UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Big Data – Concevoir et piloter un projet

Sommaire

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Fondamentaux et cadrage stratégique

Fondamentaux et cadrage stratégique

Étapes d'un projet Big Data

Un projet Big Data ne s'improvise pas : il doit suivre un **cycle méthodologique précis** qui combine **besoins métiers, faisabilité technique, gestion des données et pilotage économique**.

Fondamentaux et cadrage stratégique

Étapes d'un projet Big Data

1 Expression du besoin métier

Avant toute considération technologique, il faut **clarifier les objectifs métiers** :
Le projet doit répondre à une **problématique concrète** et non à une mode technologique.

◆ Points clés

- Identifier les **cas d'usage prioritaires** : marketing, finance, logistique, production, santé…
- Définir les **indicateurs de performance (KPI)** attendus : réduction des coûts, augmentation des ventes, rapidité de traitement, taux de satisfaction client.
- Identifier les **utilisateurs finaux** : analystes, managers, équipes opérationnelles.
- Clarifier les **contraintes métiers** : confidentialité, temps réel, précision des résultats.

Fondamentaux et cadrage stratégique

Étapes d’un projet Big Data

◆ Exemples

- Un distributeur veut réduire de 20 % ses ruptures de stock via l’analyse prédictive de la demande.
- Une banque veut détecter les fraudes en temps réel et diminuer les pertes de 15 %.
- Un hôpital veut optimiser l’occupation des lits et réduire le temps d’attente moyen aux urgences.

👉 **Livrable** : Cahier des charges fonctionnel décrivant besoins, objectifs et KPI.

Fondamentaux et cadrage stratégique

Étapes d'un projet Big Data

2 Étude de faisabilité et cadrage technique

Une fois le besoin défini, il faut vérifier **si le projet est réalisable** techniquement et organisationnellement.

◆ Étapes de cadrage

1. **Analyse des systèmes existants** : infrastructures, bases de données, outils BI déjà en place.
2. **Identification des contraintes techniques** :
   - Volumes de données à traiter (Go, To, Po).
   - Nécessité de temps réel (streaming) ou batch.
   - Compatibilité avec systèmes existants.

Fondamentaux et cadrage stratégique

Étapes d'un projet Big Data

2 Étude de faisabilité et cadrage technique

3. **Choix d'une approche technologique** (sans figer trop tôt) :
   - Hadoop vs Spark vs Cloud.
   - Data Lake vs Data Warehouse vs hybride.
   - NoSQL vs SQL.
4. **Évaluation organisationnelle** : compétences disponibles (data engineer, data scientist, architecte), besoin de formation ou de recrutement.

Fondamentaux et cadrage stratégique

Étapes d'un projet Big Data

2️⃣ Étude de faisabilité et cadrage technique

🔹 Méthodes

- **POC (Proof of Concept)** : prototype limité pour tester faisabilité.
- **Pilote** : déploiement sur un périmètre restreint avant généralisation.

🔹 Exemple

- Une compagnie aérienne lance un POC avec Spark Streaming pour analyser les données moteurs en temps réel avant un déploiement global.

👉 **Livrable** : Document de cadrage technique (solutions envisagées, contraintes, premiers choix d'architecture).

Fondamentaux et cadrage stratégique

Étapes d'un projet Big Data

3 Identification des données disponibles et manquantes

Le **cœur du Big Data** est la donnée elle-même : il faut identifier ce qui existe, ce qui est manquant et la qualité des sources.

◆ Étapes

1. **Inventaire des données existantes** : bases internes (ERP, CRM, transactions, logs).
2. **Données externes** : open data, données partenaires, réseaux sociaux, capteurs IoT.
3. **Analyse de la qualité des données** : cohérence, complétude, exactitude, doublons.
4. **Identification des manques** : quelles données sont nécessaires mais absentes ?
    - Exemple : un projet de maintenance prédictive peut nécessiter l'ajout de capteurs IoT sur les machines.
5. **Modalités d'acquisition** : achat, collecte en continu, API, web scraping.

Fondamentaux et cadrage stratégique

Étapes d'un projet Big Data

3 Identification des données disponibles et manquantes

🔹 Exemples

- Dans un projet de recommandation e-commerce : données disponibles = historique d'achats ; données manquantes = navigation anonyme des visiteurs → besoin de cookies et logs web.
- Dans un projet santé : données existantes = dossiers patients ; données manquantes = suivi en temps réel via objets connectés.

👉 **Livrable** : Cartographie des données (sources, formats, qualité, manquants).

Fondamentaux et cadrage stratégique

Étapes d’un projet Big Data

4 Estimation coûts, délais, ROI

Enfin, un projet Big Data doit être évalué sur le plan **économique et temporel** pour arbitrer et valider son lancement.

◆ Estimation des coûts

- **Infrastructures** : serveurs, cloud, stockage, licences logicielles.
- **Développement & intégration** : POC, pipelines de données, visualisations.
- **Ressources humaines** : data engineers, data scientists, formation interne.
- **Maintenance & exploitation** : monitoring, sécurité, gouvernance.

Fondamentaux et cadrage stratégique

Étapes d’un projet Big Data

4 Estimation coûts, délais, ROI

◆ Estimation des délais

- Découper le projet en **jalons** : cadrage, POC, pilote, déploiement, industrialisation.
- Méthodologie Agile (sprints, MVP) ou cycle en V selon l’organisation.
- Évaluer le **time-to-market** nécessaire : un projet de détection de fraude doit être opérationnel rapidement.

Fondamentaux et cadrage stratégique

Étapes d'un projet Big Data

4 Estimation coûts, délais, ROI

◆ Mesure du ROI (Return On Investment)

- **ROI quantitatif** :
  - Augmentation du chiffre d'affaires (ex. recommandations personnalisées).
  - Réduction des coûts (ex. optimisation logistique, maintenance prédictive).
- **ROI qualitatif** :
  - Amélioration de la satisfaction client.
  - Réputation et image de marque (innovation).
  - Meilleure conformité réglementaire.

Fondamentaux et cadrage stratégique

Étapes d'un projet Big Data

4 Estimation coûts, délais, ROI

🔹 Exemple chiffré

- Projet de maintenance prédictive :
  - Coût : 2 M€ (capteurs + développement).
  - Économies prévues : 5 M€ sur 3 ans (réduction pannes et arrêts de production).
  - ROI positif en moins de 18 mois.

👉 **Livrable** : Business case complet (budget, planning, ROI, risques).

Fondamentaux et cadrage stratégique

Étapes d'un projet Big Data

Synthèse visuelle

Étape	Objectifs	Livrables
Expression du besoin métier	Définir objectifs, KPI, utilisateurs	Cahier des charges fonctionnel
Étude de faisabilité & cadrage	Vérifier contraintes techniques, tester	Document de cadrage + POC
Identification des données	Lister sources, qualité, manquants	Cartographie des données
Estimation coûts/délais/ROI	Arbitrer lancement, valider financement	Business case complet

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Architecture et conception

Architecture et conception

Conception de l'architecture Big Data

1) Collecte : batch vs streaming (Kafka, Flume)

1.1 Décider batch vs streaming

Critères

- **Latence cible** : reporting J+1 → *batch* ; détection fraude < 2 s → *streaming*.
- **Variabilité du débit** : pics imprévisibles → *streaming* + backpressure.
- **Coûts** : batch = peu de compute en continu ; streaming = compute 24/7.
- **Exactitude** : forte exigence d'exactement-une-fois → *streaming* avec transactions / idempotence.
- **Source** : systèmes transactionnels (CDC) → *streaming* ; exports planifiés → *batch*.

Décision rapide

- **Batch** : charges **ELT/ETL planifiées**, recharges historiques, calculs lourds non temps réel.
- **Streaming** : **événements** (clics, IoT, paiements), **CDC** (Change Data Capture), alerting en temps réel.

Architecture et conception

Conception de l’architecture Big Data

1) Collecte : batch vs streaming (Kafka, Flume)

1.2 Pipeline batch (exemples)

- **Sources** : exports SQL (dump), fichiers S3/Azure Blob/GCS, SFTP, API paginées.
- **Ingestion** : jobs Airflow/Dagster/Prefect, *windowing* temporel, checksum & contrôle de volumes.
- **Contrats de données** : schéma (Avro/JSON), règles de qualité en amont, *schema-on-read* au lac.
- **Fiabilité** : dépôts *landing* immuables, renommage atomique, idempotence (écriture *overwrite* partitionnée).
- **Qualité** : Great Expectations/Deequ (complétude, unicité, ranges) + *data drift* basique (distribution).

Architecture et conception

Conception de l'architecture Big Data

1) Collecte : batch vs streaming (Kafka, Flume)

1.3 Pipeline streaming

- **Kafka** (référence industrielle)
  - **Concepts** : *topics*, **partitions** (par clé), **consumer groups** (scalabilité), **réplication**.
  - **Sémantiques** : *at-least-once* par défaut ; **exactly-once** via **transactions** + producteurs idempotents + *read-process-write* atomique.
  - **Rétention** : temps (ex. 7 jours) ou **log compaction** (dernière valeur par clé).
  - **Schémas** : **Schema Registry** (Avro/Protobuf/JSON), politiques d'évolution (*backward/forward/full*).
  - **CDC** : **Debezium** (MySQL/Postgres/Oracle) → topics Kafka (insert/update/delete), *outbox pattern* pour services.

Architecture et conception

Conception de l'architecture Big Data

2) Stockage : Data Lake vs Data Warehouse vs Lakehouse

2.1 Data Lake

- **Support** : objet (S3/ADLS/GCS, HDFS).
- **Formats** : **Parquet/ORC** (colonnaires), **Avro** (row, schéma fort).
- **Zonage (médallion)** :
  - **Bronze/raw** : données brutes, immuables.
  - **Silver/clean** : normalisées, qualité appliquée.
  - **Gold/serve** : orientées usages (BI/ML), parfois dénormalisées.
- **+** : faible coût, tous formats, idéal ML/IA & historisation longue.
- **–** : gouvernance SQL & *concurrency* difficiles sans table format ACID.

Architecture et conception

Conception de l'architecture Big Data

2) Stockage : Data Lake vs Data Warehouse vs Lakehouse

2.1 Data Lake

Tables ACID sur le lac

- **Delta Lake / Apache Iceberg / Apache Hudi** :
  - **ACID**, *time travel*, **schema evolution**, MERGE/UPSERT, **vacuum/compaction**, *data skipping*.
  - **Choix** :
    - **Delta** : écosystème Spark/Dbricks, simple et complet.
    - **Iceberg** : moteur-agnostique (Spark, Flink, Trino, Snowflake), tables évoluées, partitionnement caché.
    - **Hudi** : fort sur **UPSERT/CDC** (*Copy-On-Write*, *Merge-On-Read*).

Architecture et conception

Conception de l'architecture Big Data

2) Stockage : Data Lake vs Data Warehouse vs Lakehouse

2.2 Data Warehouse (EDW)

- **MPP SQL** (Snowflake, BigQuery, Redshift, Synapse, Vertica).
- **Modélisation** : **étoile/flocon**, *conformed dimensions*, SCD (Type 1/2).
- **Patron** : **ELT** (charger brut → transformer en SQL dans l'EDW, **dbt**).
- **+** : gouvernance SQL robuste, *optimizer*, sécurité fine, **BI** native.
- **–** : moins adapté aux **données non structurées** & workloads ML lourds ; coût selon volumétrie/scan.

Architecture et conception

Conception de l'architecture Big Data

2) Stockage : Data Lake vs Data Warehouse vs Lakehouse

2.3 Lakehouse

- **Idée** : unifier **lac (faible coût, flexibilité)** et **entreposage (gouvernance SQL/ACID)** via Delta/Iceberg/Hudi + moteur SQL (Spark SQL/Trino/DuckDB/warehouse natif).
- **Avantages** : **une seule copie** des données pour BI & ML, *time travel*, gouvernance unifiée.
- **Points d'attention** : tuning partitions, **small files problem** (→ *compaction*, *OPTIMIZE*, *bin packing*), indexation (Z-Order, clustering).

Architecture et conception

Conception de l'architecture Big Data

2) Stockage : Data Lake vs Data Warehouse vs Lakehouse

2.4 Matrice de choix (raccourci)

Critère	Data Lake	EDW	Lakehouse
Types de données	Tous (struct./non struct.)	Structuré	Tous
Coût stockage	★★ (bas)	★★★	★★
BI SQL gouvernée	★	★★★	★★–★★★
ML/IA	★★★	★★	★★★
Upsert/ACID	(avec Delta/Iceberg/Hudi)	Natif	Natif (table format)
Time-to-value	Moyen	Rapide	Rapide

Architecture et conception

Conception de l'architecture Big Data

2) Stockage : Data Lake vs Data Warehouse vs Lakehouse

Bonnes pratiques stockage

- **Partitionnement** : champs peu cardinal (date, pays) ; éviter partitions ultra fines.
- **Taille fichiers** : viser 128–1024 MB ; planifier **compaction**.
- **Catalog** : Hive Metastore/Glue/Unity/OtterTune-like ; **catalog central** + *data lineage*.
- **Sécurité** : RBAC/ABAC, chiffrement KMS, **masquage**/tokenization, *row/column-level security*.

Architecture et conception

Conception de l'architecture Big Data

3) Traitement : Spark, MapReduce, ETL

3.1 Apache Spark (standard moderne)

- **APIs** : DataFrame/Dataset, **Spark SQL**, **Structured Streaming**.
- **Moteurs** : YARN / **Kubernetes** / Standalone.
- **Pattern** : **ELT/ETL** (batch et micro-batch) + **streaming unifié**.
- **Optimisation** :
  - **AQE** (Adaptive Query Execution), **broadcast joins**, *predicate pushdown*, *cache*, *checkpointing*.
  - Gérer **shuffle** (coût), **numPartitions** cohérents, *coalesce/repartition*.
- **Streaming** : fenêtres tumbling/sliding, **watermarks**, *stateful ops*, *exactly-once* (avec sinks ACID/transactions).
- **Sinks** : Delta/Iceberg/Hudi, Kafka, JDBC, REST, Elasticsearch.

Architecture et conception

Conception de l'architecture Big Data

3) Traitement : Spark, MapReduce, ETL

3.2 MapReduce (héritage)

- **Usage résiduel** : jobs simples, historiques, où la latence n'a aucune importance.
- **Limites** : très lent (multi-phase disque), difficile à maintenir ; préférer **Spark/Flink**.

3.3 ETL/ELT & orchestration

- **Orchestrateurs** : **Airflow**, **Dagster**, **Prefect** (DAGs, *retry*, SLA, backfills).
- **ELT** : ingérer brut → **dbt** (transformations SQL versionnées, tests, *semantic layer*).
- **Flux** : **NiFi** (flow-based), **Kafka Connect** (connecteurs), **Logstash/Fluent Bit** (logs).
- **Qualité/Observabilité** : Great Expectations/Deequ, Monte Carlo/Bigeye (SLA, fraîcheur, volumétrie, anomalies).
- **SCD** : Type 1 (écrasement) / Type 2 (historisation) → MERGE sur Delta/Iceberg/Hudi.
- **Tests** : unitaires (UDF), intégration (échantillons), *data tests* (contrats), *canary runs*.

Architecture et conception

Conception de l’architecture Big Data

3) Traitement : Spark, MapReduce, ETL

3.2 MapReduce (héritage)

- **Usage résiduel** : jobs simples, historiques, où la latence n’a aucune importance.
- **Limites** : très lent (multi-phase disque), difficile à maintenir ; préférer **Spark/Flink**.

3.3 ETL/ELT & orchestration

Performance & coûts

- **Small files** : regrouper (**OPTIMIZE**, *compaction jobs*).
- **Skew** : *salting*, *skew join hints*.
- **Cache** : parcimonieux (mémoire), libérer quand inutile.
- **Autoscaling** : Kubernetes/Serverless (prévoir *warm pools* pour streaming).

Architecture et conception

Conception de l’architecture Big Data

4) Analyse & restitution : BI, ML, IA, dashboards

4.1 BI & datamarts

- **Couches Gold → datamarts** (étoile/flocon), KPI normalisés, glossaire métier.
- **Sémantique** : modèles (Looker/Power BI semantic model, dbt metrics), **RLS/CLS** (row/column-level security).
- **Actualisation** : *scheduled refresh* (batch) vs *DirectQuery/Live* (quasi-temps réel).
- **Gouvernance** : *certified datasets*, *data lineage*, tests d’acceptation métier.

Architecture et conception

Conception de l'architecture Big Data

4) Analyse & restitution : BI, ML, IA, dashboards

4.2 ML & IA

- **Feature pipelines** : jeux **Silver → Features Gold**, **feature store** (Feast/Tecton).
- **Entraînement** : Spark MLlib/Sklearn/TensorFlow/PyTorch, **tracking** MLflow (params, métriques, artefacts).
- **Déploiement** :
  - **Batch scoring** (jobs planifiés vers tables *serving*),
  - **Online** (API temps réel, Kafka stream processors).
- **MLOps** : registry, **A/B testing**, *shadow mode*, surveillance (dérive données/modèles), alertes.
- **Explicabilité** : SHAP/LIME, rapports de biais, cartes de chaleur des caractéristiques.

Architecture et conception

Conception de l’architecture Big Data

4) Analyse & restitution : BI, ML, IA, dashboards

4.3 Dashboards & apps de données

- **Design** : temps de chargement < 3 s, **drill-down**, *alerting* (seuils/KPI), annotations d’événements.
- **Caching** : *materialized views*, *result cache*, *aggregations tables*.
- **Distribution** : portail BI, email/PDF programmés, **APIs/embeds**, **reverse ETL** (HubSpot, Salesforce, outils opérationnels).

Architecture et conception

Conception de l'architecture Big Data

5) Blueprints d'architecture (références rapides)

5.1 "Batch-first" (EDW + lac)

1. **Collecte batch** (Airflow) → **Lake Bronze (Parquet)**
2. Nettoyage/standardisation → **Lake Silver (Delta/Iceberg)**
3. Modélisation **étoile** → **EDW** (dbt)
4. **BI** sur EDW ; **ML** sur Lake Silver/Gold
5. **Gouvernance** : Catalog + Qualité + Lineage

Architecture et conception

Conception de l’architecture Big Data

5) Blueprints d’architecture (références rapides)

5.2 “Streaming-first” (événementiel + lakehouse)

1. **CDC/événements → Kafka** (+ Schema Registry, DLQ)
2. **Structured Streaming (Spark) → Delta/Iceberg** (UPSERT)
3. **Gold tables** (agrégats quasi-temps réel)
4. **BI live** (Trino/warehouse sur tables ACID) & **features** pour ML online
5. **Observabilité** : métriques pipeline, SLA fraîcheur, *data contracts*

Architecture et conception

Choix technologiques et gouvernance

Le succès d'un projet Big Data dépend non seulement de l'architecture technique (collecte, stockage, traitement), mais aussi de **choix stratégiques sur l'infrastructure** et d'une **gouvernance solide** (sécurité, conformité, ownership).

Architecture et conception

Choix technologiques et gouvernance

1️⃣ Cloud (AWS, Azure, GCP) vs On-premise

🔹 Cloud Public (AWS, Azure, GCP)

Avantages :

- **Élasticité & scalabilité** : ressources quasi illimitées, ajustables en temps réel.
- **Time-to-market rapide** : services managés (pas de gestion de serveurs).
- **Écosystème riche** : IA, ML, BI intégrés (AWS SageMaker, Azure ML, BigQuery ML).
- **Pay-as-you-go** : paiement à l'usage (CapEx → OpEx).

Architecture et conception

Choix technologiques et gouvernance

1️⃣ Cloud (AWS, Azure, GCP) vs On-premise

🔹 Cloud Public (AWS, Azure, GCP)

Limites :

- Dépendance au fournisseur (**vendor lock-in**).
- Coûts parfois élevés si mauvaise gouvernance (data egress, ressources non éteintes).
- Enjeux de **souveraineté** (données sensibles, réglementations).

Architecture et conception

Choix technologiques et gouvernance

1️⃣ Cloud (AWS, Azure, GCP) vs On-premise

🔹 Cloud Public (AWS, Azure, GCP)

Exemples :

- **AWS** : S3 (lac), EMR (Spark), Redshift (DW), Glue (ETL), Kinesis (streaming).
- **Azure** : Data Lake Storage, Synapse, Databricks, Event Hub, Purview (catalogue).
- **GCP** : BigQuery (DW serverless), Dataflow (streaming batch/stream), Pub/Sub, Vertex AI.

Architecture et conception

Choix technologiques et gouvernance

1 Cloud (AWS, Azure, GCP) vs On-premise

◆ On-premise (datacenters internes)

Avantages :

- **Maîtrise complète** des données (souveraineté, conformité stricte).
- **Coûts fixes** : investissements matériels amortis sur plusieurs années.
- **Flexibilité** dans le choix technologique (open source, architectures personnalisées).

Limites :

- **Capacité limitée** : nécessité d'anticiper les besoins.
- **Complexité opérationnelle** : maintenance matérielle, mises à jour, sécurité.
- **Time-to-market plus long** (installation, paramétrage, équipes spécialisées).

Architecture et conception

Choix technologiques et gouvernance

1️⃣ Cloud (AWS, Azure, GCP) vs On-premise

🔹 On-premise (datacenters internes)

Cas d'usage typiques :

- Institutions financières soumises à régulations strictes (banques centrales).
- Organisations publiques avec données classifiées.

Architecture et conception

Choix technologiques et gouvernance

1️⃣ Cloud (AWS, Azure, GCP) vs On-premise

🔹 Hybride & Multicloud

Souvent le **meilleur compromis** :

- **Hybride** : certaines données sensibles on-premise, autres dans le cloud.
- **Multicloud** : répartition des services (ex. GCP pour IA, AWS pour data lake, Azure pour BI).

Exemples :

- Banque : données clients sensibles on-premise, analyses comportementales dans le cloud.
- Industrie : usines avec stockage local (faible latence), mais consolidation dans un cloud central.

👉 **Bonnes pratiques** :

- Définir une **politique de souveraineté** (où stocker quoi).
- Éviter le lock-in → favoriser standards (Kubernetes, Delta Lake, Kafka multi-cloud).
- Automatiser gouvernance via IaC (Terraform, Ansible).

Architecture et conception

Choix technologiques et gouvernance

2 Règles de sécurité, confidentialité, RGPD

◆ Sécurité des données Big Data

- **Chiffrement** :
  - **At rest** (données stockées) → KMS, clés gérées par client (BYOK).
  - **In transit** (réseaux) → TLS 1.2+, VPN, PrivateLink.
- **Contrôles d'accès** :
  - RBAC (Role-Based Access Control),
  - ABAC (Attribute-Based Access Control, ex. tags « sensitive:true »).
- **Audit & traçabilité** : journalisation des accès, alertes anomalies.
- **Segmentation réseau** : VPC, sous-réseaux privés, firewalls.

Architecture et conception

Choix technologiques et gouvernance

2 Règles de sécurité, confidentialité, RGPD

◆ Confidentialité & RGPD

- **Consentement** : les utilisateurs doivent accepter la collecte.
- **Minimisation** : ne collecter que les données nécessaires.
- **Droit à l'oubli** : capacité de suppression/anonymisation des données.
- **Portabilité** : export des données utilisateur dans un format lisible (JSON, CSV).
- **Data masking / anonymisation** : pseudonymisation des identifiants, hashing des emails.

Architecture et conception

Choix technologiques et gouvernance

2️⃣ Règles de sécurité, confidentialité, RGPD

🔹 Cas d'usage sécurité

- **Banque** : logs chiffrés en temps réel (Kafka + TLS + Kerberos).
- **E-commerce** : masquage des CB dans le Data Lake.
- **Santé** : anonymisation dossiers médicaux pour l'IA.

👉 **Check-list sécurité Big Data** :

- [ ] Chiffrement (stockage + transfert).
- [ ] Contrôles d'accès granulaires.
- [ ] Journalisation des accès.
- [ ] Politique RGPD documentée.
- [ ] Anonymisation systématique des données sensibles.

Architecture et conception

Choix technologiques et gouvernance

3 Gouvernance des données : catalogue, qualité, ownership

◆ Data Catalogue

- **Objectif** : permettre aux équipes de savoir **quelles données existent, où, avec quelle qualité**.
- **Fonctionnalités clés** :
  - Indexation automatique des datasets.
  - Recherche par mots-clés, métadonnées.
  - Lineage (traçabilité : origine → transformations → usage).
  - Tagging (sensible, confidentiel, PII).
- **Outils** : Apache Atlas, AWS Glue Data Catalog, Azure Purview, Collibra, Alation.

Architecture et conception

Choix technologiques et gouvernance

3 Gouvernance des données : catalogue, qualité, ownership

◆ Qualité des données

- **Dimensions** : complétude, cohérence, exactitude, unicité, fraîcheur.
- **Pratiques** :
  - Tests automatiques (Great Expectations, Deequ).
  - Alertes sur dérive (volumes anormaux, valeurs aberrantes).
  - Validation en amont (schema-on-write) ou en aval (schema-on-read).
- **Exemple** :
  - Retail : alerte si taux de remplissage email < 90 %.
  - IoT : alerte si capteurs renvoient 0 valeurs > 5 min.

Architecture et conception

Choix technologiques et gouvernance

3 Gouvernance des données : catalogue, qualité, ownership

◆ Ownership & gouvernance organisationnelle

- **Data Owner** (propriétaire métier) : responsable de la donnée, définit règles d'usage.
- **Data Steward** (référent qualité) : veille à la qualité, applique standards.
- **Data Engineer** : implémente pipelines, assure disponibilité technique.
- **Data Governance Committee** : arbitre les conflits, définit normes globales.

Bonnes pratiques organisationnelles :

- Mettre en place un **Data Governance Board** (IT + métier).
- Instaurer un **Data Catalog** obligatoire pour chaque nouveau dataset.
- Définir des **KPI qualité** : % de doublons, % de complétude, % de fraîcheur < 24h.
- Documenter chaque jeu de données (métadonnées minimales obligatoires).

Architecture et conception

Choix technologiques et gouvernance

Synthèse

Domaine	Pratiques	Outils/Technos	Bénéfices
Infra : Cloud vs On-prem	Choix selon scalabilité, coût, souveraineté	AWS/GCP/Azure, clusters Hadoop	Flexibilité, coûts maîtrisés
Sécurité & RGPD	Chiffrement, RBAC/ABAC, anonymisation, traçabilité	KMS, TLS, Ranger, Kerberos	Conformité légale, confiance clients
Gouvernance	Catalogue, qualité, ownership, comité data	Atlas, Glue, Purview, Collibra	Données fiables, accessibles, documentées

Architecture et conception

Organisation du projet

1) Méthodes de gestion : Cycle en V vs Agile (Scrum, Kanban)

Cycle en V (prédictif)

- Quand l'utiliser : exigences stables, conformité forte (banque, santé), dépendances lourdes (réseau, sécurité), budget verrouillé.
- Phases clés : Cadrage → Spécifs → Conception → Build → Tests → Recette → Mise en prod.
- ◦ : traçabilité, conformité, visibilité coûts/délais.
- − : faible flexibilité, feedback tardif, risque d'effet tunnel.

Architecture et conception

Organisation du projet

1) Méthodes de gestion : Cycle en V vs Agile (Scrum, Kanban)

Scrum (itératif/incrémental)

- Artefacts : Product Backlog, Sprint Backlog, Increment (démo à chaque fin de sprint).
- Rituels : Planning (objectif de sprint), Daily (15 min), Review (démo métier), Retrospective (améliorations).
- Quand l'utiliser : incertitude élevée (données/qualité, exploration ML), besoin de valeur fréquente (dashboards, features).
- Bonnes pratiques data : DoR/DoD orientées data (schémas versionnés, tests DQ, validation de perf modèle, runbook mis à jour).

Architecture et conception

Organisation du projet

1) Méthodes de gestion : Cycle en V vs Agile (Scrum, Kanban)

Kanban (flux tiré)

- Principes : visualiser le flux, limiter le WIP, mesurer & améliorer (lead/cycle time).
- Colonnes typiques data : Discovery → Sourcing → Transformation → Validation DQ/QA → Déploiement → Monitoring.
- Idéal pour : run, dataops, petites équipes pluridisciplinaires, flux de tickets (anomalies, petites évolutions).

Architecture et conception

Organisation du projet

1) Méthodes de gestion : Cycle en V vs Agile (Scrum, Kanban)

Choisir vite

- Forte incertitude/POC/ML ? → Scrum.
- Flux de petites demandes & incidents ? → Kanban.
- Projet réglementé à périmètre figé ? → V.
- Organisation “plateforme + domaines” ? → Scrum par domaine + Kanban pour la plateforme.

Architecture et conception

Organisation du projet

2) Rôles & responsabilités (focus principaux)

Product Owner (PO)

- Mission : maximiser la valeur métier, gérer la roadmap & le backlog, définir l'acceptation.
- Livrables : Vision, OKR, user stories & critères Gherkin, priorisation (Value/Effort, WSJF).
- KPI : time-to-value, adoption, NPS interne, objectifs produits atteints.

Architecte (Data/Entreprise)

- Mission : cible d'architecture, choix technos, normes (sécurité, RGPD), coûts/fiabilité, interopérabilité.
- Livrables : diagrammes, ADR (decisions log), standards (naming, versioning), modèles de données, data contracts.
- KPI : disponibilité/capacité, coûts unitaires, conformité, dette technique.

Architecture et conception

Organisation du projet

2) Rôles & responsabilités (focus principaux)

Data Engineer (DE)

- Mission : ingestion/ELT, qualité & fiabilité (tests, monitoring), CI/CD, MLOps/DatOps, performance & coûts.
- Livrables : pipelines, jobs, tests DQ, schémas versionnés, orchs (DAG), runbooks/alertes.
- KPI : fraîcheur, taux d'échec des jobs, SLA/SLO, coût/Go, MTTR.

Architecture et conception

Organisation du projet

2) Rôles & responsabilités (focus principaux)

Data Analyst (DA)

- **Mission** : traduire les questions métier en analyses actionnables, concevoir/maintenir les KPI, produire des insights fiables, évangéliser la data auprès des équipes (self-service & data literacy).
- **Livrables** : tableaux de bord & rapports (storytelling), requêtes SQL/notes d'analyse reproductibles, définitions de métriques (dictionnaire KPI + règles de calcul), études ad-hoc & A/B tests, cahiers de besoins BI, guide d'usage des dashboards.
- **KPI** : adoption des tableaux de bord, time-to-insight (délai demande→réponse), exactitude des KPI (écarts vs "source of truth"), taux de self-service, taux d'analyses menant à une action business (features lancées, coûts évités, revenus incrémentaux).

Architecture et conception

Organisation du projet

2) Rôles & responsabilités (focus principaux)

Qui fait quoi ?

- **Enterprise Architect** : fixe les **principes** et standards globaux (urbanisation SI, cloud strategy).
- **Data Architect** : **conçoit l'architecture data cible** (lakehouse, DWH, MDM, streaming, catalog, sécurité, formats, SLA).
- **Solution Architect** : **assemble** les briques pour **un projet** donné (choix concrets, diagrammes solution).
- **Data Platform / Cloud Engineers** : **construisent la plateforme** (infra cloud, stockage, compute, IAM, réseau, catalog, jobs schedulers).

Architecture et conception

Organisation du projet

2) Rôles & responsabilités (focus principaux)

Qui fait quoi ?

- **Data Engineers** : **développent les pipelines** d'ingestion & de transformation (batch/stream), tests DQ, contrats de données.
- **Analytics Engineers** : **modélisent le couche analytique** (staging → marts, semantic layer, tests dbt).
- **DBA/DBRE** : exploitent et optimisent les **bases/engines** (perfs, sauvegardes, réplication, RPO/RTO).
- **DataOps/SRE** : **observabilité & run** (SLI/SLO, alerting, fiabilité, MTTR).
- **Security/RSSI & DPO** : politiques **IAM, chiffrement, RGPD** (revues, audits).
- **Data Steward / Data Owner** : **gouvernance & qualité** (catalog, définitions KPI, règles DQ, propriétaires).

Architecture et conception

Pilotage opérationnel d'un projet Big Data

- Le **pilotage opérationnel** vise à assurer que le projet **avance selon la feuille de route**, respecte les contraintes (temps, coûts, qualité) et atteint les **objectifs métier**.
- Dans le contexte Big Data, il doit intégrer la **complexité technologique**, la **qualité des données** et la **conduite du changement** côté utilisateurs.

Architecture et conception

Pilotage opérationnel d'un projet Big Data

1 Roadmap, jalons, livrables

◆ Roadmap

La roadmap est une **vue globale et temporelle du projet**, structurée autour des grands objectifs métier et techniques.

- Découpée en **phases** :
  1. **Cadrage** (besoin métier, faisabilité, gouvernance)
  2. **Conception** (architecture, choix techno, sécurité)
  3. **Implémentation** (pipelines, stockage, traitements)
  4. **Validation** (tests, qualité données, POC → pilote)
  5. **Déploiement** (production, monitoring, adoption)
  6. **Industrialisation & amélioration continue**

Architecture et conception

Pilotage opérationnel d'un projet Big Data

1 Roadmap, jalons, livrables

◆ Roadmap

Exemple (6 mois) :

- Mois 1 : cadrage, définition cas d'usage, cartographie données
- Mois 2–3 : conception architecture + POC technique
- Mois 4 : développement pipelines ingestion + stockage (zone Bronze/Silver)
- Mois 5 : premiers dashboards et modèles ML → pilote
- Mois 6 : mise en production + formation utilisateurs

Architecture et conception

Pilotage opérationnel d'un projet Big Data

1 Roadmap, jalons, livrables

◆ Jalons (milestones)

Les jalons sont des **points clés de validation**.

- **Exemple Banque (fraude)** :
  - Jalon 1 : POC détection temps réel validé (<2 s latence).
  - Jalon 2 : pipeline Kafka → Data Lake opérationnel.
  - Jalon 3 : modèle ML intégré en scoring.
  - Jalon 4 : dashboards BI fraud management validés par métier.
  - Jalon 5 : mise en production sécurisée.

Architecture et conception

Pilotage opérationnel d'un projet Big Data

1 Roadmap, jalons, livrables

◆ Livrables

Chaque phase doit produire des **artefacts concrets** validés.

- **Cadrage** : cahier de charges, business case, estimation ROI.
- **Conception** : schéma d'architecture, RACI (rôles), plan sécurité/RGPD.
- **Implémentation** : pipelines ETL, data lake structuré, scripts Terraform.
- **Validation** : rapports de tests, tableaux qualité données.
- **Déploiement** : dashboards BI, API ML, manuel utilisateur.
- **Industrialisation** : documentation, monitoring (Grafana/Prometheus), playbooks incident.

Architecture et conception

Pilotage opérationnel d'un projet Big Data

2 Indicateurs clés (KPI)

Le pilotage doit s'appuyer sur des **indicateurs de suivi précis et mesurables**.

◆ Qualité des données

- **Complétude** : % de champs remplis (objectif > 95%).
- **Exactitude** : écart toléré entre données sources et données traitées (<1%).
- **Fraîcheur** : délai entre génération et disponibilité (ex. <5 min pour fraude, <24h pour reporting).
- **Taux d'anomalies détectées** : données invalides, doublons, incohérences.

👉 *Exemple* : sur un projet IoT, 99% des capteurs doivent envoyer des données toutes les 5 minutes.

Architecture et conception

Pilotage opérationnel d'un projet Big Data

2 Indicateurs clés (KPI)

Le pilotage doit s'appuyer sur des **indicateurs de suivi précis et mesurables**.

◆ Performance technique

- **Latence de traitement** : temps ingestion → mise à dispo (batch J+1 ou streaming <2 s).
- **Débit** : nombre d'événements/s traités (scalabilité).
- **Disponibilité** : SLA (ex. 99,9 % uptime du pipeline Kafka/Spark).
- **Temps de réponse** : API de recommandation < 300 ms (p95).

👉 *Exemple* : moteur de recommandation retail → 95% des requêtes < 200 ms.

Architecture et conception

Pilotage opérationnel d'un projet Big Data

2 Indicateurs clés (KPI)

🔹 Adoption métier

- **Taux d'utilisation des dashboards** : nombre de connexions hebdo / utilisateurs cibles.
- **Taux d'appropriation** : % de métiers formés qui utilisent réellement la solution.
- **Impact métier** :
  - Banque : % fraudes détectées supplémentaires.
  - Retail : augmentation du panier moyen.
  - Industrie : réduction des arrêts machine.

👉 *Exemple* : objectif 80% des managers logistique connectés chaque semaine au dashboard prédictif.

Architecture et conception

Pilotage opérationnel d'un projet Big Data

3 Suivi budgétaire et risques

◆ Suivi budgétaire

- **CapEx (investissement initial)** : serveurs, licences, mise en place cloud.
- **OpEx (coûts récurrents)** : stockage cloud (S3, ADLS), compute (Spark, EMR, Databricks), monitoring.
- **Ressources humaines** : Data Engineers, Data Scientists, formation.
- **Outils de suivi** :
  - Tableaux de bord financiers (mensuels).
  - Alertes budget cloud (AWS Budgets, Azure Cost Management, GCP Billing).

👉 *Exemple* : seuil d'alerte si coûts cloud mensuels > +10% du prévisionnel.

Architecture et conception

Pilotage opérationnel d'un projet Big Data

3 Suivi budgétaire et risques

◆ Gestion des risques

Typologie des risques Big Data :

1. **Techniques** :
   - Pipeline non scalable (Kafka saturé).
   - Dérive de schéma → job cassé.
   - Données manquantes → modèles biaisés.
   - → *Parade* : tests de charge, data contracts, monitoring qualité.
2. **Organisationnels** :
   - Manque de compétences internes.
   - Dépendance à un seul fournisseur cloud (**vendor lock-in**).
   - → *Parade* : formation, multicloud, adoption standards (Kubernetes, Delta Lake).

Architecture et conception

Pilotage opérationnel d'un projet Big Data

3 Suivi budgétaire et risques

◆ Gestion des risques

Typologie des risques Big Data :

3. **Réglementaires et éthiques** :
   - Non-conformité RGPD (données sensibles stockées sans consentement).
   - Biais algorithmiques (discrimination dans scoring crédit).
   - → *Parade* : DPO impliqué, audits, IA responsable (explicabilité).
4. **Adoption métier** :
   - Outils livrés mais non utilisés.
   - Métiers non impliqués dans conception.
   - → *Parade* : ateliers réguliers, feedback utilisateurs, accompagnement au changement.

Architecture et conception

Pilotage opérationnel d'un projet Big Data

Dimension	Bonnes pratiques	Exemple
Roadmap & jalons	Découpage phases + milestones clairs	POC validé en 2 mois, pilote en 4 mois
Livrables	Artefacts concrets validés par sponsor	Schéma archi, pipelines, dashboards
KPI Qualité	Complétude, fraîcheur, anomalies	Fraîcheur <5 min, exactitude 99%
KPI Performance	Latence, débit, SLA	API reco <300 ms (p95)
KPI Adoption	Usage réel, impact métier	80% managers connectés/semaine
Budget	Suivi CapEx/OpEx + alertes cloud	Alerte > +10% budget prévu
Risques	Typologie + registre + mitigation	Vendor lock-in → multicloud