

NoSQL, systèmes distribués et passage en production de projets Data

Thierry GAMEIRO MARTINS

Séances

1. Introduction et prise en main d'Onyxia
2. Le stockage des données en NoSQL
3. Les systèmes de traitement distribués
4. Le passage en production
5. Orchestration et pratique DevOps
6. Déploiement conteneurisé sous Kubernetes

Modalité d'évaluation

Objectifs

Présenter par groupe (de 4 ou 5 personnes) un POC (*Proof of Concept*) d'une chaîne de traitement de la données comme solution pour un client

- **Présentation des travaux** : exposé de 15 minutes
- **Questions/réponses** : 10 minutes de questions
- **Livrable** : slides détaillant votre solution, à envoyer avant le jour de la présentation

Barème de la présentation

- Présentation du sujet et de la problématique
- Explication des différentes étapes de traitements de la donnée (collecte, pré-processing, valorisation ou d'exposition de la donnée)
- Les briques technologiques utilisées et les raisons de leurs choix
- Comment la solution répond à la problématique
- Forme de la présentation

Sujets

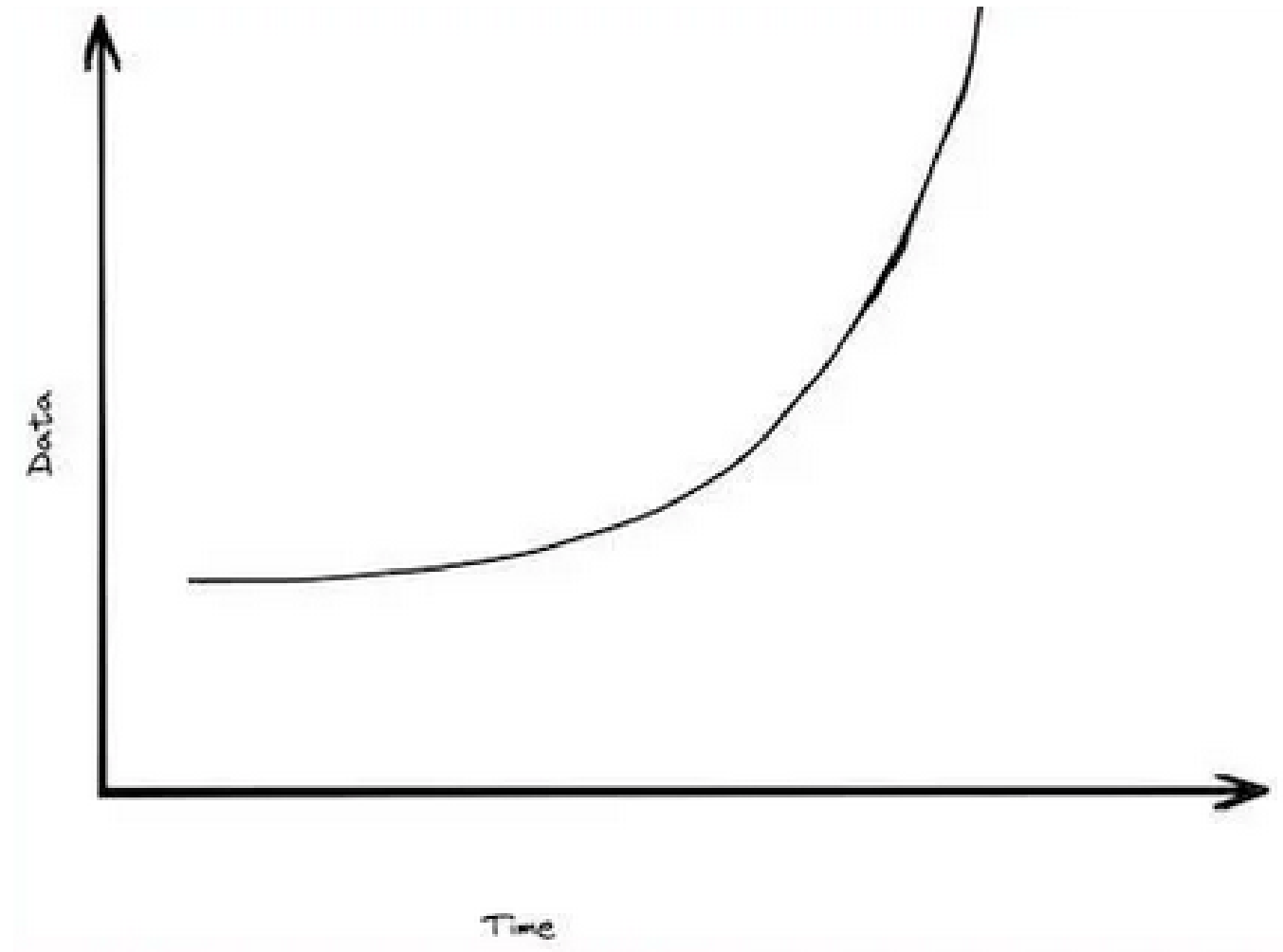
- 6 sujets sont proposés
- Possibilité de proposer son propre sujet mais à valider en amont
- Date limite pour le choix du sujet avant la quatrième séance

Quelques sujets

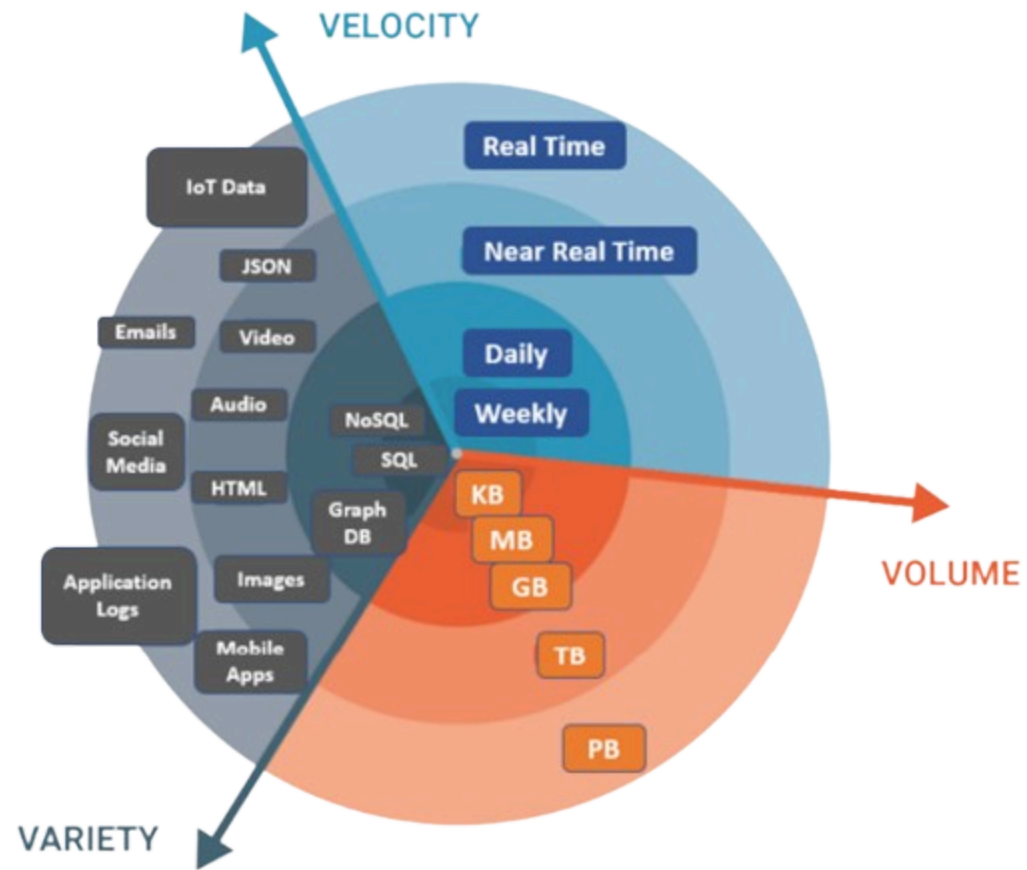
- Analyse de tweets <https://tinyurl.com/y5v4j8f6>
- Parsing de données IOT (Airparif) <https://tinyurl.com/y6xdod7p>
- Analyse des données de disponibilité des vélib à Paris <https://tinyurl.com/yykzr6hv>
- Analyse des données de subventions aux associations parisiennes
<https://tinyurl.com/mv5fkp5j>
- Analyse et comparaison des trajets uber / Taxi à New York
<https://tinyurl.com/y29k2jco>
- Système de recommandation de films <https://tinyurl.com/v2oynmf>

Introduction

La massification des données



Définition du Big Data par les 3V



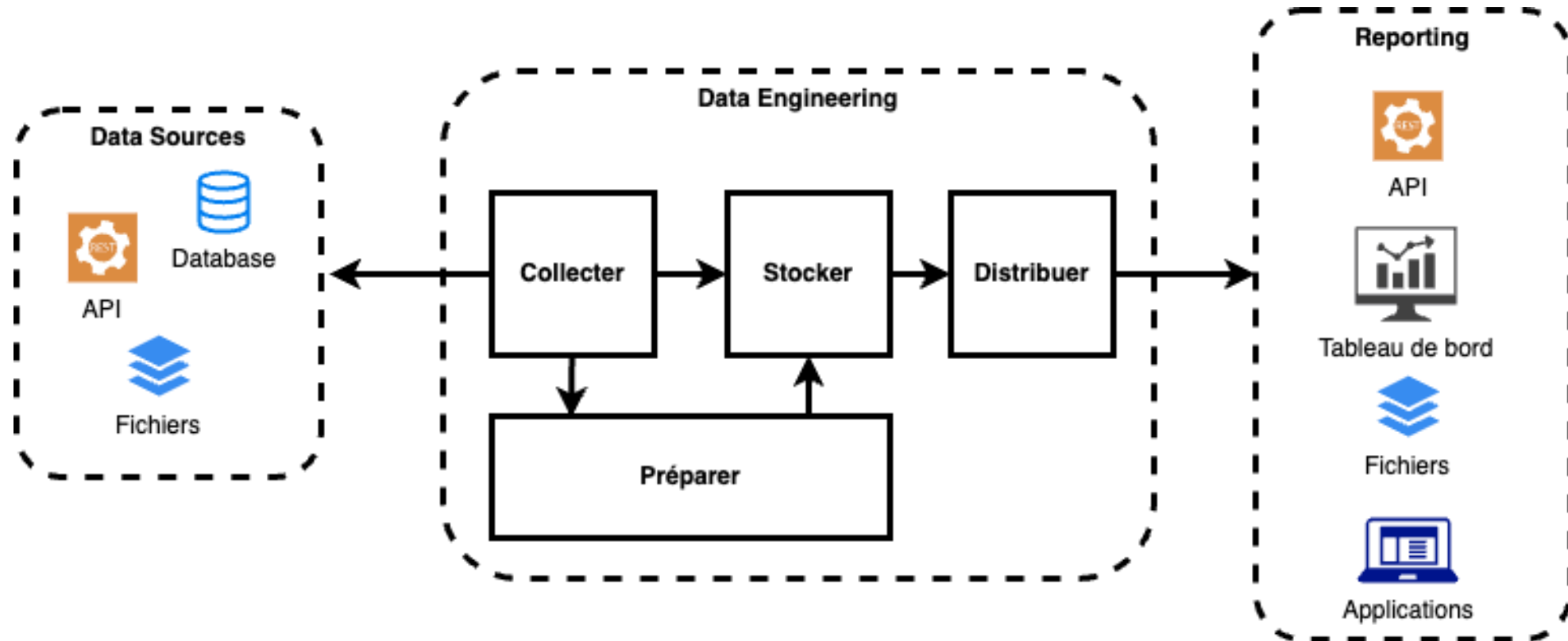
Qu'est ce que le *Data Engineering* ?

- **Gérer le cycle de vie de la données** (collecte, traitement et mise à disposition)
- **Garantir** la qualité, l'utilisabilité et la gouvernance des données
- **Concevoir et maintenir** les infrastructures de données (base de données, briques d'ETL, exposition de données, etc.)

Pourquoi est-ce important ?

- Ne pas se concentrer seulement sur les algorithmes mais aussi au cas d'usage
- Est le socle du *Machine Learning* pour les Data Scientist
- Importance de savoir récupérer, traiter et industrialiser des données pour ses projets

Les étapes d'une *pipeline* de données



Concept : Qu'est qu'une API

Une API (application programming interface ou « interface de programmation d'application ») est une interface logicielle qui permet de « connecter » un logiciel ou un service à un autre logiciel ou service afin d'échanger des données et des fonctionnalités.

CNIL

API RESTful : API conforme au style d'architecture REST

- Communication via le protocole HTTP
 - On requête un endpoint (API Sirene)
 - Avec des méthodes HTTP (GET, POST, etc.)
- Spécification OpenAPI

Définition peu informative car scikit-learn, Docker, etc. sont des APIs

La collecte des données

Les méthodes de collecte de données varient en fonction de la source, du volume, de la vitesse et des types de données

- **Collecte par API** : récupération de données à la demande en utilisant des requêtes HTTP

Outils : requests, Postman, HTTPie

Prendre en compte le rate limiting, pas adapté pour de gros volumes, utile pour de l'enrichissement

- **Extraction de fichiers** : importation de fichiers de différents formats (CSV, JSON, XML, images, vidéos, etc)

Outils : Apache Nifi, Airflow, Talend

Solution simple, risque sur le parsing, s'adapter au format

Concept : les types de données

Structuré avec un format fixe (table relationnelles, csv , etc.)

```
firstname,lastname,birthdate,address  
Leonardo,DiCaprio,19741111,NY 2 River Terrace
```

Customer

CustomerID	CustomerName	LastName	Country	Age	Phone
1	Shubham	Thakur	India	23	xxxxxxxxxx
2	Aman	Chopra	Australia	21	xxxxxxxxxx
3	Naveen	Tulasi	Sri lanka	24	xxxxxxxxxx
4	Aditya	Arpan	Austria	21	xxxxxxxxxx
5	Nishant. Salchichas S.A.	Jain	Spain	22	xxxxxxxxxx

Semi-structuré avec une structure, mais flexible (JSON , XML)

```
{  
  "firstname": "Leonardo",  
  "lastname": "DiCaprio",  
  "birthdate": 19741111,  
  "address":  
    {"city": "NY",  
     "street": "2 River Terrace"}  
}
```

Non-structuré sans format de données (image, texte, vidéos, etc.)



Text



Image



Video



Audio



Email



Social Media



Sensor

- **Collecte en temps réel** : récupération des données en continue

Outils : Apache Kafka, Apache Flink

Orienté détection de fraude, surveillance de systèmes

- **Webscraping** : collecte des données des pages web

Outils : BeautifulSoup, Scrapy, Selenium

Maintenance difficile, très sensible aux changements, cadre légal encore flou

- **Base de données** : mécanismes d'extraction des données de bases de données transactionnelles

Outils : Sqoop, Debezium, Talend, Spark, Apache Flink, Airflow

Synchronisation incrémentale (CDC) ou totale, processus souvent lourd et nécessite de la puissance de calcul

Stockage de la donnée

Le stockage des données peut être de différentes formes

- **Bases de données relationnelles :**
utilisées pour des données structurées
avec des relations bien définies entre
tables

Exemple : MySQL, PostgreSQL, Oracle,
SQL Server

*Applications pour du requêtage de
système opérationnel (API, web
application, etc.)*

- **Bases de données NoSQL :** utilisées
pour les données semi-structurées,
offrant une grande flexibilité

Exemple : MongoDB, Cassandra,
ElasticSearch, Redis

*Applications web ou mobiles en constante
évolution, IoT, réseaux sociaux*

- **Data Warehouse** : stockage de données organisé et optimisé pour du calcul analytique
 - coûteux à mettre en place
 - limité aux données structurées

Exemple : Amazon Redshift, Google BigQuery, Snowflake, VerticaDB

Stockage de grandes quantités de données, efficace sur des requêtes analytiques

- **Data Lake** : stockage de données au format brute de tous type
 - permet les sources variés
 - moins coûteux à mettre en place

Exemple : HDFS, Amazon S3

Utile pour stocker la donnée brute

Concept : OLAP vs OLTP

	OLAP	OLTP
Définition	<i>OnLine Analytical Processing</i>	<i>OnLine Transactional Processing</i>
Objectif	Traiter le maximum de lignes pour de l'analyse	Lire et modifier les données le plus rapidement possible
Requête	Requêtes simples	Requêtes complexes
Temps	En millisecondes	Entre la seconde et la minute
Stockage	Base relationnelle	Data Warehouse
Source	Source de la donnée	Bases OLTP

Préparer les données

Traitement par lots : les données sont traitées en gros volumes, généralement à des intervalles réguliers

Exemples : Apache Spark, Apache Hadoop, Data Warehouse en SQL

Traitement de données historiques, création de rapports, pipelines journalières

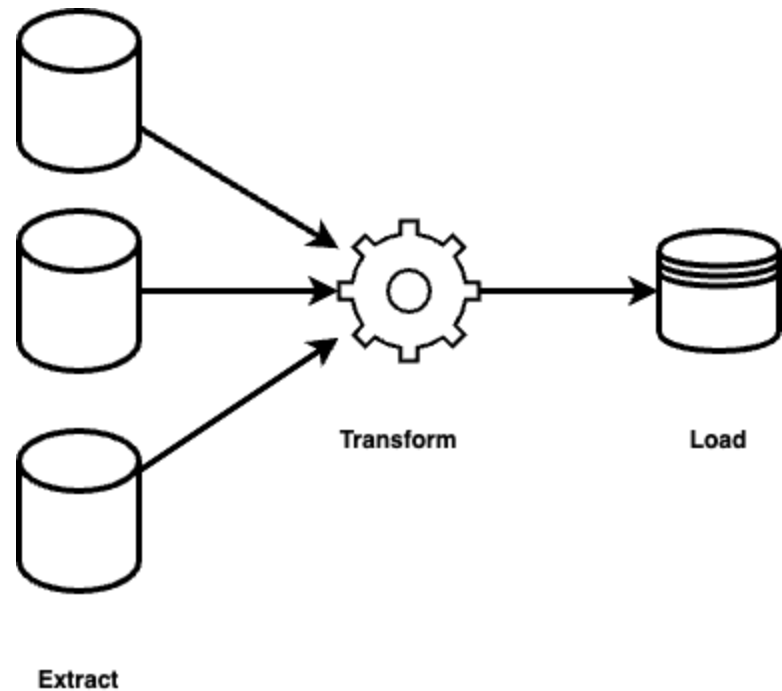
Traitement en temps réel : les données sont traitées en continu, au fur et à mesure qu'elles sont générées.

Exemples : Apache Kafka, Apache Flink

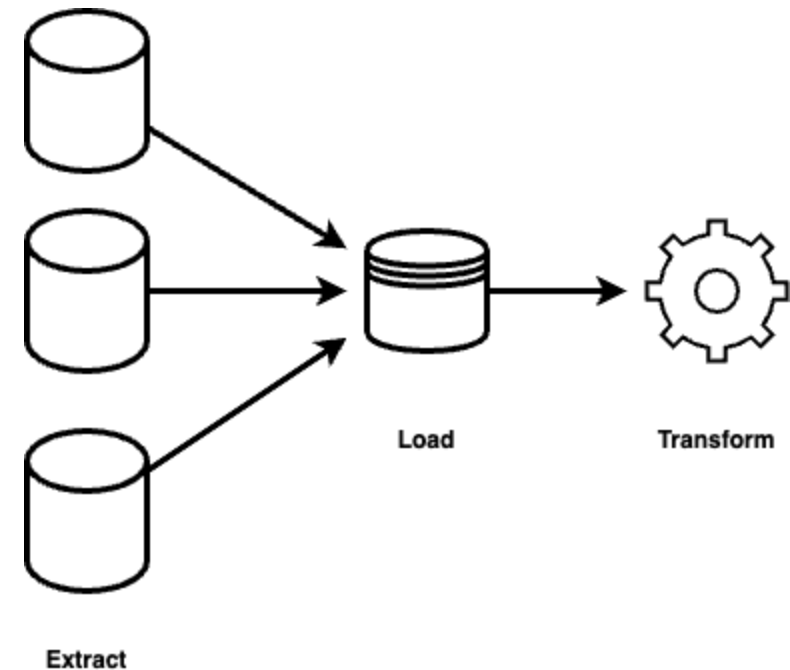
Analyse de données en temps réel (ex. détection de fraude, suivi des événements IoT)

Concept : ETL vs ELT

ETL : les transformations ont lieu **AVANT** le chargement des données



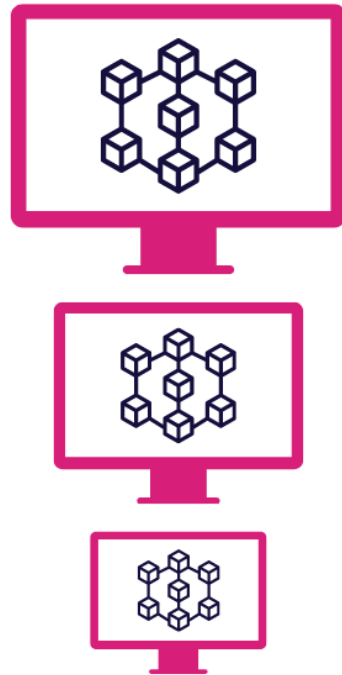
ELT : les transformations ont lieu **APRÈS** le chargement des données



Concept : Scalabilité Horizontale vs Verticale

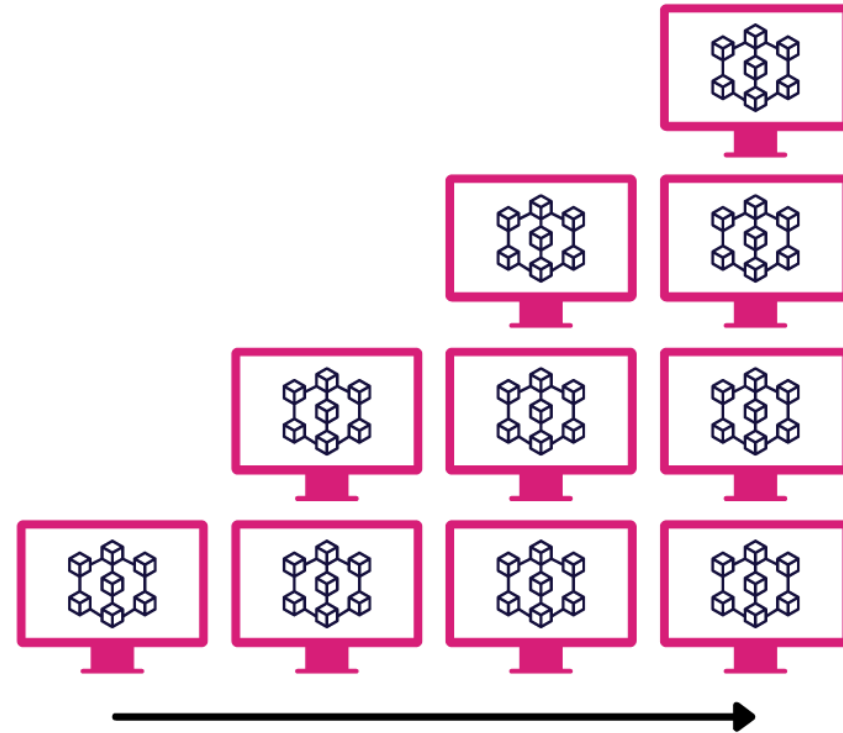
Scalabilité verticale

Augmenter la taille des entités (RAM, CPU, etc.)



Scalabilité horizontale

Ajouter des entités

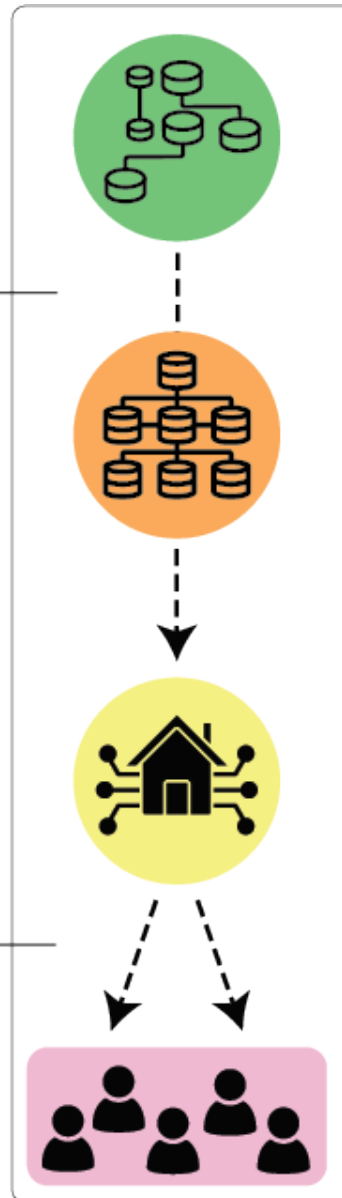


Concept : Architecture Data Lake vs Data Warehouse

Data Warehouse

Incoming data is cleaned then organized into a single, consistent schema before being moving to the data warehouse

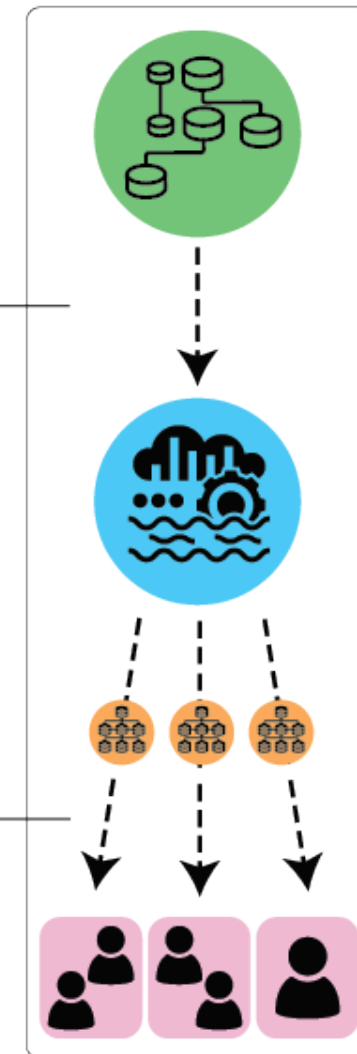
Analysis is done directly on the curated warehouse data



Data Lake

Incoming data goes into the lake in its raw form

Data is selected and organized based on individual needs



Diffuser et exposer les données

Diffuser les données sous la forme de **tableaux de bords** exploitables (*Data Analysis*)

Retravailler la donnée afin de **la redistribuer** sous une forme compréhensible (via API, ou fichiers)



Prise en main d'Onyxia

Onyxia est une application web développée par l'INSEE qui permet aux data scientists et data ingénieurs de :

- Lancer des services (éditeur de code, base de données, outils d'orchestration, etc.)
- D'explorer des données et d'entraîner des modèles
- Déployer des applications
- Se former en Data Science

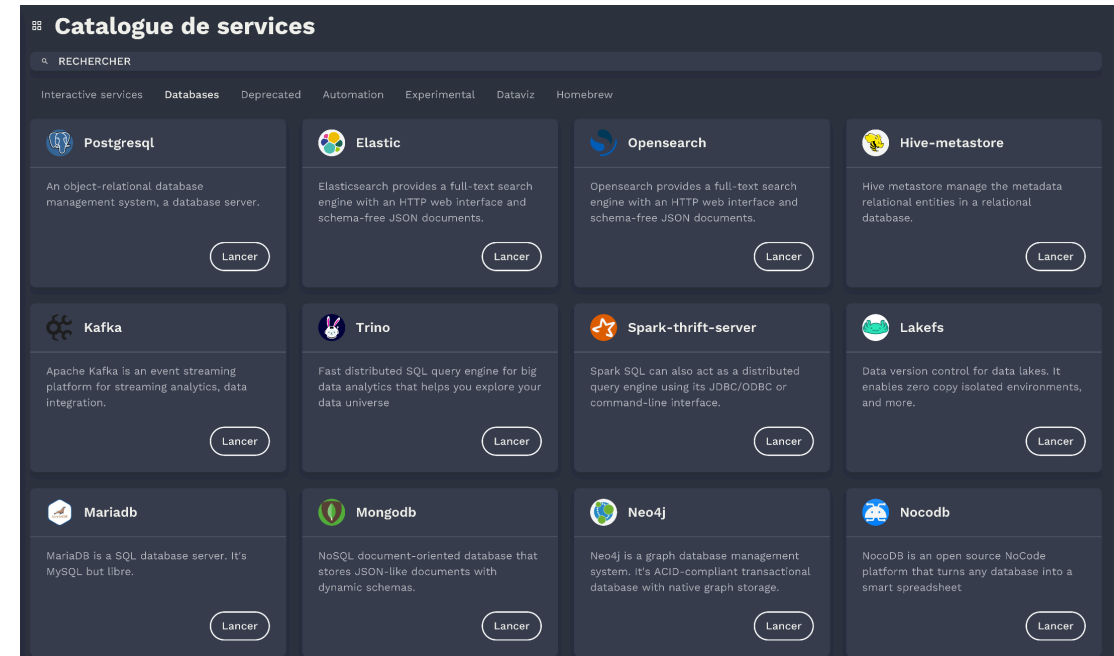


Lien pour se connecter disponible sur :
<https://datalab.sspcloud.fr>

Catalogue de services

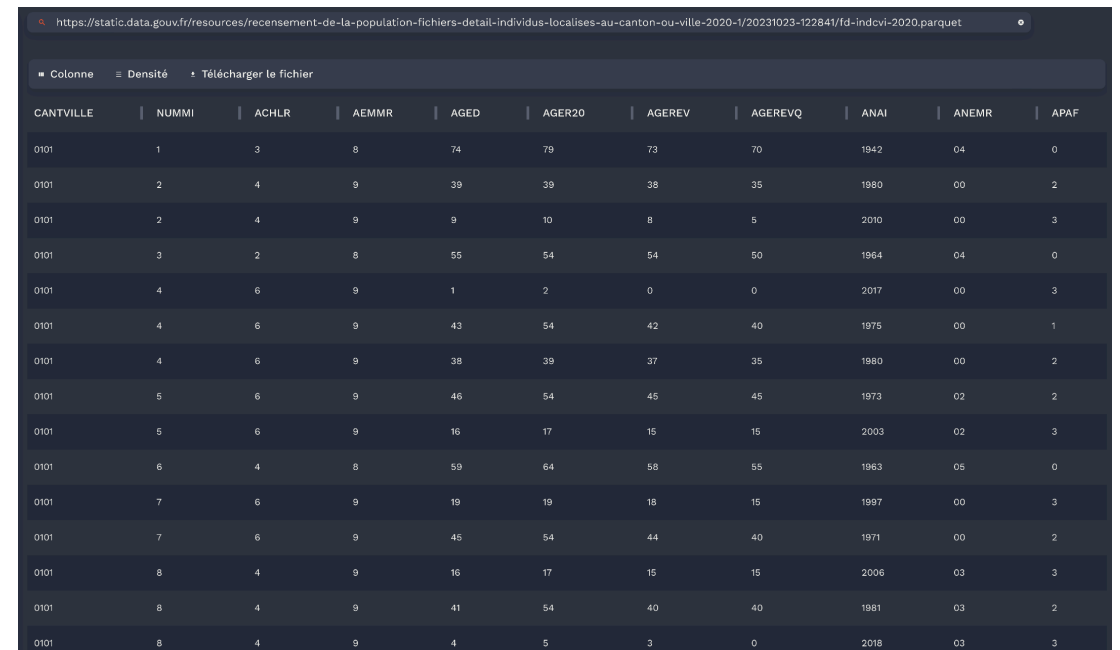
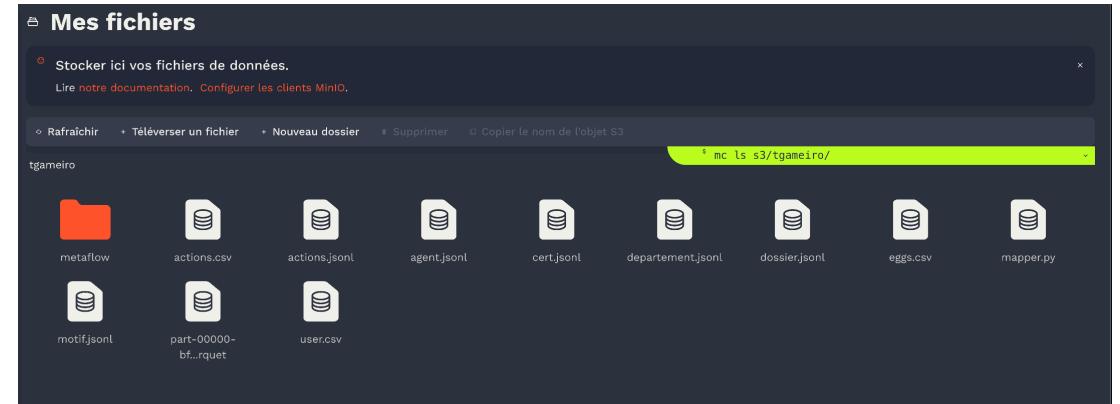
- **Base de données** (MongoDB, Elastic, PostgreSQL, etc.)
- **Outils d'orchestration** (ArgoCD, Argo Workflow, MLFlow etc.)
- **Environnement de développement** (Jupyter, VSCode, RStudio)
- **Visualisation** (Superset, Redash)

Configurable (initscript, ressources, stockage, git, secret, etc.)



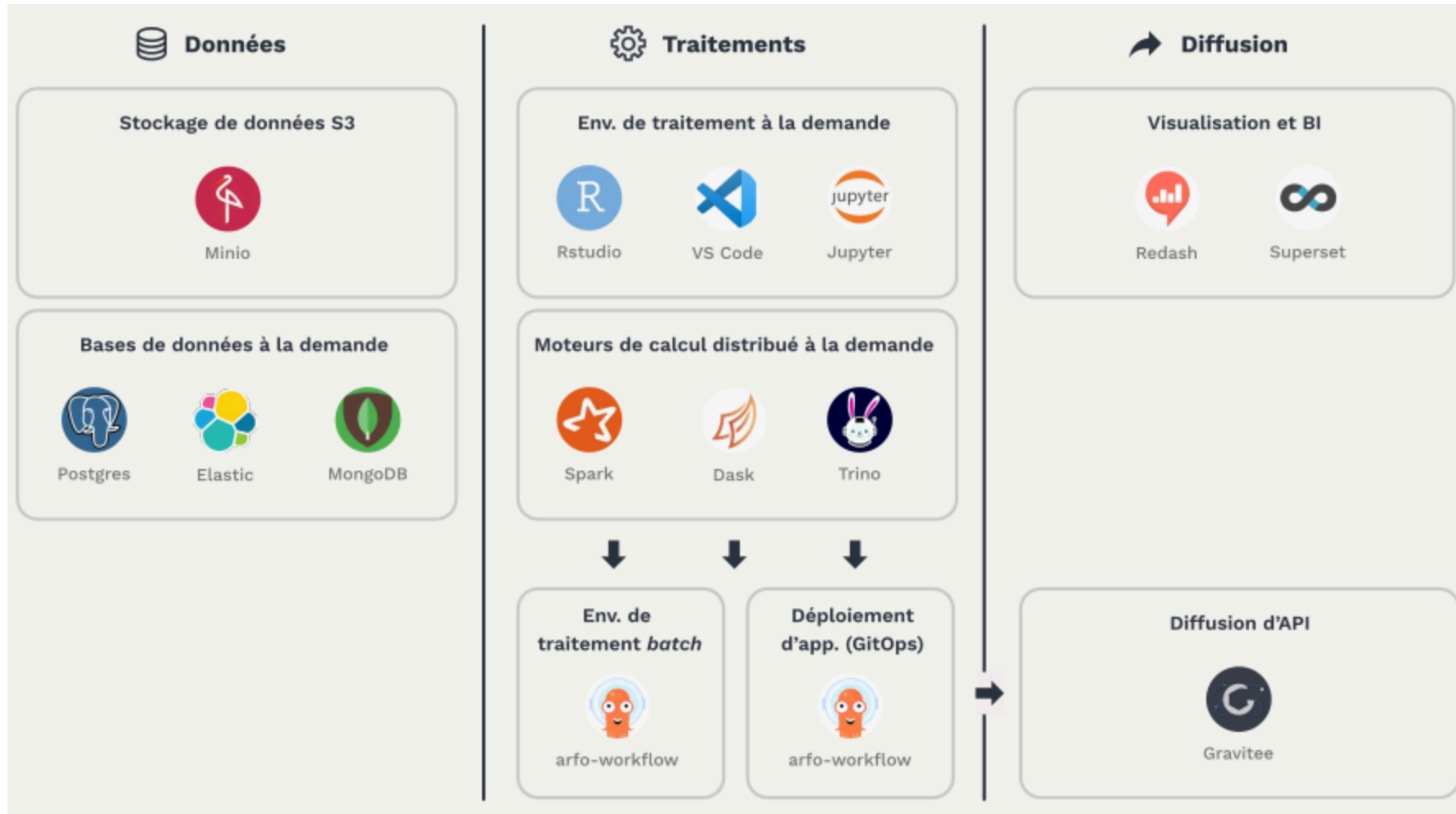
Stockage des secrets et fichiers

- Un explorateur S3 est disponible pour envoyer des fichiers ou récupérer ses fichiers depuis le Datalab
- Un outil d'exploration de données pour visualiser directement des fichiers au format `.csv` ou `.parquet`
- Un gestionnaire de secret pour les injecter dans les services
- Gestion des configurations (git, customisations de services, etc.)



CANTVILLE	NUMMI	ACHLR	AEMMR	AGED	AGER20	AGEREV	AGEREVQ	ANAI	ANEMR	APAF
0101	1	3	8	74	79	73	70	1942	04	0
0101	2	4	9	39	39	38	35	1980	00	2
0101	2	4	9	9	10	8	5	2010	00	3
0101	3	2	8	55	54	54	50	1964	04	0
0101	4	6	9	1	2	0	0	2017	00	3
0101	4	6	9	43	54	42	40	1975	00	1
0101	4	6	9	38	39	37	35	1980	00	2
0101	5	6	9	46	54	45	45	1973	02	2
0101	5	6	9	16	17	15	15	2003	02	3
0101	6	4	8	59	64	58	55	1963	05	0
0101	7	6	9	19	19	18	15	1997	00	3
0101	7	6	9	45	54	44	40	1971	00	2
0101	8	4	9	16	17	15	15	2006	03	3
0101	8	4	9	41	54	40	40	1981	03	2
0101	8	4	9	4	5	3	0	2018	03	3

Un catalogue de services complet pour les projets de data science



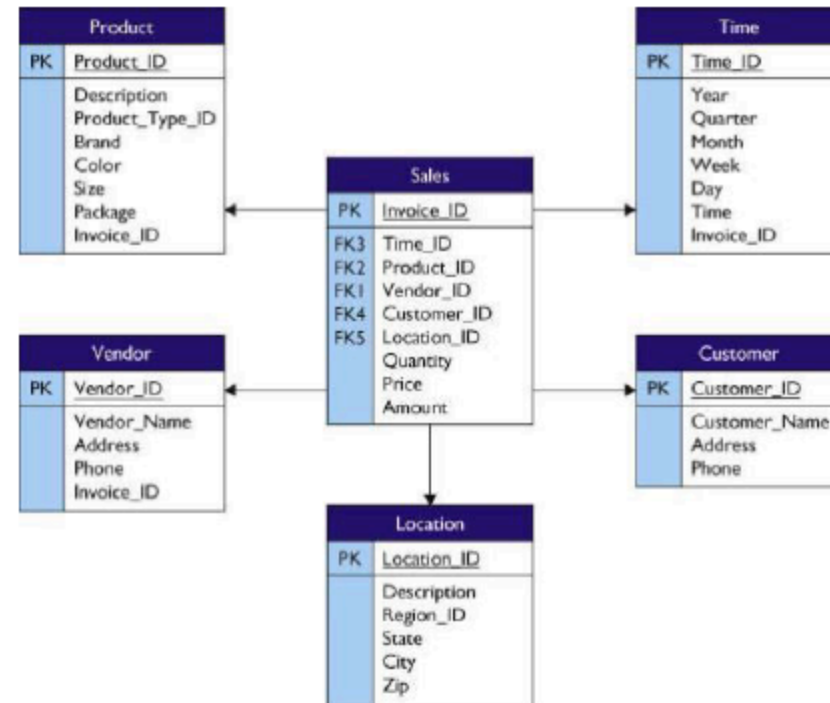
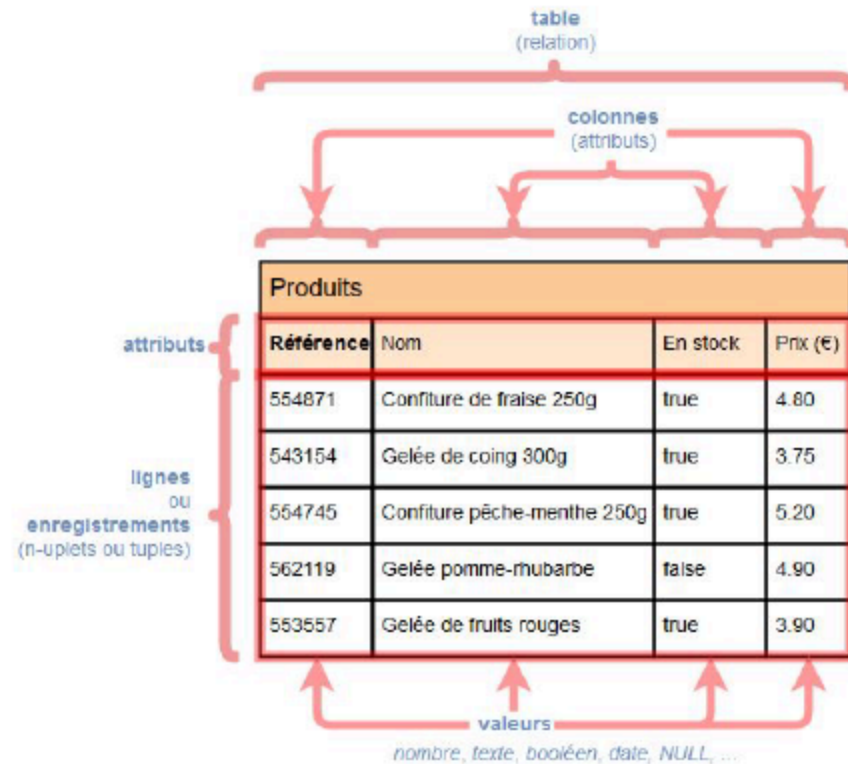
Rappel : les bases de données relationnelles

Les base de données relationnels (SGBDR)

- Logiciel de stockage et de gestion de données régit par des **transactions**
- Organise les données sous la forme d'un **schémas relationnel** visant à éviter la redondance
- Interrogeable par du **SQL** : *Structured Query Langage*



Schéma relationnel



Les contraintes ACID des SGBDR

- **Atomicité** : Une transaction se fait au complet ou pas du tout, sinon remettre les données dans l'état où elles étaient (rollback)
- **Cohérence** : Tout changement doit être valide selon toutes les règles définies en base (contraintes d'intégrité)
- **Isolation** : Toute transaction doit s'exécuter comme si elle était la seule sur le système. Aucune dépendance possible entre les transactions
- **Durabilité** : Lorsqu'une transaction a été confirmée, elle demeure enregistrée

Langage SQL

- **DDL** (Data Definition Language) : CREATE , ALTER , DROP
- **DML** (Data Manipulation Language) : SELECT , INSERT , UPDATE , DELETE , JOIN
- **DCL** (Data Control Language) : GRANT , REVOKE
- **TCL** (Transaction Control Language) : COMMIT , ROLLBACK , SAVEPOINT