Bases de données NoSQL: une introduction

Module BDA et GL

Dr I. Gueye

Nota

- Ce cours fera appelle souvent à des notions de BD avancées à très avancées (Transactions ACID, CAP, distributed computing, traitements distribués, MapReduce, etc.). Ces notions sont développées en DIC2 et DIC3.

- Nous prendrons donc ce qui nous y intéresse seulement.

Les SGBD sont Universelles

- Systèmes « SQL »
  - Facilité d'utilisation
  - Cohérence des données
  - Persistance des données
  - Fiabilité (Pannes)
  - Efficacité
  - **Universalité**

- Vues SQL
- Données Structurées
- Transactions
- Optimisation des requêtes
- Indexation des données

Peut-on répondre à tous les besoins avec un système?

« One size fits all »

Caractéristiques du Big Data

Variety: Manage and benefit from diverse data types and data structures

Velocity: Analyze streaming data and large volumes of persistent data

Volume: Scale from terabytes to zettabytes

Tout à changé… évolution

Nouvelles **Données** :

- Web 2.0 : Facebook, Twitter, news, blogs, ...
- LOD : graphes, ontologies, ...
- Flux : capteurs, GPS, …

➔ très gros volumes, données pas ou faiblement structurées

Nouveaux **Traitements** :

- Moteurs de recherche
- Extraction, analyse, ...
- Recommandation, filtrage collaboratif, …

➔ transformation, agrégation, indexation

Nouvelles **Infrastructures** :

- Cluster, réseaux mobiles, microprocesseurs multi-coeurs, …

➔ distribution, parallélisation, redondance

L'évolution conduit à la spécialisation

Systèmes « noSQL » (**n**ot **o**nly **SQL**):

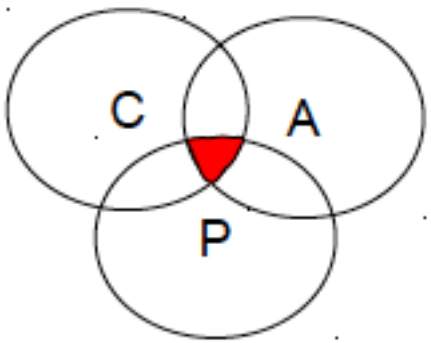
- **Facilité d'utilisation**
- Cohérence des données
- **Persistance des données**
- Fiabilité (Pannes)
- **Efficacité**
- ~~Universalité~~

- Langages spécialisés
- Données hétérogènes
- Réplication
- Parallélisation
- Indexation de contenus

« Systèmes sur mesure »

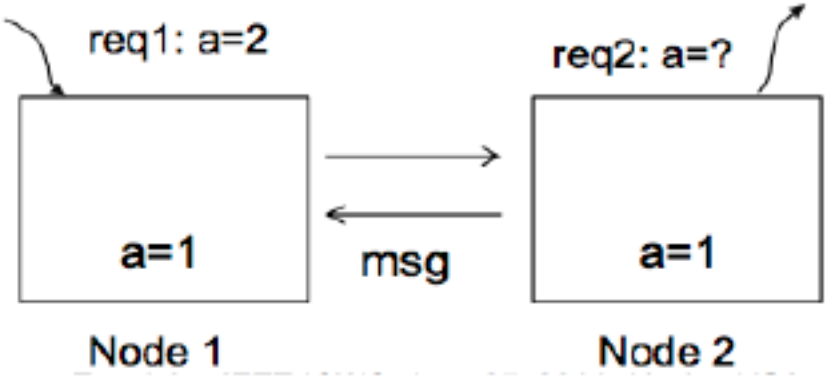
Cohérence, Disponibilité, Pannes

- Théorème CAP : dans un système distribué il est impossible de garantir à chaque instant t plus que *deux parmi les trois* propriétés suivantes :
- **C**onsistency (cohérence) :
  - tous les noeuds voient la même version
- **A**vailability (disponibilité) :
  - chaque requête obtient une réponse
- **P**artition tolerance (résistance à une panne partielle) :
  - la perte de messages n'empêche pas le système de continuer à fonctionner



Cohérence, Disponibilité, Pannes

- Impossibilité d'assurer C (cohérence), A (Disponibilité) et P (tolérance aux pannes) en même temps

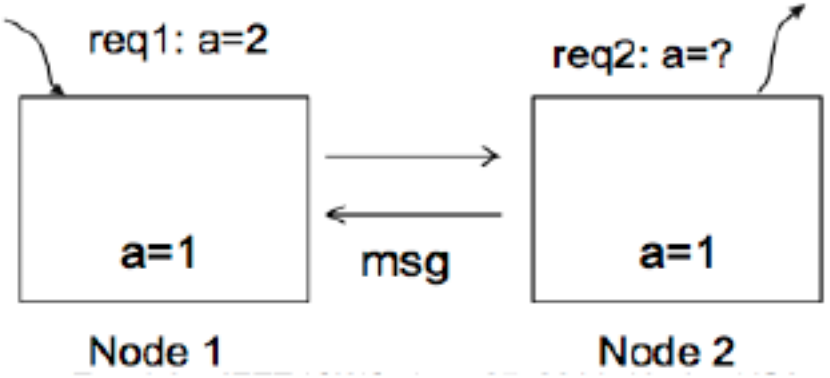


A et C : si A alors Node2 réponds à req2

si C alors la variable **a** de Node2 contient 2

Cohérence, Disponibilité, Pannes

- Impossibilité d'assurer C (cohérence), A (Disponibilité) et P (tolérance aux pannes) en même temps

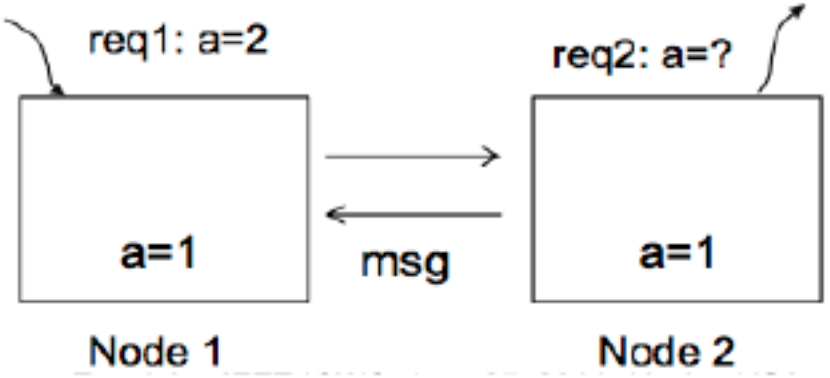


A et P : si A alors Node2 réponds à req2

si P alors le msg de Node1 vers Node2 perdu et donc a=1 sur ce dernier

Cohérence, Disponibilité, Pannes

- Impossibilité d'assurer C (cohérence), A (Disponibilité) et P (tolérance aux pannes) en même temps



C et P : Exercice

SQL ← vs → NoSQL

Cohérence forte :

- Logique : Schémas, contraintes
- Physique : Transactions ACID

Distribution des **données**

- Transactions distribuées

Ressources limitées

- Optimisation de requêtes

Langage **standard** : SQL

Cohérence faible :

- Schémas, contraintes
- Cohérence « à terme »

Distribution des **traitements** :

- Traitements « Batch »
- MapReduce

Ressources « **illimitées** »

- Passage à L'échelle horizontal

Langages **spécialisés**, API

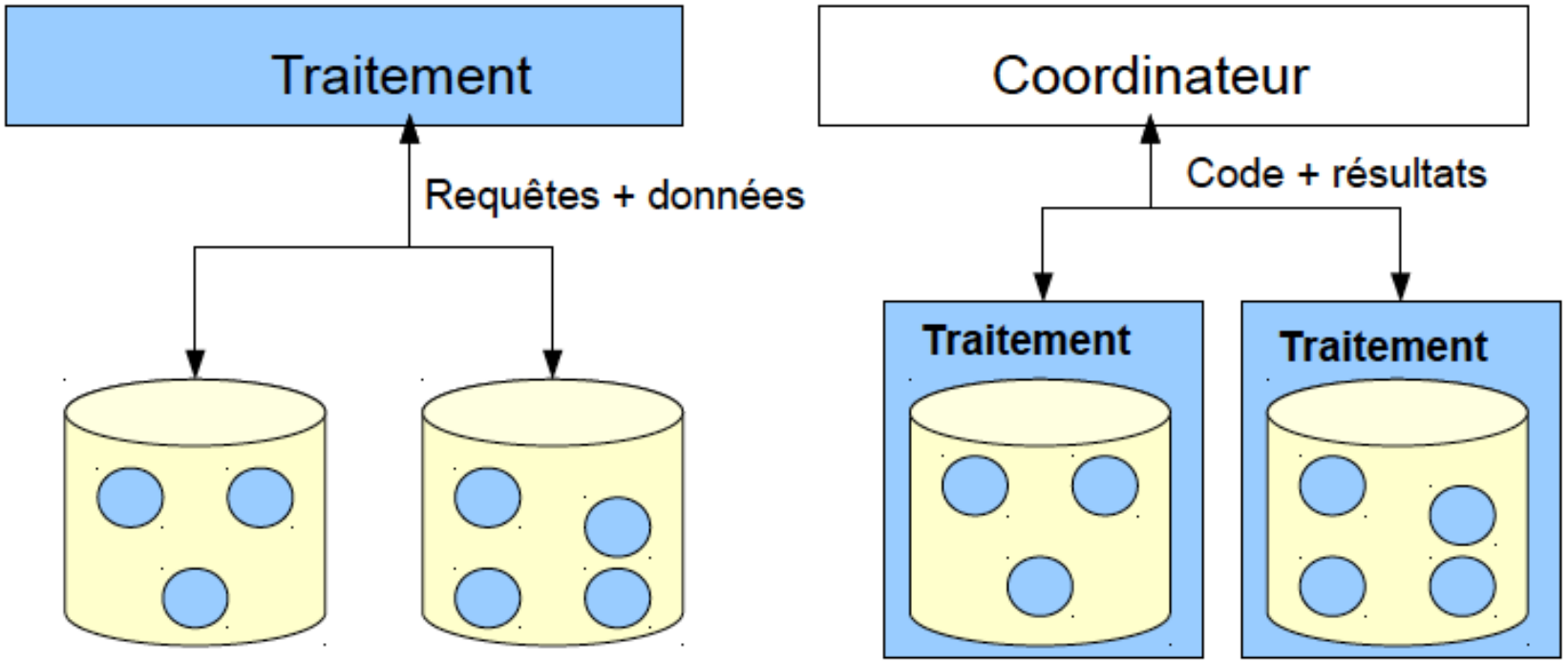
Infrastructures RAIN: le Cloud

- **R**edundant **A**rray of **I**ndependent **N**odes (cloud)
- Infrastructure à faible coût :
  - PC, open-source, LAN générique
- Tolérance aux fautes :
  - redondance du matériel, des données et des traitements
- Administration facile :
  - Architecture « shared-nothing »
  - Virtualisation
- Utilisation facile :
  - Modèles de programmation restreint : MapReduce

SQL ← vs → NoSQL

- Traitements centralisés
- Accès distribué

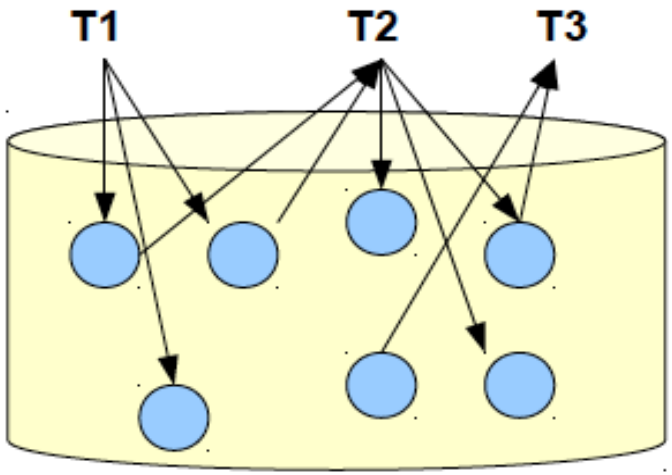
- Traitements distribués
- Accès local



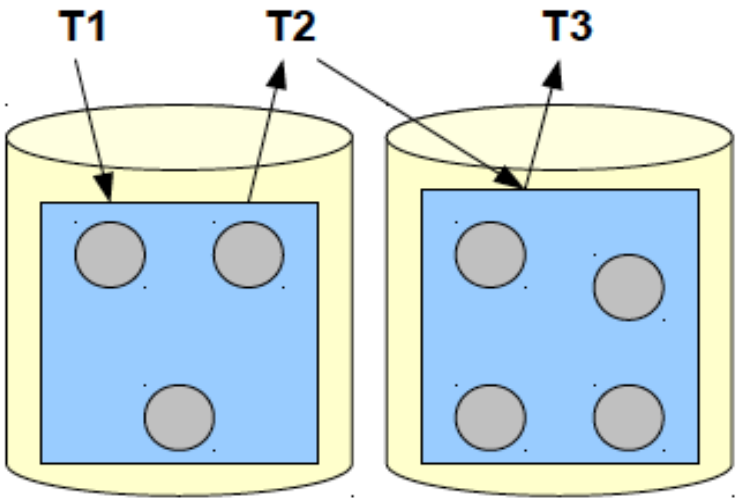
SQL ← vs → NoSQL

Accès à grain fin :

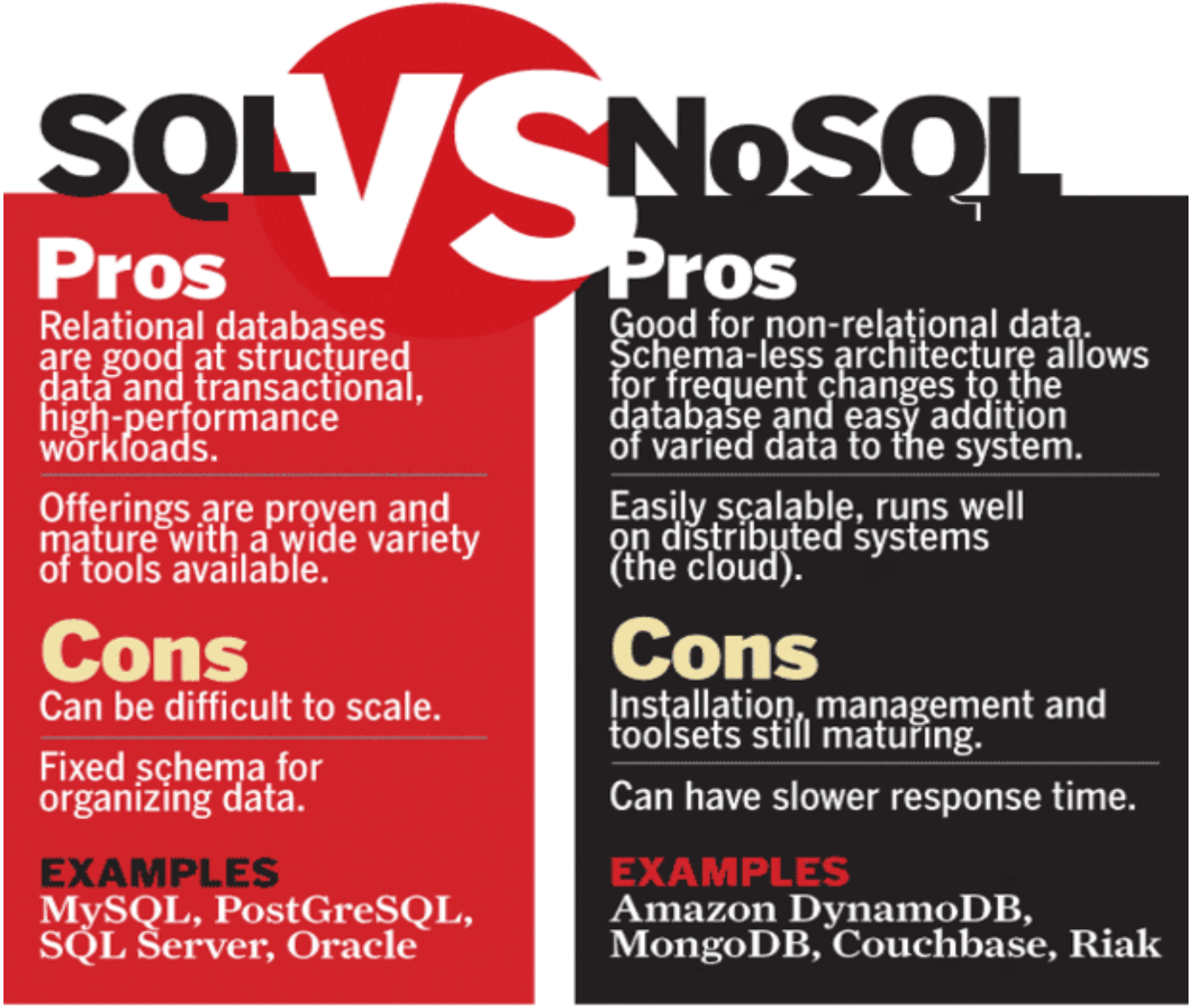
- beaucoup de lectures / écritures de de petits objets



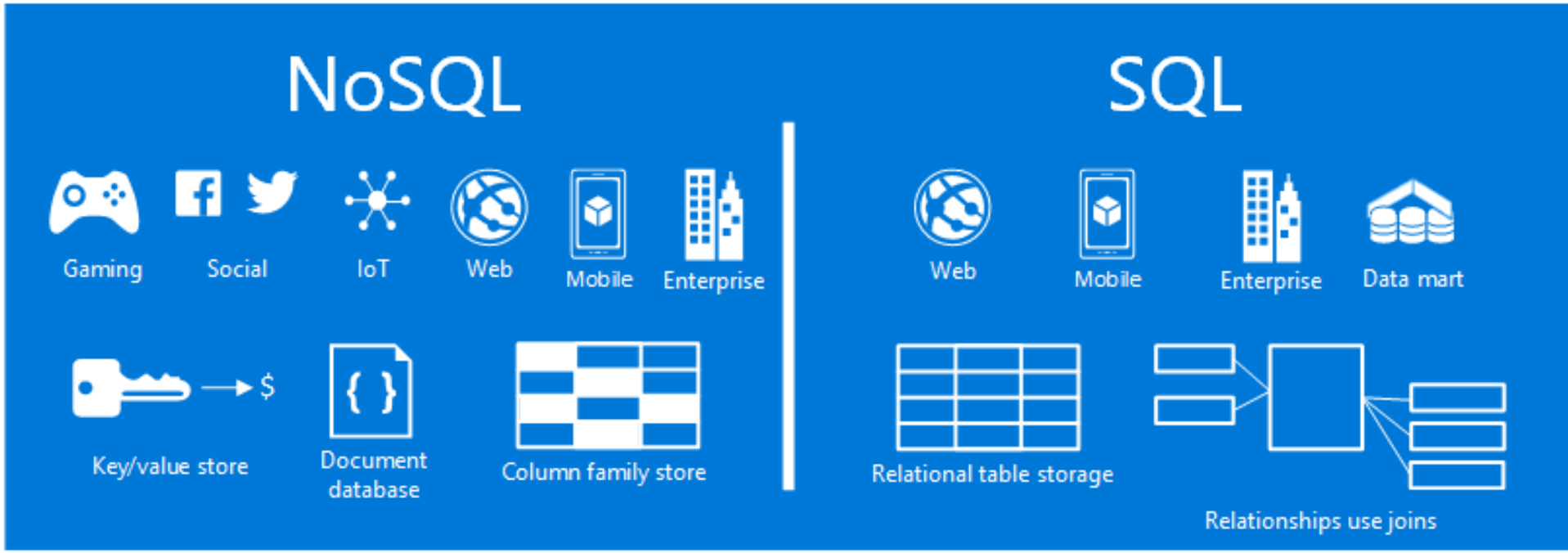
- Accès « batch » :
  - peu de lectures / écritures de grands objets



Vue sur le Web



Vue sur le Web



Vers le NoSQL

- Etendre / adapter un SGBD traditionnel :
  - niveaux de concurrence, indexes, stockage
- Définir des systèmes spécialisés pour
  - une infrastructure (distribuée) : cloud, clusters
  - un type de données : profils, documents XML, RDF, ...
  - un type de traitements : partage, analyse/agrégation, visualisation

Classification de systèmes

- Classer selon:
  - Les types de données : tables, clés/valeurs, arbres, graphes, documents
  - Le paradigme (langages) : map/reduce (PIG, Hive)
  - L'API / Protocole : JSON/REST
  - La persistance : mémoire, disque, Cloud...
  - La gestion de concurrence / cohérence
  - La réplication, protocoles
  - Le langage d'implémentation, ...

Liste (non exhaustive) des systèmes NoSQL

- Document store (DS) :

  Collections de documents

  – Natifs : CouchDB, **MongoDB**, TerraStore, ...

  – Soft : eXist, Virtuoso

- Key-value store (KVS) :

  Absence de schéma

  – DynamoDB, Voldemort, Azure Table Storage, MongoDB, Bigtable, Oracle KV-Store, ...

- Tabular store (TS) :

  Tables

  – Cassandra, Hadoop / Hbase, Hypertable (Bigtable)

- Graph store (GS) :

  Graphes

  – Neo4j, AllegroGraph, InfiniteGraph

Autres types de systèmes NoSQL: XML, Triplestore (RDF), orientés objets, etc.

Vue sur le Web

Acteurs majeurs du NoSQL

- Amazon : DynamoDB, SimpleDB
- Microsoft : Azure Table Storage
- Google : BigTable, Datastore, GFS
- Apache : CouchDB, Cassandra, Hadoop / Hbase
- Beaucoup de start-ups...

Hands on MongoDB

- Install via Atlas
  - https://www.mongodb.com/download-center?jmp=nav#atlas
  - Modèle IaaS: Payant à l'utilisation avec une part gratuite, 100% Cloud
  - Possibilité de choisir son Cloud Provider
  - … et sa région de déploiement