

A large, semi-transparent aerial photograph of a coastal city, likely Marseille, France, is visible in the background. The image shows the city's urban sprawl along the Mediterranean Sea, with green hills to the west and a large body of water to the east.

Formation CQP Bloc 2 - U5 - S18  
Introduction à la plateforme  
Open Source Kalisio

<https://kalisio.com>

- **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.

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

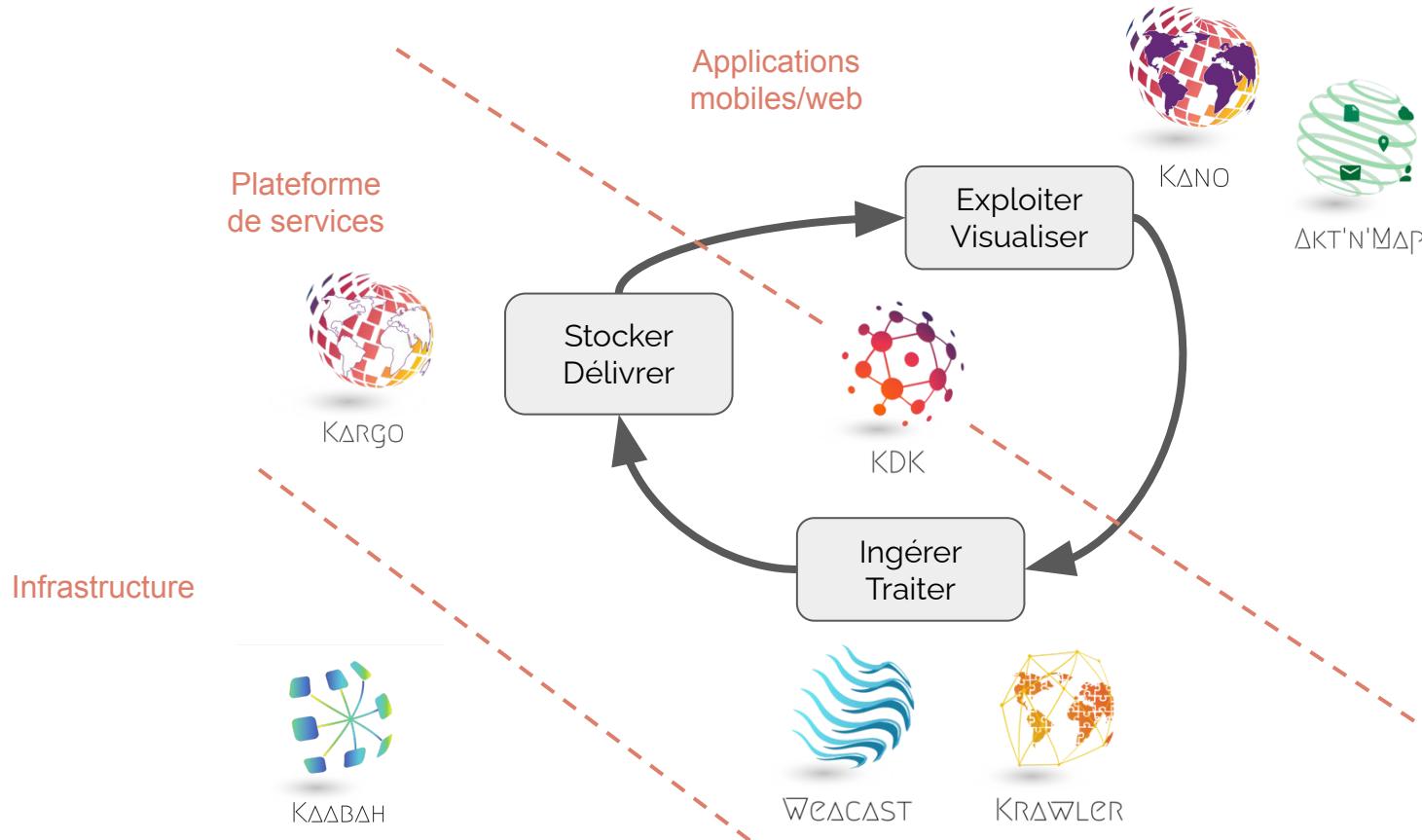


Superviser des assets géolocalisés (biens & personnes)  
Visualiser les données pour prendre des décisions  
Exploiter les données dans des processus métier

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

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

# Notre écosystème de solutions



# Notre écosystème de solutions

## Open Data

Récupération de données à partir de multiples sources ouvertes variées.



## Krawler

Préparation et uniformisation des données.



## Weacast

Récupération spécialisée et simplifiée de sources météorologiques.



## Kargo

Mise en place de l'infrastructure nécessaire pour véhiculer les données.

## Kano

Affichage des données reçues de manière sélective sur une carte modulable.



## Akt'n'Map

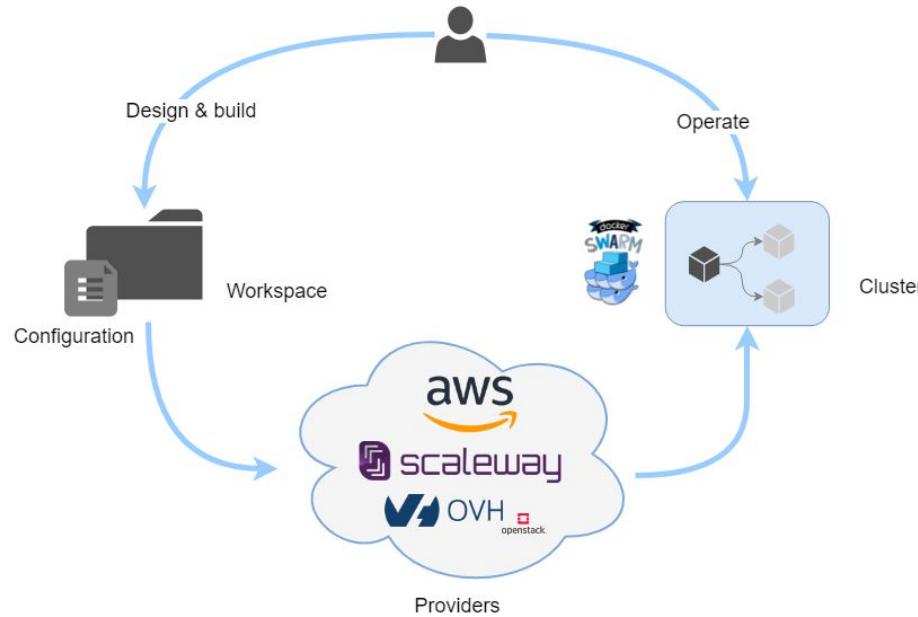
Géolocalisez, mobilisez et interagissez avec vos équipes en temps réel sur le terrain.



Kaabah



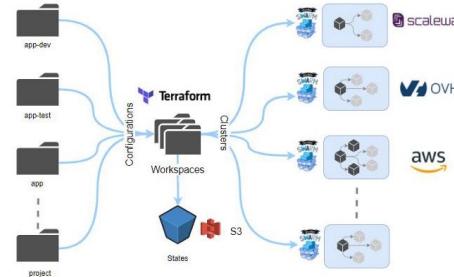
Une solution pour générer et maintenir des infrastructures Docker Swarm dans le cloud



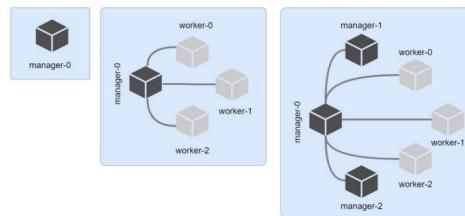
<https://medium.com/@christophe.nouguier/introduction-%C3%A0-kaabah-a435ce0296d0>



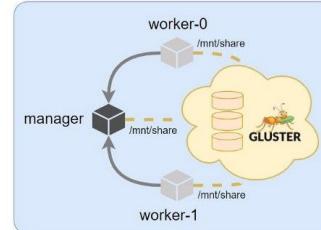
- Une solution pour gérer de multiples plateformes adaptée à une stratégie multi-cloud



- Permet de construire des clusters selon différentes topologies

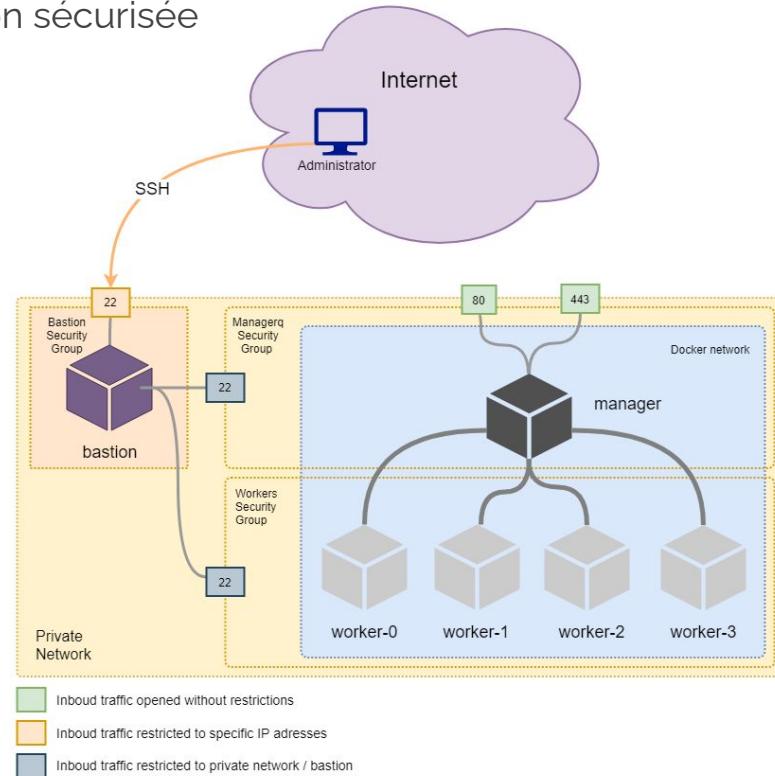


- Permet de construire des clusters hautement résilients



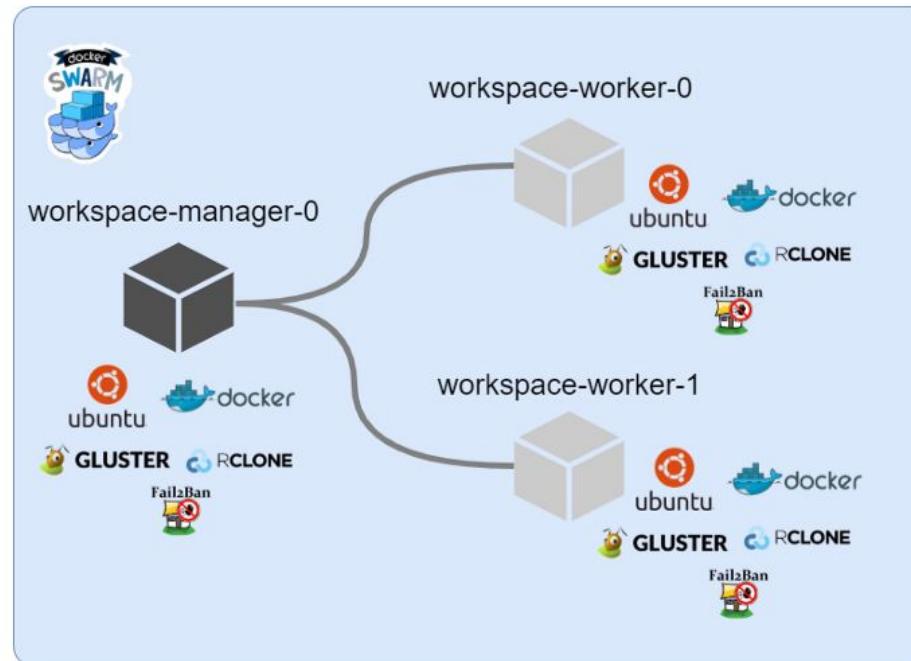


→ Une solution sécurisée





→ Une solution qui repose sur des technologies pérennes, robustes et libres

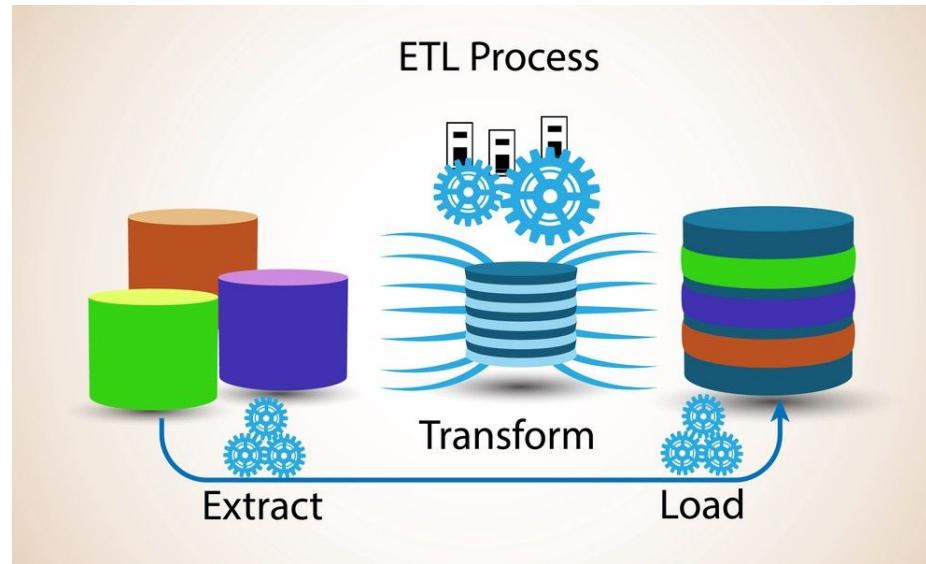




Krawler



Une solution pour réaliser des services d'ingestion et traitement de la données géospatiales :



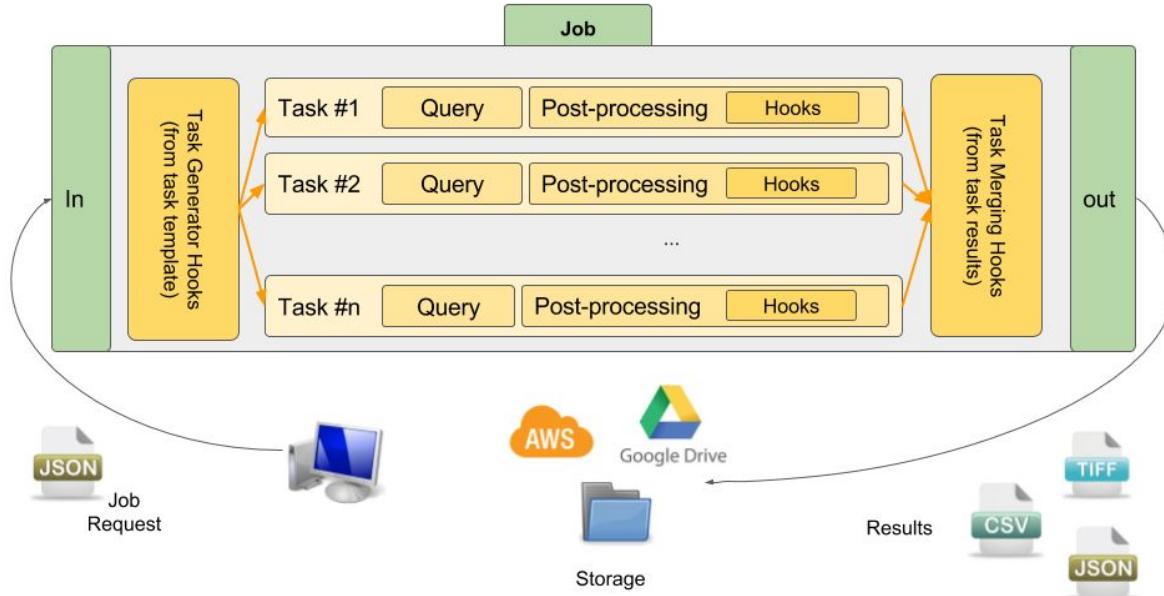
<https://blog.feathersjs.com/a-minimalist-etl-using-feathersjs-part-1-1d56972d6500>

<https://blog.feathersjs.com/a-minimalist-etl-using-feathersjs-part-2-6aa89bd73d66>

<https://towardsdatascience.com/how-to-build-a-map-print-service-in-minutes-c5a24b1fof41>



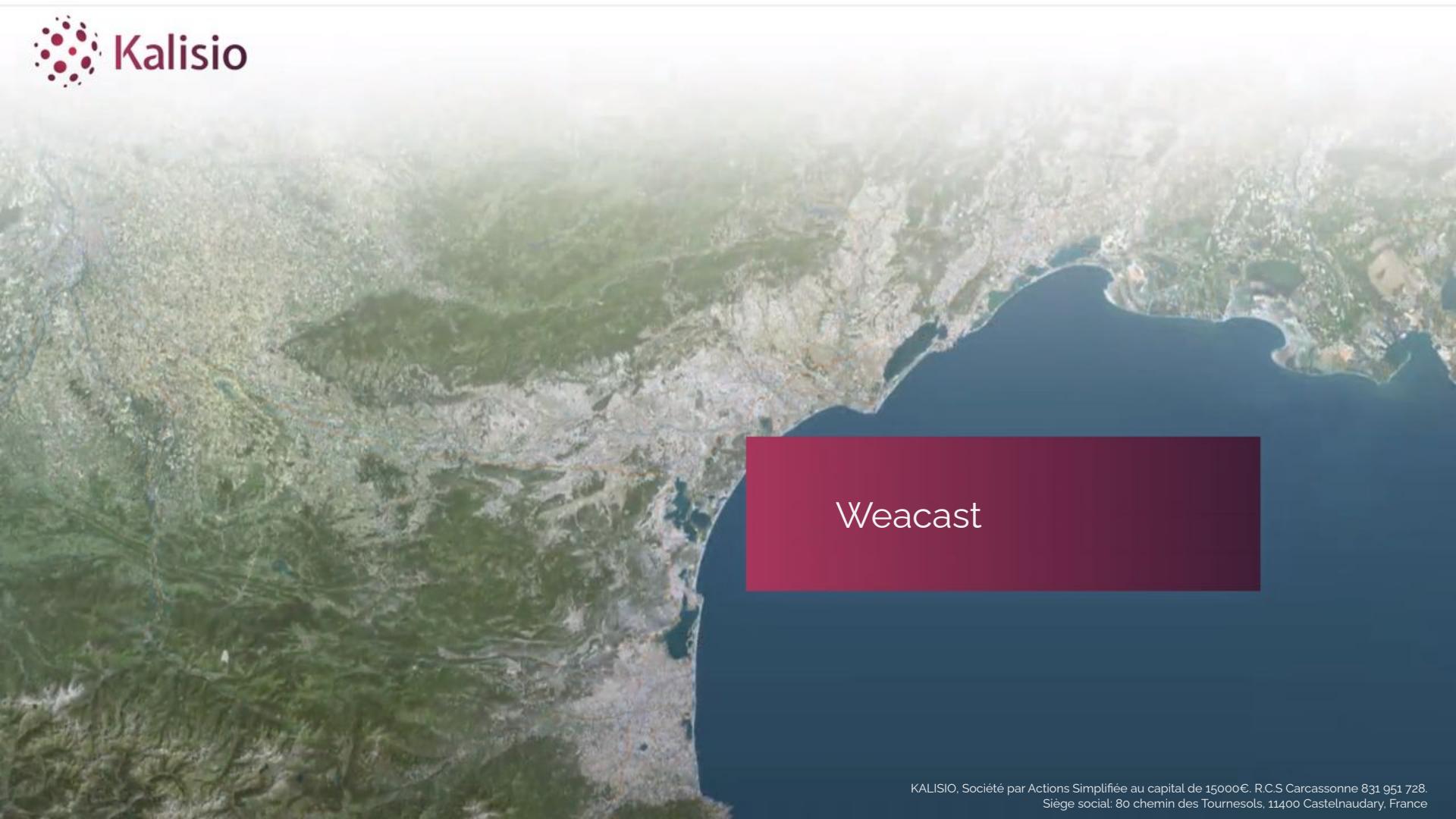
KRAWLER



Feathers



- Support des formats de données : CSV, JSON, GeoJSON, XML, YML....
- Support des bases de données MongoDB et PostGRES SQL
- Exploitation des services cloud de stockage : AWS S3, Google Drive...
- Support du service FTP
- Support des protocoles HTTP, WMS, WFS, WCS, OverPass
- Exploitation de l'API Docker
- Gestion de données maillées et météorologiques

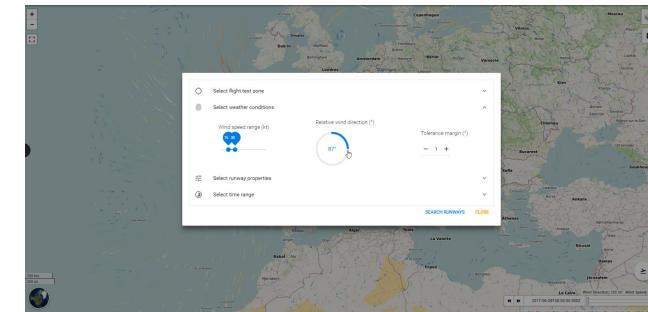


A large, semi-transparent red rectangular overlay is positioned in the lower right quadrant of the image, covering approximately the bottom third of the frame. The text 'Weacast' is centered within this red area in a white, sans-serif font.

Weacast



- Une solution pour ingérer les données de modèles météorologiques
  - Support des modèles :
    - AROME/ARPEGE (Météo France)
    - GFS (NOAA)
  - Fonctionnalités d'analyse avancée :
    - capteurs virtuels
    - recherche de conditions
    - visualisation avancée



- Architecture ouverte et extensible basée sur la solution



KRAWLER



WEACAST





A satellite map of the Black Sea region, showing the coastline of the Black Sea and the surrounding landmasses. The terrain is depicted in various shades of green and brown, with human-made structures like roads and buildings visible. A large, solid red rectangular box is overlaid on the map, covering the area from approximately the 30th to 45th degree of longitude and the 40th to 50th degree of latitude. The word "Kargo" is centered within this red box.

Kargo



KARGO

Une solution qui permet de déployer des services sur une infrastructure Swarm

Geospatial  
services



Docker Swarm  
infrastructure





→ Une solution qui permet de déployer de nombreux services :

- géospatiaux :

- Services d'accès à la donnée suivants les standards WMS, WFS, WMTS, TMS... (MapServer, MapProxy....)
- Service d'accès à la donnée d'élévation 3D
- Services avancées de traitement d'ingestion et traitement de la données basés sur la solution **Krawler**
- Gestionnaire de base de données (PostGIS, MongoDB...)



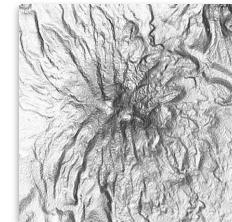
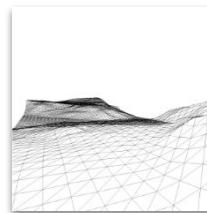
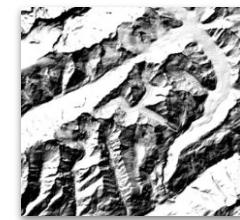
- fonctionnels :

- Réaliser du routage de requêtes (Routing)
- Contrôler l'accès aux services (API)
- Supervision des services (Monitoring)



KARGO

→ Une solution qui permet de déployer un catalogue de données :

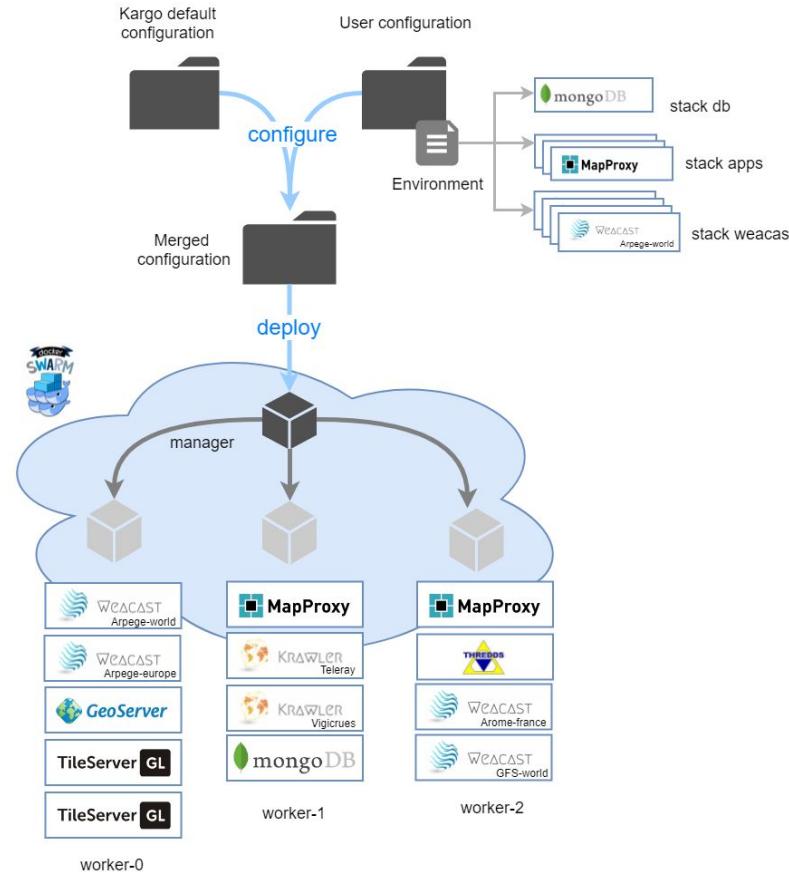


IGN  
PLANET  
OBSERVER

# Kargo



KARGO





KDK

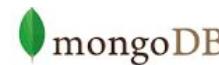


Un framework de développement d'applications géospatiales Web/Mobiles

- Une architecture modulaire proposant des fonctionnalités avancées :
  - Gestion de l'authentification
  - Gestion des utilisateurs, groupes et organisations
  - Gestion des notifications
  - Gestion des solutions de stockage cloud
  - Visualisation cartographique 2D et 3D
  - Gestion d'éléments géographiques
  - Gestion d'évènements temps réels
  - Gestion de paiement en ligne
- Un framework reposant sur un ensemble de solutions OpenSource éprouvées :



Feathers

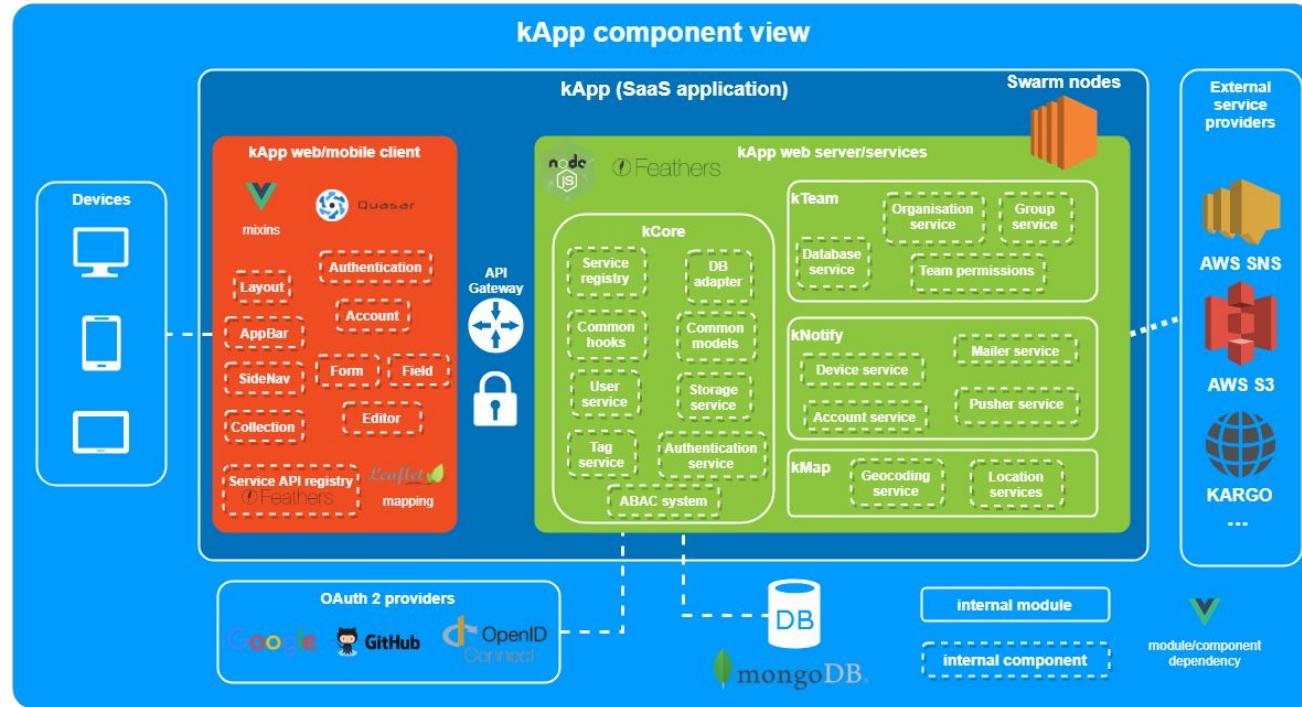


Quasar Framework



Leaflet







- Une solution Open Source qui exploite au mieux les services géospatiaux exposés par une plateforme de services **Kargo**
- Une solution qui repose sur une méthodologie de développement bien définie



A satellite map showing the coastline of the Mediterranean Sea. The land is depicted in various shades of green and brown, representing vegetation and terrain. A large, dark red rectangular box is overlaid on the map, centered on the coast of Turkey. The city of Kano is labeled in white text within this red box.

Kano



**K**ano 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.

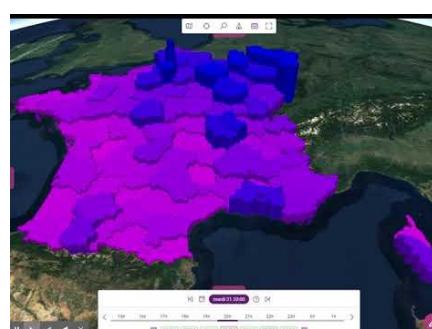
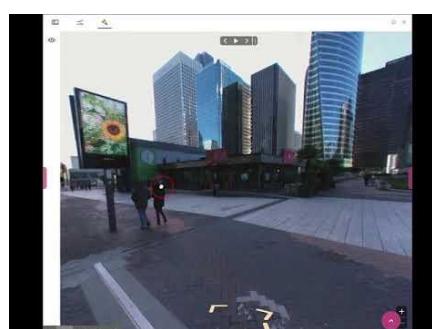
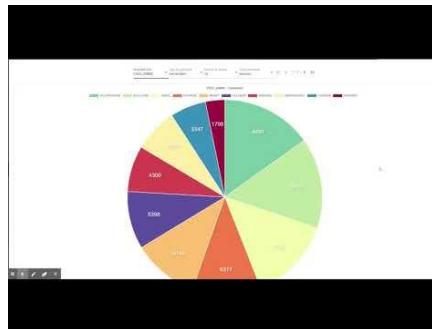
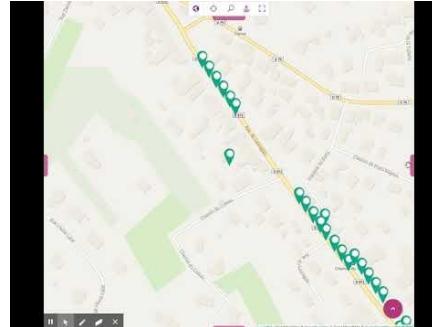
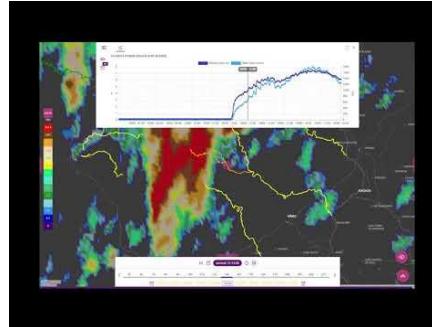
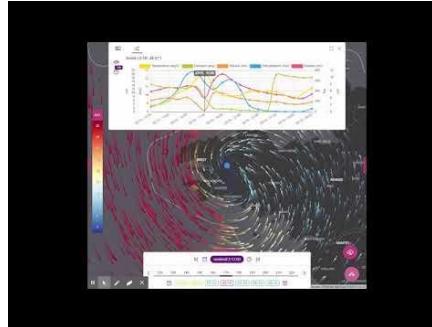
Tirez enfin le meilleur parti de vos différents jeux de données et confrontez-les dans un contexte spécifique grâce à cet outil modulable et adaptable à tout environnement.

Kano fait partie intégrante d'[Akt'nMap](#).

- 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



Kano





## Kano

Kano peut également être intégré via un IFrame et opérée via une API dédiée.

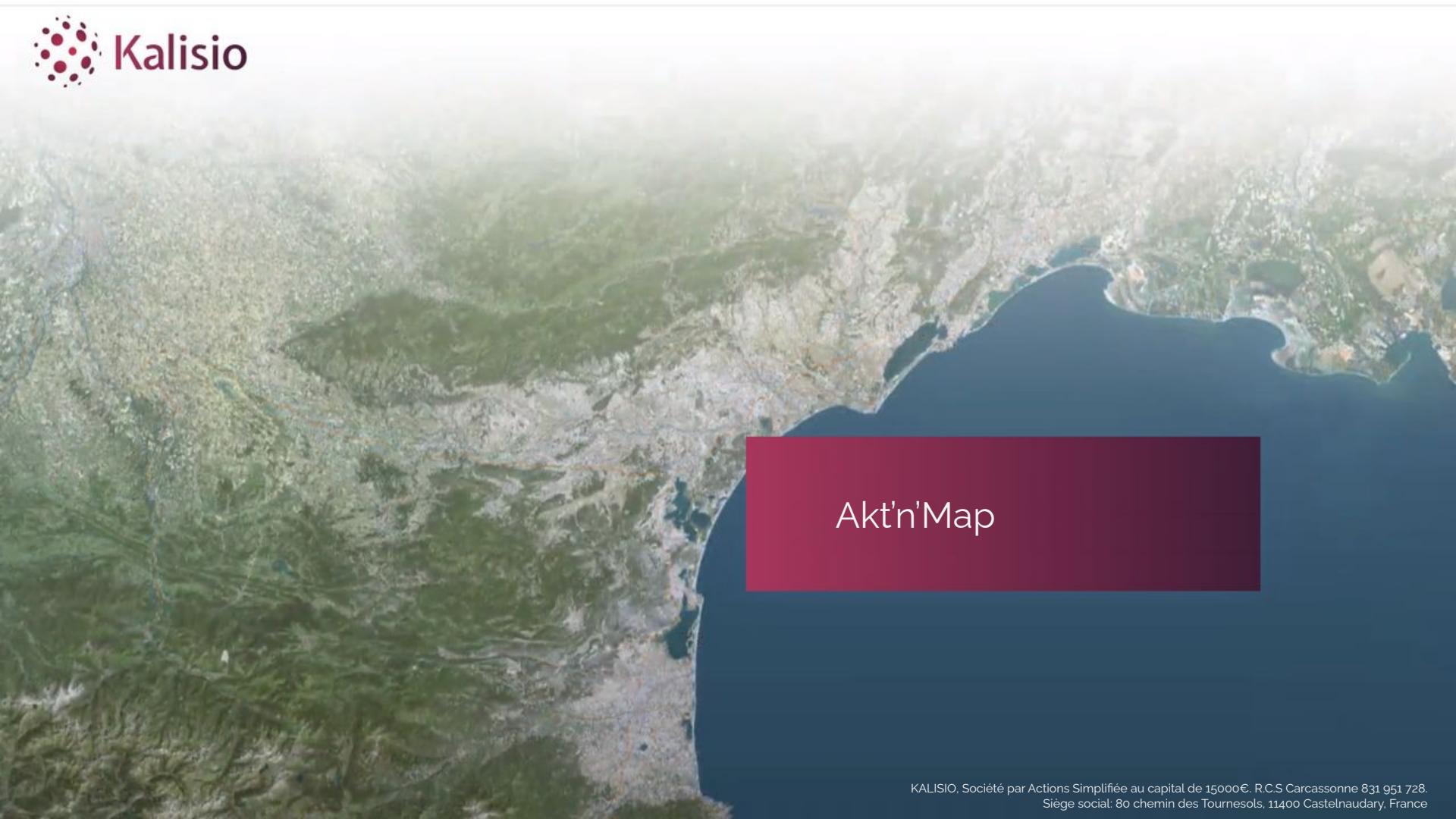
### Kano IFrame Integration Sample

Switch 2D/3D  
Switch background  
Switch airports  
Switch METAR  
Update METAR  
Switch aircraft  
Update aircraft  
Switch ACARS  
Update ACARS  
Switch flightpath  
Update flightpath  
Switch gradient flightpath  
Update gradient flightpath  
Switch flightplan  
Update forecast level

da Tiko

500 km  
300 mi

Leaflet | OpenMapTiles © OpenStreetMap © OpenStreetMap contributors



Akt'n'Map



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

# A

kt'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.

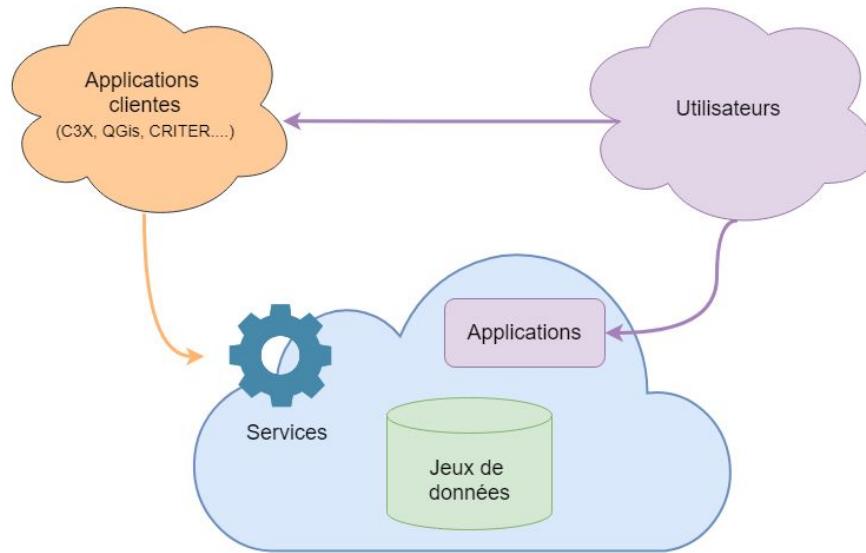
- Géolocalisez vos équipes en temps réel
- Notifiez et partagez des informations
- Communiquez en temps réel
- Centralisez l'information
- Structurez votre organisation
- Facilitez la collaboration entre équipes





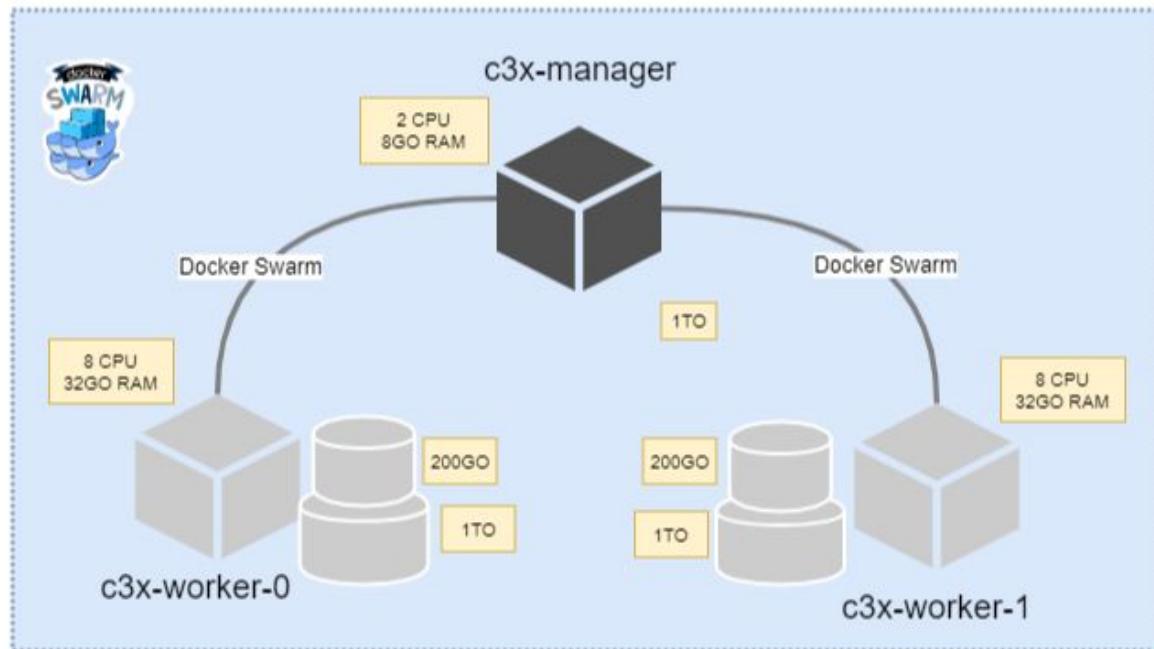


Cas d'usage

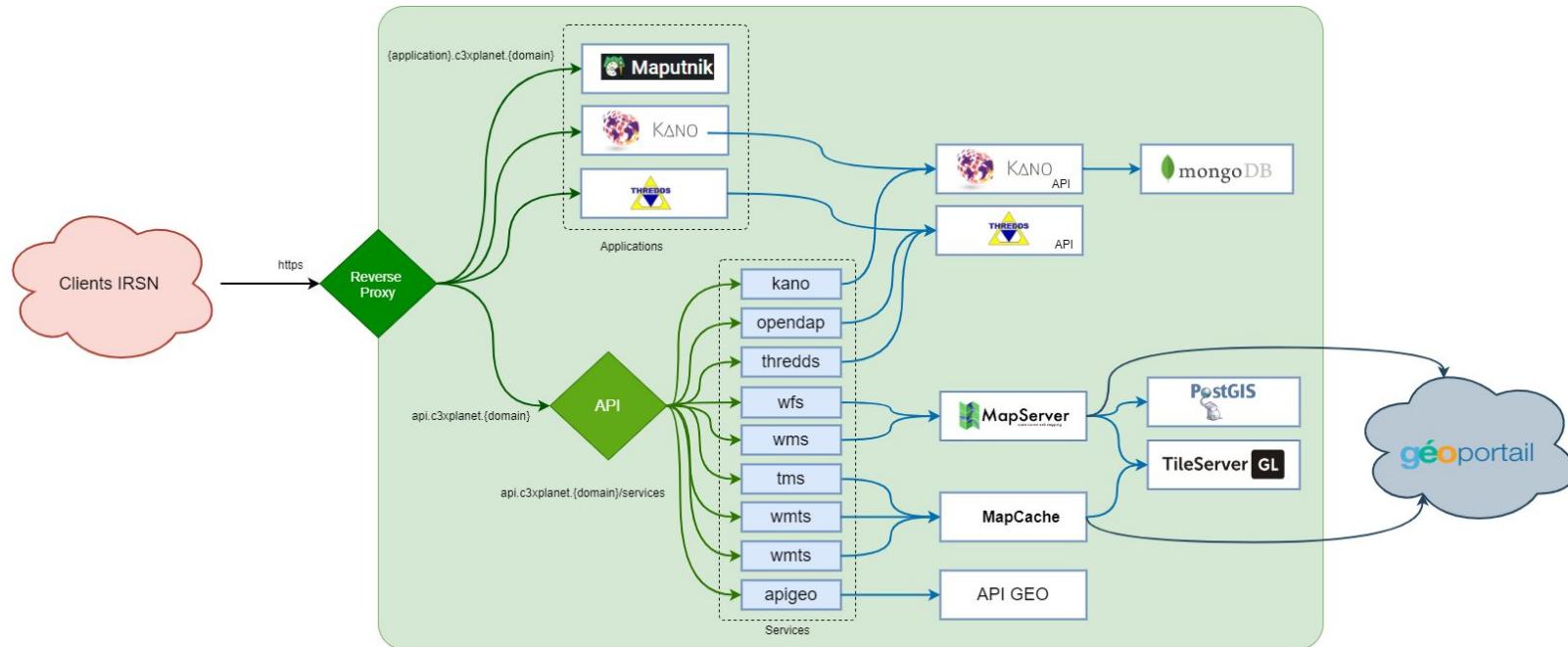


C3[X]  
Planet

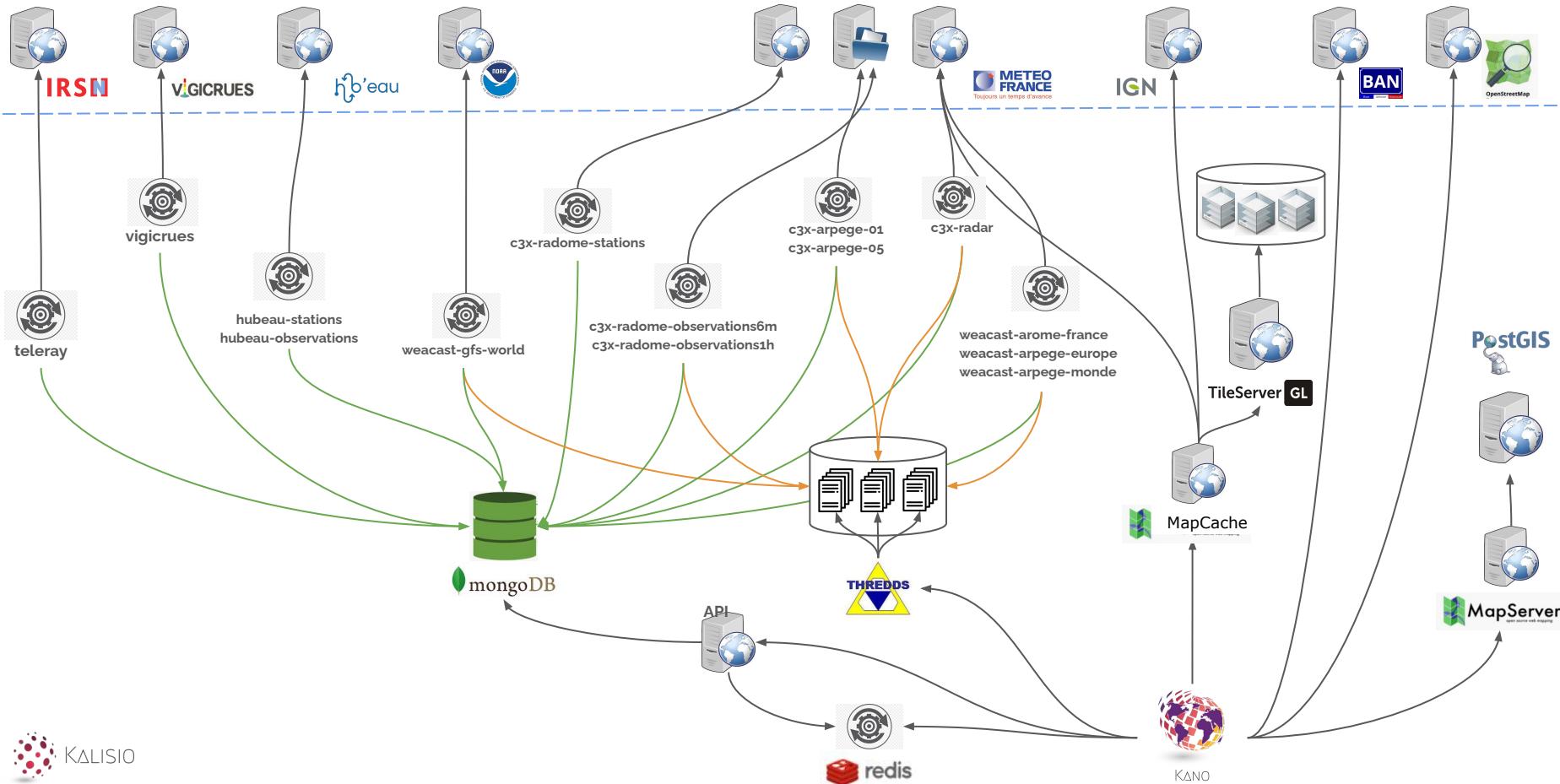
# Infrastructures



# Routing des requêtes



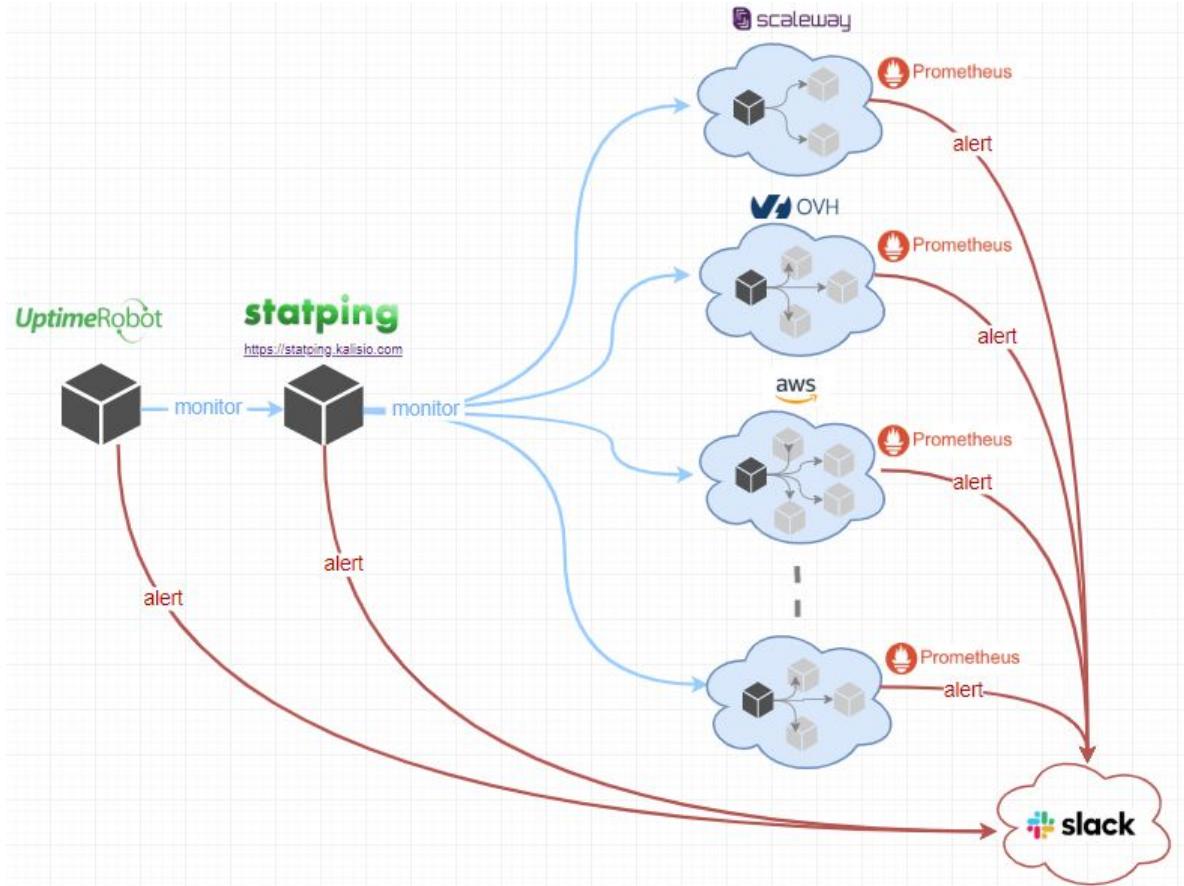
# Architecture de services



## Supervision des services



# Supervision des plateformes

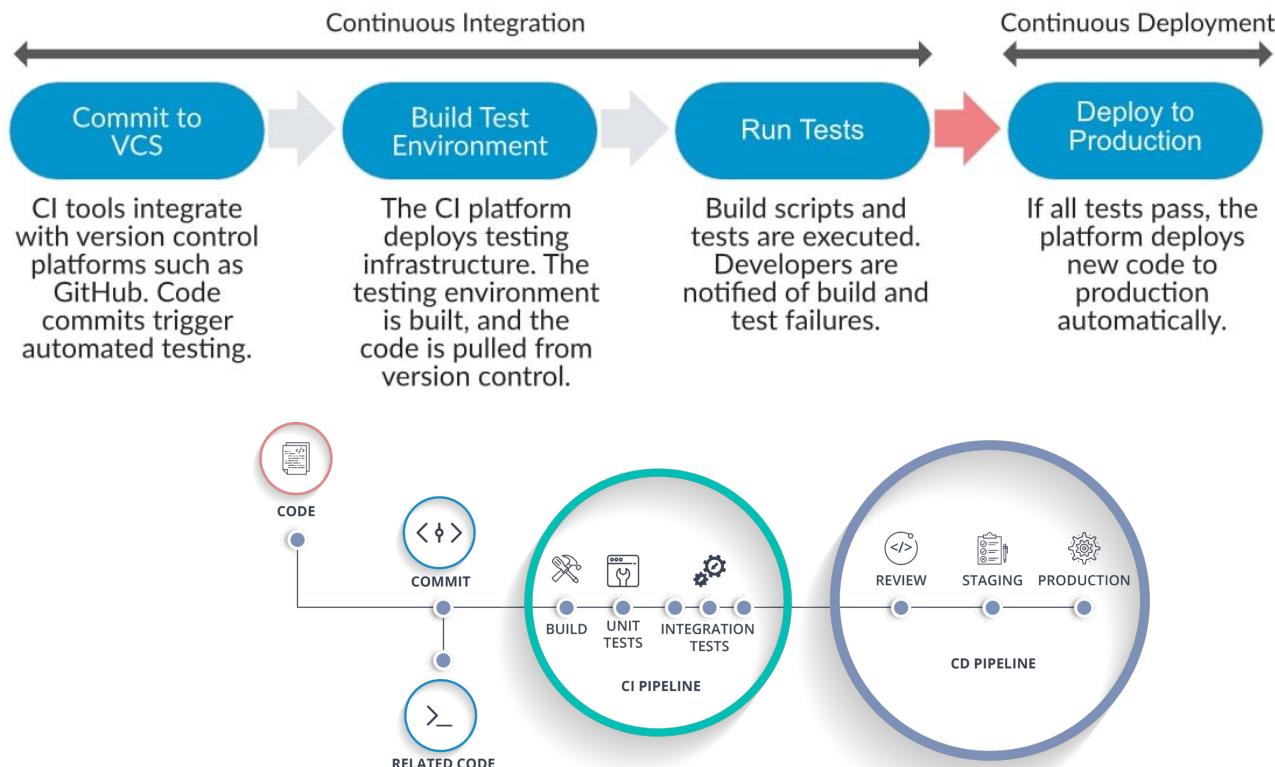




## Méthodologie de Développement

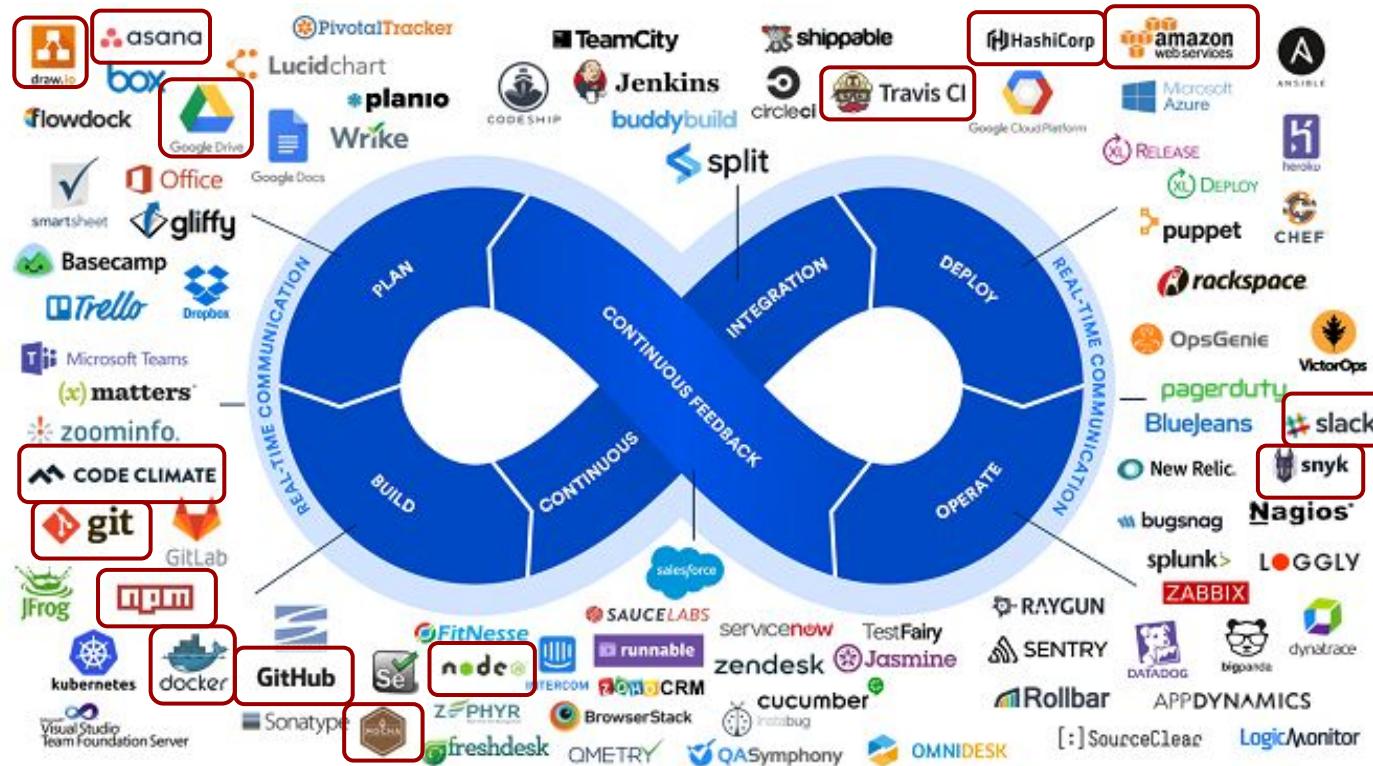
# Avec quoi peut-on construire tout cela ?

## 1. Des méthodes de développement



# Méthodologies de développement: les outils

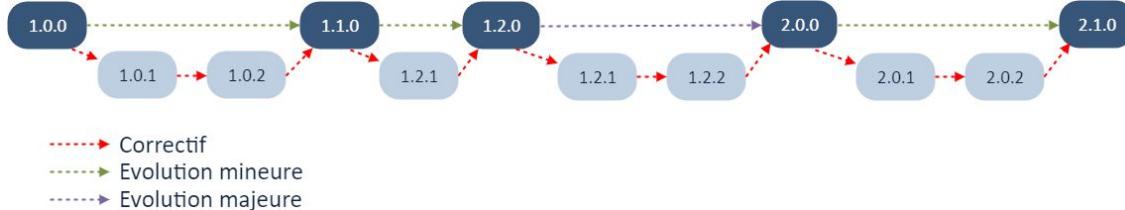
## 2. Des outils pour les mettre en place



# Méthodologies de développement: gestions des versions

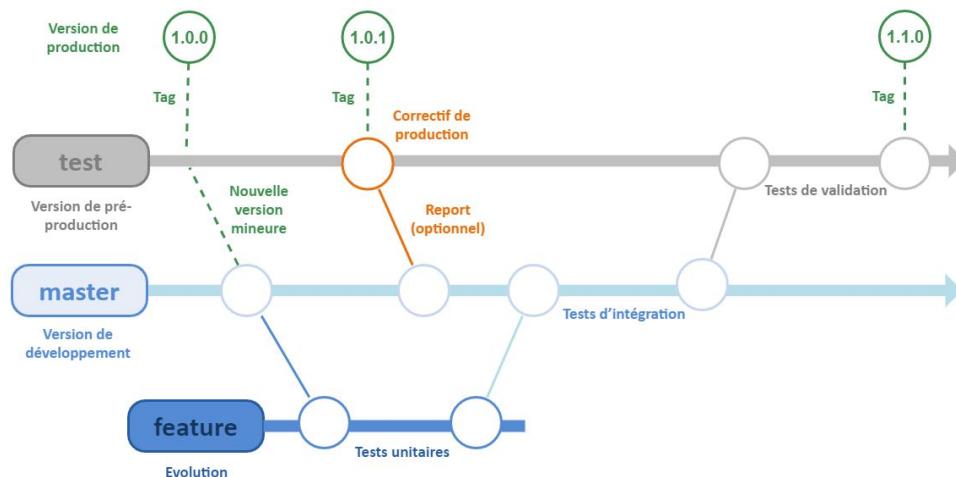
1

## Semantic versioning (semver)

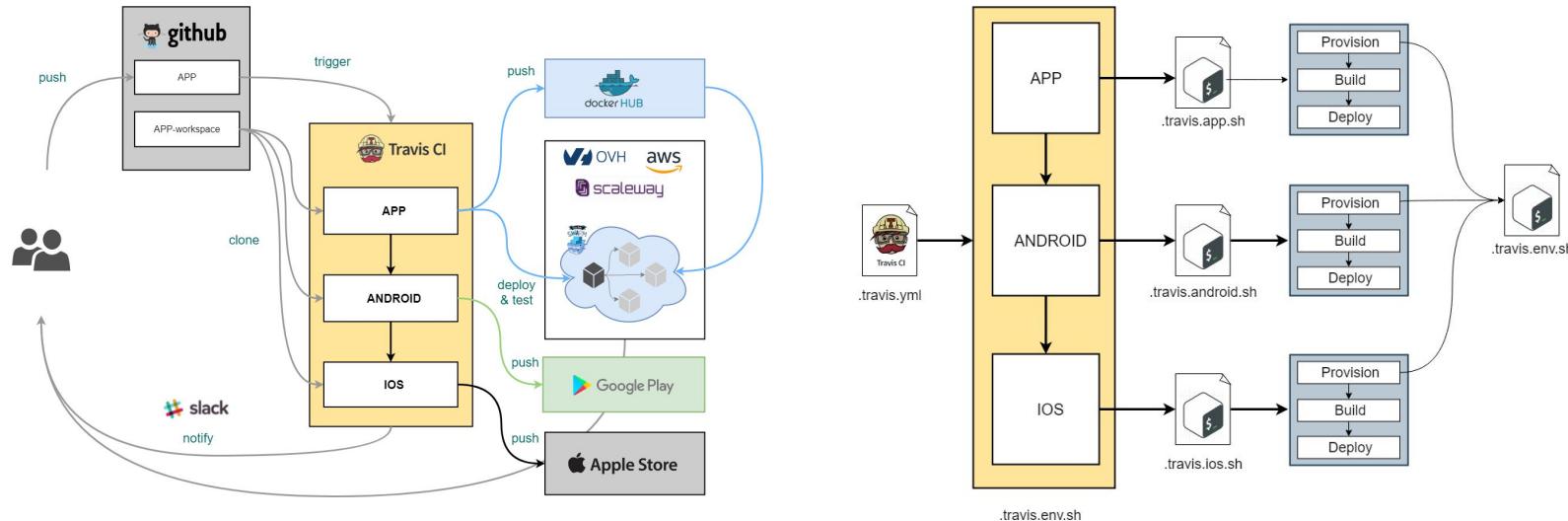


2

## Gitflow



# Méthodologies de développement: déploiement continu



Plus de détails [ici](#)

<https://medium.com/better-programming/why-we-stopped-using-so-called-best-practices-in-our-ci-cd-process-2ff09811f633>

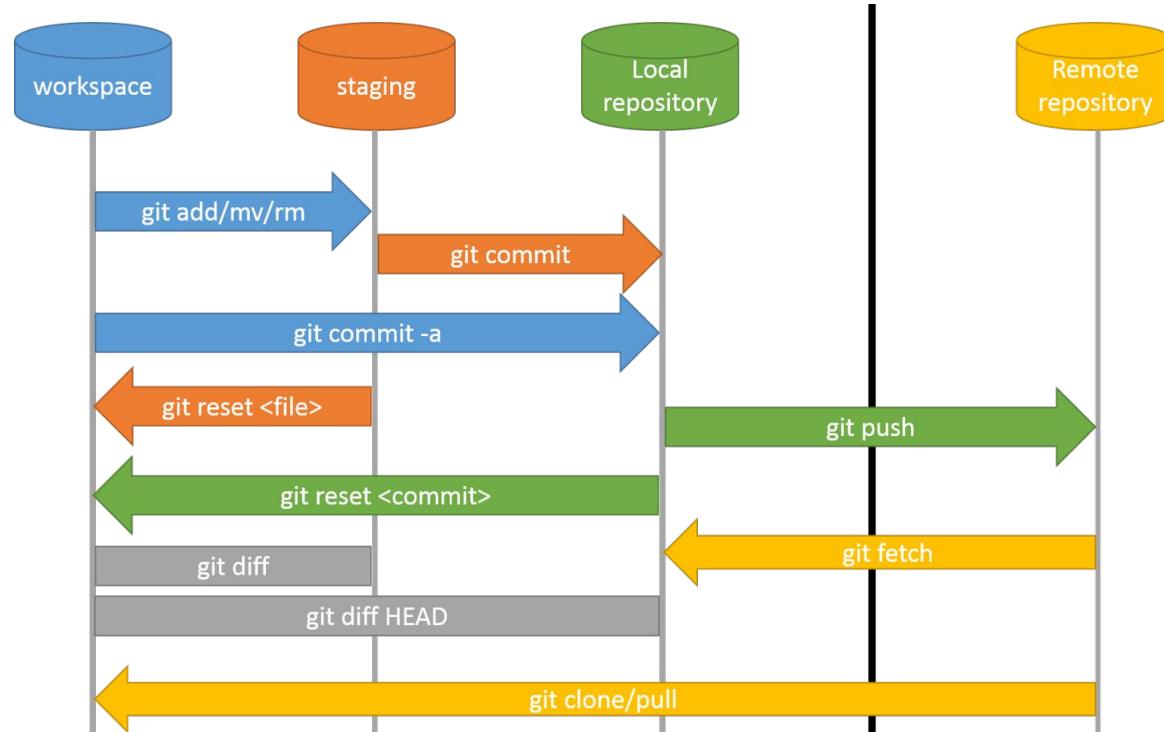
A large, semi-transparent aerial photograph of a coastal city, likely Marseille, France, is visible in the background. The image shows the city's urban sprawl along the Mediterranean Sea, with green hills to the west and a dense concentration of buildings and infrastructure. A dark red rectangular box is overlaid on the lower right portion of the image, containing the text.

Exercices de  
découverte

→ 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 le job **adsb**
  - d. Lancer l'exemple **csv2db** MongoDB via NodeJS
  - e. Lancer l'exemple via un **container Docker**
  - f. **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/>

# Exercises: Yarn

## A TINY CHEATSHEET ON



### yarn add [package]

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

### yarn upgrade [package]

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

**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 remove [package]

Remove a [package]  
Removes *devDependency* as well

### yarn [script]

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

### yarn list

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

### 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

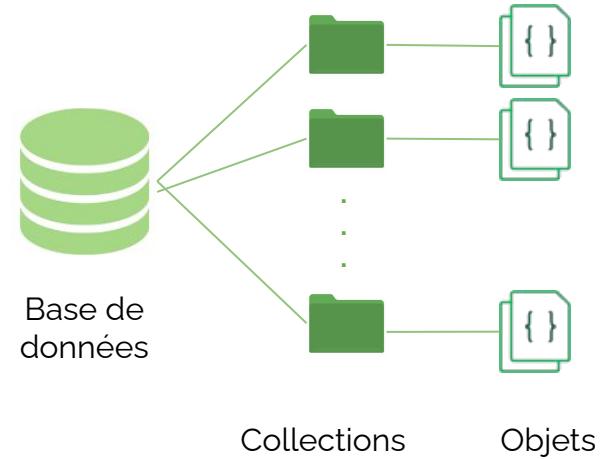
Show current version of yarn

# Exercices: MongoDB



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.



# Exercises: MongoDB

Basic Mongo DB		Filters	
db	Show name of current database	{"key": "value"}	Used for filter arguments to filter collection
mongod	Start database	{}key: {\$operator: value} }	Operators for querying data
mongo	Connect to database	{}key: {\$exists: true}}	Matches all documents containing subdocument key
show dbs	Show databases	\$eq	Matches values that are equal to a specified value.
use db	Switch to database db	\$gt	Matches values that are greater than a specified value.
show collections	Display current database collections	\$gte	Matches values that are greater than or equal to a specified value.
<b>Create</b>		\$in	Matches any of the values specified in an array
insert(data)	insert document(s) returns write result	syntax: {key:{\$in: [array of values] }}	
insertOne (data, options)	insert one document	\$lt	Matches values that are less than a specified value.
insertMany(data, options)	insert many documents	\$lte	Matches values that are less than or equal to a specified value.
insertMany([{},{}],{})	needs square brackets	\$ne	Matches all values that are not equal to a specified value.
<b>Read</b>		\$nin	Matches none of the values specified in an array.
db.collection.find()	Display documents from collection	\$and	Performs AND operation
find(filter, options)	find all matching documents	syntax: {\$and: [ {},{} ] }	
findOne(filter, options)	find first matching document	{}key: {\$op: filter}, {filter}}	\$and operator is necessary when the same field or operator has to be specified in multiple expressions
<b>Update</b>		find({doc.subdoc:-value})	Filter sub documents
updateOne(filter, data, options)	Change one document		
updateMany(filter, data, options)	Change many documents		
replaceOne(filter, data, options)	Replace document entirely		
<b>Delete</b>			
deleteOne(filter, options)	Delete one document	.count()	Counts how many results
deleteMany(filter, options)	Delete many documents	.sort(filter)	Sort ascend:1 descend:-1
<b>Functions</b>			

# Exercises: 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 (output sorted by distance)	[x,y]	R	-
	\$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: [ .... ] }}});
```

# Exercises: Docker

→ <https://devhints.io/docker> - <https://devhints.io/dockerfile> - <https://devhints.io/docker-compose>



## Docker Cheat Sheet



### 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.

**Registry / 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
```

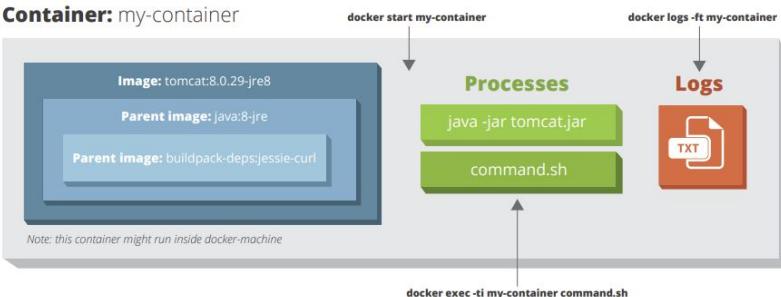
Create and start containers  
`docker-compose up`

### INTERACTING WITH A CONTAINER

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

Follow the container logs  
`docker logs -f container_name`

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



**Container: my-container**

**Image:** tomcat:8.0.29-jre8

**Parent image:** java:8-jre

**Parent image:** buildpack-deps:jessie-curl

*Note: this container might run inside docker-machine*

`docker start my-container`

`docker logs -f my-container`

`docker exec -ti my-container command.sh`

**Processes**

`java -jar tomcat.jar`

`command.sh`

**Logs**

`TXT`

**LEARN HOW JREBEL AND XREBEL TRANSFORM ENTERPRISE SOFTWARE DEVELOPMENT.**

Try for free at [jrebel.com](http://jrebel.com)

# Exercices

## → Solution

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

```
yarn install
yarn link
```

```
krawler examples/adsb/jobfile.js
```

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

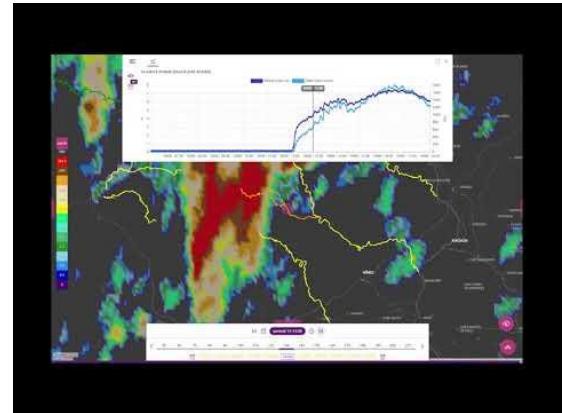
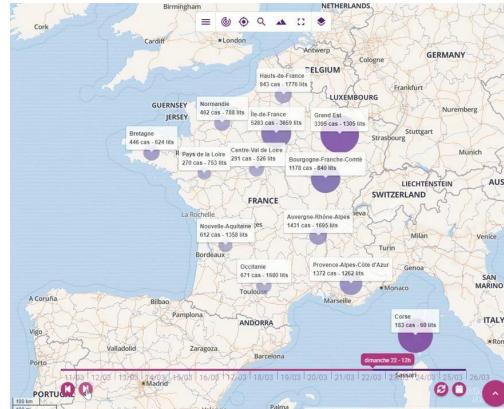
```
krawler examples/csv2db/jobfile.mongo.js
use krawler-test
db.world_cities_csv.find({}).sort({ 'properties.pop': -1 })
```

A high-angle, aerial satellite photograph of a coastal region. The left side shows a range of green hills and mountains. The right side shows a large, irregularly shaped body of water, likely a bay or a large lake, with a city built along its shore. The city area is a mix of green spaces and urban development. A dark red rectangular box is overlaid on the image, containing the text.

Données  
spatio-temporelles

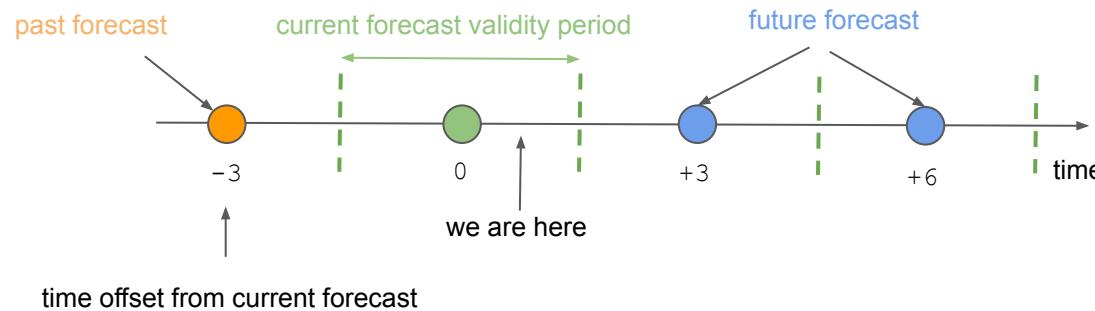
# Spécificité

- 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



# 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):

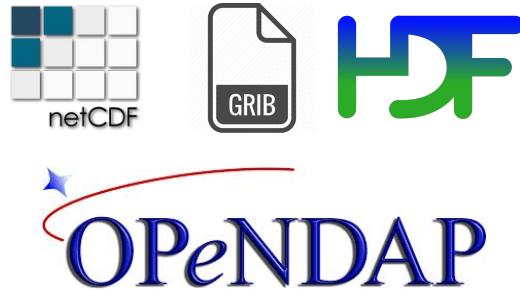


forecast data at a given time is a 2D grid

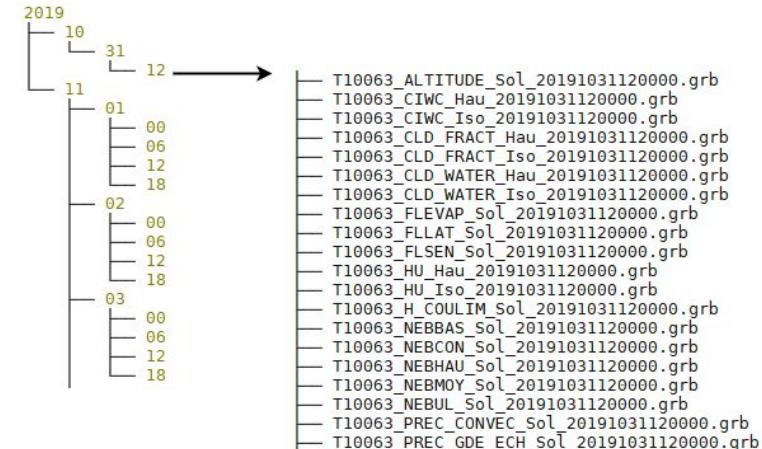
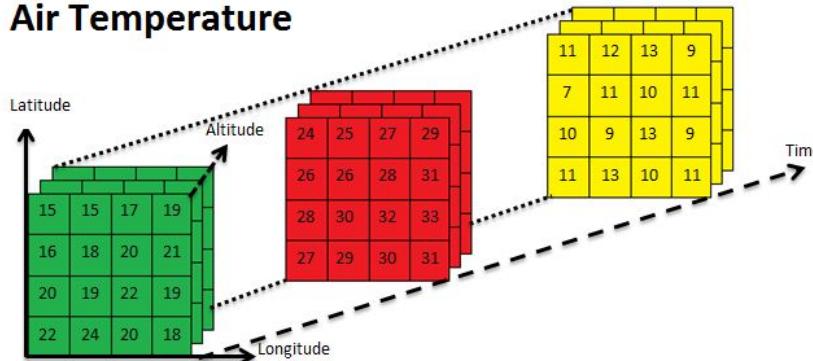


# Example: Meteo Data

- 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)



## Air Temperature



# 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\\_op50.pl](https://nomads.ncep.noaa.gov/cgi-bin/filter_gfs_op50.pl)

run time

Directory: /gfs.20200930/06

\*\*NEW\*\* Select one file only (size in bytes)

gfs.t06z.pgrb2full.op50.f000 (163157071) ▾

forecast time

### GRIB Filter

For GRIB data you have to option to filter the data.

#### Extract Levels and Variables

You may select some or all levels and variables. The selections below represent common choices which may or may not be relevant to the files that you have selected. For example choosing RH (relative humidity) would be pointless in file of sea-surface temperatures. In addition, not all possibilities are allowed. For example, suppose you only want the virtual temperature at the tropopause at 01Z. In this case you'd have to transfer the entire file.

##### For GRIB-2 data only:

Select the levels desired:

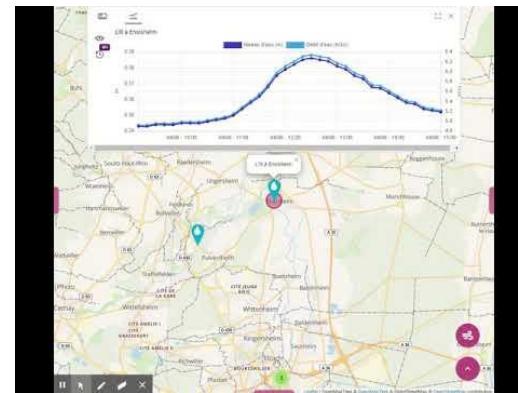
all  0-0.1 m below ground  0.1-0.4 m below ground  0.33-1 sigma layer  0.4-1 m below ground  0.44-0.72 sigma layer  0.4 m  0.72-0.94 sigma layer  0.995 sigma level  0C isotherm  1000 mb  100 m above ground  100 mb  10 m above ground  10 m above mean sea level  10 mb  120-90 mb above ground  125 mb  1-2 m below ground  150-120 mb above ground  150 mb  15 mb  175 mb  180-0 mb above ground  180-150 mb above ground  180 mb  2 m above ground  250 mb  255-0 mb above ground  2743 m above mean sea level  275 mb  2 m above ground  2 mb  3000-0 m above ground  300 mb  30-0 mb above ground  305 m above mean sea level  30 m above ground  30 mb  325 mb  350 mb  375 mb  3 mb  400 mb  40 m above ground  40 mb  425 mb  450 mb  4572 m above mean sea level  457 m above mean sea level  475 mb  500 mb  50 m above ground  50 mb  525 mb  550 mb  575 mb  5 mb  6000-0 m above ground  600 mb  60-30 mb above ground  610 m above mean sea level  625 mb  650 mb  675 mb  700 mb  725 mb  750 mb  775 mb  7 mb  800 mb  80 m above ground  825 mb  850 mb  875 mb  900 mb  90-60 mb above ground  914 m above mean sea level  925 mb  950 mb  975 mb  boundary layer cloud layer  convective cloud bottom level  convective cloud layer  convective cloud top level  entire atmosphere  entire atmosphere (considered as a single layer)  high cloud bottom level  high cloud layer  high cloud top level  highest tropospheric freezing level  low cloud bottom level  low cloud layer  low cloud top level  max wind  mean sea level  middle cloud bottom level  middle cloud layer  middle cloud top level  planetary boundary layer  PV=1.5e-06 (Km^2/kg/s) surface  PV=1.5e-06 (Km^2/kg/s) surface  PV=1e-06 (Km^2/kg/s) surface  PV=1e-06 (Km^2/kg/s) surface  PV=2e-06 (Km^2/kg/s) surface  PV=2e-06 (Km^2/kg/s) surface  PV=5e-07 (Km^2/kg/s) surface  PV=5e-07 (Km^2/kg/s) surface  PV=7e-07 (Km^2/kg/s) surface  PV=7e-07 (Km^2/kg/s) surface  PV=1e-06 (Km^2/kg/s) surface  PV=1e-06 (Km^2/kg/s) surface  PV=2e-06 (Km^2/kg/s) surface  PV=2e-06 (Km^2/kg/s) surface  PV=5e-07 (Km^2/kg/s) surface  PV=5e-07 (Km^2/kg/s) surface  PV=7e-07 (Km^2/kg/s) surface  PV=7e-07 (Km^2/kg/s) surface  surface  top of atmosphere  tropopause

##### Select the variables desired:

all  4LFTX  5WAVH  ABSV  ACPCC  ALBDO  APCP  CAPE  CDUVB  CFRZR  CICEP  CIN  CLWMR  CNWAT  CPOFP  CPRAT  CRAIN  CSNOW  CWAT  CWORK  DLWRF  DPT  DSWR  DVBF  DZDT  FLDCP  GFLX  GRLE  GUST  HGT  HINDEX  HLCY  HPBL  ICAIT  ICEC  ICEG  ICETK  ICML  ICSEV  LAND  LANDN  LFTX  LHFL  MSLET  OSMR  PEVPR  PLPL  POT  PRATE  PRES  PRMSL  PWAT  REFC  RH  RWMR  SHFTL  SNMR  SNOD  SOILW  SPFH  SUNSD  TCDC  TMAX  TMIN  TMP  TOZNE  TSOIL  UFLX  UGRD  U-GWD  ULWRF  USTM  USWRF  VFLX  VGRD  V-GWD  VIS  VRATE  VSTM  VVEL  VWSH  WATR  WEASD  WILT

## 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.test.kalisio.xyz/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](#)

- ➔ 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> ou utiliser <https://reqbin.com>
  - b. Test de l'accès sécurisé avec un token => <https://kano.test.kalisio.xyz>  
user: kalisio@kalisio.xyz  
password: Pass;word1
  - 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

## → Solutions

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

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

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

curl.exe --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.exe --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"
```

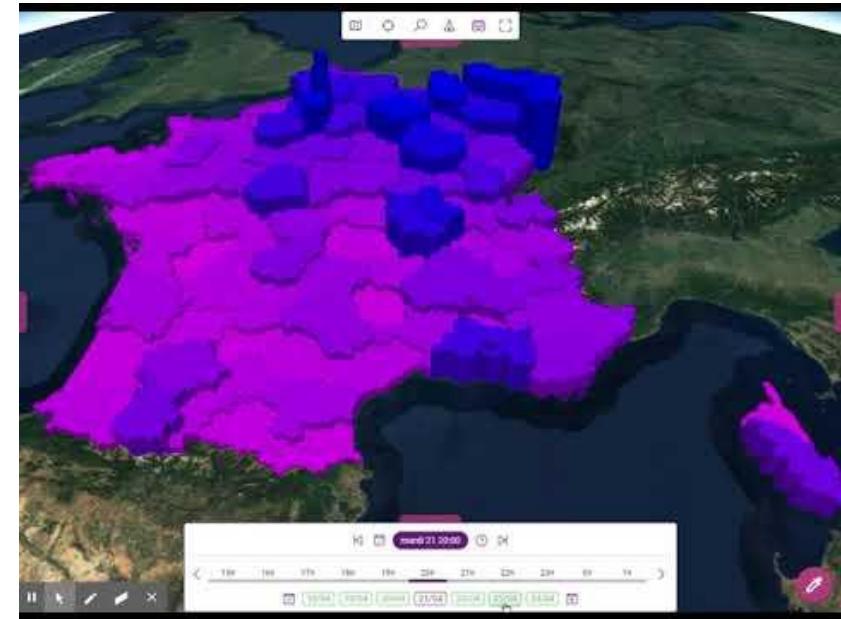
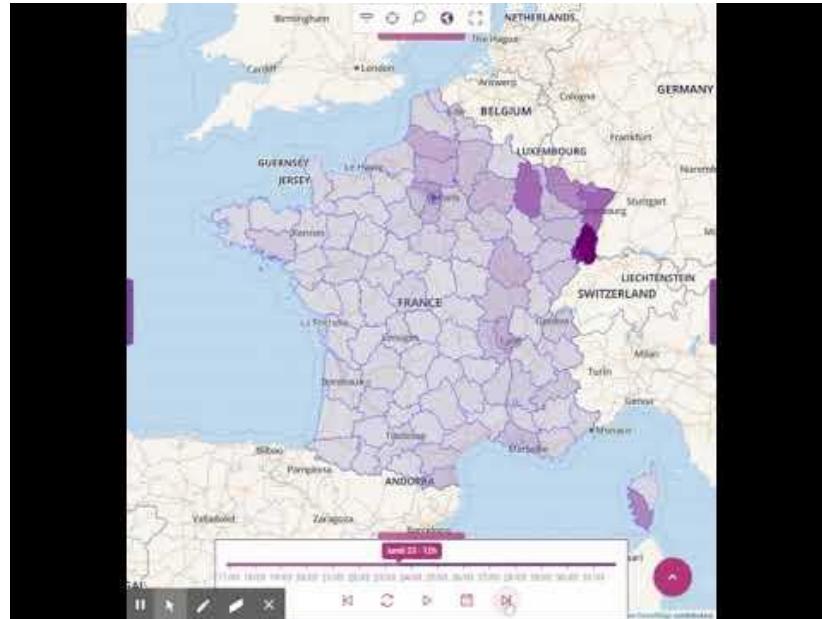


A high-angle, aerial photograph of a coastal city. The city is built on a peninsula, with a large body of water to the right and a green, hilly landscape to the left. The urban area is a mix of green spaces and grey buildings, with a prominent road or railway line running through it. The sky is overcast with a light grey mist.

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



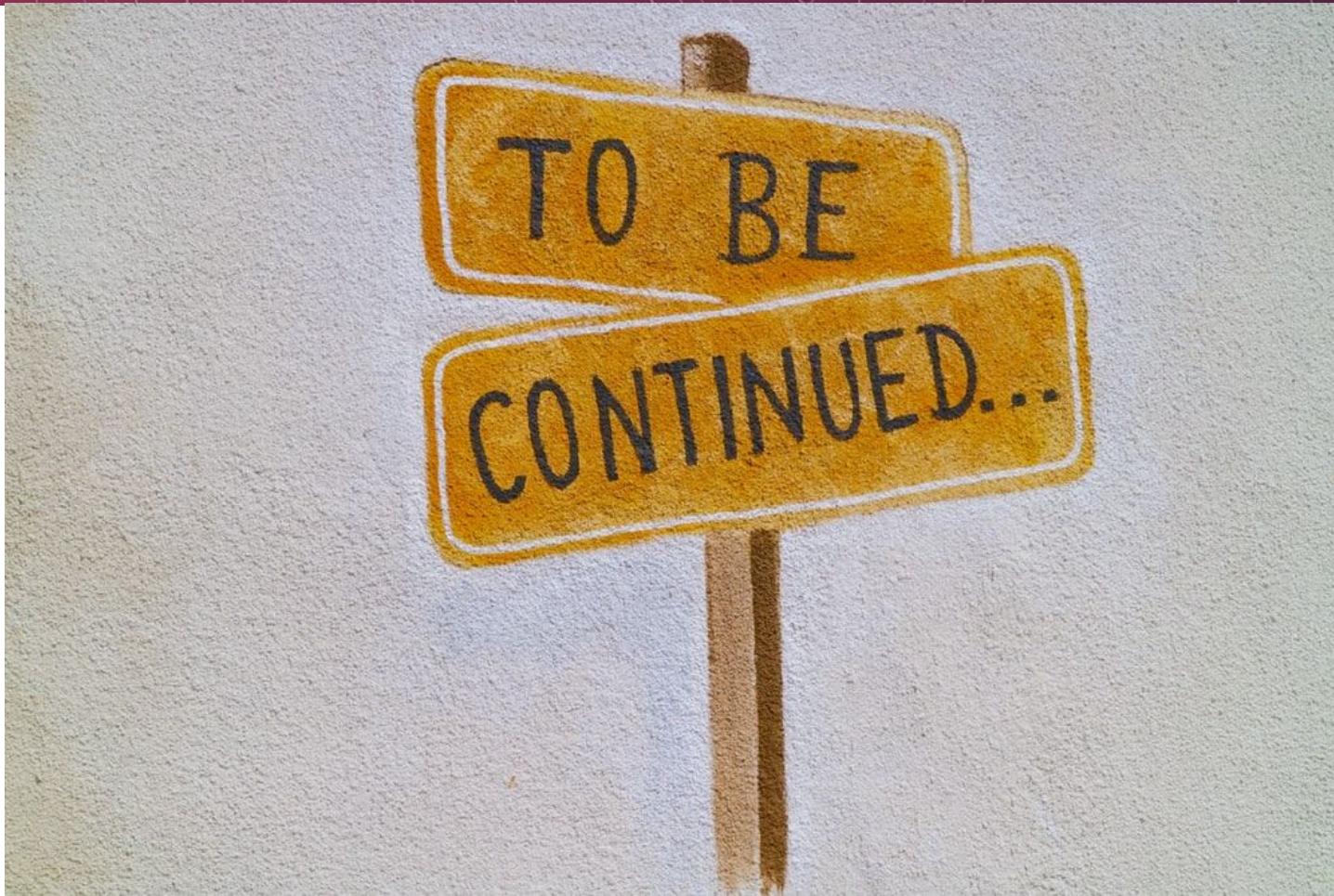
# Objectifs du fil rouge

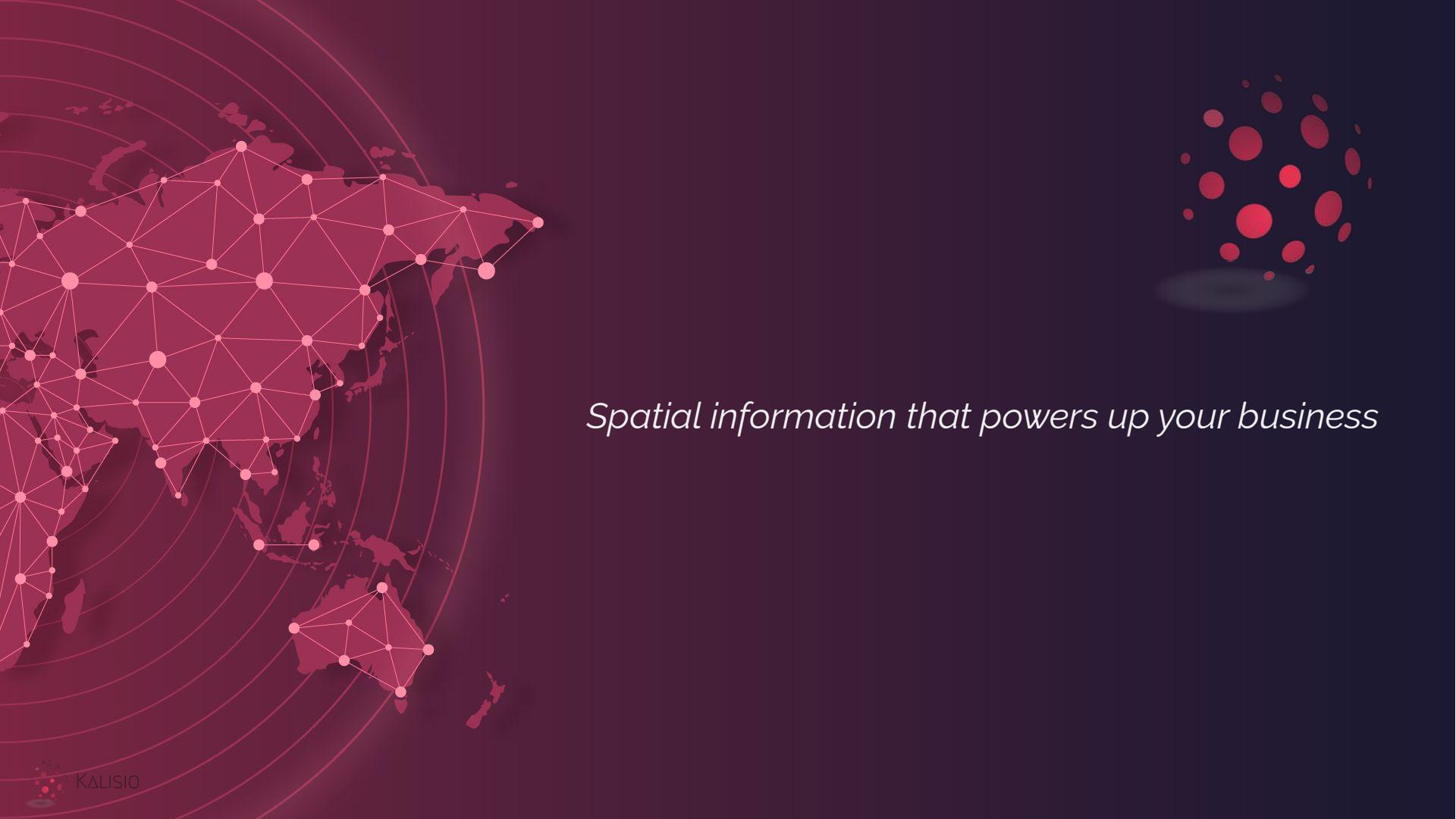
- Calculer le taux d'hospitalisation par département et le représenter sur une carte avec une symbologie adaptée
  - a. Recenser les données disponibles
    - nombre d'hospitalisations
    - limites administratives
    - population
  - b. Définir les traitements appropriés
    - extraction des données
    - transformation des données
  - c. Définir la représentation
    - symbologie
    - configuration de l'outil de visualisation
- Retrouvez le contenu des exercices sur <https://github.com/kalisio/cqp-geom>

# 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







*Spatial information that powers up your business*