, list** to global packages

yarn audit
Scan and list all the vulnerabilities in the project. yarn cannot fix them yet. Run **npm audit fix** to fix

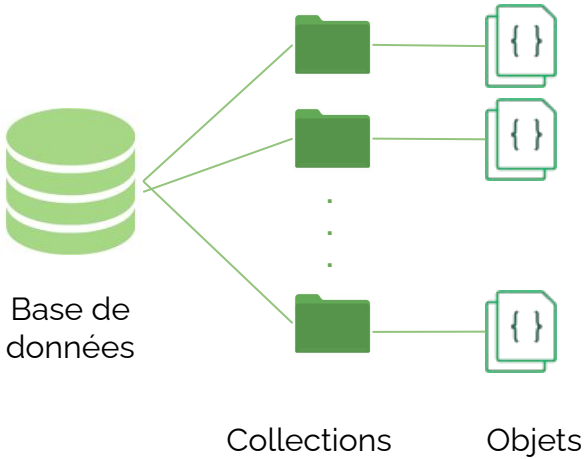
yarn version
Show current version of yarn

Exercices: MongoDB

mongoDB

SSPL

MongoDB est un système de base de données dit NoSQL orienté document. Il ne nécessite pas de schémas prédéfinis et complexes. Il permet de structurer la donnée de façon très simple. Une base de données est structurée en collections et les collections sont elles même constituées d'objets au format JSON.



Exercices: MongoDB

Basic Mongo DB	
`db`	Show name of current database
`mongod`	Start database
`mongo`	Connect to database
`show dbs`	Show databases
`use db`	Switch to database `db`
`show collections`	Display current database collections

Create	
`insert(data)`	insert document(s) returns write result
`insertOne (data, options)`	insert one document
`insertMany(data, options)`	insert many documents
`insertMany([{},{},{}])`	needs square brackets

Read	
`db.collection.find()`	Display documents from `collection`
`find(filter, options)`	find all matching documents
`findOne(filter, options)`	find first matching document

Update	
`updateOne(filter, data, options)`	Change one document
`updateMany(filter, data, options)`	Change many documents
`replaceOne(filter, data, options)`	Replace document entirely

Delete	
`deleteOne(filter, options)`	Delete one document
`deleteMany(filter, options)`	Delete many documents

Filters	
`{"key": "value"}`	Used for filter arguments to filter collection
`{key: {$operator: value} }`	Operators for querying data
`{key: {$exists: true}}`	Matches all documents containing subdocument `key`
`$eq`	Matches values that are equal to a specified value.
`$gt`	Matches values that are greater than a specified value.
`$gte`	Matches values that are greater than or equal to a specified value.
`$in`	Matches any of the values specified in an array
`syntax:`	`{key:{$in: [array of values] } }`
`$lt`	Matches values that are less than a specified value.
`$lte`	Matches values that are less than or equal to a specified value.
`$ne`	Matches all values that are not equal to a specified value.
`$nin`	Matches none of the values specified in an array.
`$and`	Performs AND operation
`syntax:`	`{$and: [ {},{} ] }`
`{key: {$op: filter}, {filter}}`	$and operator is necessary when the same field or operator has to be specified in multiple expressions
`find({doc.subdoc:-value})`	Filter sub documents

Functions	
`.count()`	Counts how many results
`.sort(filter)`	Sort ascend:1 descend: -1

KALISIO

Exercices: MongoDB (geospatial operators)

➜ MongoDB data types are geo-friendly: GeoJson is a native way to encode and query geospatial data

Operator	Geometry Arg Type	2d	2dsphere
$geoWithin	$box,$center,$polygon	Y	N
	$geometry: { type, coordinates }	N	Y
	$centerSphere: [[x,y], **radians**]	Y	Y
$geoIntersects	$geometry only	N	Y
$near,$nearSphere	[x,y]	**R**	-
(output sorted by distance)	$geometry: {type, coordinates}	-	**R**
	+ $minDistance	N	Y
	+ $maxDistance	Y	Y

Y = will assist
N = will not assist
R = REQUIRED

Syntax helper:
find("loc":{$geoWithin: {$box: [[x0,y0], [x1,y2]]}});
find("loc":{$geoWithin: {$geometry: { type: "Polygon", coordinates: [....] }}});

Exercices: Docker

➜ https://devhints.io/docker - https://devhints.io/dockerfile - https://devhints.io/docker-compose

docker Cheat Sheet

JRebel | XRebel

GLOSSARY

Layer
A set of read-only files to provision the system.

Image
A read-only layer that is the base of your container. Might have a parent image.

Container
A runnable instance of the image.

Registery / Hub
Central place where images live.

Docker machine
A VM to run Docker containers (Linux does this natively).

Docker compose
A utility to run multiple containers as a system.

USEFUL ONE-LINERS

Download an image
```
docker pull image_name
```

Start and stop the container
```
docker [start|stop] container_name
```

Create and start container, run command
```
docker run -ti --name container_name
    image_name command
```

Create and start container, run command, destroy container
```
docker run --rm -ti image_name command
```

Example filesystem and port mappings
```
docker run -it --rm -p 8080:8080 -v
/path/to/agent.jar:/agent.jar -e
JAVA_OPTS="-javaagent:/agent.jar"
tomcat:8.0.29-jre8
```

DOCKER CLEANUP COMMANDS

Kill all running containers
```
docker kill $(docker ps -q)
```

Delete dangling images
```
docker rmi $(docker images -q -f
    dangling=true)
```

Remove all stopped containers
```
docker rm $(docker ps -a -q)
```

DOCKER MACHINE COMMANDS

Use docker-machine to run the containers

Start a machine
```
docker-machine start machine_name
```

Configure docker to use a specific machine
```
eval "$(docker-machine env machine_name)"
```

DOCKER COMPOSE SYNTAX

docker-compose.yml file example
```
version: "2"
services:
web:
  container_name: "web"
  image: java:8 # image name
  # command to run
  command: java -jar /app/app.jar
  ports: # map ports to the host
    - "4567:4567"
  volumes: # map filesystem to the host
    - ./myapp.jar:/app/app.jar
mongo: # container name
  image: mongo # image name
```

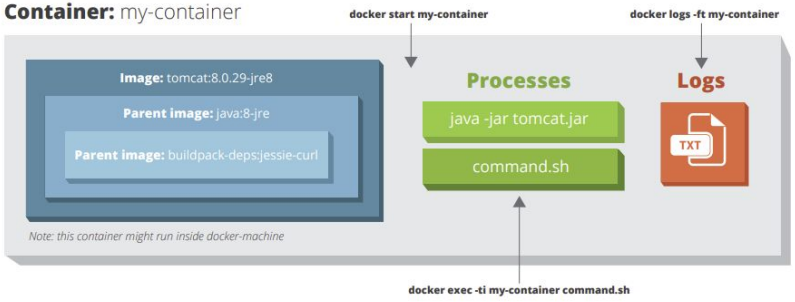
Create and start containers
```
docker-compose up
```

INTERACTING WITH A CONTAINER

Run a command in the container
```
docker exec -ti container_name command.sh
```

Follow the container logs
```
docker logs -ft container_name
```

Save a running container as an image
```
docker commit -m "commit message" -a "author"
    container_name username/image_name:tag
```



Exercices

➜ Solution

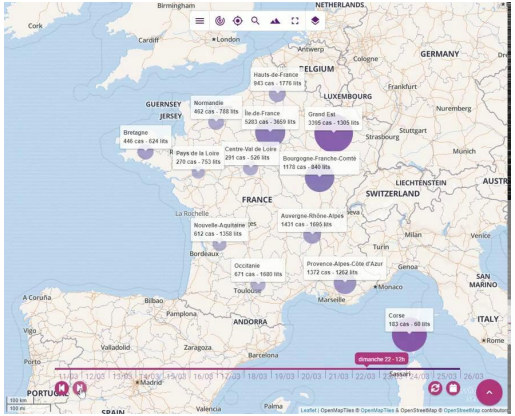
```
git clone https://github.com/kalisio/krawler.git

cd krawler

yarn install

yarn link

krawler example/adsb/jobfile.js

docker pull kalisio/krawler

docker run --name krawler --rm -v
C:\krawler\examples:/opt/krawler/examples -e
"ARGS=/opt/krawler/examples/adsb/jobfile.js" kalisio/krawler

use krawler-test
db.world_cities_csv.find({}).sort({ 'properties.pop': -1 })
```

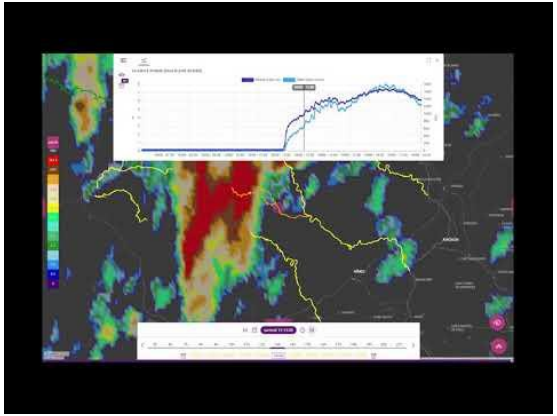
Kalisio

Données spatio-temporelles

Spécificité

- Une donnée spatio-temporelle est une données dont
  a. La géométrie change au cours du temps
     - e.g. un mobile se déplaçant ou un zonage dynamique
  b. Les propriétés changent au cours du temps
     - symbologie ou méta-données
     - e.g. une sonde/un capteur mesurant une valeur
  c. La géométrie et les propriétés changent au cours du temps
     - e.g. une sonde mobile

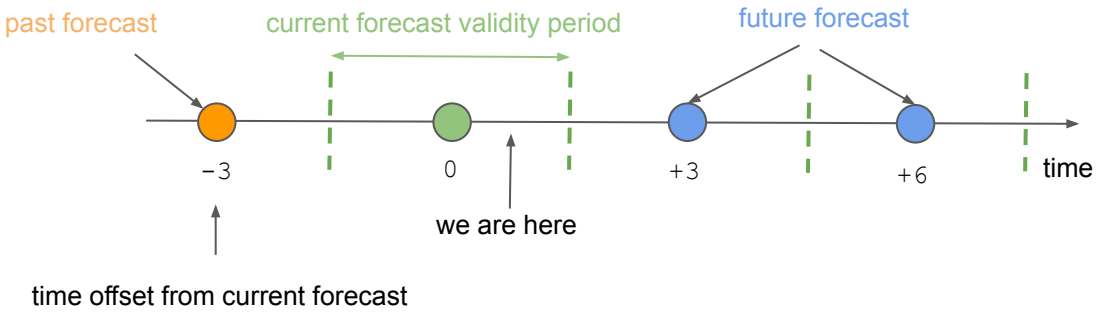


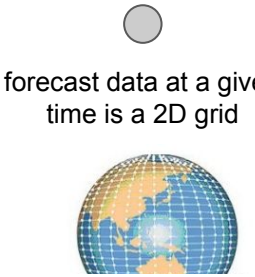


KALISIO

Example: Meteo Data

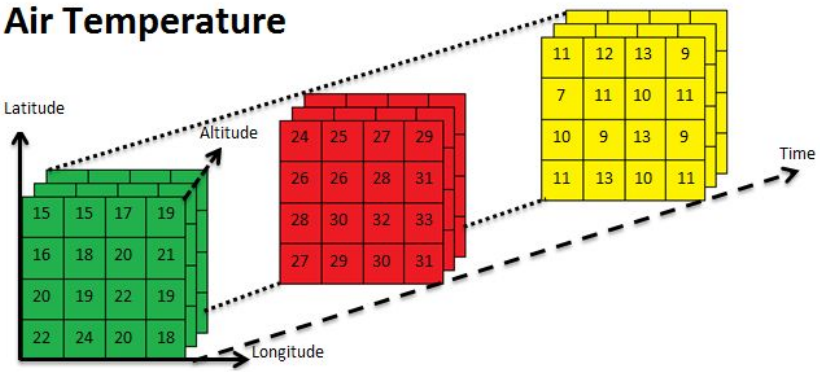
- Forecast models output hundreds of forecast elements (e.g. wind, temperature)
- Production of a set of forecast data is called a **run** of the model
  - occurs on a regular daily basis, e.g. every 6 hours, a.k.a. **run interval/time**
- Each element is a 4D dataset: **forecast time**, **level**, 2D **geographic grid** (lat/long)
  - level can be a meter or pressure scale
  - time is regularly sampled, a.k.a. **forecast interval**, e.g. every 3 hours
- If we consider a given element (e.g. temperature) at a given level (e.g. 100m):

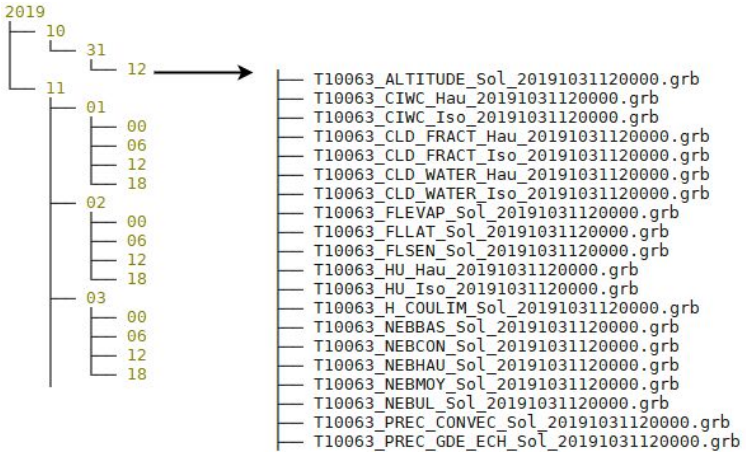




Example: Meteo Data

- Requires dedicated techniques to manage
  - data storage (specific data formats)
  - data volume (eg archiving required hundreds of TB)
  - data velocity (eg updated every 3h hours)
  - data access (eg specific protocols with subsetting)





Exercices

➜ Explorer des données météorologiques

a. Installer Panoply
https://www.giss.nasa.gov/tools/panoply/download/

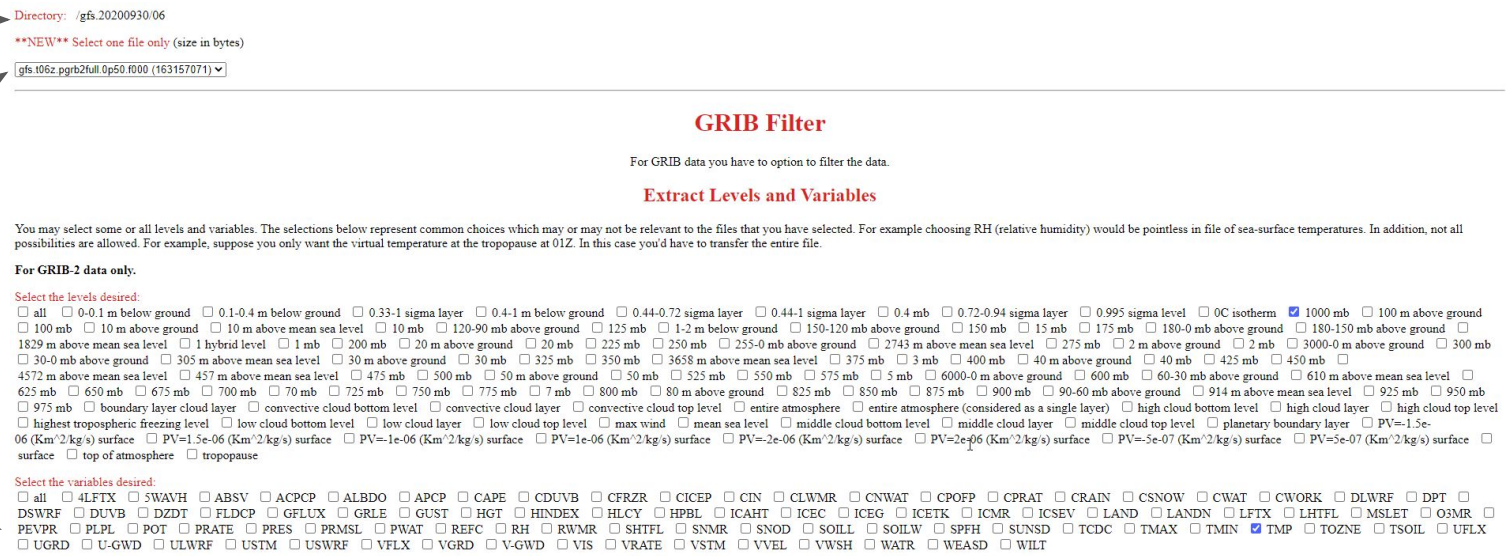
b. Télécharger des jeux de données GFS et visualisez-les
https://nomads.ncep.noaa.gov/cgi-bin/filter_gfs_0p50.pl

run time

forecast time

level (eg 1000 mb)

target element
(eg temperature)



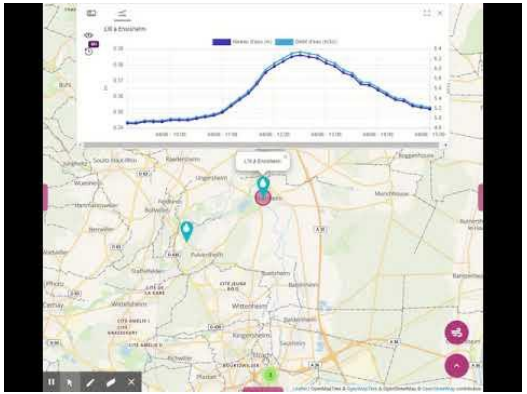
KALISIO

Example: Réseaux de capteurs

- Pour chaque réseau de mesure Kano expose deux points d'entrées via des web services dédiés, ils permettent de récupérer:
  - la liste des **stations** de mesure sur le réseau,
  - les **observations** (i.e. mesures) réalisées par les stations au cours du temps sur le réseau.

- Par exemple, les données hydrométriques Hub'Eau disposent de ces deux points d'entrée:
  - `hubeau-stations` pour les stations,
  - `hubeau-observations` pour les observations.

- Ces deux points d'entrée donnent accès à deux collections MongoDB stockant in-fine les données.



Example: Réseaux de capteurs

- Kano API base URL https://kano.kalisio.fr/ api

- Standards query parameters
  - `$limit` will return only the number of results
  - `$skip` will skip the specified number of results
  - `$sort` will sort based on a list of properties by which to sort mapped to the order (1 ascending, -1 descending)
  - `$select` allows to pick which fields to include in the result
  - Find records where properties do (`$in`) or do not (`$nin`) match any of the given values
  - Find all records where the value is less (`$lt`) or less and equal (`$lte`) to a given value
  - Find all records where the value is more (`$gt`) or more and equal (`$gte`) to a given value

- Geospatial query parameters
  - `$groupBy,$aggregate` will return results grouped according to a property with aggregated values over time
  - `south,north,east,west` will return only the results in a given bounding box
  - `centerLon,centerLat,distance` will return only the results within a given radius from a location

Plus de détails ici

Exercices

➜ Réaliser des requêtes spatio-temporelles sur l'API de Kano et le service Hub'Eau

a. Installer cURL https://github.com/curl/curl-for-win

b. Test de l'accès sécurisé avec un token

c. Récupération des 10 premières stations

d. Récupération de la liste des stations dans une zone donnée (i.e. boîte englobante)

e. Récupération des mesures brutes dans une plage de temps et une zone donnée (i.e. boîte englobante) avec les plus récentes en premier

f. Récupération des mesures agrégées sur une station et une plage de temps données

Exercices

➜ Solution

```
curl --get "https://kano.test.kalisio.xyz/api/users" --header "Authorization: Bearer JWT"

curl --get "https://kano.test.kalisio.xyz/api/hubeau-stations" --data-urlencode "\$limit=10"
--data-urlencode "\$skip=10" --header "Authorization: Bearer JWT"

curl --get "https://kano.test.kalisio.xyz/api/hubeau-stations?south=44&north=45&west=-0.5&east=0 "
--header "Authorization: Bearer JWT"

curl --get
"https://kano.test.kalisio.xyz/api/hubeau-observations?south=44&north=45&west=-0.5&east=0 "
--data-urlencode "time[\$gte]=2020-09-30T07:01:00Z "
--data-urlencode "time[\$lte]=2020-09-30T07:31:00Z "
--data-urlencode "\$sort[time]=1" --header "Authorization: Bearer JWT"

curl --get "https://kano.test.kalisio.xyz/api/hubeau-observations"
--data-urlencode "\$aggregate[0]=H" --data-urlencode "\$aggregate[1]=Q"
--data-urlencode "time[\$gte]=2020-09-30T07:01:00Z "
--data-urlencode "time[\$lte]=2020-09-30T07:31:00Z "
--data-urlencode "properties.code_station=#X331001001"
--data-urlencode "\$groupBy=code_station" --header "Authorization: Bearer JWT"
```

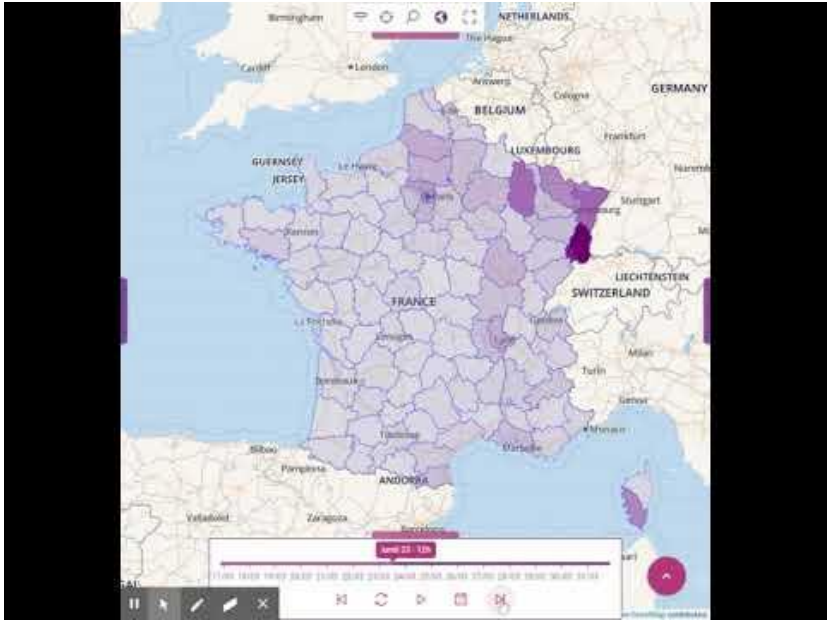
Kalisio

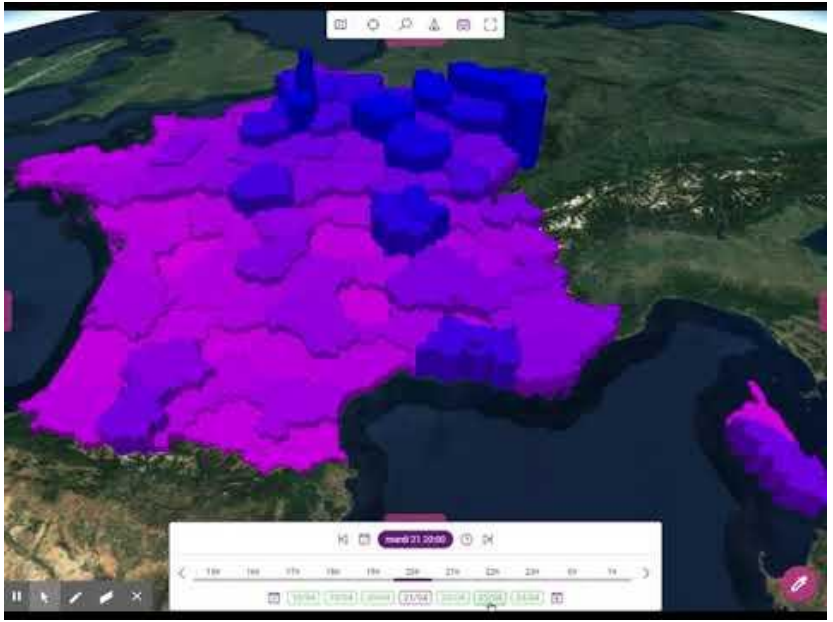
Un cas d'usage concret

Créer une cartographie liée à la pandémie COVID-19

Objectifs du fil rouge

➜ Calculer le taux d'hospitalisation par département et le représenter sur une carte avec une symbologie adaptée





Objectifs du fil rouge

➜ Calculer le taux d'hospitalisation par département et le représenter sur une carte avec une symbologie adaptée

a. Recenser les données disponibles
   - nombre d'hospitalisations
   - limites administratives
   - population

b. Définir les traitements appropriés
   - extraction des données
   - transformation des données

c. Définir la représentation
   - symbologie
   - configuration de l'outil de visualisation

Exercices

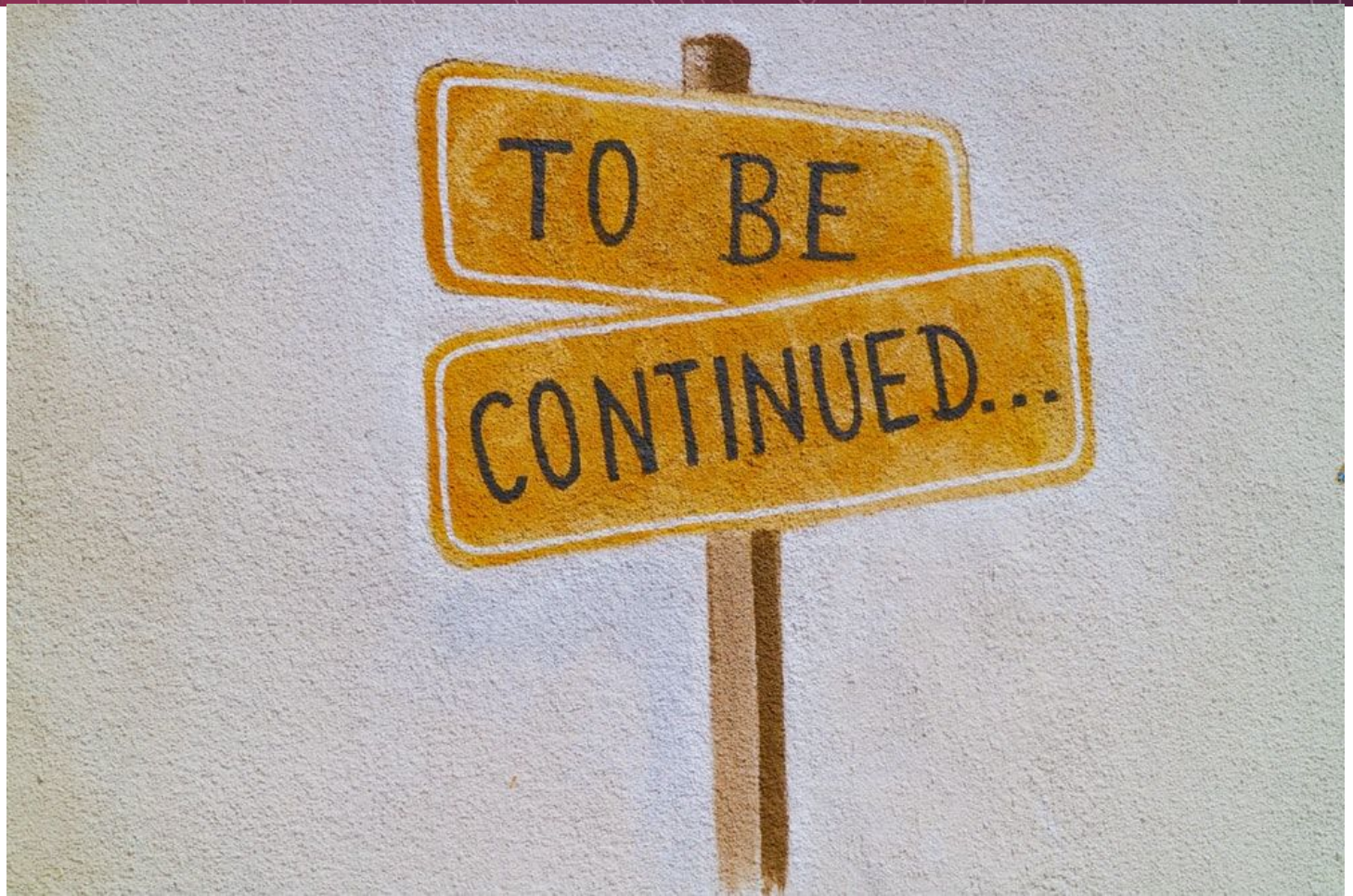
- Trouver les données Santé Publique France permettant d'obtenir le nombre d'hospitalisations sur le portail Open Data de l'Etat https://www.data.gouv.fr/fr/organizations/sante-publique-france/

- Identifier les méta-données pertinentes et le moyen d'automatiser le téléchargement des données

- Télécharger les limites administratives les plus appropriées sur https://france-geojson.gregoiredavid.fr/

- Télécharger les informations de population sur le portail de l'INSEE https://www.insee.fr/fr/statistiques
  - convertir les données au format CSV

Fil rouge



Spatial information that powers up your business

KALISIO