



# Big Data : enjeux, stockage et extraction

## Module R6.01

2025



# A propos de ce module

Ce cours est une introduction aux problématiques d'**exploitation des données** sur les plateformes modernes, dans le contexte aujourd’hui standard de “**Big Data**”.

Il aborde des sujets qui vont concerner plusieurs profils au sein de l’entreprise : data engineer, data scientist, architecte, mais aussi data manager... et les autres modules du semestre lui sont bien sûr complémentaires.



R6.01  
**Big Data : enjeux,  
stockage et extraction**



# Déroulement du module R6.01

01

**Introduction**

02

**Anatomie des données**

03

**Architecture d'une plateforme**

04

**Traitements de données**

05

**Le cloud**

06

**Pour un usage responsable des données**

1

**Eval écrite**  
(fin du 2ème CM)

3

**TP/TD non notés**

1

**1 TP/TD noté**



# Who am I?



Thomas VIAL

[thomas.vial@octo.com](mailto:thomas.vial@octo.com)

+33 6 28 50 07 64

- **Consultant** chez OCTO Technology (filiale d'Accenture) depuis 2007
  - > On fait officiellement du Big Data depuis 2013 chez OCTO !
- Au sein l'équipe **SmartIndustry**, qui a pour vocation d'apporter des solutions innovantes de data et d'IA aux clients industriels d'OCTO
- **Profil hybride** Architecte de SI, Architecte de données, Data engineering, Data science
- Certifié Hadoop et Google Cloud Professional Architect



01

# Introduction



# Qu'entend-on par “Big Data” ?

- “Big data” (ou “mégadonnées”) fait référence à des données qui sont trop volumineuses pour être traitées facilement
- Il est souvent résumé par des séries de “V” dont les 3 principaux :
  - > **Volume** : quantité importante de données, croissante par accumulation au fil du temps
  - > **Variété** : formats très différents, allant du très structuré au complètement déstructuré
  - > **Vélocité** : prise en compte rapide (selon le besoin) pour en tirer le maximum de valeur
- Ces 3 (et quelque) “V” sont des frontières techniques pour les serveurs au début des années 2000, période où l’usage du web explose : pages, blogs, commentaires, réseaux sociaux, ...
- Les premiers utilisateurs du big data viennent donc du web, ils s’appellent Google, Amazon, Facebook (les GAFA), mais aussi Yahoo!, eBay, LinkedIn, Twitter (X), etc.



# Qu'entend-on par “Big Data” ?

- Les entreprises ont adopté les pratiques de ces “géants” du web pour enfin exploiter les données dispersées dans leur système d’information : bases de données, emails, procédures, bases documentaires, ...
- Puis le cloud est arrivé, avec des solutions clef en main pour traiter les données “Big data”
- **En 2025**, le cloud est pour beaucoup la solution par défaut, capable de **stocker et traiter** tout type de données, “big” ou pas



# Quelques chiffres et ordres de grandeur



<https://www.g2.com/articles/big-data-statistics>



# Avantages par rapport à l'informatique “traditionnelle”

Business Intelligence,  
Analytique

Data mining

Internet des objets  
(IoT)

Machine learning

IA générative

Brassage rapide de très grands volumes de données  
Prise en charge des données non structurées  
Croisement de nombreuses sources  
Interrogation en langage naturel

Mémorisation d'historiques profonds  
Corrélations d'un grand nombre de capteurs  
Contextualisation d'événements

Apprentissage sur des volumes de données “sans limite”  
Détection de motifs rares (anomalies, incidents, ...)  
Réduction des intervalles de confiance  
Prise en charge de données non structurées (texte, son, image, vidéo, ...)  
Apprentissage de gros modèles (variables, paramètres)  
Nouveaux algorithmes de plus en plus performants



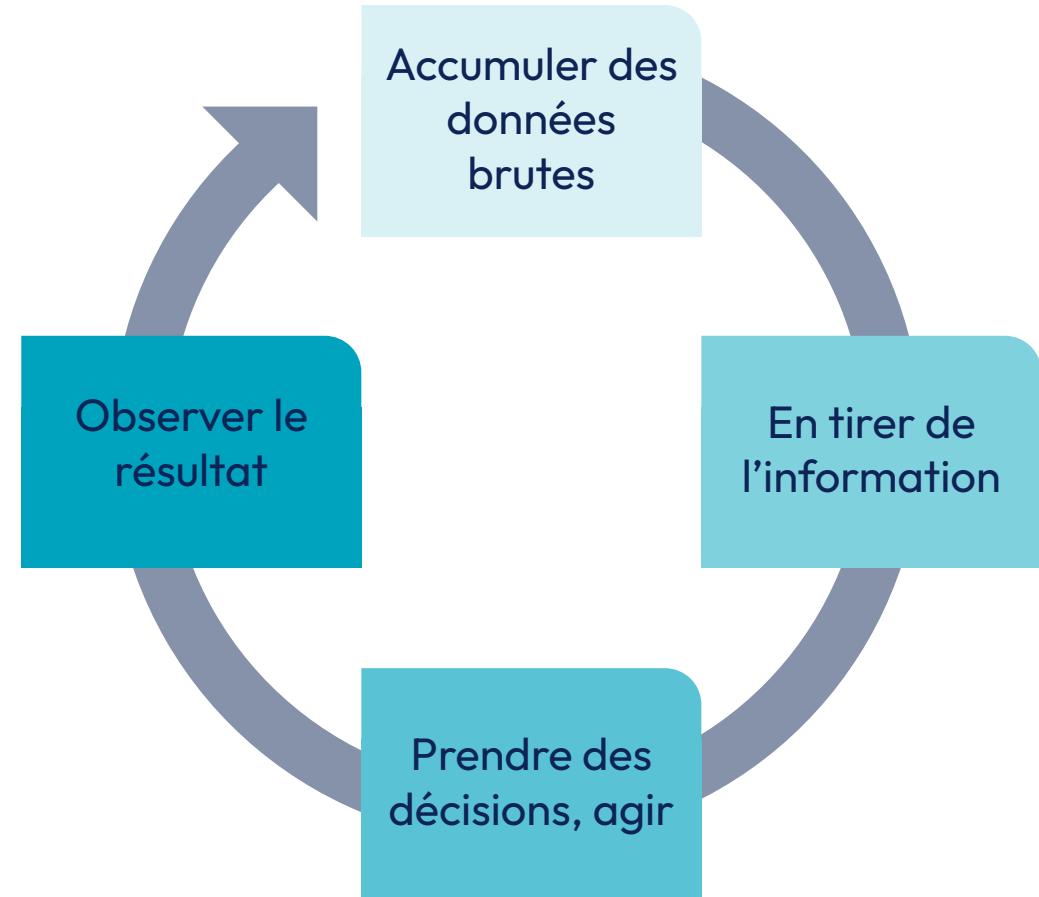
## Notion de plateforme



# Cycle de vie des données

Un système (big) data **centralise** de la donnée brute, la **transforme** en informations, qui servent d'appui à la **prise de décision** (automatique ou humaine).

Les **actions** consécutives à la décision ont un effet reflété dans de nouvelles observations, sous forme de données qui alimentent le cycle.

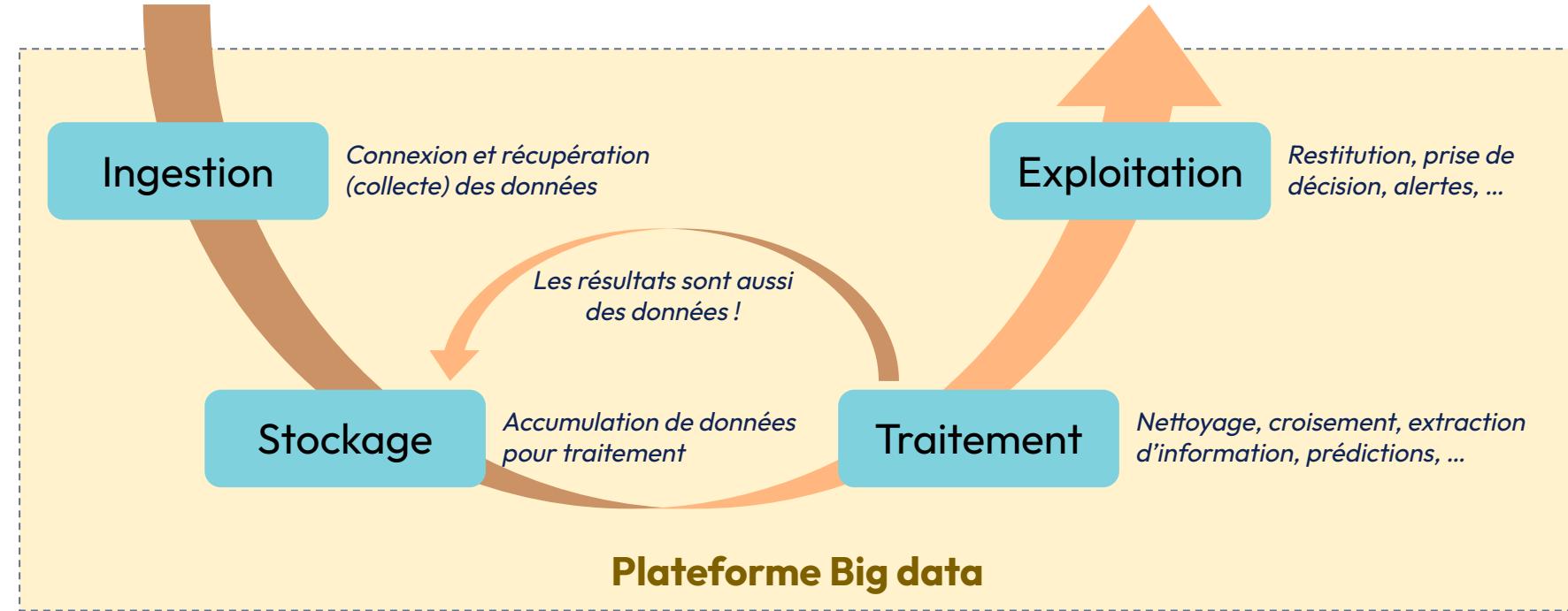




# Plateforme Big data



Environnement produisant et consommant des données  
(site web, application smartphone, capteurs, ...)





# Quelques applications

**Citer des applications concrètes de Big data**





# Quelques applications

## Commerce et services aux particuliers

- Profilage et rétention client
- Recommandation de produits
- Personnalisation d'expérience





# Quelques applications

## Banque, Finance, Assurance

- Gestion de budget personnalisée
- Détection de fraude
- Lecture de documents administratifs
- Actuariat



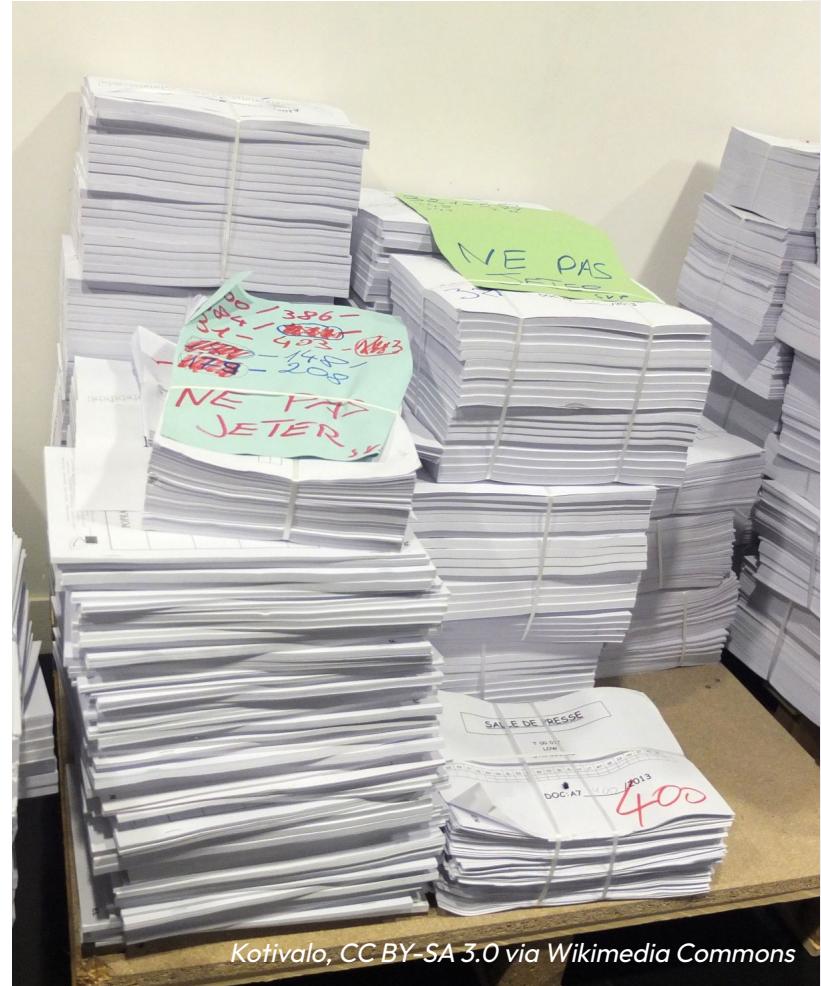
Nick Youngson CC BY-SA 3.0 Pix4free.org



# Quelques applications

## Documentation (législatif, juridique, procédures, ...)

- Moteur de recherche et bases de connaissances
- Assistants virtuels (chatbots)



Kotivalo, CC BY-SA 3.0 via Wikimedia Commons



# Quelques applications

## Industrie

- Automatisation de contrôle qualité
- Optimisation de production
- Réduction d'émissions polluantes
- Détection de situations à risque (fuite, ...)



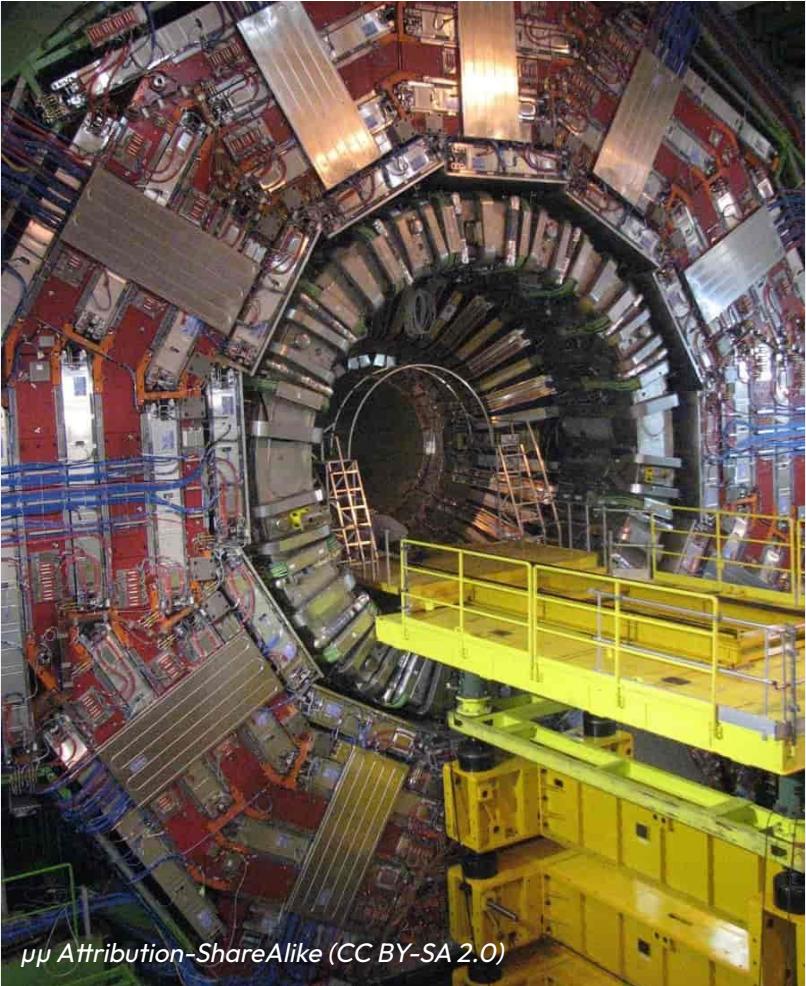
Bruno Glätsch



# Quelques applications

## Sciences fondamentales

- Analyse de sondages géophysiques
- Analyse d'ADN/ARN (génomique, métagénomique)
- Dépouillement de résultats d'expériences (ex. boson de Higgs)





02

# Anatomie des données

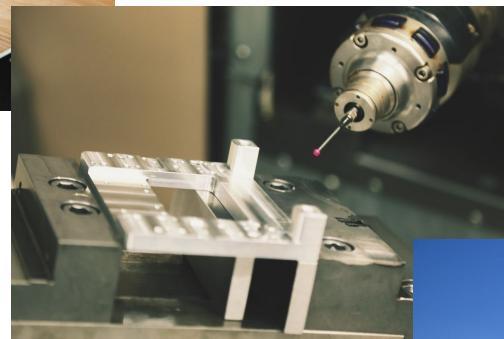


# Provenance et types de données (exemples)



## Humains

- Réseaux sociaux
- Formulaires, documentation d'entreprise
- Navigation & usage de sites ou applications



## Machines (y compris ordinateurs)

- Traces d'exécution
- Rapports d'erreur
- Consommation énergétique



## Capteurs

- Météo & environnement
- Caméras
- Puces RFID



# Données : Modélisation et format

## Concept

## Modélisation :

Comment la donnée est-elle décrite ?

## Format :

Comment la donnée est-elle physiquement stockée (fichiers) ?



### Mesure ::=

- *date* (timestamp AAAAMMJJ)
- *longitude* (entier)
- *latitude* (entier)
- *valeur* (flottant)

Champs & types

```
date;longitude;latitude;valeur  
20240112;157;-23;8.9193  
20240112;157;-22;8.4410  
20240113;157;-23;9.0012
```

CSV

```
01B6D7AA89E7C710BF611...
```

ZIP



# Modélisation

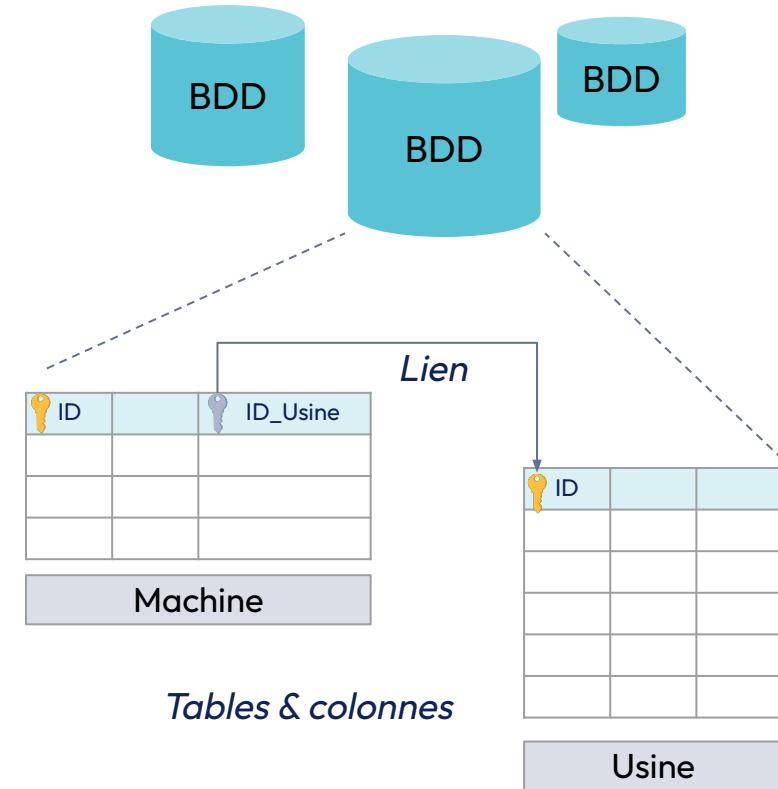
**Quels types de modélisation connaissez-vous ?**





# Modélisation : Zoom sur le modèle relationnel

- Le **modèle relationnel** est celui des bases de données SQL : PostgreSQL, MySQL, SQLite, ...
- Il est très présent dans les systèmes informatiques des entreprises
- Caractéristiques :
  - > Notion de **tables** composées de **colonnes**
  - > Liens entre les tables via la notion de clef primaire / clef étrangère

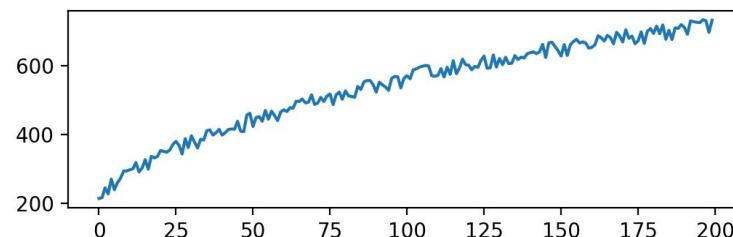




# Modélisation : Zoom sur les séries temporelles

- Une **série temporelle** (*time series*) est une suite de points de mesure, modélisée par au moins un timestamp et une valeur
- Le timestamp peut être régulier ou irrégulier
- Opérations intéressantes :
  - > Rééchantillonnage (*resampling*)
  - > Prédiction des prochains points (*forecast*)
  - > Elimination de valeurs aberrantes
  - > Lissage
  - > Analyse fréquentielle
  - > ...

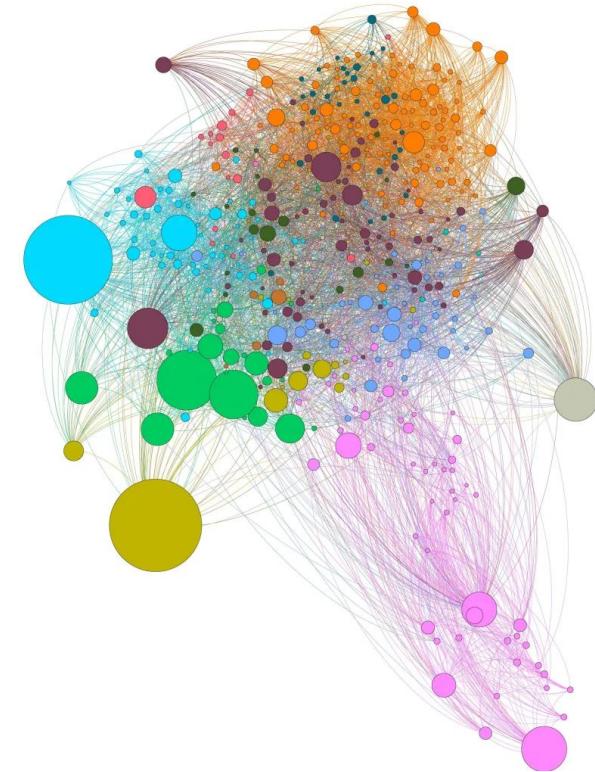
timestamp	machine	valeur (°C)
2024-01-12 17:00:00	Four #2	412.7
2024-01-12 17:00:10	Four #2	413.1
2024-01-12 17:00:20	Four #2	-19901.8
2024-01-12 17:00:30	Four #2	419.2
...	...	...





# Modélisation : Zoom sur les graphes

- ➊ Les **graphes** (ou réseaux) représentent des objets ou des concepts (*noeuds*), liés entre eux par des relations (*arcs*)
- ➋ Opérations intéressantes :
  - > Parcours en largeur (BFS) ou en profondeur (DFS)
  - > Recherche de chemins
  - > Calculs d'influence de noeuds (ex. PageRank)
  - > Recherche de sous-graphes
  - > ...
  - > Voir des modèles de deep learning (GNN) !

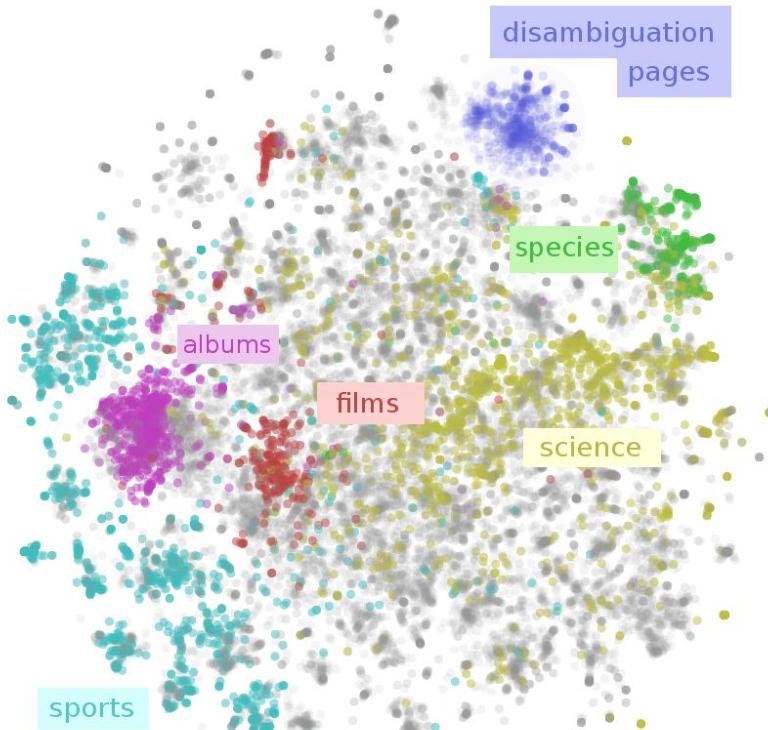


<https://studentwork.pratssi.org/infovis/visualization/weak-ties-network-analysis-of-twitch-data/>



# Modélisation : Zoom sur les vecteurs

- Les **vecteurs** ou **embeddings** encodent des données non structurées (mots d'un texte, documents, images, ...) au sein d'un corpus
  - > Exemple : [0.1, 2.32, -1.76, 0.8]
- Ils sont au cœur des modèles d'IA type ChatGPT
- Opérations intéressantes :
  - > Calcul de proximité via la distance entre 2 vecteurs
  - > Recherche des voisins proches
  - > Classification de documents ou concepts





# Modélisation : Notion de structuration

Au niveau de la modélisation, on distingue en général 3 niveaux de structuration :

## Non structuré

La donnée ne peut pas être interprétée par une machine sans traitement d'extraction préalable.

### Formats associés :

Documents (.txt, .docx...)  
Images, son, vidéo  
Flux d'octets bruts

## Semi-structuré

La donnée est un agrégat d'informations structurées et non structurées.

### Formats associés :

XML/JSON  
HTML  
Logs machines

## Structuré

La donnée possède une structure documentée (schéma) permettant d'interpréter ses constituants.

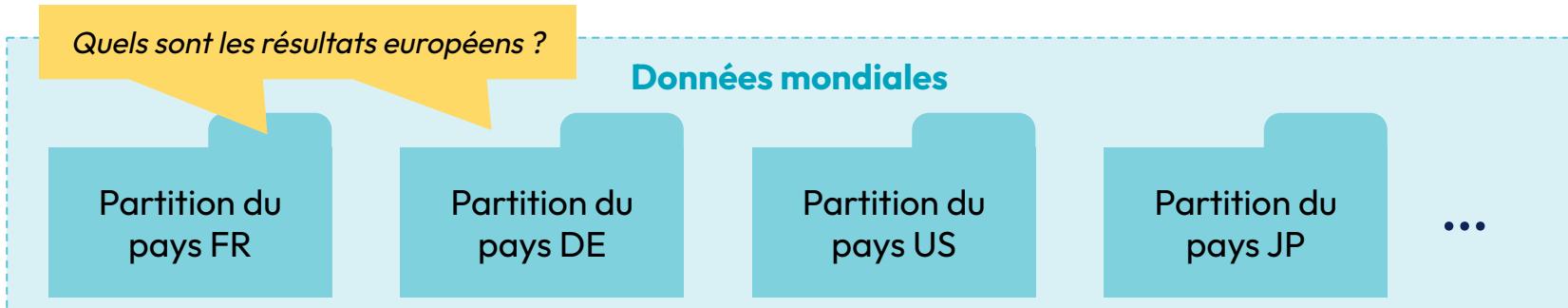
### Formats associés :

Bases de données  
CSV, Excel  
XML/JSON  
Parquet, Avro, HDF5



# Modélisation : Partitionnement

- ⦿ **Principe :** “découper” les données selon un ou plusieurs champs, la **clé de partitionnement**
  - > Exemples fréquents : partitionnement par année (temporel), par pays (géographique)
- ⦿ Cela permet aux traitements de cibler le sous-ensemble des données utiles en ignorant le reste : gain de performance
- ⦿ Les partitions peuvent être imbriquées à plusieurs niveaux (pays > année > mois > jour > ...)
- ⦿ La clé de partitionnement doit être adaptée aux traitements, et présenter un bon niveau de finesse (ni trop grossier, ni trop fin)





# Format et modélisation par l'usage

- L'efficacité d'un traitement dépend en grande partie de **l'adéquation** entre la logique du traitement, la modélisation des données et le format dans lequel elle est encodée

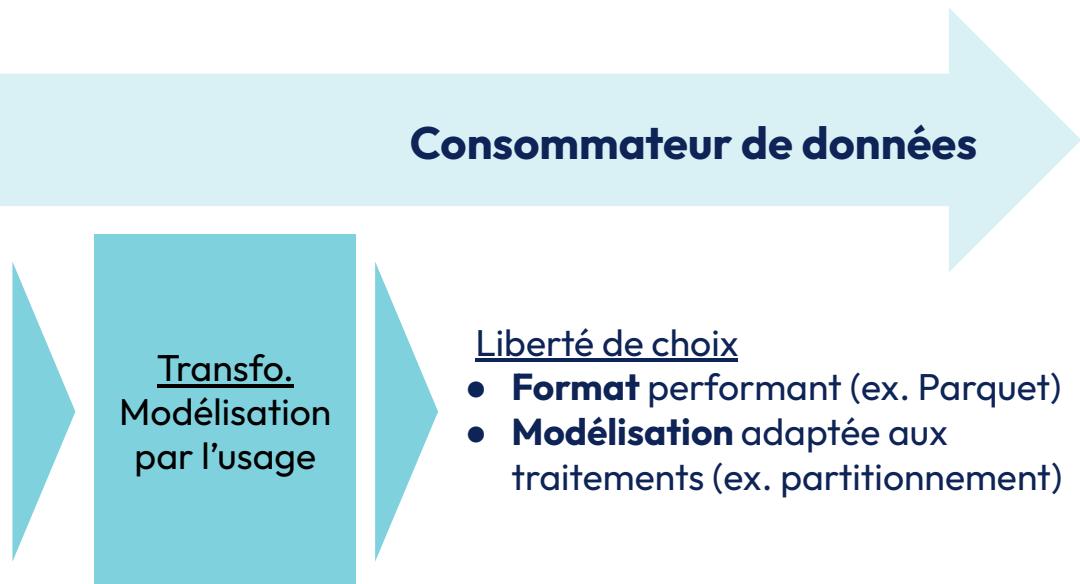
## Fournisseur de données

Modélisation et format imposés par la source

## Consommateur de données

### Liberté de choix

- **Format** performant (ex. Parquet)
- **Modélisation** adaptée aux traitements (ex. partitionnement)

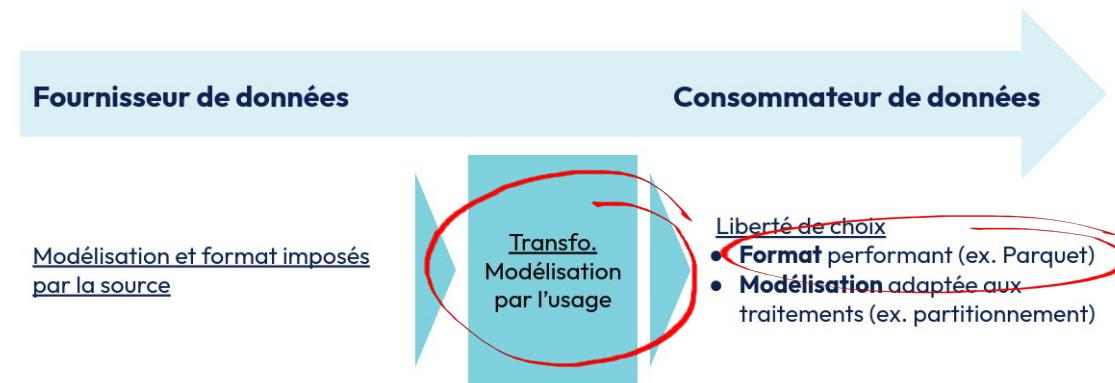




# Format : Zoom sur Parquet

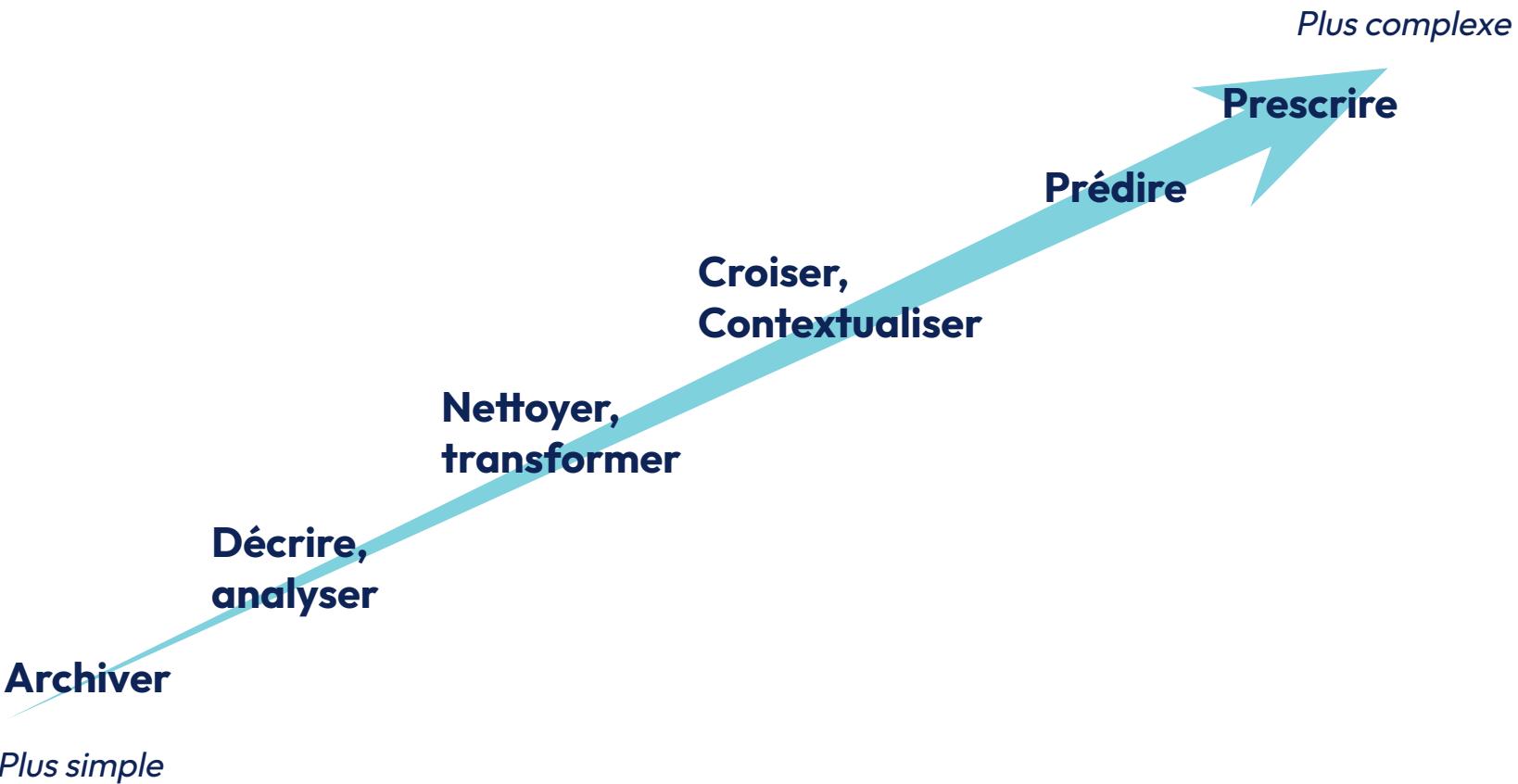


- ➊ Le format **Parquet** est optimisé pour les traitements de type “big data”, notamment :
  - > Souple pour la modélisation : structuré, semi-structuré, partitionnable
  - > Hautement compressé donc rapide à lire
  - > Optimisé pour les calculs analytiques (somme, min, max, ...)
  - > Compatible avec les algorithmes de traitement parallèle
- ➋ Tip : profiter de l’étape de transformation des données brutes pour sauvegarder une copie en Parquet



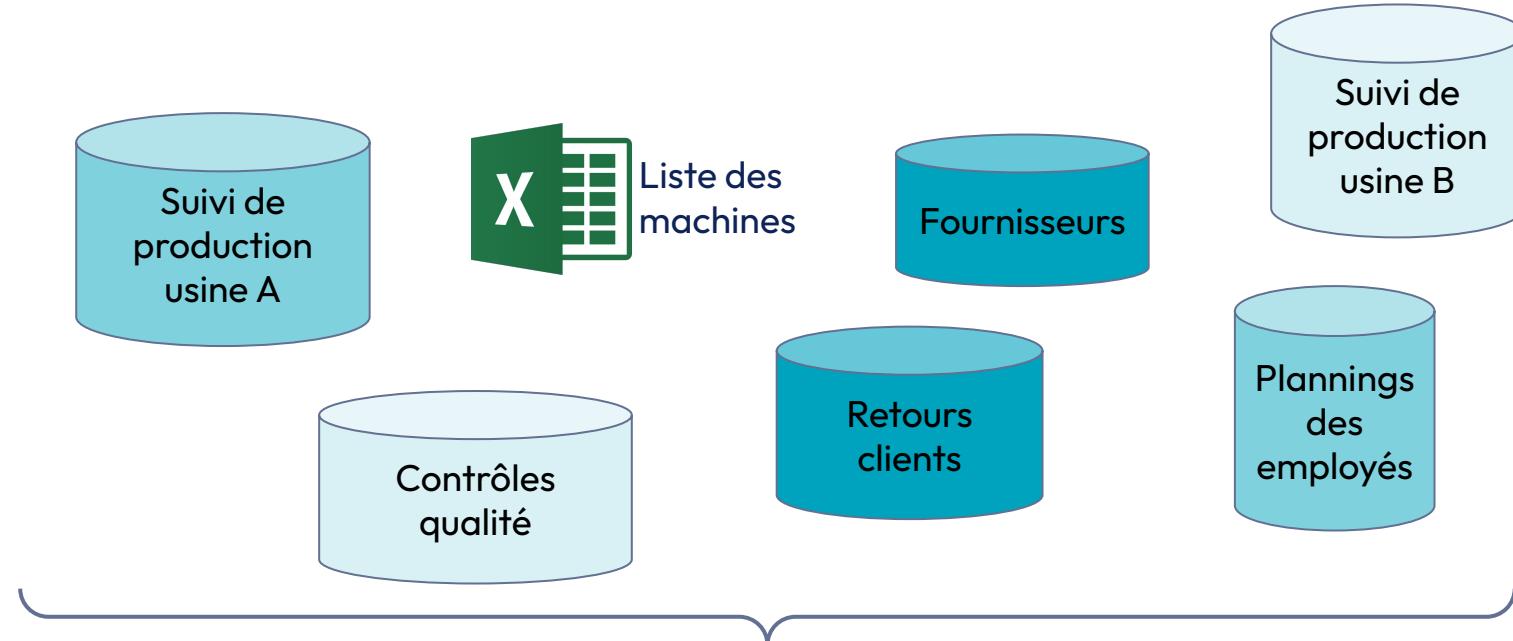


# Catégories de manipulation de données





# Centralisation des données

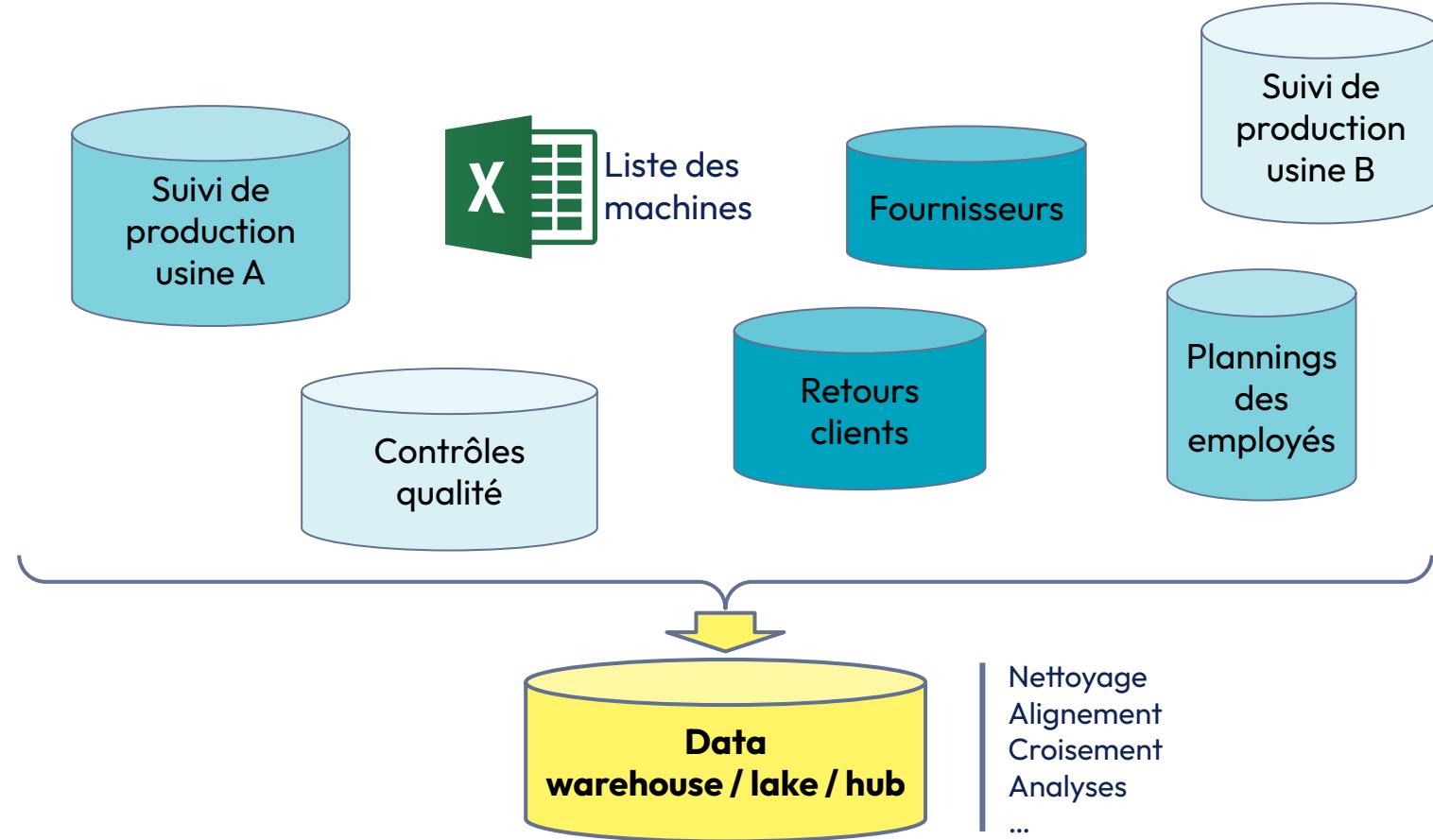


Comment répondre à la question :

***“Quelles sont les causes des insatisfactions clients ?”***

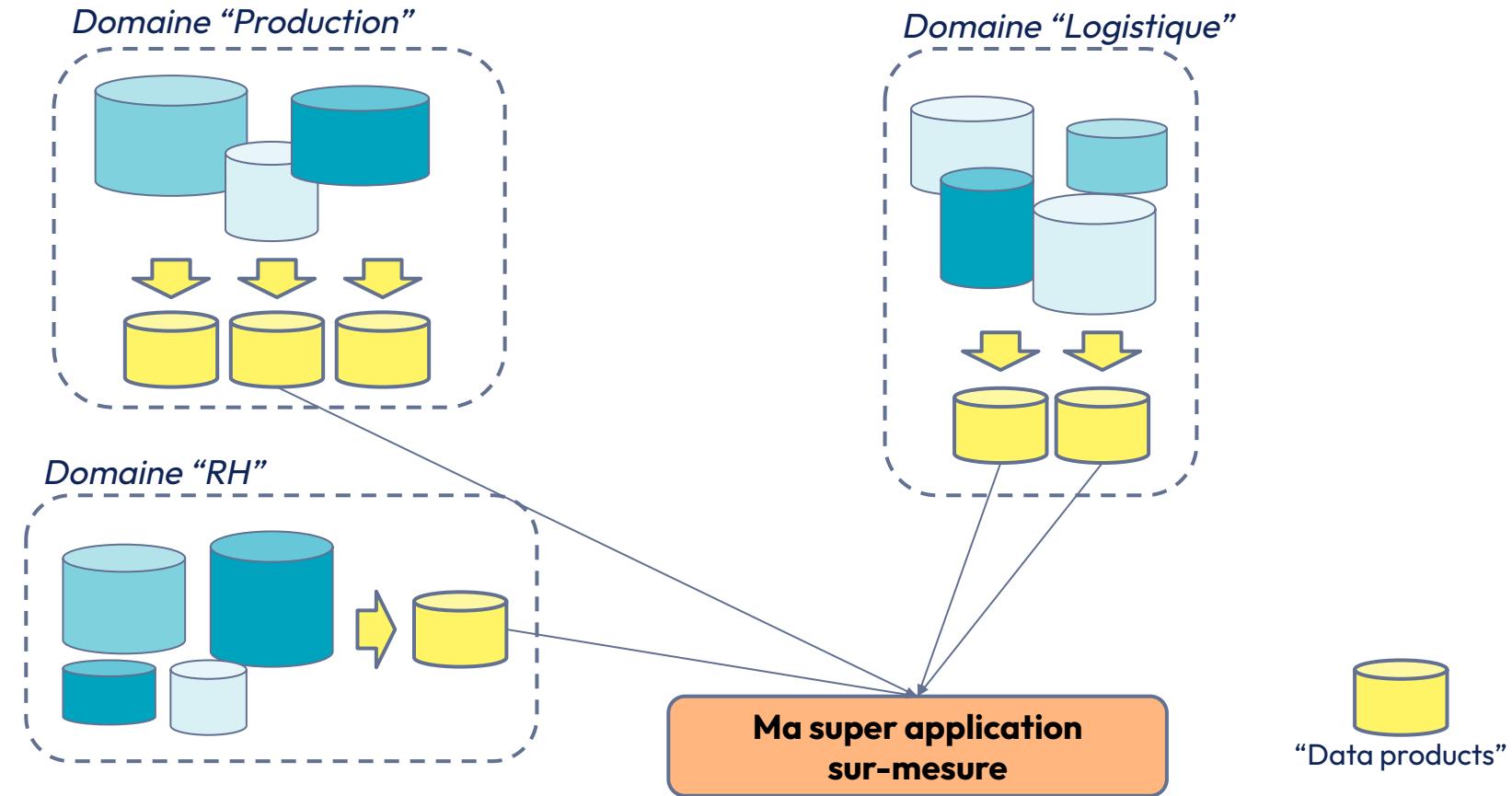


# Centralisation des données





# Equilibrer centralisation et autonomie : le modèle “data mesh”





# Qualité de données

Les problèmes de qualité sont inévitables et biaisen les analyses. Quand on ne peut corriger à la source, on compense :

Problème	Suggestions
Valeurs manquantes	Interpoler, remplir (“imputer”) avec des moyennes ou du machine learning
Bruit ou valeurs aberrantes	Lisser (moyenne ou médiane), éliminer les anomalies (outliers statistiques), utiliser des modèles robustes
Doublons	Stratégie à choisir (choisir au hasard, garder la version la plus récente, fusionner les doublons, choisir au hasard...)
Liens (clefs étrangères) manquants	Ajouter une source de données qui possède les liens, rapprocher selon des critères heuristiques (ex. date)
Problèmes de texte (abus d'abréviations, orthographe, mélange de langues, ...)	Pré-traiter, utiliser des modèles pré-entraînés sur ces corpus problématiques

Quand les données sont suffisamment nombreuses, et les exceptions rares, on peut aussi les écarter.



# Gouvernance des données

## Principe de la gouvernance des données :

*Mettre en place des règles et des processus garantissant un usage efficace des données, dans le respect des contraintes de qualité, de sécurité, de confidentialité, etc.*

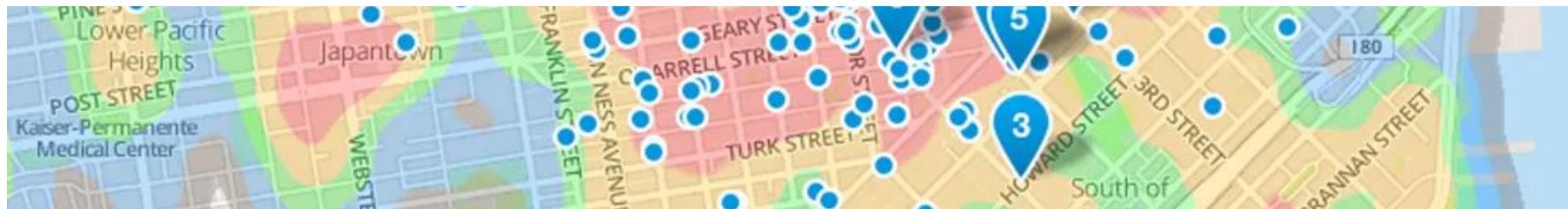


# Gouvernance des données

- Supposons qu'un opérateur téléphonique vende à des tiers les cartes de fréquentation des rues d'une ville, en s'appuyant sur le bornage GPS ou cellulaire de ses abonnés
  - Usage : optimiser la localisation des commerces dans les quartiers fréquentés



**Quelles questions de  
gouvernance cela soulève-t-il ?**

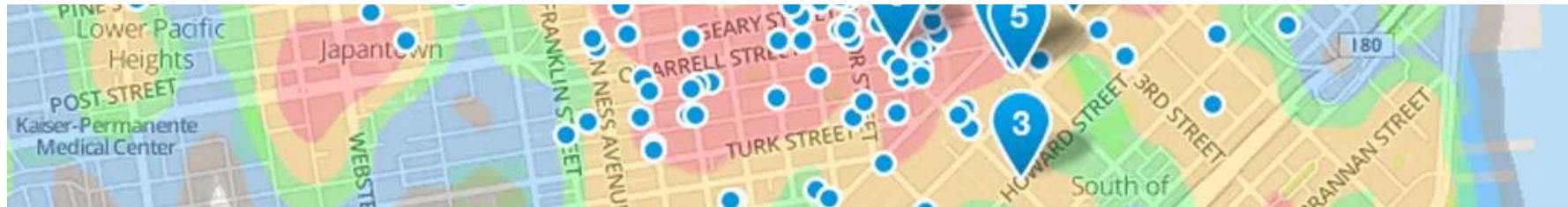




# Gouvernance des données

- Quelques idées :

- > Respect de la vie privée par anonymisation des terminaux et bornages
- > Obligation de restituer ou supprimer les données d'un abonné sur demande (RGPD)
- > Suivi et maintien d'une qualité de données en cohérence avec le prix du service
- > Procédures de correction de données aberrantes
- > Maîtrise des coûts en évitant l'accumulation ad vitam aeternam de données devenues caduques
- > Garantie que d'autres applications internes "sauvages" ne détournent pas le jeu de données (RGPD, encore)
- > Documentation à jour du jeu de données et des moyens techniques d'accès (API)





03

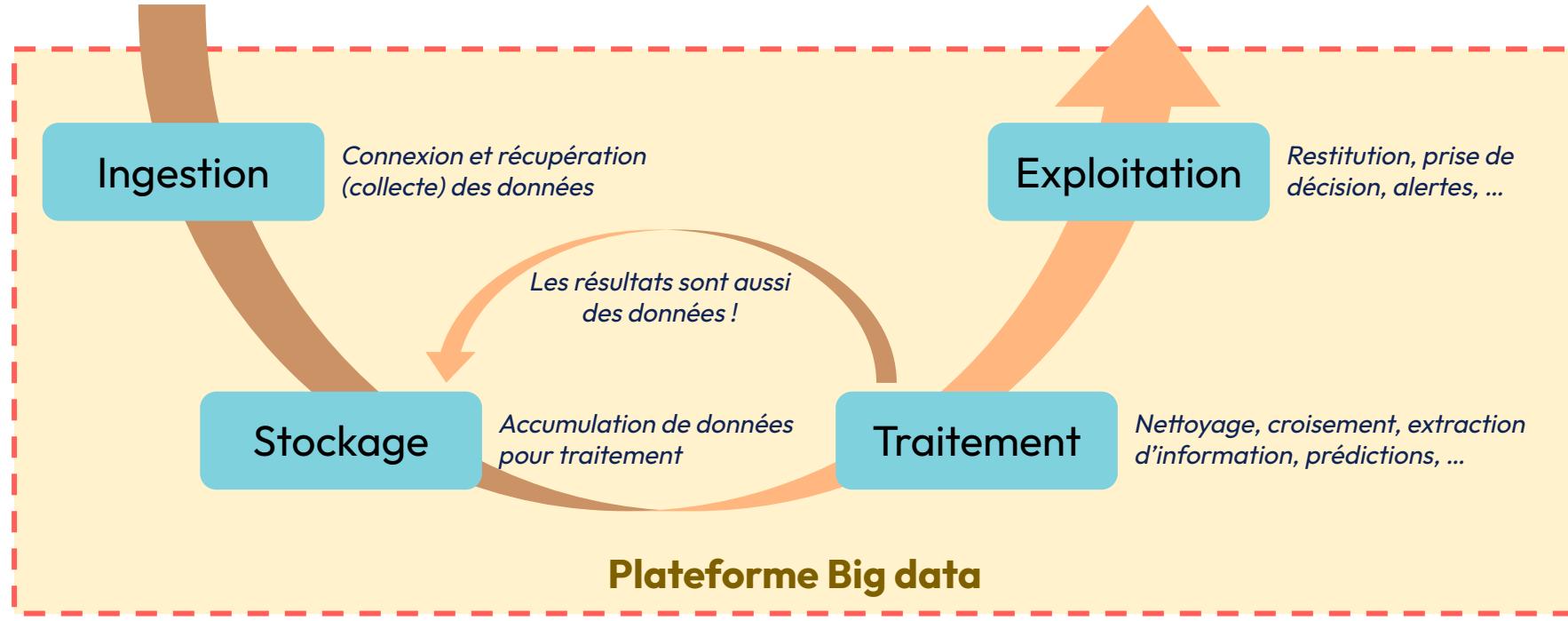
# Architecture d'une plateforme



# Plateforme Big data (rappel)

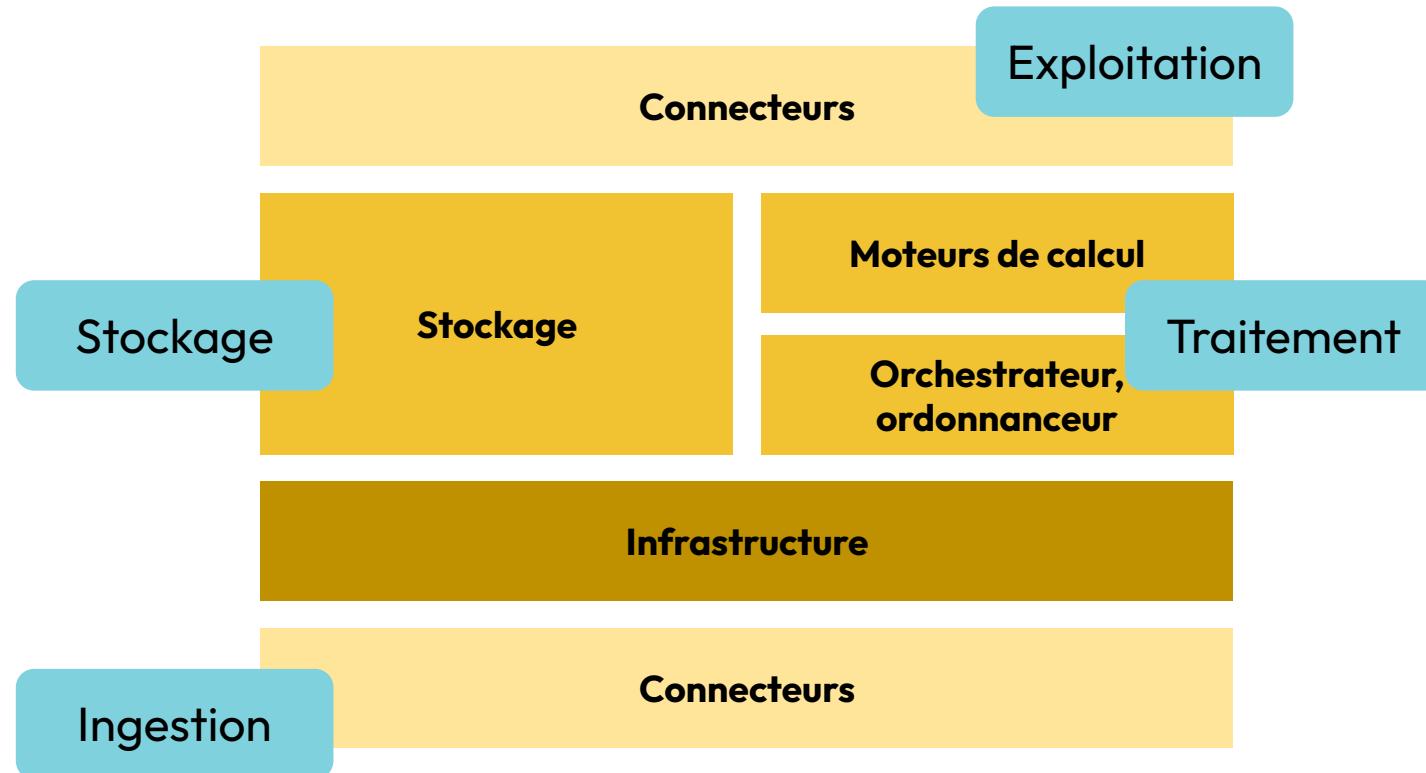


Environnement produisant et consommant des données  
(site web, application smartphone, capteurs, ...)





# Représentation en couches

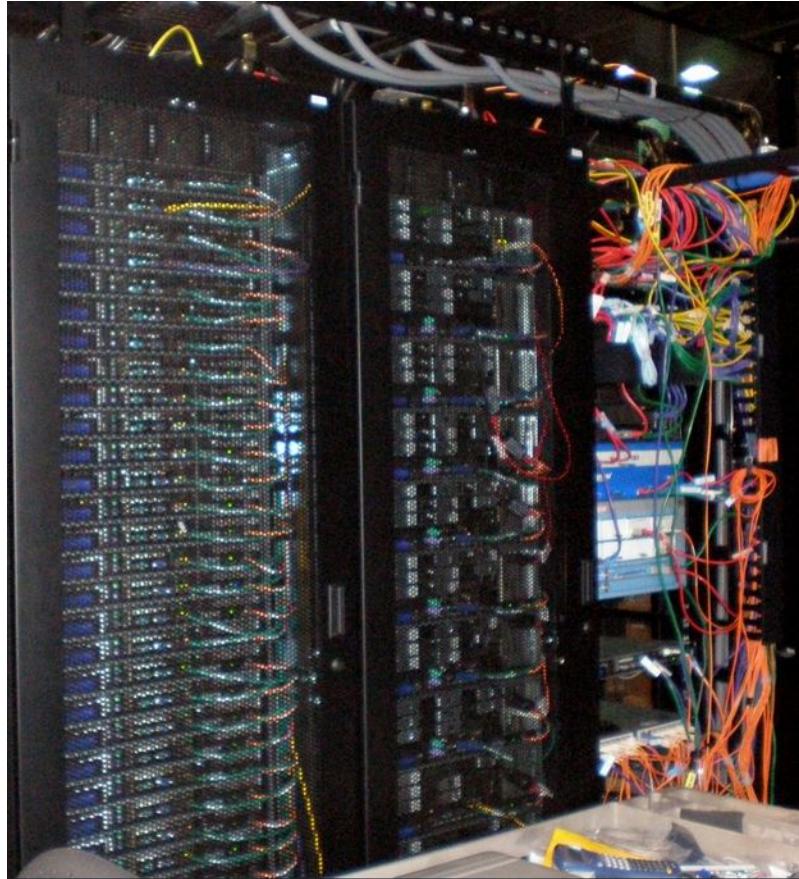




# Infrastructure

Les machines qui font tourner la bête !

- Des serveurs pour le calcul (CPU, GPU, RAM)
- Des baies de disques pour le stockage
- Du réseau rapide entre tout ce monde



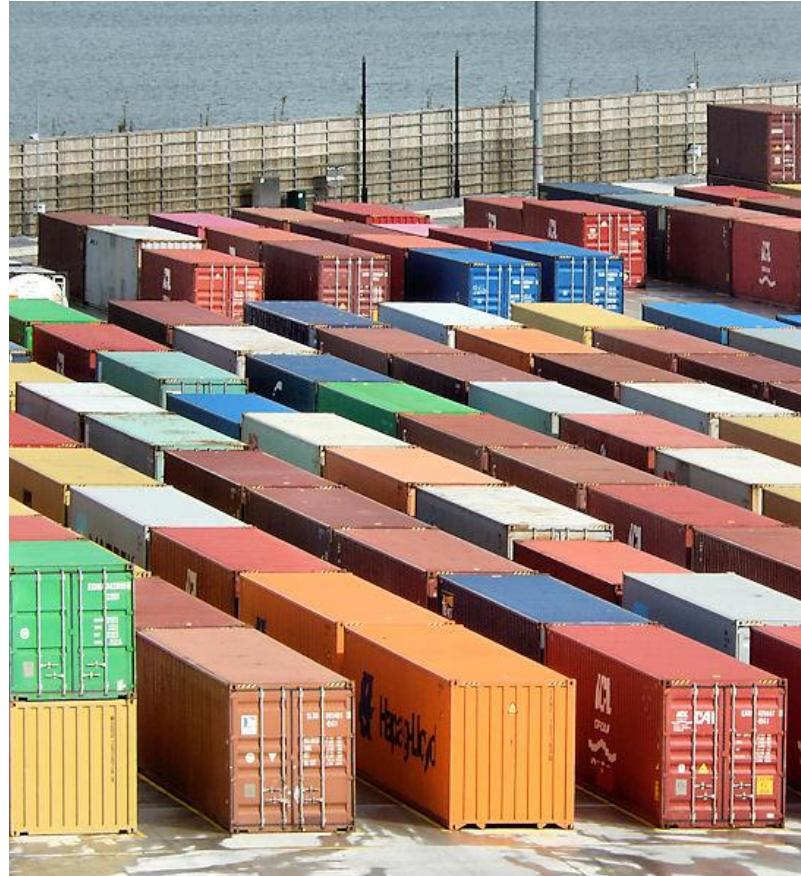
[https://www.flickr.com/photos/s\\_w\\_ellis/3877534599/in/photostream/](https://www.flickr.com/photos/s_w_ellis/3877534599/in/photostream/)



# Stockage

Exploiter l'infrastructure de stockage pour :

- Offrir des services de persistance fiables et performants des données ingérées ou produites (fichiers, bases de données)
- Gérer le cycle de vie des données, leurs modèles, le chiffrement, les droits d'accès



<https://www.geograph.ie/photo/7319143>



# Moteurs de calcul

Des logiciels généralistes, utilisés par les connecteurs pour :

- Exécuter du code dans le langage choisi par l'utilisateur (Python, SQL, ...)
- Utiliser les algorithmes les plus efficaces pour exploiter la puissance de la plateforme
- Permettre la mise en oeuvre de cas d'usage variés (data viz, IA, streaming, ...)



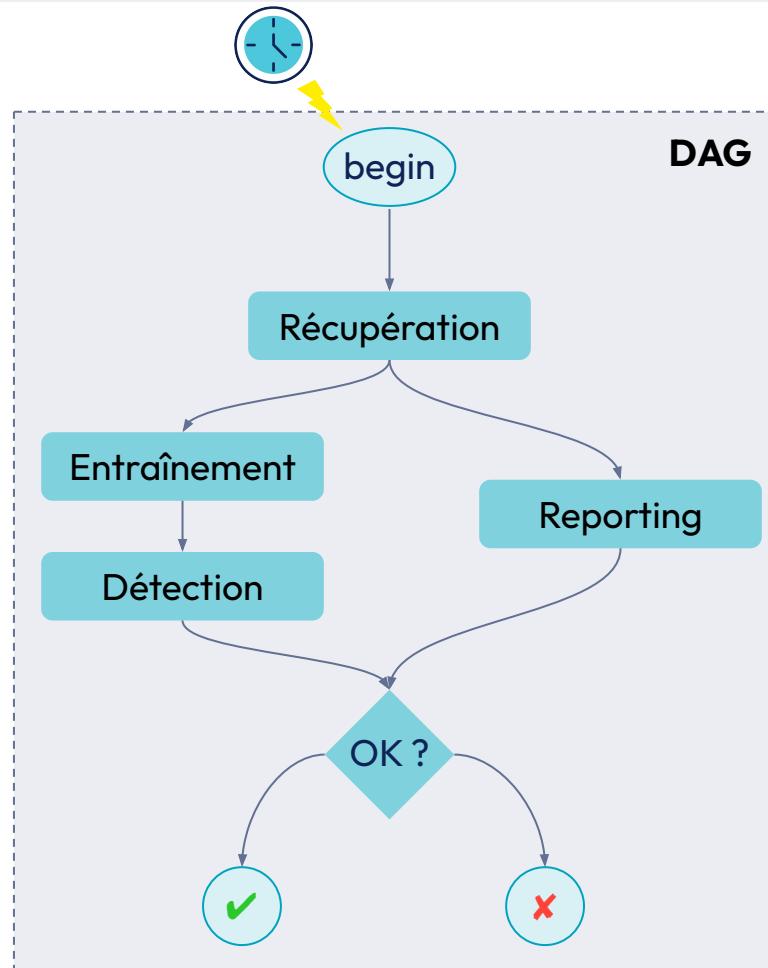
<https://negativespace.co/industrial-gears/>



# Orchestrator et ordonnanceur

Gestion des ressources de calcul de la plateforme :

- Distribuer les traitements sur les noeuds de calcul
- Suivre leur bonne exécution
- Enchaîner les étapes de traitement complexes (“pipelines”, “workflows”, etc.)





# Connecteurs

Logiciels qui permettent à la plateforme :

- D'interagir avec l'environnement producteur et consommateur de données
- D'interagir avec les utilisateurs (data scientists, ...)

Exemples :

- Dépôt de fichiers en entrée/sortie
- Import/export bases de données (ETL/ELT\*)
- Librairies Python, R, Java (voir plus loin)
- PowerBI, Tableau, Excel
- Jupyter Notebooks, Dataiku

(\*) ETL = Extract, Transform, Load, ELT = Extract, Load, Transform



<https://www.flickr.com/photos/shieldconnectors/4251069678>



# Les plateformes de données en entreprise

La plateforme est en général imposée par l'entreprise. Elle peut être sur le cloud, ou maintenue en interne.

Ce qui vient en général avec une telle plateforme :

- de la **documentation** sur son architecture
- des **règles de bonne utilisation** (gouvernance des données, sécurité, ...)
- des **connecteurs** prêts à l'emploi pour créer de nouvelles applications rapidement
- une **équipe** qui assure son bon fonctionnement, et vous accompagne à l'utilisation



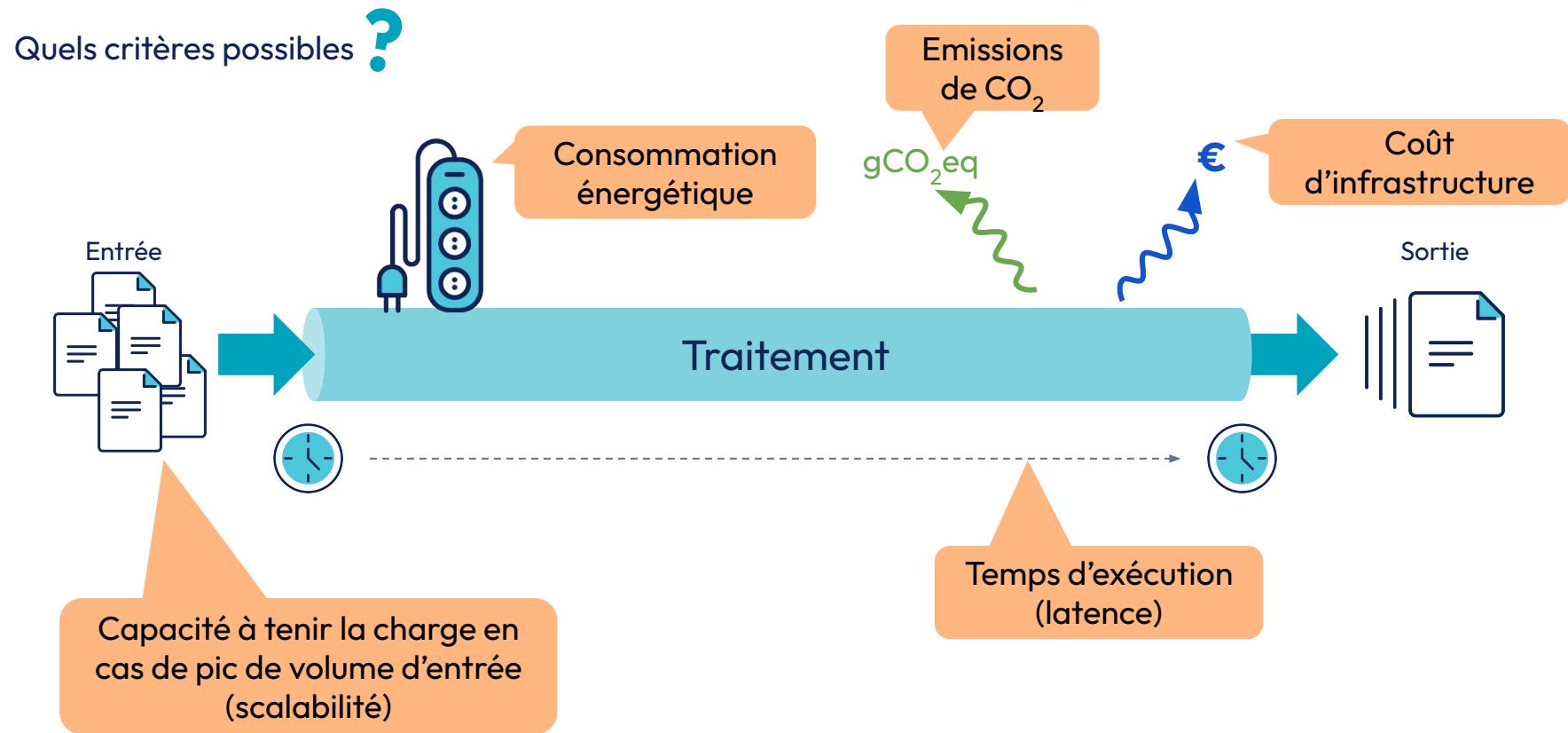
04

# Traitements de données



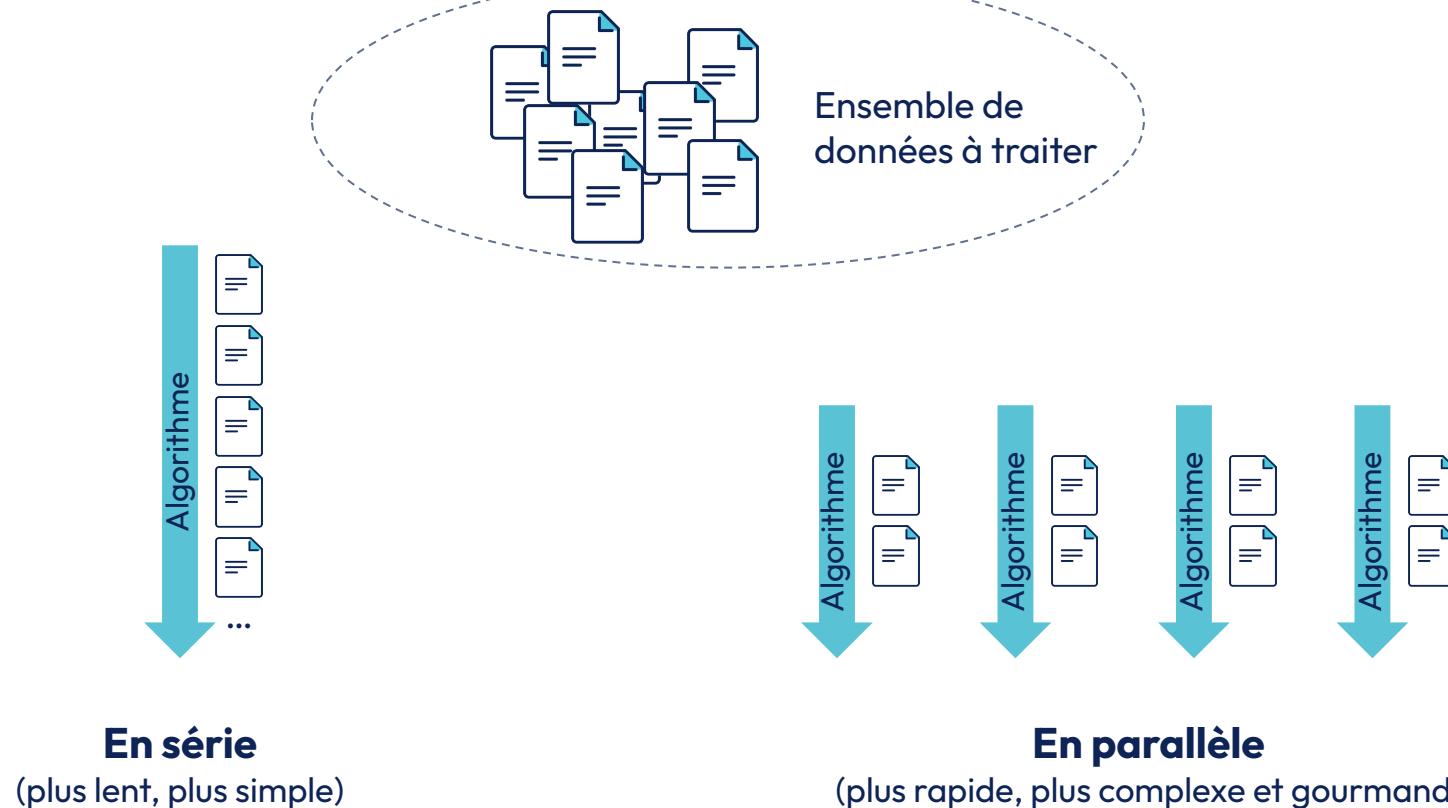
# Comment mesure-t-on la performance d'un traitement ?

Quels critères possibles ?





# Calcul en série ou en parallèle ?

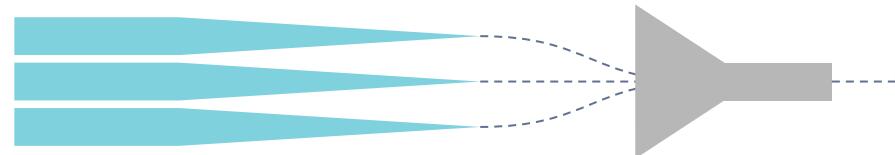




# Quand peut-on paralléliser ?

Pour paralléliser efficacement, les algorithmes utilisés doivent respecter certaines règles :

- La partie qui prend le plus de temps doit pouvoir être **découpée** en sous-problèmes plus petits, indépendants les uns des autres : “diviser pour mieux régner”
- Les parties non indépendantes sont des goulets d’étranglement et doivent travailler sur les plus petites données possibles, pré-mâchées par les parties parallélisables



## ✓ Les conditions les plus glop :

- Données complètement indépendantes de bout en bout (problème “embarrassingly parallel”)
- Traitements commutatifs et associatifs

## ✗ Les conditions les plus pas glop :

- Fortes interactions entre les données (ex. données très “connectées”)
- Algorithmes nécessitant plusieurs passes

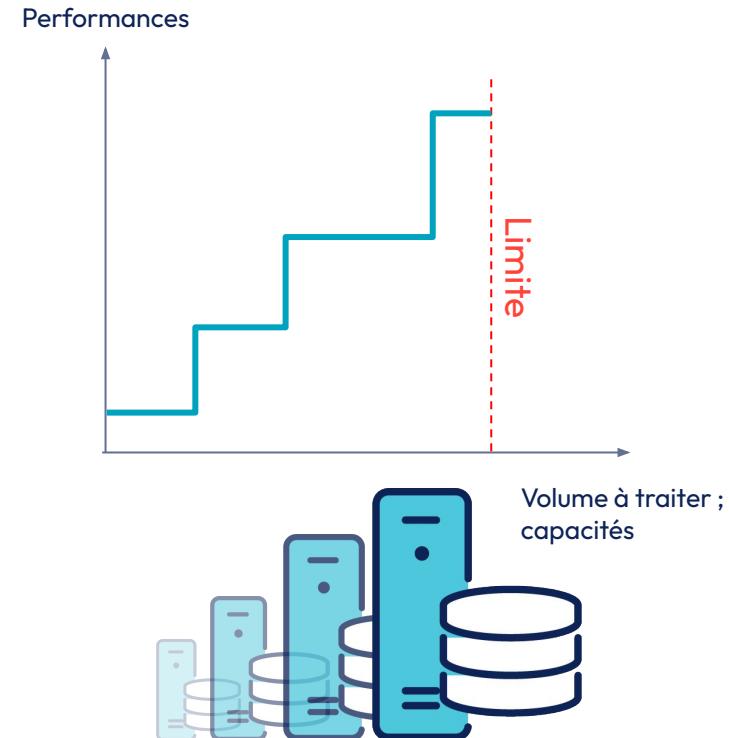


# Concilier volume et performance : le “scale-up”

- ➊ Augmenter la puissance des composants

- > Des disques plus gros
- > Un CPU plus rapide
- > Un CPU avec plus de coeurs pour paralléliser dessus
- > Un GPU pour paralléliser dessus
- > Plus de RAM
- > Réseau plus rapide

- ➋ ... mais on finit par atteindre une limite de capacité quand le volume dépasse un seuil maximal

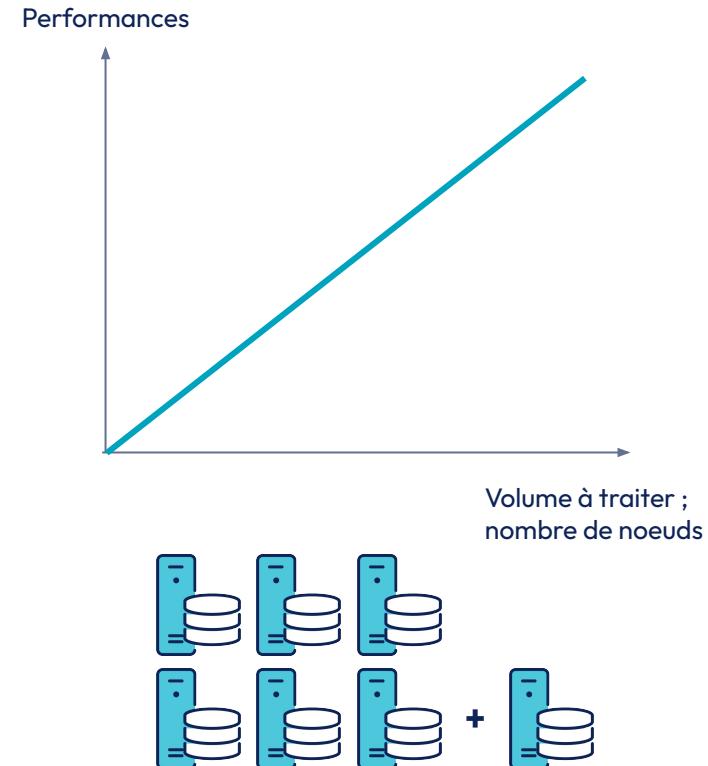




# Concilier volume et performance : le “scale-out”

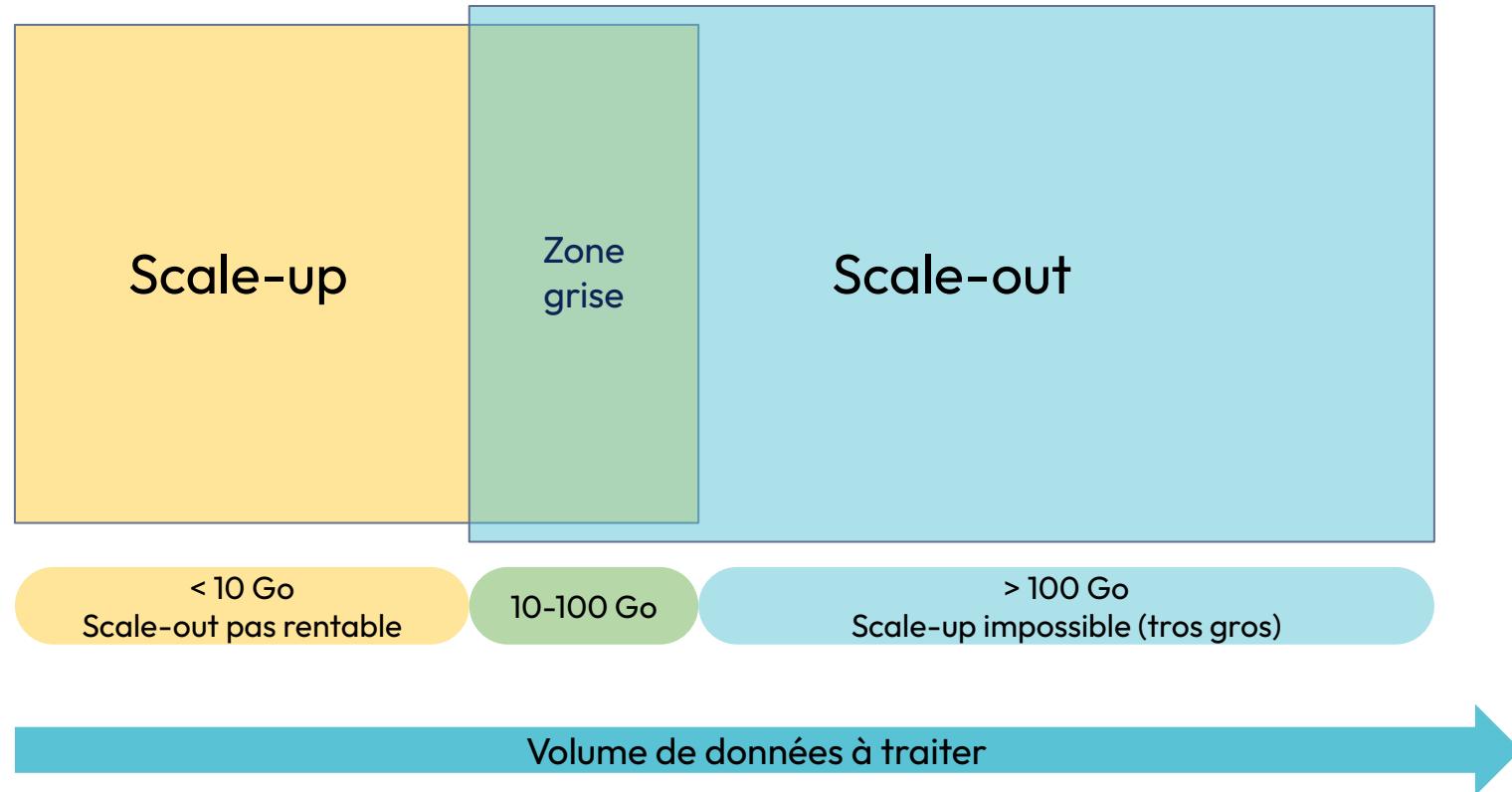
## ➊ Multiplier les composants

- > On parle alors de **cluster** ou **système distribué**
- > On parle de **scalabilité linéaire** quand la performance est proportionnelle (en théorie) au nombre de noeuds identiques
  - > Calcul parallélisé sur tous les noeuds
- > Toutes les plateformes big data sont distribuées, et sur le cloud, la taille d'un cluster est facilement réglable





# Mettre le curseur au bon endroit





# Estimer le volume pour faire le meilleur choix

Les traitements les plus rapides sont ceux qui manipulent des données en mémoire. Les accès fréquents aux disques, au réseau peuvent les ralentir drastiquement !

Fichier CSV :

Artiste	Album	Année	Pistes
Queen	Innuendo	1991	12
Queen	Hot Space	1982	11
R.E.M.	Out of Time	1991	11
Etienne Daho	Paris, Ailleurs	1991	11
...	...	...	...

Tableau NumPy

ID Artiste	ID Album	Année	Pistes
181	7776	1991	12
181	7189	1982	11
217	9561	1991	11
94	453	1991	11
...	...	...	...



# Volume

Comment estimeriez-vous l'occupation mémoire de ces données, selon le format ?

Artiste	Album	Année	Pistes
Queen	Innuendo	1991	12
Queen	Hot Space	1982	11
R.E.M.	Out of Time	1991	11
Etienne Daho	Paris, Ailleurs	1991	11
...	...	...	...

ID Artiste	ID Album	Année	Pistes
181	7776	1991	12
181	7189	1982	11
217	9561	1991	11
94	453	1991	11
...	...	...	...





# Version CSV

En CSV, tout est sous forme de chaînes de caractères, même les nombres.

Il faut estimer la taille de chaque champ, et le nombre d'enregistrements.

Artiste	Album	Année	Pistes
Queen	Innuendo	1991	12
Queen	Hot Space	1982	11
R.E.M.	Out of Time	1991	11
Etienne Daho	Paris, Ailleurs	1991	11
...	...	...	...

~ 10 000  
enregistrements

17 caractères en moyenne, maximum 34

24 caractères en moyenne, maximum 47

4 caractères

1 ou 2 caractères

**Moyenne** :  $10\ 000 \times (17 + 24 + 4 + 2) = 459\text{ Ko}$   
**Pessimiste** :  $10\ 000 \times (34 + 47 + 4 + 2) = 850\text{ Ko}$



Simplification : 1 caractère = 1 octet



# Version NumPy

En NumPy (tableau en mémoire), **les nombres sont stockés en représentation machine**. On choisit le type de représentation en fonction des contraintes connues.

Il faut estimer la taille de chaque champ, et le nombre d'enregistrements.

ID Artiste	ID Album	Année	Pistes
181	7776	1991	12
181	7189	1982	11
217	9561	1991	11
94	453	1991	11
...	...	...	...

~ 10 000  
enregistrements

Entier sur 2 octets  
( $< 2^{15}$ )

Entier sur 4 octets  
( $< 2^{31}$ )

Entier sur 2 octets

1 octet  
( $< 2^8$ )

**Estimation :**  $10\,000 \times (2 + 4 + 2 + 1) = 88\, \text{Ko}$



# Un exemple plus réaliste

Des séries temporelles :

- 67 équipements
- 14 capteurs par équipement
- Une mesure par seconde
- On souhaite traiter 5 ans d'historique
- Format des séries indiqué ci-contre

## Enregistrements :

67 équipements  
x 14 capteurs  
x 10 ans  
x 365 jours  
x 24 heures  
x 60 minutes  
x 60 secondes  
= 295 807 680 000 mesures

ID Equipment	ID Capteur	Date & Heure	Mesure
84	4091	2024-12-01 12:32:17	7.33991
84	4091	2024-12-01 12:32:18	7.42819
84	4091	2024-12-01 12:32:19	7.35710
84	4091	2024-12-01 12:32:20	7.32399
...	...	...	...

## Taille unitaire encodée (en octets) :

4 ID Equipment (<  $2^{31}$ )  
+ 4 ID Capteur (<  $2^{31}$ )  
+ 4 Date & Heure (entier “époque”)  
+ 4 Mesure (flottant simple précision)  
= 16 octets par mesure

**Total = 4 408 Go**



# Typologies de traitements

Classification liée à la **cinématique** de prise en compte des données :

## Par lots (batch)

Les données sont traitées **par paquets**, comme un tout. Les paquets sont de taille arbitraire.

### Exemples :

Fichier de synthèse quotidienne des transactions de la veille  
Clôture comptable

## Au fil de l'eau (streaming)

Les données sont traitées **individuellement** au gré de leur arrivée dans la plateforme. Elles doivent être indépendantes.

### Exemples :

Message avec une transaction pour détection de fraude  
Enchère publicitaire web sur un clic

NB : on peut collecter des données au fil de l'eau, puis les traiter par paquets de temps en temps



# Typologies de traitements

Classification liée à la **sémantique** des données, principalement pour le traitement par lots :

## “Annule et remplace”

On traite une “photo” d’un état cohérent.

### Exemples :

Liste des produits disponibles à la vente avec leurs prix  
Etat des stocks d'un entrepôt

## “Delta” (ou incrémental)

Les données contiennent des ordres de mises à jour à appliquer à un état précédent mémorisé.

### Exemples :

Liste des arrivées et départs d'employés dans une entreprise  
Capteur renvoyant la variation d'une grandeur et non sa mesure absolue

NB : on peut aussi comparer 2 photos successives pour calculer le delta entre elles



# Tableau des typologies de traitements et complexité

Une évaluation à la louche des compromis relatifs de mise oeuvre (quand on a le choix...) :

	Annule & remplace	Delta
Batch		
Streaming		

## Détails sur la complexité

- Le streaming nécessite une prise en compte rapide des données, et les algorithmes sont plus compliqués
- Le delta exige de la fiabilité dans la mise à jour de l'état (pas de seconde chance si on rate un fichier). De plus, en streaming, l'ordre d'arrivée des données n'est pas toujours garanti



# Tableau des typologies de traitements et complexité

Une évaluation à la louche des compromis relatifs de mise oeuvre (quand on a le choix...) :

	Annule & remplace	Delta
Batch	<ul style="list-style-type: none"><li>Simple à développer</li><li>Volume et latence élevés</li></ul>	<ul style="list-style-type: none"><li>Encore plus complexe</li><li>Volume faible</li></ul>
Streaming	<ul style="list-style-type: none"><li>Plus complexe</li><li>Latence faible</li></ul>	<ul style="list-style-type: none"><li>Très complexe</li><li>Volume et latence faibles</li></ul>

## Détails sur la complexité

- Le streaming nécessite une prise en compte rapide des données, et les algorithmes sont plus compliqués
- Le delta exige de la fiabilité dans la mise à jour de l'état (pas de seconde chance si on rate un fichier). De plus, en streaming, l'ordre d'arrivée des données n'est pas toujours garanti



# Ecrire un traitement : le monde R / Python



**En mémoire  
(local)**

Des dataframes pour  
Python



Opérateurs de  
manipulation de données  
composables



dplyr, tidyverse (“R packages  
for data science”)

Deep learning



**Distribué**  
(NB : qui peut le plus  
peut le moins)

Analogue à Pandas, pour  
du calcul distribué



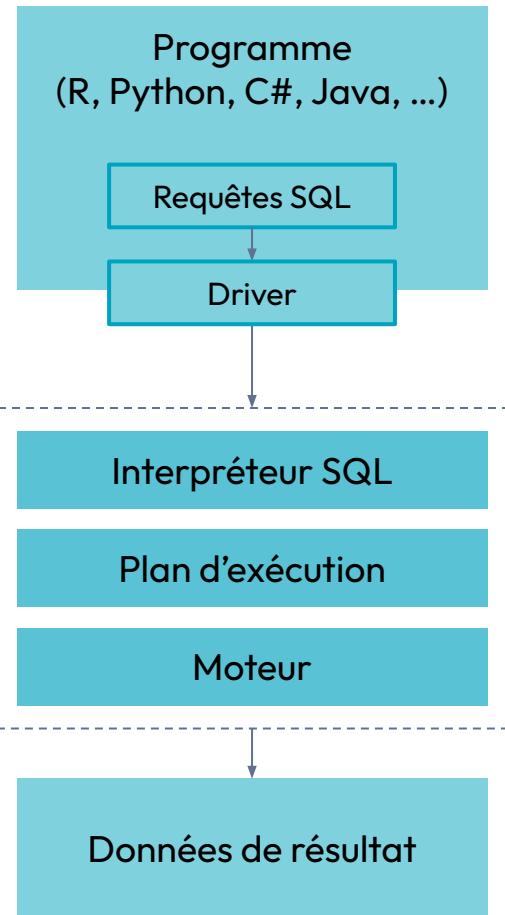
Traitement distribué  
basé sur la prog.  
fonctionnelle





# Ecrire un traitement : le monde SQL

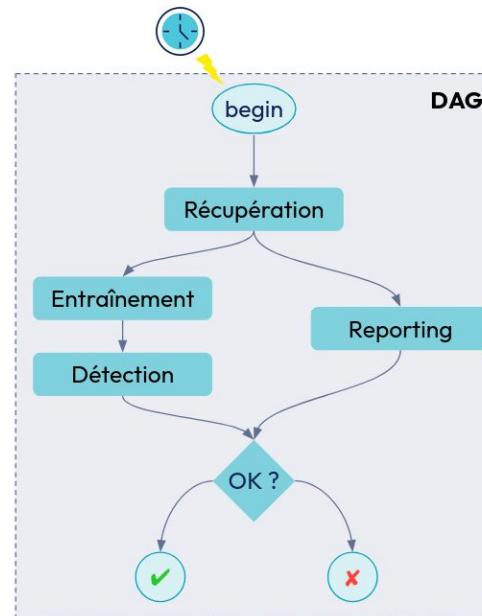
- Le langage SQL est aujourd’hui offert par de nombreuses plateformes big data. Les requêtes sont transformées en programmes qui s’exécutent en distribué
  - > Exemples : Spark SQL, BigQuery, Snowflake, ...
- SQL a été enrichi pour les données non structurées, le streaming, le machine learning, ...





# Ecrire un traitement : traitements composés

- Les traitements complexes sont découpés en étapes simples, regroupées en **DAGs / workflows / pipelines de données**
- Ils sont exécutés par la plateforme quand l'événement souhaité se produit : fréquence temporelle, réception d'un fichier, etc.



Conception graphique

```
@dag(  
    dag_id="process-employees",  
    schedule_interval="0 0 * * *",  
    start_date=datetime(2021, 1, 1, tz="UTC")  
)  
def ProcessEmployees():  
    create_table = PostgresOperator(...)  
  
    create_temp_table = ...  
  
    @task  
    def get_data():  
        ...  
  
    @task  
    def merge_data():  
        ...  
  
    [create_table, create_temp_table] >> \  
        get_data() >> \  
        merge_data()  
  
dag = ProcessEmployees()
```

Conception via code



# Robustesse et fiabilité

Soit une plateforme big data distribuée, sur laquelle on stocke des données et exécute des traitements.

**Que peut-il se passer d'imprévu ? Comment y remédier ?**





# Robustesse et fiabilité : problèmes possibles

Plus le système est gros, plus la probabilité est grande qu'un composant fasse défaut. Les problèmes peuvent survenir à tous les niveaux :

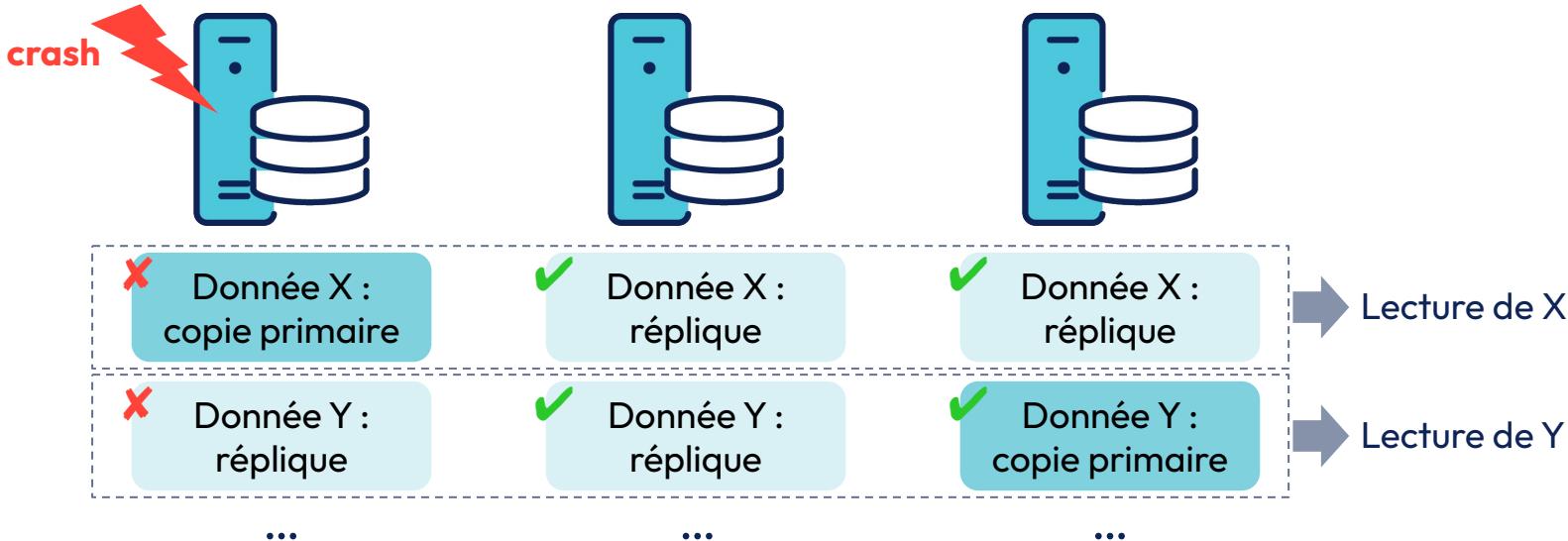
- **Stockage** : crash d'un disque dur, ou d'un serveur qui en contient un
- **Calcul** : défaillance d'un processeur, crash d'un serveur de calcul
- **Réseau** : perte de lien entre les noeuds du cluster sur lequel les traitements se répartissent
- **Code de traitement** : bug ou présence de données inattendues



# Robustesse et fiabilité : point de vue de la plateforme

## Pour le stockage :

- Le stockage utilise la technique de **réPLICATION** pour éviter la perte de données en cas de crash

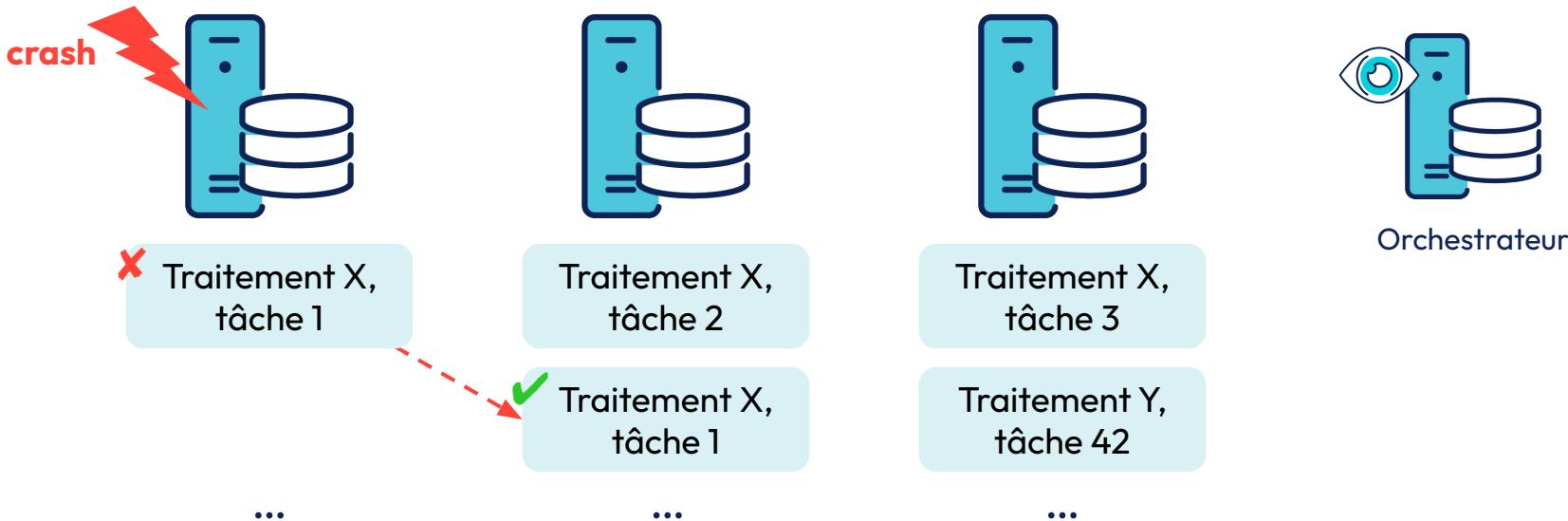




# Robustesse et fiabilité : point de vue de la plateforme

## Pour le calcul :

- Quand une partie d'un traitement est interrompue de manière abrupte, l'orchestrateur surveille et **relance la tâche en échec**, éventuellement sur un autre noeud si le premier est hors service





# Robustesse et fiabilité : point de vue de la plateforme

En résumé, on applique de la **redondance** sur les composants d'infrastructure.

Cela permet d'en faire un système à **haute disponibilité** : la plateforme continue de “tourner” le temps qu'un incident soit réparé.

Sur le cloud, la plupart des services sont à haute disponibilité.



# Robustesse et fiabilité : typologie d'incidents

On distingue 2 types d'erreurs pouvant survenir lors d'un traitement de données :

## Erreurs transitoires

Une erreur transitoire est causée par un problème passager dû à l'infrastructure, externe au traitement.

### Exemples :

Incidents des pages précédentes  
Traitement : manque de mémoire *pour cause de cohabitation entre traitements*

## Erreurs permanentes

Une erreur permanente surviendra à chaque fois que le même traitement sera relancé sur les mêmes données (déterminisme).

### Exemples :

Bug dans le code du traitement  
Données d'entrée incorrectes  
Manque de mémoire *pour cause de données trop volumineuses*



# Robustesse et fiabilité : notion de rejeu

- ⦿ On a vu que l'orchestrateur pouvait relancer une tâche (partie d'un traitement) en échec. Il peut aussi arriver qu'on relance complètement un traitement complexe. Ce sont des **rejeux**
- ⦿ Un rejeu après une erreur permanente est assuré d'échouer à nouveau : mêmes causes, mêmes effets
  - > Les logs sont indispensables pour l'analyse !
  - > Risque de boucle infinie si un humain n'est pas là pour décider du rejeu
  - > En général, les chaînes de traitement automatisées sont paramétrées pour lâcher l'affaire après  $N$  tentatives



# Robustesse et fiabilité : point de vue du traitement

- Quand on écrit le code d'un traitement, il faut prévoir le cas d'un rejet partiel ou total du programme qui l'exécute
- Quelques techniques aux noms barbares :
  - > **Points de reprise** (ou **checkpoints**) : le traitement enregistre régulièrement son avancement pour reprendre là où il s'était arrêté sans tout recommencer
  - > **Idempotence** : le traitement est conçu pour donner le même résultat, rejet ou pas. Exemple : traitement de type "annule & remplace"
- L'idempotence n'est pas facile à mettre en place. Attention en particulier :
  - > Au code non déterministe (qui dépend de l'heure courante, de nombres aléatoires...)
  - > Aux programmes qui mettent à jour des bases de données en cours de traitement (état de départ qui change) : stateful vs



# Résumé des bonnes pratiques

## Performances

Modélisation par l'usage

Traiter en parallèle ce qui est indépendant

Réduire le volume le plus tôt possible

Travailler en mémoire vive

Scale-out quand le volume n'est pas gérable sur un seul noeud

Optimiser le code

## Robustesse & fiabilité

Eviter de complexifier (batch, annule & remplace quand c'est possible)

Tracer les erreurs (logs) et les résultats intermédiaires

Attention aux rejeux automatiques

Mettre les bonnes conditions de rejet : checkpoints, idempotence

Mettre en place des tests automatisés



05

# Le cloud



# Principe

**Sous-traiter** la gestion de l'infrastructure à un fournisseur extérieur (**provider**), qui va fournir des services de stockage, calcul, traitement, sécurité, ... à composer pour faire des applications.

Vous souscrivez un abonnement et payez pour l'usage des ressources utilisées :

- Nombre et temps d'utilisation des machines
- Volume stocké et traité
- Nombre de requêtes
- + plein d'autres options selon les services





# Différentes “saveurs” de cloud

## IaaS

***Infrastructure as a Service***

Location de machines virtuelles (VM), de disque durs virtuels, de bande passante réseau, ...



## PaaS

***Platform as a Service***

Location de composants logiciels servant à construire des applications : bases de données, ingestion, moteurs, IA, ...



*Focus principal de ce cours*

## SaaS

***Software as a Service***

Location d'un logiciel “métier” sous licence : gestion client, commerce en ligne, simulation scientifique, graphisme, ...





# Les grands providers de cloud

IaaS



IaaS + PaaS



IaaS + PaaS + SaaS





# Les services essentiels

## IAM : Identity & Access Management

- Authentifier les utilisateurs (*qui est-tu ?*)
- Vérifier les droits d'accès (*as-tu le droit d'utiliser ce service ?*)

## Billing

- Etablir les factures d'utilisation du cloud
- Ventiler par service et par projet
- Alerter quand le budget dérape





# Les services essentiels

## Monitoring

- Donner de la visibilité sur ce qu'il se passe avec les services utilisés (traces, logs)
- Alerter quand une situation anormale se produit





# Les services essentiels

## Machines virtuelles (“VM”)

- ➊ Mettre à disposition des serveurs de tout type (CPU, GPU, RAM, disques, ...) avec lesquels on fait ce qu'on veut
- ➋ Les organiser en réseau
- ➌ Sécuriser le réseau pour éviter le piratage

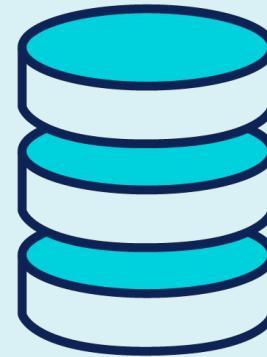




# Les services essentiels

## Stockage objet

- Permettre de stocker des fichiers de tout type (images, données, ...) facilement et à coût modéré
- Organiser ces fichiers en collections (“buckets”)





# Les services essentiels

## Bases de données

- Offrir des systèmes de gestion de bases de données modernes pour des applications plus complexes





# Les services essentiels

## Serveurs web

- Déployer des serveurs web dynamiques ou des API HTTP





# Les services essentiels

## Services de fonction (“lambda”, “serverless”)

- Déclencher des petits bouts de code sans se préoccuper des serveurs où il s'exécutera

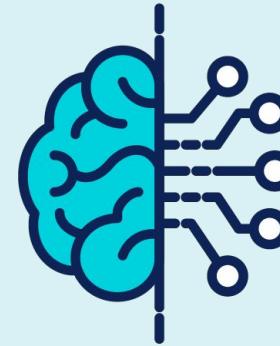




# Les services essentiels

## Machine learning et IA

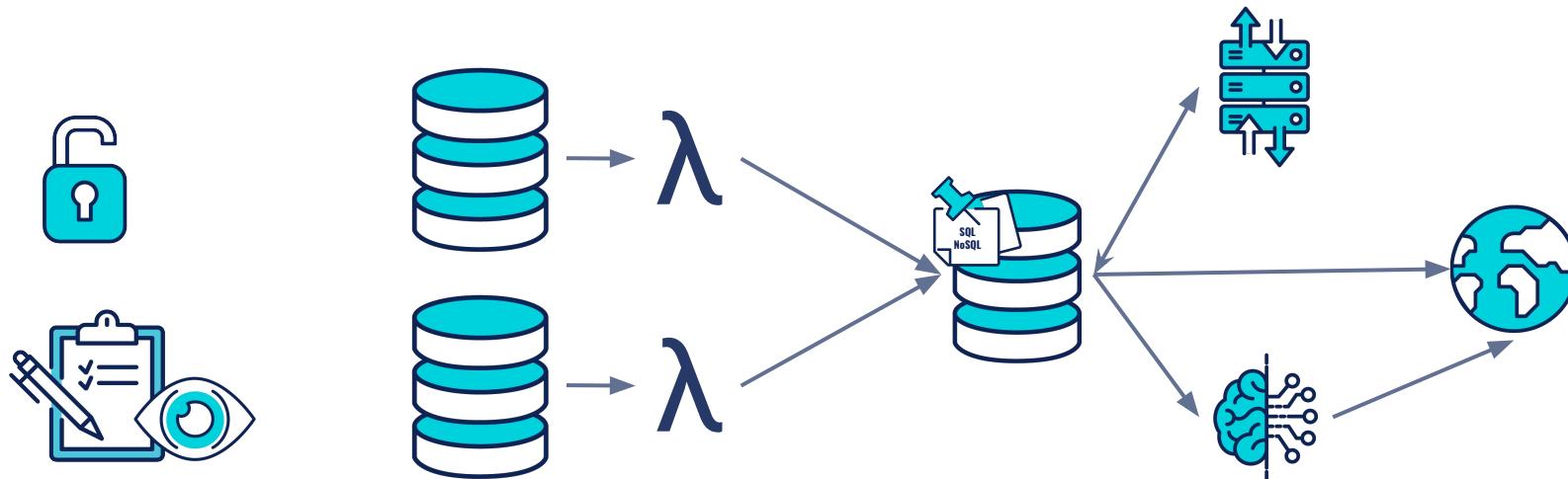
- Offrir aux data scientists les outils de leur quotidien, performants, avec un accès simple aux données stockées sur le cloud
- Gérer les modèles et leur cycle de réentraînement
- Faciliter la mise au point des traitements et leur déploiement final





# Une application ou un projet sur le cloud

... C'est le **choix et l'assemblage** de services configurés pour répondre à un besoin particulier, simple ou complexe.





# Quiz : les briques de cloud





# Quiz



? Apache Hive, Spark SQL

 Redshift, Athena

 Azure Synapse Analytics

 Big Query

# Big Query

Requêteage SQL de gros volumes de données



# Quiz



? N/A

 AWS Lambda

 Azure Functions

 Cloud Functions

# Lambda

Fonctions “serverless”



# Quiz



MySQL, PostgreSQL,  
MSSQL, ...



Azure SQL



Cloud SQL

## Relational Data Store

Base de données relationnelle (type MySQL ou PostgreSQL) managée sur le cloud



# Quiz



? cf. Hugging Face

 AWS Bedrock

 Cognitive Services

 Vertex AI Studio

# Cognitive Services

Modèles de machine learning variés, pré-entraînés prêts à l'emploi



# Quiz



? Jupyter + MLFlow + ...

SageMaker

Azure ML

Vertex AI

# Vertex AI

Boîte à outils de data science sur le cloud



# Quiz



? Elasticsearch, Algolia

 CloudSearch

 Azure AI Search

 Cloud Search

# Azure AI Search

Moteur de recherche



# Quiz



? Apache Spark

 Amazon EMR

 Synapse Analytics,  
Databricks

 Dataproc

# Dataproc

Moteur Spark géré par le provider



# Quiz



? Apache Ozone, Ceph

 Amazon S3

 Azure Blob Storage

 Google Cloud Storage

S3

Stockage objet



# Quiz



# Diplodocus

Rien du tout



# Quiz



? VMWare, Docker

 Amazon EC2

 Azure VM

 Compute Engine

# EC2

Machines virtuelles



# Quiz



? ffmpeg, Handbrake

 Elastic Transcoder

 Media Services

 Transcoder API

## Transcoder API

Transcodification audio & vidéo



# Quiz



? Data Galaxy

 Macie (?)

 Azure Purview

 Sensitive Data Protection

## Purview

Gouvernance de la donnée : catalogue, classification, généalogie, ...



# Quiz



? Apache Airflow

 SageMaker Pipelines

 Azure ML

 Cloud Composer

# SageMaker Pipelines

Ordonnanceur de workflows de data science



# Quiz



? Kafka

 Kinesis

 Event Hub

 PubSub

## Event Hub

Distribution de messages au fil de l'eau (pour l'ingestion en streaming)



# Quiz



# Cosmos DB

Base NoSQL

? MongoDB, Couchbase, ...

DynamoDB

**Cosmos DB**

Firestore, Datastore



06

# Pour un usage responsable des données



# De nombreux risques avec la numérisation croissante

Confiance envers les providers

Usurpation d'identité

Publicité abusive

Empreinte environnementale du numérique

Spam, phishing

Vol de données bancaires ou personnelles

Désinformation et propagande

Espionnage industriel

Appareils mal sécurisés

Deep fakes

Ransomwares



6.1

# Les règlementations



# Le RGPD

- Le RGPD est le **Règlement Général sur la Protection des Données** (GDPR en anglais)
- Il encadre l'usage des données personnelles et sensibles **en Europe**, et hors d'Europe **pour les ressortissants européens**
- Il existe un analogue américain : le Cloud Act





# RGPD : classification des données

Rappeler la différence entre **données personnelles** et **données sensibles**





# RGPD : Données personnelles, données sensibles

- Les **données personnelles** ou à caractère personnel identifient une personne
  - > Directement (nom, date de naissance, email, adresse IP, photo, dessin, ...)
  - > ... ou en recoupant des données diverses via des ID
  - > ... ou même par appréciation subjective sur la personne, permettant de la reconnaître
  - > Elles doivent être hébergées en Union Européenne, sauf clause contractuelle particulière
- Les **données sensibles** sont les données personnelles susceptibles d'entraîner une discrimination
  - > Origine ethnique, opinions politiques, affiliation syndicale, orientation sexuelle, religion, santé, biométrie, sanctions administratives
  - > Elles ne peuvent **pas** être hébergées en dehors de l'Union Européenne
  - > Toute prise en compte de ces données doit donner lieu à une analyse d'impact

**L'usage de ces données est encadré, qu'elles soient informatisées ou non, traitées de manière automatique ou non, ou même si elles sont collectées et non exploitées**



# RGPD : la CNIL

- La CNIL (**Commission Nationale de l'Informatique et des Libertés**) veille entre autres au respect du RGPD en France.
- Elle peut être saisie par les individus qui souhaitent faire valoir leur **droit de consultation et de rectification** de leurs données, et énonce les sanctions en cas de manquement
- Une entreprise ou un organisme traitant des données personnelles doit pouvoir prouver à la CNIL qu'elle respecte la réglementation. Elle doit pour cela tenir un **registre des traitements**

## Registre des traitements

### Traitements X

- ✓ Désignation du responsable
- ✓ Activités et finalité
- ✓ Catégories de données traitées
- ✓ Durée de conservation prévue
- ✓ Destinataires internes, externes
- ✓ Transferts hors UE
- ✓ Mesures de protection des données



# RGPD : Protection des identités

Anonymisation, pseudonymisation ?





# RGPD : pseudonymisation, anonymisation

**Principe** : empêcher l'identification des personnes à partir des données récoltées. Ces pratiques peuvent être exigées par la loi

- ⦿ **Pseudonymisation** : empêcher l'identification en l'absence de données complémentaires
  - > Exemple : remplacer les noms & prénoms par un ID ou des valeurs aléatoires (data masking)
  - > Exemple : brouter des coordonnées GPS
- ⦿ **Anonymisation** : détruire complètement l'information permettant de cibler les personnes
  - > Exemple : agrégation statistique
  - > Attention aux recouplements, aux corrélations, aux données non personnelles mais avec des valeurs rares

Identifier une personne ne signifie pas nécessairement retrouver son nom, mais peut consister à la “cibler” anonymement pour des usages non prévus par le traitement initial.



# L'AI Act

- ⦿ L'**AI Act** est un règlement européen encadrant la mise sur le marché de solutions d'intelligence artificielle
- ⦿ Il impose des exigences à de tels produits, en fonction de leur niveau de risque
- ⦿ Certains usages sont exclus : militaire, recherche scientifique, usages non professionnels





# AI Act : niveaux de risque

Niveau	Exemples	Exigences



# AI Act : niveaux de risque

Niveau	Exemples	Exigences
Inacceptable	Reconnaissance faciale Manipulation	Interdiction pure et simple
Haut	Santé, éducation Infrastructures vitales	Transparence, qualité, sécurité, supervision humaine (évaluation continue)
Généraliste	ChatGPT et consorts	Selon capacités et risques estimés, et type de licence
Limité	Manipulation d'images, sons ou vidéos	Transparence
Minimal	Jeux vidéos, filtres de spam	Aucune



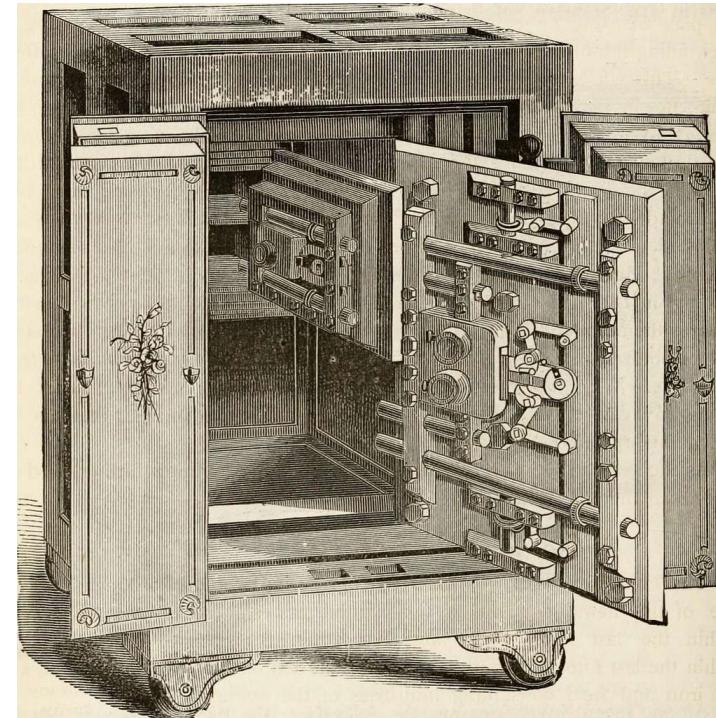
6.2

# Sécurisation des données



# Sécurisation

- Ce sont les mesures prises pour protéger les données des usages indus : fuite, identification de personnes, détournement de finalité, etc.
- Exemples de données **non personnelles** à protéger :
  - > Secrets des affaires (législation, délit d'initié)
  - > Secret industriel (concurrence, souveraineté)
- Les entreprises ont un barème de classification des données
  - > Ex. "C1" (public) à "C4" (top secret)
  - > Ex. de règle : "Pas de donnée C4 sur le cloud"





# Quelques pratiques indispensables

(en plus de la protection physique des infrastructures)

- **Chiffrer les données de bout en bout :**
  - > Stockage (au repos ou “at rest”) ou flux (“in motion”)
  - > Le chiffrement est systématique sur le cloud
- Mettre en place un **contrôle d'accès** :
  - > Authentifier les personnes et machines (comptes de service)
  - > Moindre privilège : attribuer à chaque compte les seuls droits nécessaires
- **Tracer** les opérations sensibles, et s'outiller pour des **audits** réguliers
  - > Etre capable de détecter les “trous dans la raquette”





6.3

# Eco-conception pour la donnée



# L'éco-conception en général

L'éco-conception s'inscrit dans la démarche générale de **réduction de notre empreinte environnementale**

Elle s'attaque en particulier à **l'empreinte du numérique**, qui croît à un rythme soutenu.

Nos “besoins” numériques imposent une pression environnementale à plusieurs niveaux :

- **Matières premières** (fabrication des équipements) : pollution (terre, eau), exploitation humaine, perte de souveraineté
- **Emissions de gaz à effet de serre (GES)** : utilisation de sources d'électricité carbonées pour la fabrication et l'usage

Contribution du numérique à l'empreinte carbone mondiale :

**3,8% en 2020**  
  
**8% en 2025**

Les émissions de GES du numérique dépasseront bientôt celles du transport.



# Quels sont les grands postes d'émissions de GES ?

La **fabrication des équipements** est loin devant l'usage : multiplication des serveurs, laptops, smartphones, objets connectés divers.

Dit autrement :

**Exécuter un traitement sur un serveur**

<<

**Fabriquer le serveur**

<<

**Fabriquer tous les terminaux avec lequel les utilisateurs vont consulter les résultats**

Casser le cercle vicieux de l'infrastructure qui suscite de nouveaux usages permet de gagner sur tous les tableaux. Le cloud est un bon moyen pour “louer” des infrastructures déjà fabriquées



# La data en particulier

La majorité des émissions vient donc du **renouvellement des terminaux**, “la data” n'est pas le principal contributeur direct.

## CEPENDANT

Un questionnement en profondeur du service numérique aura une influence bénéfique sur sa consommation, y compris sur la partie liée aux données.

Parmi les technologies qui participent à la construction d'une application, le machine learning n'est en général pas la moins émettrice !



# Ordres de grandeur

Classer ces différents usages par puissance consommée, du plus faible au plus gourmand :

- Une machine virtuelle sur le cloud
- Stocker 1 To de données sur le cloud
- Servir les utilisateurs de ChatGPT
- Utiliser son cerveau :-)





# Ordres de grandeur

Usage	Puissance consommée (estimation)
Une machine virtuelle sur le cloud	
Stocker 1 To de données sur le cloud	
Servir les utilisateurs de ChatGPT	
Utiliser son cerveau :-)	



# Ordres de grandeur

<b>Usage</b>	<b>Puissance consommée (estimation)</b>
Une machine virtuelle sur le cloud	420 W
Stocker 1 To de données sur le cloud	3,6 W
Servir les utilisateurs de ChatGPT	23 500 000 W (23,5 MW)
Utiliser son cerveau :-)	20-40 W



# Les bonnes questions à se poser pour l'usage

- (question 0 : est-ce que j'estime mes émissions de GES ?)
  
- Moi ou mes clients / utilisateurs ont-ils vraiment besoin de ce service ? De toutes ces fonctionnalités ?
- Du point de vue data, le service peut-il être rendu sans machine learning ? Avec moins de données et sans calcul distribué ?
- Si j'ai besoin de machine learning, faut-il entraîner un modèle ou puis-je en utiliser un déjà fait ?
- Mes infrastructures sont-elles bien utilisées ? (*merci d'éteindre la lumière en partant*)
- Est-ce que mon traitement est efficace, puis-je l'optimiser ?



Plus d'impact  
“Facteur 4”

Moins d'impact  
“Facteur 1”



# Estimer ses émissions de GES

Ce n'est pas facile mais on peut s'outiller un minimum.

Les estimations se calculent en **gCO<sub>2</sub>eq** (“grammes équivalents-carbone”)

- Sur des machines (virtuelles ou non), on peut **instrumenter le code** pour mesurer la consommation CPU / GPU, en déduire l'énergie consommée, et les émissions de CO<sub>2</sub>
  - > Emissions = Energie x Intensité carbone de l'approvisionnement électrique (mix énergétique)
  - > Attention l'intensité varie dans l'espace et dans le temps
- Sur le PaaS, c'est plus compliqué
  - > Certains providers fournissent un tableau de bord adapté
  - > Sinon, il faut faire des hypothèses simplificatrices
  - > La région d'hébergement joue un rôle, par le biais du mix énergétique
  - > Optimiser pour réduire la facture (€) est toujours bénéfique sur les émissions (gCO<sub>2</sub>eq)



<http://codecarbon.io/>



# Projeter les émissions liées à l'usage du machine learning

- Si on sait estimer l'émission d'un traitement, on sait en particulier estimer **celle d'une inférence unique d'un modèle de machine learning**
- Multipliée par le nombre d'inférences prévues (ou mesurées), on peut avoir une idée des émissions de ce poste consommateur qu'est le machine learning, à l'usage
- Attention aux modèles type ChatGPT & co, dont les émissions sont opaques et se cachent derrière des fonctionnalités anodines (chatbot sur un site web)
  - > Pour les modèles de LLM, la consommation dépend de la langue d'utilisation



# MERCI !



*There  
is  
a Better  
Way*