Big Data

Master 2

Komlan Mawuli FOOVI,Doctorant en Intelligence Artificielle(Depuis 2023)

Groupe SUPINFO

09 Février 2025

Définition et Concepts Fondamentaux

- **Big Data :** Ensemble des techniques, technologies et méthodologies permettant de collecter, stocker, traiter et analyser de très grands volumes de données.

Définition et Concepts Fondamentaux

- **Big Data :** Ensemble des techniques, technologies et méthodologies permettant de collecter, stocker, traiter et analyser de très grands volumes de données.
- **Nature des données :**

Définition et Concepts Fondamentaux

- **Big Data :** Ensemble des techniques, technologies et méthodologies permettant de collecter, stocker, traiter et analyser de très grands volumes de données.
- **Nature des données :**
  - **Structurées :** Données organisées dans des bases relationnelles.

Définition et Concepts Fondamentaux

- **Big Data :** Ensemble des techniques, technologies et méthodologies permettant de collecter, stocker, traiter et analyser de très grands volumes de données.
- **Nature des données :**
  - **Structurées :** Données organisées dans des bases relationnelles.
  - **Semi-structurées :** Formats comme XML ou JSON.

Définition et Concepts Fondamentaux

- **Big Data :** Ensemble des techniques, technologies et méthodologies permettant de collecter, stocker, traiter et analyser de très grands volumes de données.
- **Nature des données :**
  - **Structurées :** Données organisées dans des bases relationnelles.
  - **Semi-structurées :** Formats comme XML ou JSON.
  - **Non structurées :** Textes, images, vidéos, logs, etc.

Définition et Concepts Fondamentaux

- **Big Data :** Ensemble des techniques, technologies et méthodologies permettant de collecter, stocker, traiter et analyser de très grands volumes de données.
- **Nature des données :**
  - **Structurées :** Données organisées dans des bases relationnelles.
  - **Semi-structurées :** Formats comme XML ou JSON.
  - **Non structurées :** Textes, images, vidéos, logs, etc.
- **Origines multiples :** Générées par les réseaux sociaux, les capteurs IoT, les transactions financières, etc.

Définition et Concepts Fondamentaux

- **Big Data :** Ensemble des techniques, technologies et méthodologies permettant de collecter, stocker, traiter et analyser de très grands volumes de données.
- **Nature des données :**
  - **Structurées :** Données organisées dans des bases relationnelles.
  - **Semi-structurées :** Formats comme XML ou JSON.
  - **Non structurées :** Textes, images, vidéos, logs, etc.
- **Origines multiples :** Générées par les réseaux sociaux, les capteurs IoT, les transactions financières, etc.
- **Objectif principal :** Extraire des informations stratégiques pour soutenir l'innovation, l'optimisation des processus métiers et la prise de décision.

Définition et Concepts Fondamentaux

- **Évolution historique :**

Définition et Concepts Fondamentaux

- **Évolution historique :**
  - Passage des systèmes traditionnels aux technologies distribuées (ex. Hadoop, Spark).

Définition et Concepts Fondamentaux

- **Évolution historique :**
  - Passage des systèmes traditionnels aux technologies distribuées (ex. Hadoop, Spark).
  - Émergence des infrastructures cloud pour gérer des volumes massifs.

Définition et Concepts Fondamentaux

- **Évolution historique :**
  - Passage des systèmes traditionnels aux technologies distribuées (ex. Hadoop, Spark).
  - Émergence des infrastructures cloud pour gérer des volumes massifs.
- **Approche de Stéphane Tufféry :** Mise en lumière des évolutions technologiques qui transforment l'économie numérique et ouvrent la voie à de nouvelles applications (machine learning, intelligence artificielle, etc.).

Les 5V du Big Data

- **Volume :**

Les 5V du Big Data

- **Volume :**
  - Gestion de quantités de données souvent exprimées en téraoctets, pétaoctets, voire zettaoctets.

Les 5V du Big Data

- **Volume :**
  - Gestion de quantités de données souvent exprimées en téraoctets, pétaoctets, voire zettaoctets.
  - Nécessite des infrastructures massives et évolutives (clusters, cloud computing).

Les 5V du Big Data

- **Volume :**
  - Gestion de quantités de données souvent exprimées en téraoctets, pétaoctets, voire zettaoctets.
  - Nécessite des infrastructures massives et évolutives (clusters, cloud computing).
- **Vélocité :**

Les 5V du Big Data

- **Volume :**
  - Gestion de quantités de données souvent exprimées en téraoctets, pétaoctets, voire zettaoctets.
  - Nécessite des infrastructures massives et évolutives (clusters, cloud computing).
- **Vélocité :**
  - Vitesse à laquelle les données sont générées et doivent être traitées.

Les 5V du Big Data

- **Volume :**
  - Gestion de quantités de données souvent exprimées en téraoctets, pétaoctets, voire zettaoctets.
  - Nécessite des infrastructures massives et évolutives (clusters, cloud computing).
- **Vélocité :**
  - Vitesse à laquelle les données sont générées et doivent être traitées.
  - Exemples : flux en temps réel des réseaux sociaux, transactions financières instantanées.

Les 5V du Big Data

- **Volume :**
  - Gestion de quantités de données souvent exprimées en téraoctets, pétaoctets, voire zettaoctets.
  - Nécessite des infrastructures massives et évolutives (clusters, cloud computing).
- **Vélocité :**
  - Vitesse à laquelle les données sont générées et doivent être traitées.
  - Exemples : flux en temps réel des réseaux sociaux, transactions financières instantanées.
- **Variété :**

Les 5V du Big Data

- **Volume :**
  - Gestion de quantités de données souvent exprimées en téraoctets, pétaoctets, voire zettaoctets.
  - Nécessite des infrastructures massives et évolutives (clusters, cloud computing).
- **Vélocité :**
  - Vitesse à laquelle les données sont générées et doivent être traitées.
  - Exemples : flux en temps réel des réseaux sociaux, transactions financières instantanées.
- **Variété :**
  - Diversité des formats et sources (texte, images, vidéos, données capteurs).

Les 5V du Big Data

- **Volume :**
  - Gestion de quantités de données souvent exprimées en téraoctets, pétaoctets, voire zettaoctets.
  - Nécessite des infrastructures massives et évolutives (clusters, cloud computing).
- **Vélocité :**
  - Vitesse à laquelle les données sont générées et doivent être traitées.
  - Exemples : flux en temps réel des réseaux sociaux, transactions financières instantanées.
- **Variété :**
  - Diversité des formats et sources (texte, images, vidéos, données capteurs).
  - Implique l'utilisation d'outils spécifiques (bases NoSQL, systèmes de gestion de contenu multimédia).

Les 5V du Big Data

- **Véracité :**

Les 5V du Big Data

- **Véracité :**
  - Qualité, fiabilité et cohérence des données.

Les 5V du Big Data

- **Véracité :**
  - Qualité, fiabilité et cohérence des données.
  - Importance de la gestion des erreurs, des données manquantes ou erronées pour garantir la pertinence des analyses.

Les 5V du Big Data

- **Véracité :**
  - Qualité, fiabilité et cohérence des données.
  - Importance de la gestion des erreurs, des données manquantes ou erronées pour garantir la pertinence des analyses.
- **Valeur :**

Les 5V du Big Data

- **Véracité :**
  - Qualité, fiabilité et cohérence des données.
  - Importance de la gestion des erreurs, des données manquantes ou erronées pour garantir la pertinence des analyses.
- **Valeur :**
  - Capacité à extraire des informations stratégiques à partir des données brutes.

Les 5V du Big Data

- **Véracité :**
  - Qualité, fiabilité et cohérence des données.
  - Importance de la gestion des erreurs, des données manquantes ou erronées pour garantir la pertinence des analyses.
- **Valeur :**
  - Capacité à extraire des informations stratégiques à partir des données brutes.
  - Impact direct sur la performance économique et la prise de décisions éclairées.

Importance du Big Data dans la Transformation Numérique

- **Innovation et compétitivité :**

Importance du Big Data dans la Transformation Numérique

- **Innovation et compétitivité :**
  - Identification de nouvelles opportunités de marché grâce à l'analyse des tendances.

Importance du Big Data dans la Transformation Numérique

- **Innovation et compétitivité :**
  - Identification de nouvelles opportunités de marché grâce à l'analyse des tendances.
  - Développement de produits et services personnalisés.

Importance du Big Data dans la Transformation Numérique

- **Innovation et compétitivité :**
  - Identification de nouvelles opportunités de marché grâce à l'analyse des tendances.
  - Développement de produits et services personnalisés.
- **Optimisation des processus métiers :**

Importance du Big Data dans la Transformation Numérique

- **Innovation et compétitivité :**
  - Identification de nouvelles opportunités de marché grâce à l'analyse des tendances.
  - Développement de produits et services personnalisés.
- **Optimisation des processus métiers :**
  - Utilisation de l'analyse prédictive pour anticiper les besoins et les dysfonctionnements.

Importance du Big Data dans la Transformation Numérique

- **Innovation et compétitivité :**
  - Identification de nouvelles opportunités de marché grâce à l'analyse des tendances.
  - Développement de produits et services personnalisés.
- **Optimisation des processus métiers :**
  - Utilisation de l'analyse prédictive pour anticiper les besoins et les dysfonctionnements.
  - Amélioration de la gestion des ressources et de la chaîne logistique.

Importance du Big Data dans la Transformation Numérique

- **Innovation et compétitivité :**
  - Identification de nouvelles opportunités de marché grâce à l'analyse des tendances.
  - Développement de produits et services personnalisés.
- **Optimisation des processus métiers :**
  - Utilisation de l'analyse prédictive pour anticiper les besoins et les dysfonctionnements.
  - Amélioration de la gestion des ressources et de la chaîne logistique.
- **Décision stratégique :**

Importance du Big Data dans la Transformation Numérique

- **Innovation et compétitivité :**
  - Identification de nouvelles opportunités de marché grâce à l'analyse des tendances.
  - Développement de produits et services personnalisés.
- **Optimisation des processus métiers :**
  - Utilisation de l'analyse prédictive pour anticiper les besoins et les dysfonctionnements.
  - Amélioration de la gestion des ressources et de la chaîne logistique.
- **Décision stratégique :**
  - Appui sur des données concrètes pour orienter les stratégies d'entreprise.

Importance du Big Data dans la Transformation Numérique

- **Innovation et compétitivité :**
  - Identification de nouvelles opportunités de marché grâce à l'analyse des tendances.
  - Développement de produits et services personnalisés.
- **Optimisation des processus métiers :**
  - Utilisation de l'analyse prédictive pour anticiper les besoins et les dysfonctionnements.
  - Amélioration de la gestion des ressources et de la chaîne logistique.
- **Décision stratégique :**
  - Appui sur des données concrètes pour orienter les stratégies d'entreprise.
  - Réduction des risques par une meilleure connaissance des comportements clients et des processus opérationnels.

Importance du Big Data dans la Transformation Numérique

- **Exemples sectoriels :**

Importance du Big Data dans la Transformation Numérique

- **Exemples sectoriels :**
  - **Santé :** Diagnostics, suivi des patients et recherche médicale.

Importance du Big Data dans la Transformation Numérique

- **Exemples sectoriels :**
  - **Santé :** Diagnostics, suivi des patients et recherche médicale.
  - **Industrie :** Maintenance prédictive, optimisation de la production.

Importance du Big Data dans la Transformation Numérique

- **Exemples sectoriels :**
  - **Santé :** Diagnostics, suivi des patients et recherche médicale.
  - **Industrie :** Maintenance prédictive, optimisation de la production.
  - **Administration publique :** Gestion intelligente des villes et amélioration des services citoyens.

Éthique et Responsabilité dans l'Exploitation des Données Massives

- **Protection de la vie privée :**

Éthique et Responsabilité dans l'Exploitation des Données Massives

- **Protection de la vie privée :**
  - Respect des réglementations (RGPD, Loi Informatique et Libertés).

Éthique et Responsabilité dans l'Exploitation des Données Massives

- **Protection de la vie privée :**
  - Respect des réglementations (RGPD, Loi Informatique et Libertés).
  - Mise en place de mécanismes d'anonymisation et de sécurisation des données sensibles.

Éthique et Responsabilité dans l'Exploitation des Données Massives

- **Protection de la vie privée :**
  - Respect des réglementations (RGPD, Loi Informatique et Libertés).
  - Mise en place de mécanismes d'anonymisation et de sécurisation des données sensibles.
- **Transparence et consentement :**

Éthique et Responsabilité dans l'Exploitation des Données Massives

- **Protection de la vie privée :**
  - Respect des réglementations (RGPD, Loi Informatique et Libertés).
  - Mise en place de mécanismes d'anonymisation et de sécurisation des données sensibles.
- **Transparence et consentement :**
  - Information claire des utilisateurs quant aux modalités de collecte et d'utilisation de leurs données.

Éthique et Responsabilité dans l'Exploitation des Données Massives

- **Protection de la vie privée :**
  - Respect des réglementations (RGPD, Loi Informatique et Libertés).
  - Mise en place de mécanismes d'anonymisation et de sécurisation des données sensibles.
- **Transparence et consentement :**
  - Information claire des utilisateurs quant aux modalités de collecte et d'utilisation de leurs données.
  - Obtention d'un consentement explicite et éclairé.

Éthique et Responsabilité dans l'Exploitation des Données Massives

- **Protection de la vie privée :**
  - Respect des réglementations (RGPD, Loi Informatique et Libertés).
  - Mise en place de mécanismes d'anonymisation et de sécurisation des données sensibles.
- **Transparence et consentement :**
  - Information claire des utilisateurs quant aux modalités de collecte et d'utilisation de leurs données.
  - Obtention d'un consentement explicite et éclairé.
- **Biais et discrimination :**

Éthique et Responsabilité dans l'Exploitation des Données Massives

- **Protection de la vie privée :**
  - Respect des réglementations (RGPD, Loi Informatique et Libertés).
  - Mise en place de mécanismes d'anonymisation et de sécurisation des données sensibles.
- **Transparence et consentement :**
  - Information claire des utilisateurs quant aux modalités de collecte et d'utilisation de leurs données.
  - Obtention d'un consentement explicite et éclairé.
- **Biais et discrimination :**
  - Veiller à l'absence de biais dans les algorithmes d'analyse.

Éthique et Responsabilité dans l'Exploitation des Données Massives

- **Protection de la vie privée :**
  - Respect des réglementations (RGPD, Loi Informatique et Libertés).
  - Mise en place de mécanismes d'anonymisation et de sécurisation des données sensibles.
- **Transparence et consentement :**
  - Information claire des utilisateurs quant aux modalités de collecte et d'utilisation de leurs données.
  - Obtention d'un consentement explicite et éclairé.
- **Biais et discrimination :**
  - Veiller à l'absence de biais dans les algorithmes d'analyse.
  - Assurer l'équité dans les processus de décision automatisée.

Éthique et Responsabilité dans l'Exploitation des Données Massives

- **Responsabilité sociétale :**

Éthique et Responsabilité dans l'Exploitation des Données Massives

- **Responsabilité sociétale :**
  - Impact sur l'emploi et la transformation des métiers.

Éthique et Responsabilité dans l'Exploitation des Données Massives

- **Responsabilité sociétale :**
  - Impact sur l'emploi et la transformation des métiers.
  - Nécessité d'une gouvernance éthique et transparente dans l'exploitation des données.

Éthique et Responsabilité dans l'Exploitation des Données Massives

- **Responsabilité sociétale :**
  - Impact sur l'emploi et la transformation des métiers.
  - Nécessité d'une gouvernance éthique et transparente dans l'exploitation des données.
- **Conformité légale :**

Éthique et Responsabilité dans l'Exploitation des Données Massives

- **Responsabilité sociétale :**
  - Impact sur l'emploi et la transformation des métiers.
  - Nécessité d'une gouvernance éthique et transparente dans l'exploitation des données.
- **Conformité légale :**
  - Collaboration avec les autorités de régulation pour s'assurer du respect des normes en vigueur.

Conclusion sur Big Data et Enjeux Actuels

- Le Big Data transforme fondamentalement la manière dont les entreprises et les institutions exploitent l'information.

Conclusion sur Big Data et Enjeux Actuels

- Le Big Data transforme fondamentalement la manière dont les entreprises et les institutions exploitent l'information.
- Les 5V offrent un cadre analytique pour appréhender les défis technologiques et stratégiques liés aux données massives.

Conclusion sur Big Data et Enjeux Actuels

- Le Big Data transforme fondamentalement la manière dont les entreprises et les institutions exploitent l'information.
- Les 5V offrent un cadre analytique pour appréhender les défis technologiques et stratégiques liés aux données massives.
- Une exploitation éthique et responsable des données est essentielle pour assurer une transformation numérique durable et respectueuse des droits individuels.

Modèles d'Architecture : Lambda, Kappa et Zeta

- **Architecture Lambda :**

Modèles d'Architecture : Lambda, Kappa et Zeta

- **Architecture Lambda :**
  - Architecture à deux couches : couche de traitement par lots et couche de traitement en temps réel.

Modèles d'Architecture : Lambda, Kappa et Zeta

- **Architecture Lambda :**
  - Architecture à deux couches : couche de traitement par lots et couche de traitement en temps réel.
  - Permet de traiter les données en temps réel et par lots de manière parallèle.

Modèles d'Architecture : Lambda, Kappa et Zeta

- **Architecture Lambda :**
  - Architecture à deux couches : couche de traitement par lots et couche de traitement en temps réel.
  - Permet de traiter les données en temps réel et par lots de manière parallèle.
  - Utilisation d'un système de traitement en temps réel (par exemple, Apache Kafka) et d'un moteur de traitement par lots (ex. Apache Hadoop).

Modèles d'Architecture : Lambda, Kappa et Zeta

- **Architecture Lambda :**
  - Architecture à deux couches : couche de traitement par lots et couche de traitement en temps réel.
  - Permet de traiter les données en temps réel et par lots de manière parallèle.
  - Utilisation d'un système de traitement en temps réel (par exemple, Apache Kafka) et d'un moteur de traitement par lots (ex. Apache Hadoop).
  - Exemple d'utilisation : traitement des logs d'activité, analyse des données en temps réel et historique.

Modèles d'Architecture : Lambda, Kappa et Zeta

- **Architecture Lambda :**
  - Architecture à deux couches : couche de traitement par lots et couche de traitement en temps réel.
  - Permet de traiter les données en temps réel et par lots de manière parallèle.
  - Utilisation d'un système de traitement en temps réel (par exemple, Apache Kafka) et d'un moteur de traitement par lots (ex. Apache Hadoop).
  - Exemple d'utilisation : traitement des logs d'activité, analyse des données en temps réel et historique.
- **Architecture Kappa :**

Modèles d'Architecture : Lambda, Kappa et Zeta

- **Architecture Lambda :**
  - Architecture à deux couches : couche de traitement par lots et couche de traitement en temps réel.
  - Permet de traiter les données en temps réel et par lots de manière parallèle.
  - Utilisation d'un système de traitement en temps réel (par exemple, Apache Kafka) et d'un moteur de traitement par lots (ex. Apache Hadoop).
  - Exemple d'utilisation : traitement des logs d'activité, analyse des données en temps réel et historique.
- **Architecture Kappa :**
  - Simplification de l'architecture Lambda : tout est traité en temps réel à l'aide de technologies comme Apache Kafka et Apache Flink.

Modèles d'Architecture : Lambda, Kappa et Zeta

- **Architecture Lambda :**
  - Architecture à deux couches : couche de traitement par lots et couche de traitement en temps réel.
  - Permet de traiter les données en temps réel et par lots de manière parallèle.
  - Utilisation d'un système de traitement en temps réel (par exemple, Apache Kafka) et d'un moteur de traitement par lots (ex. Apache Hadoop).
  - Exemple d'utilisation : traitement des logs d'activité, analyse des données en temps réel et historique.
- **Architecture Kappa :**
  - Simplification de l'architecture Lambda : tout est traité en temps réel à l'aide de technologies comme Apache Kafka et Apache Flink.
  - Élimine la complexité de la gestion des traitements par lots, en se concentrant uniquement sur le traitement en temps réel.

Modèles d'Architecture : Lambda, Kappa et Zeta

- **Architecture Lambda :**
  - Architecture à deux couches : couche de traitement par lots et couche de traitement en temps réel.
  - Permet de traiter les données en temps réel et par lots de manière parallèle.
  - Utilisation d'un système de traitement en temps réel (par exemple, Apache Kafka) et d'un moteur de traitement par lots (ex. Apache Hadoop).
  - Exemple d'utilisation : traitement des logs d'activité, analyse des données en temps réel et historique.
- **Architecture Kappa :**
  - Simplification de l'architecture Lambda : tout est traité en temps réel à l'aide de technologies comme Apache Kafka et Apache Flink.
  - Élimine la complexité de la gestion des traitements par lots, en se concentrant uniquement sur le traitement en temps réel.
  - Avantages : simplicité de la gestion et réduction du nombre d'outils

Modèles d'Architecture : Lambda, Kappa et Zeta

- **Architecture Lambda :**
  - Architecture à deux couches : couche de traitement par lots et couche de traitement en temps réel.
  - Permet de traiter les données en temps réel et par lots de manière parallèle.
  - Utilisation d'un système de traitement en temps réel (par exemple, Apache Kafka) et d'un moteur de traitement par lots (ex. Apache Hadoop).
  - Exemple d'utilisation : traitement des logs d'activité, analyse des données en temps réel et historique.
- **Architecture Kappa :**
  - Simplification de l'architecture Lambda : tout est traité en temps réel à l'aide de technologies comme Apache Kafka et Apache Flink.
  - Élimine la complexité de la gestion des traitements par lots, en se concentrant uniquement sur le traitement en temps réel.
  - Avantages : simplicité de la gestion et réduction du nombre d'outils

Modèles d'Architecture : Lambda, Kappa et Zeta

- **Architecture Zeta :**

Modèles d'Architecture : Lambda, Kappa et Zeta

- **Architecture Zeta :**
  - Combine les meilleures caractéristiques de Lambda et Kappa, tout en introduisant une gestion plus flexible des données.

Modèles d'Architecture : Lambda, Kappa et Zeta

- **Architecture Zeta :**
  - Combine les meilleures caractéristiques de Lambda et Kappa, tout en introduisant une gestion plus flexible des données.
  - Permet de traiter des données en temps réel tout en maintenant une vue globale de l'historique.

Modèles d'Architecture : Lambda, Kappa et Zeta

- **Architecture Zeta :**
  - Combine les meilleures caractéristiques de Lambda et Kappa, tout en introduisant une gestion plus flexible des données.
  - Permet de traiter des données en temps réel tout en maintenant une vue globale de l'historique.
  - Recommandée pour des applications où la flexibilité et l'adaptabilité sont primordiales.

Bases de données NoSQL : MongoDB, Cassandra, HBase

- **MongoDB :**

Bases de données NoSQL : MongoDB, Cassandra, HBase

- **MongoDB :**
  - Base de données orientée document.

Bases de données NoSQL : MongoDB, Cassandra, HBase

- **MongoDB :**
  - Base de données orientée document.
  - Utilise le format BSON (Binary JSON) pour stocker les documents.

Bases de données NoSQL : MongoDB, Cassandra, HBase

- **MongoDB :**
  - Base de données orientée document.
  - Utilise le format BSON (Binary JSON) pour stocker les documents.
  - Permet de gérer des données semi-structurées et non structurées.

Bases de données NoSQL : MongoDB, Cassandra, HBase

- **MongoDB :**
  - Base de données orientée document.
  - Utilise le format BSON (Binary JSON) pour stocker les documents.
  - Permet de gérer des données semi-structurées et non structurées.
  - Avantages : évolutivité horizontale, flexibilité dans le schéma de données, prise en charge des requêtes complexes.

Bases de données NoSQL : MongoDB, Cassandra, HBase

- **MongoDB :**
  - Base de données orientée document.
  - Utilise le format BSON (Binary JSON) pour stocker les documents.
  - Permet de gérer des données semi-structurées et non structurées.
  - Avantages : évolutivité horizontale, flexibilité dans le schéma de données, prise en charge des requêtes complexes.
  - Cas d'utilisation : gestion des données utilisateur dans des applications mobiles ou Web.

Bases de données NoSQL : MongoDB, Cassandra, HBase

- **MongoDB :**
  - Base de données orientée document.
  - Utilise le format BSON (Binary JSON) pour stocker les documents.
  - Permet de gérer des données semi-structurées et non structurées.
  - Avantages : évolutivité horizontale, flexibilité dans le schéma de données, prise en charge des requêtes complexes.
  - Cas d'utilisation : gestion des données utilisateur dans des applications mobiles ou Web.
- **Cassandra :**

Bases de données NoSQL : MongoDB, Cassandra, HBase

- **MongoDB :**
  - Base de données orientée document.
  - Utilise le format BSON (Binary JSON) pour stocker les documents.
  - Permet de gérer des données semi-structurées et non structurées.
  - Avantages : évolutivité horizontale, flexibilité dans le schéma de données, prise en charge des requêtes complexes.
  - Cas d'utilisation : gestion des données utilisateur dans des applications mobiles ou Web.
- **Cassandra :**
  - Base de données distribuée, orientée colonnes.

Bases de données NoSQL : MongoDB, Cassandra, HBase

- **MongoDB :**
  - Base de données orientée document.
  - Utilise le format BSON (Binary JSON) pour stocker les documents.
  - Permet de gérer des données semi-structurées et non structurées.
  - Avantages : évolutivité horizontale, flexibilité dans le schéma de données, prise en charge des requêtes complexes.
  - Cas d'utilisation : gestion des données utilisateur dans des applications mobiles ou Web.
- **Cassandra :**
  - Base de données distribuée, orientée colonnes.
  - Permet une scalabilité horizontale et une gestion efficace des gros volumes de données.

Bases de données NoSQL : MongoDB, Cassandra, HBase

- **MongoDB :**
  - Base de données orientée document.
  - Utilise le format BSON (Binary JSON) pour stocker les documents.
  - Permet de gérer des données semi-structurées et non structurées.
  - Avantages : évolutivité horizontale, flexibilité dans le schéma de données, prise en charge des requêtes complexes.
  - Cas d'utilisation : gestion des données utilisateur dans des applications mobiles ou Web.
- **Cassandra :**
  - Base de données distribuée, orientée colonnes.
  - Permet une scalabilité horizontale et une gestion efficace des gros volumes de données.
  - Avantages : disponibilité, tolérance aux pannes, haute performance.

Bases de données NoSQL : MongoDB, Cassandra, HBase

- **MongoDB :**
  - Base de données orientée document.
  - Utilise le format BSON (Binary JSON) pour stocker les documents.
  - Permet de gérer des données semi-structurées et non structurées.
  - Avantages : évolutivité horizontale, flexibilité dans le schéma de données, prise en charge des requêtes complexes.
  - Cas d'utilisation : gestion des données utilisateur dans des applications mobiles ou Web.
- **Cassandra :**
  - Base de données distribuée, orientée colonnes.
  - Permet une scalabilité horizontale et une gestion efficace des gros volumes de données.
  - Avantages : disponibilité, tolérance aux pannes, haute performance.
  - Cas d'utilisation : gestion des données financières, applications nécessitant une haute disponibilité.

Bases de données NoSQL : MongoDB, Cassandra, HBase

- **HBase :**

Bases de données NoSQL : MongoDB, Cassandra, HBase

- **HBase :**
  - Base de données distribuée, compatible avec Hadoop.

Bases de données NoSQL : MongoDB, Cassandra, HBase

- **HBase :**
  - Base de données distribuée, compatible avec Hadoop.
  - Conçue pour le traitement en temps réel des données massives.

Bases de données NoSQL : MongoDB, Cassandra, HBase

- **HBase :**
  - Base de données distribuée, compatible avec Hadoop.
  - Conçue pour le traitement en temps réel des données massives.
  - S'appuie sur HDFS pour le stockage de données.

Bases de données NoSQL : MongoDB, Cassandra, HBase

- **HBase :**
  - Base de données distribuée, compatible avec Hadoop.
  - Conçue pour le traitement en temps réel des données massives.
  - S'appuie sur HDFS pour le stockage de données.
  - Cas d'utilisation : analyse de logs en temps réel, systèmes de recommandation.

Stockage Distribué : HDFS, Amazon S3, Google BigQuery

- **HDFS (Hadoop Distributed File System) :**

Stockage Distribué : HDFS, Amazon S3, Google BigQuery

- **HDFS (Hadoop Distributed File System) :**
  - Système de fichiers distribué permettant de stocker des données massives de manière fiable.

Stockage Distribué : HDFS, Amazon S3, Google BigQuery

- **HDFS (Hadoop Distributed File System) :**
  - Système de fichiers distribué permettant de stocker des données massives de manière fiable.
  - Utilisé dans l'écosystème Hadoop pour le stockage des données avant leur traitement.

Stockage Distribué : HDFS, Amazon S3, Google BigQuery

- **HDFS (Hadoop Distributed File System) :**
  - Système de fichiers distribué permettant de stocker des données massives de manière fiable.
  - Utilisé dans l'écosystème Hadoop pour le stockage des données avant leur traitement.
  - Avantages : tolérance aux pannes, évolutivité horizontale, architecture décentralisée.

Stockage Distribué : HDFS, Amazon S3, Google BigQuery

- **HDFS (Hadoop Distributed File System) :**
  - Système de fichiers distribué permettant de stocker des données massives de manière fiable.
  - Utilisé dans l'écosystème Hadoop pour le stockage des données avant leur traitement.
  - Avantages : tolérance aux pannes, évolutivité horizontale, architecture décentralisée.
  - Cas d'utilisation : stockage des données de traitement par lots dans les systèmes Hadoop.

Stockage Distribué : HDFS, Amazon S3, Google BigQuery

- **HDFS (Hadoop Distributed File System) :**
  - Système de fichiers distribué permettant de stocker des données massives de manière fiable.
  - Utilisé dans l'écosystème Hadoop pour le stockage des données avant leur traitement.
  - Avantages : tolérance aux pannes, évolutivité horizontale, architecture décentralisée.
  - Cas d'utilisation : stockage des données de traitement par lots dans les systèmes Hadoop.
- **Amazon S3 (Simple Storage Service) :**

Stockage Distribué : HDFS, Amazon S3, Google BigQuery

- **HDFS (Hadoop Distributed File System) :**
  - Système de fichiers distribué permettant de stocker des données massives de manière fiable.
  - Utilisé dans l'écosystème Hadoop pour le stockage des données avant leur traitement.
  - Avantages : tolérance aux pannes, évolutivité horizontale, architecture décentralisée.
  - Cas d'utilisation : stockage des données de traitement par lots dans les systèmes Hadoop.
- **Amazon S3 (Simple Storage Service) :**
  - Solution de stockage cloud offrant une capacité de stockage illimitée.

Stockage Distribué : HDFS, Amazon S3, Google BigQuery

- **HDFS (Hadoop Distributed File System) :**
  - Système de fichiers distribué permettant de stocker des données massives de manière fiable.
  - Utilisé dans l'écosystème Hadoop pour le stockage des données avant leur traitement.
  - Avantages : tolérance aux pannes, évolutivité horizontale, architecture décentralisée.
  - Cas d'utilisation : stockage des données de traitement par lots dans les systèmes Hadoop.
- **Amazon S3 (Simple Storage Service) :**
  - Solution de stockage cloud offrant une capacité de stockage illimitée.
  - Permet de stocker et de récupérer des données via des API RESTful.

Stockage Distribué : HDFS, Amazon S3, Google BigQuery

- **HDFS (Hadoop Distributed File System) :**
  - Système de fichiers distribué permettant de stocker des données massives de manière fiable.
  - Utilisé dans l'écosystème Hadoop pour le stockage des données avant leur traitement.
  - Avantages : tolérance aux pannes, évolutivité horizontale, architecture décentralisée.
  - Cas d'utilisation : stockage des données de traitement par lots dans les systèmes Hadoop.
- **Amazon S3 (Simple Storage Service) :**
  - Solution de stockage cloud offrant une capacité de stockage illimitée.
  - Permet de stocker et de récupérer des données via des API RESTful.
  - Avantages : durabilité des données, faible coût, sécurité intégrée.

Stockage Distribué : HDFS, Amazon S3, Google BigQuery

- **HDFS (Hadoop Distributed File System) :**
  - Système de fichiers distribué permettant de stocker des données massives de manière fiable.
  - Utilisé dans l'écosystème Hadoop pour le stockage des données avant leur traitement.
  - Avantages : tolérance aux pannes, évolutivité horizontale, architecture décentralisée.
  - Cas d'utilisation : stockage des données de traitement par lots dans les systèmes Hadoop.
- **Amazon S3 (Simple Storage Service) :**
  - Solution de stockage cloud offrant une capacité de stockage illimitée.
  - Permet de stocker et de récupérer des données via des API RESTful.
  - Avantages : durabilité des données, faible coût, sécurité intégrée.
  - Cas d'utilisation : stockage des données brutes, sauvegarde des fichiers volumineux.

Stockage Distribué : HDFS, Amazon S3, Google BigQuery

- **Google BigQuery :**

Stockage Distribué : HDFS, Amazon S3, Google BigQuery

- **Google BigQuery :**
  - Service de stockage et d'analyse de données massives dans le cloud.

Stockage Distribué : HDFS, Amazon S3, Google BigQuery

- **Google BigQuery :**
  - Service de stockage et d'analyse de données massives dans le cloud.
  - Permet des analyses en temps réel sur des ensembles de données massifs avec SQL.

Stockage Distribué : HDFS, Amazon S3, Google BigQuery

- **Google BigQuery :**
  - Service de stockage et d'analyse de données massives dans le cloud.
  - Permet des analyses en temps réel sur des ensembles de données massifs avec SQL.
  - Avantages : haute performance, scalabilité, intégration avec l'écosystème Google Cloud.

Stockage Distribué : HDFS, Amazon S3, Google BigQuery

- **Google BigQuery :**
  - Service de stockage et d'analyse de données massives dans le cloud.
  - Permet des analyses en temps réel sur des ensembles de données massifs avec SQL.
  - Avantages : haute performance, scalabilité, intégration avec l'écosystème Google Cloud.
  - Cas d'utilisation : analyse de grandes bases de données, analyse des logs.

Virtualisation et Conteneurisation : Docker, Kubernetes

- **Docker :**

Virtualisation et Conteneurisation : Docker, Kubernetes

- **Docker :**
  - Outil de conteneurisation permettant d'exécuter des applications dans des environnements isolés appelés conteneurs.

Virtualisation et Conteneurisation : Docker, Kubernetes

- **Docker :**
  - Outil de conteneurisation permettant d'exécuter des applications dans des environnements isolés appelés conteneurs.
  - Avantages : portabilité, efficacité des ressources, rapidité d'exécution.

Virtualisation et Conteneurisation : Docker, Kubernetes

- **Docker :**
  - Outil de conteneurisation permettant d'exécuter des applications dans des environnements isolés appelés conteneurs.
  - Avantages : portabilité, efficacité des ressources, rapidité d'exécution.
  - Utilisé dans le développement et le déploiement d'applications Big Data.

Virtualisation et Conteneurisation : Docker, Kubernetes

- **Docker :**
  - Outil de conteneurisation permettant d'exécuter des applications dans des environnements isolés appelés conteneurs.
  - Avantages : portabilité, efficacité des ressources, rapidité d'exécution.
  - Utilisé dans le développement et le déploiement d'applications Big Data.
  - Cas d'utilisation : création d'environnements pour Apache Kafka, Hadoop, ou Spark.

Virtualisation et Conteneurisation : Docker, Kubernetes

- **Docker :**
  - Outil de conteneurisation permettant d'exécuter des applications dans des environnements isolés appelés conteneurs.
  - Avantages : portabilité, efficacité des ressources, rapidité d'exécution.
  - Utilisé dans le développement et le déploiement d'applications Big Data.
  - Cas d'utilisation : création d'environnements pour Apache Kafka, Hadoop, ou Spark.
- **Kubernetes :**

Virtualisation et Conteneurisation : Docker, Kubernetes

- **Docker :**
  - Outil de conteneurisation permettant d'exécuter des applications dans des environnements isolés appelés conteneurs.
  - Avantages : portabilité, efficacité des ressources, rapidité d'exécution.
  - Utilisé dans le développement et le déploiement d'applications Big Data.
  - Cas d'utilisation : création d'environnements pour Apache Kafka, Hadoop, ou Spark.
- **Kubernetes :**
  - Orchestrateur de conteneurs permettant de gérer, déployer et scaler des applications conteneurisées.

Virtualisation et Conteneurisation : Docker, Kubernetes

- **Docker :**
  - Outil de conteneurisation permettant d'exécuter des applications dans des environnements isolés appelés conteneurs.
  - Avantages : portabilité, efficacité des ressources, rapidité d'exécution.
  - Utilisé dans le développement et le déploiement d'applications Big Data.
  - Cas d'utilisation : création d'environnements pour Apache Kafka, Hadoop, ou Spark.
- **Kubernetes :**
  - Orchestrateur de conteneurs permettant de gérer, déployer et scaler des applications conteneurisées.
  - Permet une gestion avancée des ressources et de la haute disponibilité.

Virtualisation et Conteneurisation : Docker, Kubernetes

- **Docker :**
  - Outil de conteneurisation permettant d'exécuter des applications dans des environnements isolés appelés conteneurs.
  - Avantages : portabilité, efficacité des ressources, rapidité d'exécution.
  - Utilisé dans le développement et le déploiement d'applications Big Data.
  - Cas d'utilisation : création d'environnements pour Apache Kafka, Hadoop, ou Spark.
- **Kubernetes :**
  - Orchestrateur de conteneurs permettant de gérer, déployer et scaler des applications conteneurisées.
  - Permet une gestion avancée des ressources et de la haute disponibilité.
  - Avantages : automatisation des tâches de gestion des conteneurs, résilience, scalabilité.

Virtualisation et Conteneurisation : Docker, Kubernetes

- **Docker :**
  - Outil de conteneurisation permettant d'exécuter des applications dans des environnements isolés appelés conteneurs.
  - Avantages : portabilité, efficacité des ressources, rapidité d'exécution.
  - Utilisé dans le développement et le déploiement d'applications Big Data.
  - Cas d'utilisation : création d'environnements pour Apache Kafka, Hadoop, ou Spark.
- **Kubernetes :**
  - Orchestrateur de conteneurs permettant de gérer, déployer et scaler des applications conteneurisées.
  - Permet une gestion avancée des ressources et de la haute disponibilité.
  - Avantages : automatisation des tâches de gestion des conteneurs, résilience, scalabilité.

Cloud Computing et Big Data : AWS, Azure, Google Cloud

- **AWS (Amazon Web Services) :**

Cloud Computing et Big Data : AWS, Azure, Google Cloud

- **AWS (Amazon Web Services) :**
  - Offre de services cloud permettant de déployer des solutions Big Data (par exemple, Amazon EMR pour Hadoop, Amazon Redshift pour les bases de données).

Cloud Computing et Big Data : AWS, Azure, Google Cloud

- **AWS (Amazon Web Services) :**
  - Offre de services cloud permettant de déployer des solutions Big Data (par exemple, Amazon EMR pour Hadoop, Amazon Redshift pour les bases de données).
  - Avantages : flexibilité, évolutivité, large éventail de services spécialisés.

Cloud Computing et Big Data : AWS, Azure, Google Cloud

- **AWS (Amazon Web Services) :**
  - Offre de services cloud permettant de déployer des solutions Big Data (par exemple, Amazon EMR pour Hadoop, Amazon Redshift pour les bases de données).
  - Avantages : flexibilité, évolutivité, large éventail de services spécialisés.
  - Cas d'utilisation : analyse des données à grande échelle, gestion de l'infrastructure Big Data.

Cloud Computing et Big Data : AWS, Azure, Google Cloud

- **AWS (Amazon Web Services) :**
  - Offre de services cloud permettant de déployer des solutions Big Data (par exemple, Amazon EMR pour Hadoop, Amazon Redshift pour les bases de données).
  - Avantages : flexibilité, évolutivité, large éventail de services spécialisés.
  - Cas d'utilisation : analyse des données à grande échelle, gestion de l'infrastructure Big Data.
- **Microsoft Azure :**

Cloud Computing et Big Data : AWS, Azure, Google Cloud

- **AWS (Amazon Web Services) :**
  - Offre de services cloud permettant de déployer des solutions Big Data (par exemple, Amazon EMR pour Hadoop, Amazon Redshift pour les bases de données).
  - Avantages : flexibilité, évolutivité, large éventail de services spécialisés.
  - Cas d'utilisation : analyse des données à grande échelle, gestion de l'infrastructure Big Data.
- **Microsoft Azure :**
  - Plateforme cloud de Microsoft avec des services comme Azure HDInsight et Azure Databricks pour Big Data.

Cloud Computing et Big Data : AWS, Azure, Google Cloud

- **AWS (Amazon Web Services) :**
  - Offre de services cloud permettant de déployer des solutions Big Data (par exemple, Amazon EMR pour Hadoop, Amazon Redshift pour les bases de données).
  - Avantages : flexibilité, évolutivité, large éventail de services spécialisés.
  - Cas d'utilisation : analyse des données à grande échelle, gestion de l'infrastructure Big Data.
- **Microsoft Azure :**
  - Plateforme cloud de Microsoft avec des services comme Azure HDInsight et Azure Databricks pour Big Data.
  - Avantages : intégration avec les outils Microsoft, sécurité avancée, gestion simplifiée.

Cloud Computing et Big Data : AWS, Azure, Google Cloud

- **AWS (Amazon Web Services) :**
  - Offre de services cloud permettant de déployer des solutions Big Data (par exemple, Amazon EMR pour Hadoop, Amazon Redshift pour les bases de données).
  - Avantages : flexibilité, évolutivité, large éventail de services spécialisés.
  - Cas d'utilisation : analyse des données à grande échelle, gestion de l'infrastructure Big Data.
- **Microsoft Azure :**
  - Plateforme cloud de Microsoft avec des services comme Azure HDInsight et Azure Databricks pour Big Data.
  - Avantages : intégration avec les outils Microsoft, sécurité avancée, gestion simplifiée.
  - Cas d'utilisation : analyse des données en temps réel, stockage dans le cloud.

Cloud Computing et Big Data : AWS, Azure, Google Cloud

- **AWS (Amazon Web Services) :**
  - Offre de services cloud permettant de déployer des solutions Big Data (par exemple, Amazon EMR pour Hadoop, Amazon Redshift pour les bases de données).
  - Avantages : flexibilité, évolutivité, large éventail de services spécialisés.
  - Cas d'utilisation : analyse des données à grande échelle, gestion de l'infrastructure Big Data.
- **Microsoft Azure :**
  - Plateforme cloud de Microsoft avec des services comme Azure HDInsight et Azure Databricks pour Big Data.
  - Avantages : intégration avec les outils Microsoft, sécurité avancée, gestion simplifiée.
  - Cas d'utilisation : analyse des données en temps réel, stockage dans le cloud.
- **Google Cloud :**

Cloud Computing et Big Data : AWS, Azure, Google Cloud

- **AWS (Amazon Web Services) :**
  - Offre de services cloud permettant de déployer des solutions Big Data (par exemple, Amazon EMR pour Hadoop, Amazon Redshift pour les bases de données).
  - Avantages : flexibilité, évolutivité, large éventail de services spécialisés.
  - Cas d'utilisation : analyse des données à grande échelle, gestion de l'infrastructure Big Data.
- **Microsoft Azure :**
  - Plateforme cloud de Microsoft avec des services comme Azure HDInsight et Azure Databricks pour Big Data.
  - Avantages : intégration avec les outils Microsoft, sécurité avancée, gestion simplifiée.
  - Cas d'utilisation : analyse des données en temps réel, stockage dans le cloud.
- **Google Cloud :**

Cloud Computing et Big Data : AWS, Azure, Google Cloud

- **AWS (Amazon Web Services) :**
  - Offre de services cloud permettant de déployer des solutions Big Data (par exemple, Amazon EMR pour Hadoop, Amazon Redshift pour les bases de données).
  - Avantages : flexibilité, évolutivité, large éventail de services spécialisés.
  - Cas d'utilisation : analyse des données à grande échelle, gestion de l'infrastructure Big Data.
- **Microsoft Azure :**
  - Plateforme cloud de Microsoft avec des services comme Azure HDInsight et Azure Databricks pour Big Data.
  - Avantages : intégration avec les outils Microsoft, sécurité avancée, gestion simplifiée.
  - Cas d'utilisation : analyse des données en temps réel, stockage dans le cloud.
- **Google Cloud :**

Cloud Computing et Big Data : AWS, Azure, Google Cloud

- **AWS (Amazon Web Services) :**
  - Offre de services cloud permettant de déployer des solutions Big Data (par exemple, Amazon EMR pour Hadoop, Amazon Redshift pour les bases de données).
  - Avantages : flexibilité, évolutivité, large éventail de services spécialisés.
  - Cas d'utilisation : analyse des données à grande échelle, gestion de l'infrastructure Big Data.
- **Microsoft Azure :**
  - Plateforme cloud de Microsoft avec des services comme Azure HDInsight et Azure Databricks pour Big Data.
  - Avantages : intégration avec les outils Microsoft, sécurité avancée, gestion simplifiée.
  - Cas d'utilisation : analyse des données en temps réel, stockage dans le cloud.
- **Google Cloud :**

Conclusion sur l'architecture et Infrastructures du Big Data

- Le Big Data repose sur une architecture complexe et distribuée, adaptée à des volumes de données massifs.

Conclusion sur l'architecture et Infrastructures du Big Data

- Le Big Data repose sur une architecture complexe et distribuée, adaptée à des volumes de données massifs.
- Les technologies modernes comme les bases de données NoSQL et les solutions cloud facilitent l'intégration et le traitement des données massives.

Conclusion sur l'architecture et Infrastructures du Big Data

- Le Big Data repose sur une architecture complexe et distribuée, adaptée à des volumes de données massifs.
- Les technologies modernes comme les bases de données NoSQL et les solutions cloud facilitent l'intégration et le traitement des données massives.
- La virtualisation et la conteneurisation offrent des solutions flexibles et évolutives pour le déploiement des systèmes Big Data.

Apache Hadoop

- **HDFS** : Système de fichiers distribué tolérant aux pannes
- **YARN** : Gestionnaire de ressources et planification des tâches
- **MapReduce** : Modèle de programmation pour le traitement distribué

Installation d'Apache Hadoop

- Télécharger Hadoop depuis `https://hadoop.apache.org/releases.html`
- Installer Java :

```
sudo apt update
sudo apt install openjdk-11-jdk
java -version
```

- Décompresser Hadoop et configurer les variables d'environnement

```
tar -xvzf hadoop-3.3.1.tar.gz
export HADOOP_HOME=~/hadoop-3.3.1
export PATH=$PATH:$HADOOP_HOME/bin
```

Apache Spark

- **Architecture** : Cluster, Driver, Executors
- **RDDs** : Résilience et traitement en mémoire
- **DataFrames** : API optimisée pour les données structurées
- **Spark SQL** : Interface pour les requêtes SQL sur Spark

Installation d'Apache Spark

- Télécharger Spark depuis `https://spark.apache.org/downloads.html`
- Installer Scala et Spark :

```
sudo apt install scala
tar -xvzf spark-3.2.1-bin-hadoop3.2.tgz
export SPARK_HOME=~/spark-3.2.1-bin-hadoop3.2
export PATH=$PATH:$SPARK_HOME/bin
```

- Tester l'installation :

```
spark-shell
```

Data Warehousing et Data Lakes

- **Snowflake** : Entrepôt de données cloud scalable
- **Delta Lake** : Gestion des données transactionnelles dans les Data Lakes

Utilisation de Snowflake et Delta Lake

- Snowflake : Inscription sur `https://www.snowflake.com/` et utilisation en SQL
- Delta Lake : Installation avec Spark :

```
pyspark --packages io.delta:delta-core_2.12:1.0.0
```

Traitement Batch et Temps Réel

- **Batch** : Hadoop MapReduce, Spark Batch
- **Temps Réel** : Spark Streaming, Apache Flink, Apache Storm

Installation des outils de traitement temps réel

- Flink :

```
wget https://dlcdn.apache.org/flink/flink-1.15.2-bin-sc
tar -xvzf flink-1.15.2-bin-scala_2.12.tgz
cd flink-1.15.2
bin/start-cluster.sh
```

- Spark Streaming :

```
spark-submit --packages org.apache.spark:spark-streamir
```

Utilisation de Python dans le Big Data

Python est l'un des langages les plus utilisés pour le Big Data en raison de sa simplicité, de sa richesse en bibliothèques et de son intégration avec les principaux frameworks comme Hadoop, Spark et Kafka. Grâce à ses nombreuses API et à son écosystème puissant, il permet de manipuler efficacement des données massives en batch ou en temps réel.

Hadoop : HDFS, YARN, MapReduce

HDFS (Hadoop Distributed File System)

- Système de fichiers distribué.
- Stockage en mode bloc pour la tolérance aux pannes.

YARN (Yet Another Resource Negotiator)

- Gestion des ressources et des tâches.

MapReduce : Programmation Distribuée

```
from mrjob.job import MRJob

class WordCount(MRJob):
def mapper(self, _, line):
for word in line.split():
yield word, 1
```

Hadoop : HDFS, YARN, MapReduce

```
def reducer(self, key, values):
yield key, sum(values)

if __name__ == "__main__":
WordCount.run()
```

Spark : RDDs, DataFrames, Spark SQL

RDDs (Resilient Distributed Datasets)

```
from pyspark.sql import SparkSession
spark =
 ↪  SparkSession.builder.appName("example").getOrCreate()
rdd = spark.sparkContext.parallelize([1,2,3,4,5])
squared = rdd.map(lambda x: x**2).collect()
print(squared)
```

Spark SQL : Interrogation des données

```
df = spark.createDataFrame([(1, "Alice"), (2,
 ↪  "Bob")], ["id", "name"])
df.createOrReplaceTempView("people")
spark.sql("SELECT * FROM people").show()
```

Kafka : Gestion des flux de données en temps réel

```
from kafka import KafkaProducer, KafkaConsumer

producer =
 ↪  KafkaProducer(bootstrap_servers='localhost:9092')
producer.send('my_topic', b'Hello Kafka!')

consumer = KafkaConsumer('my_topic',
 ↪  bootstrap_servers='localhost:9092')
for msg in consumer:
print(msg.value)
```

Snowflake, Delta Lake

Snowflake : Stockage et Analyse SQL

```
import snowflake.connector
conn = snowflake.connector.connect(user='user',
↪  password='pass', account='account')
cursor = conn.cursor()
cursor.execute("SELECT * FROM my_table")
print(cursor.fetchall())
```

Delta Lake : Stockage structuré

```
from delta.tables import DeltaTable
deltaTable = DeltaTable.forPath(spark, "/mnt/delta_table")
deltaTable.toDF().show()
```

Flink et Spark Streaming

Apache Flink

```
from pyflink.datastream import
 ↪  StreamExecutionEnvironment
env =
 ↪  StreamExecutionEnvironment.get_execution_environment()
ds = env.from_collection([("Alice", 23), ("Bob",
 ↪  34)])
ds.print()
env.execute("flink_job")
```

Flink et Spark Streaming

Spark Streaming

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import explode, split

spark =
 ↪  SparkSession.builder.appName("streaming").getOrCreate()
lines =
 ↪  spark.readStream.format("socket").option("host",
 ↪  "localhost").option("port", 9999).load()
words = lines.select(explode(split(lines.value, "
 ↪  ")).alias("word"))
query =
 ↪  words.writeStream.outputMode("append").format("console"
query.awaitTermination()
```

Conclusion

- Le Big Data repose sur un écosystème d'outils variés et puissants.
- Python permet d'interagir facilement avec ces technologies.
- Hadoop, Spark, Kafka, Snowflake et Flink sont des piliers du traitement massif des données.

Collecte, Nettoyage et Prétraitement des Données

- **Contexte :** Face à l'explosion des volumes de données, l'intégration et la préparation des données deviennent indispensables pour des analyses fiables et performantes.
- **Objectifs :**
  - Identifier et exploiter divers types de sources de données.
  - Comparer les approches ETL et ELT adaptées aux environnements Big Data.
  - Appliquer des techniques de data wrangling et de data cleaning en utilisant Python (Pandas et PySpark).
  - Assurer la qualité et la véracité des données pour des analyses robustes.
  - Automatiser et orchestrer les pipelines de données avec Apache Airflow.

Collecte, Nettoyage et Prétraitement des Données

- **Références recommandées :**
  - *Python for Data Analysis* (W. McKinney)
  - *Learning Spark* (Chambers & Zaharia)
  - *Data Analysis with Python and PySpark* (J. Rioux)

Sources de Données

Principaux types de sources

- **Web** : Récupération via API REST, web scraping, flux RSS.
- **IoT** : Données issues de capteurs, dispositifs connectés et logs temps réel.
- **Logs** : Fichiers journaux générés par applications et serveurs.
- **Bases de données** : Systèmes relationnels (SQL) et NoSQL (MongoDB, Cassandra, etc.).

Sources de Données

Pratiques recommandées

- Utiliser les bibliothèques Python comme `requests` ou `BeautifulSoup` pour extraire les données web.
- S'appuyer sur *Python for Data Analysis* pour normaliser et transformer des formats hétérogènes.
- Adapter les protocoles d'extraction en fonction de la vélocité et du volume (cf. *Learning Spark* pour le traitement de gros volumes).

ETL vs ELT

ETL (Extraction, Transformation, Chargement)

- Transformation préalable aux chargements.
- Approche classique pour entrepôts de données.
- Permet des traitements complexes hors de la cible.

ELT (Extraction, Chargement, Transformation)

- Chargement des données brutes dans un Data Lake ou entrepôt moderne.
- Transformation exécutée « in situ » en tirant profit de la puissance de calcul du système cible.
- Très adapté aux environnements Big Data et au cloud.

Data Wrangling et Data Cleaning

Définitions et enjeux

- **Data Wrangling** : Processus de conversion, reformatage et structuration de données brutes pour les rendre exploitables.
- **Data Cleaning** : Identification et correction des erreurs, valeurs manquantes et incohérences.

Pratiques issues des meilleurs ouvrages

- **Pandas** (cf. *Python for Data Analysis*) :
  - Utilisation des fonctions `fillna`, `drop_duplicates` et de la conversion de types pour normaliser les données.
  - Optimisation de la gestion mémoire pour des jeux de données de taille modérée.

Data Wrangling et Data Cleaning

Pratiques issues des meilleurs ouvrages

- **Pandas** (cf. *Python for Data Analysis*) :
  - Utilisation des fonctions `fillna`, `drop_duplicates` et de la conversion de types pour normaliser les données.
  - Optimisation de la gestion mémoire pour des jeux de données de taille modérée.
- **PySpark** (cf. *Learning Spark* et *Data Analysis with Python and PySpark*) :
  - Traitement distribué avec les DataFrames et RDDs pour manipuler de très gros volumes.
  - Transformation de données en appliquant des fonctions SQL-like et l'utilisation d'User-Defined Functions (UDFs).

Exemple de Data Cleaning avec Pandas

```python
import pandas as pd

# Chargement d'un DataFrame depuis un fichier CSV
df = pd.read_csv('donnees.csv')
# Gestion des valeurs manquantes : propagation en
↪  avant
df.fillna(method='ffill', inplace=True)
# Suppression des doublons
df.drop_duplicates(inplace=True)
# Conversion de la colonne 'date' en type datetime
df['date'] = pd.to_datetime(df['date'])
# Normalisation des chaînes de caractères
df['nom'] = df['nom'].str.strip().str.lower()
```

Exemple de Data Cleaning avec PySpark

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, to_date, trim,
↪  lower
spark =
↪  SparkSession.builder.appName("DataCleaning").getOrCreat
# Chargement d'un DataFrame Spark depuis un fichier
↪  CSV
df = spark.read.option("header",
↪  "true").csv("donnees.csv")
# Remplissage des valeurs nulles avec une valeur par
↪  défaut
df = df.na.fill({"colonne": "inconnu"})
# Conversion d'une colonne en format date
df = df.withColumn("date", to_date(col("date"),
```

Qualité et Véracité des Données

Aspects critiques à surveiller

- **Exactitude** : Les données doivent refléter fidèlement la réalité.
- **Complétude** : Éviter les données manquantes qui peuvent biaiser les analyses.
- **Cohérence** : Assurer l'uniformité entre diverses sources et formats.
- **Actualité** : Garantir que les données soient à jour et pertinentes.

Qualité et Véracité des Données

Méthodes et outils recommandés

- **Profilage des données** : Utiliser des bibliothèques (ex. `pandas-profiling`) pour générer des rapports automatisés.
- **Règles de validation** : Implémenter des tests unitaires et des contrôles d'intégrité dans vos pipelines.
- **Surveillance continue** : Déployer des dashboards et alertes pour détecter rapidement les anomalies.

Ces approches sont détaillées dans plusieurs ouvrages de référence sur la Data Engineering.

Automatisation avec Apache Airflow

Introduction et concepts clés

- Apache Airflow est une plateforme open source d'orchestration de workflows.
- Les pipelines sont définis comme des DAGs (Directed Acyclic Graphs), ce qui facilite la gestion des dépendances.
- Il offre une interface web interactive pour la planification et le suivi des tâches.

Automatisation avec Apache Airflow

Pourquoi Airflow ?

- **Scalabilité** : Capable de gérer des milliers de tâches réparties sur un cluster.
- **Flexibilité** : S'intègre facilement avec divers outils Big Data (Spark, Hadoop, etc.).
- **Extensibilité** : Possibilité d'écrire des opérateurs personnalisés en Python.

Pour une mise en œuvre concrète, consultez la documentation officielle d'Airflow ainsi que les cas d'usage présentés dans Learning Spark.

Installation d'Apache Hadoop dans Docker

Utiliser une image Hadoop préconfigurée

```
docker pull
 ↪  bde2020/hadoop-namenode:2.0.0-hadoop3.2.1-java8
docker pull
 ↪  bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8
docker pull
 ↪  bde2020/hadoop-resourcemanager:2.0.0-hadoop3.2.1-ja
docker pull
 ↪  bde2020/hadoop-nodemanager:2.0.0-hadoop3.2.1-java8
docker pull
 ↪  bde2020/hadoop-historyserver:2.0.0-hadoop3.2.1-java
```

Installation d'Apache Hadoop dans Docker

Utiliser une image Hadoop préconfigurée

```
PS C:\WINDOWS\system32> docker pull bde2020/hadoop-namenode:2.0.0-hadoop3.2.1-java8
2.0.0-hadoop3.2.1-java8: Pulling from bde2020/hadoop-namenode
73b8c0ccb707: Pulling fs layer
77920a3e82af: Pulling fs layer
f373218fec59: Pulling fs layer
c3a84a3e49c8: Pulling fs layer
8b1800105b98: Pulling fs layer
facffb3a6de3: Pulling fs layer
a65640a64a76: Pulling fs layer
3192219afd04: Pulling fs layer
92329e81aec4: Pulling fs layer
c71a6df73788: Pulling fs layer
7127a1d8cced: Pulling fs layer
883a89599900: Pulling fs layer
aa53513fe997: Pulling fs layer
73b8c0ccb707: Download complete
a65640a64a76: Download complete
c3a84a3e49c8: Download complete
8b1800105b98: Download complete
f373218fec59: Download complete
c71a6df73788: Download complete
facffb3a6de3: Download complete
aa53513fe997: Download complete
883a89599900: Download complete
77920a3e82af: Download complete
3192219afd04: Download complete
3192219afd04: Pull complete
```

Installation d'Apache Hadoop dans Docker

Utiliser une image Hadoop préconfigurée

```
c3a84a3e49c8: Pull complete
aa53513fe997: Pull complete
8b1800105b98: Pull complete
a65640a64a76: Pull complete
facffb3a6de3: Pull complete
73b8c0ccb707: Pull complete
c71a6df73788: Pull complete
Digest: sha256:51ad9293ec52083c5003ef0aaab00c3dd7d6335ddf495cc1257f97a272cab4c0
Status: Downloaded newer image for bde2020/hadoop-namenode:2.0.0-hadoop3.2.1-java8
docker.io/bde2020/hadoop-namenode:2.0.0-hadoop3.2.1-java8

PS C:\WINDOWS\system32> docker pull bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8
2.0.0-hadoop3.2.1-java8: Pulling from bde2020/hadoop-datanode
13c9c87a46cb: Pulling fs layer
3ca2ec07878c: Pulling fs layer
26c2dd45430e: Pulling fs layer
26c2dd45430e: Download complete
13c9c87a46cb: Download complete
3ca2ec07878c: Download complete
26c2dd45430e: Pull complete
13c9c87a46cb: Pull complete
3ca2ec07878c: Pull complete
Digest: sha256:ddf6e9ad55af4f73d2ccb6da31d9e3331ffb94d5f046126db4f40aa348d484bf
Status: Downloaded newer image for bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8
docker.io/bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8

PS C:\WINDOWS\system32>
```

Figure

Installation d'Apache Hadoop dans Docker

Création d'un fichier docker-compose.yml

```
PS C:\WINDOWS\system32> mkdir hadoop-docker


    Répertoire : C:\WINDOWS\system32


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        01/04/2025     11:44                hadoop-docker


PS C:\WINDOWS\system32> cd .\hadoop-docker
PS C:\WINDOWS\system32\hadoop-docker> New-Item -Path . -Name "docker-compose.yml" -ItemType "file"


    Répertoire : C:\WINDOWS\system32\hadoop-docker


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        01/04/2025     11:45              0 docker-compose.yml


PS C:\WINDOWS\system32\hadoop-docker>
```

Installation d'Apache Hadoop dans Docker

Utiliser une image Hadoop préconfigurée

```
PS C:\WINDOWS\system32\hadoop-docker> notepad .\docker-compose.yml

PS C:\WINDOWS\system32\hadoop-docker>
```

Figure –

Installation d'Apache Hadoop dans Docker

Lancer les services Hadoop

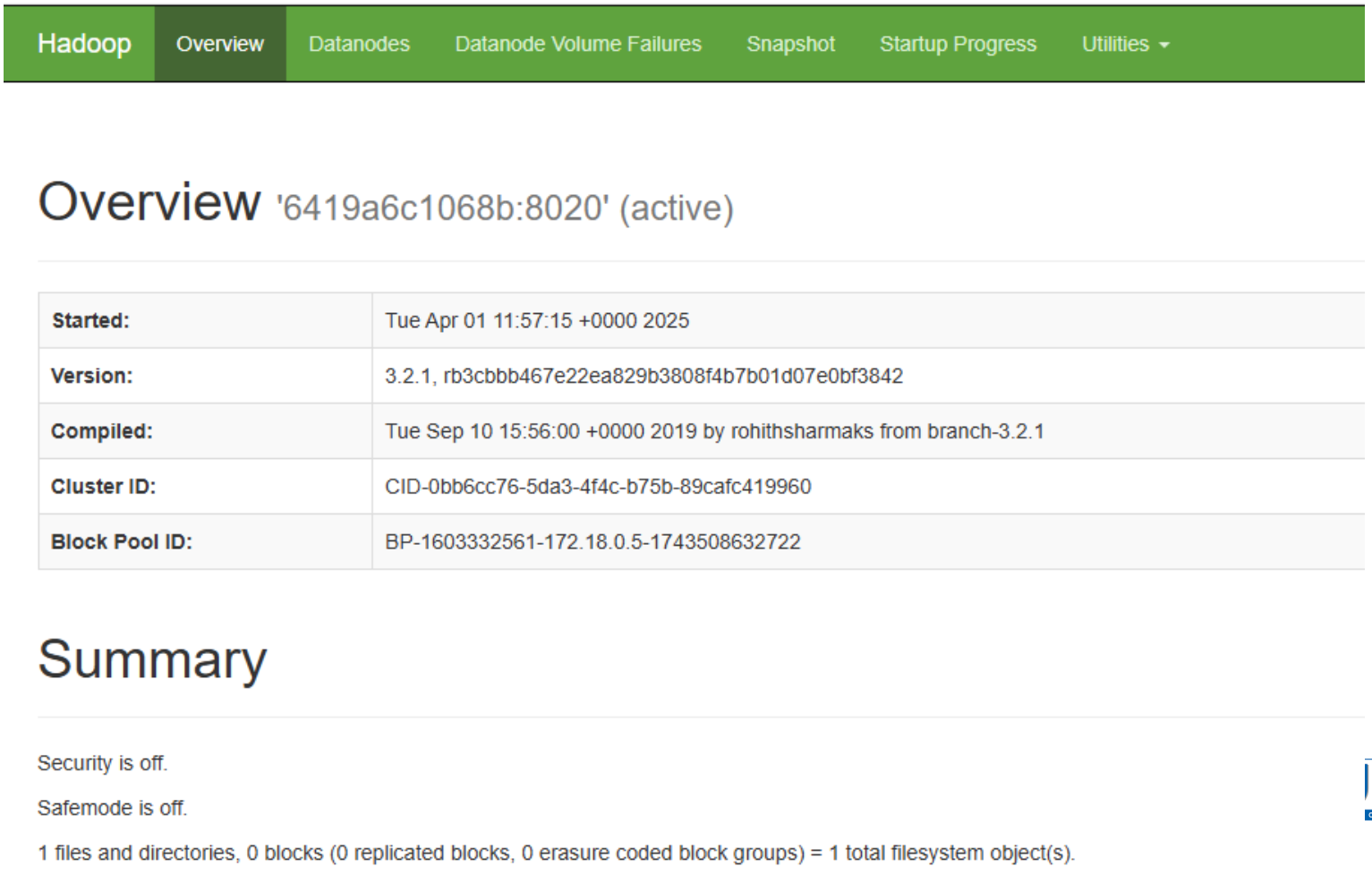
```
PS C:\WINDOWS\system32\hadoop-docker> docker-compose up -d

docker-compose : time="2025-04-01T11:57:07Z" level=warning msg="C:\\WI
attribute `version` is obsolete, it will be ignored, please remove it
Au caractère Ligne:1 : 1
+ docker-compose up -d
+ ~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : NotSpecified: (time="2025-04-0...tial co
    + FullyQualifiedErrorId : NativeCommandError

 Network hadoop-docker_default  Creating
 Network hadoop-docker_default  Created
 Volume "hadoop-docker_datanode_data"  Creating
 Volume "hadoop-docker_datanode_data"  Created
 Volume "hadoop-docker_namenode_data"  Creating
 Volume "hadoop-docker_namenode_data"  Created
 Container nodemanager  Creating
 Container resourcemanager  Creating
 Container namenode  Creating
 Container historyserver  Creating
 Container datanode  Creating
 Container nodemanager  Created
 Container resourcemanager  Created
 Container historyserver  Created
 Container datanode  Created
```

Installation d'Apache Hadoop dans Docker

Accéder à l'interface Web



Installation d'Apache Hadoop dans Docker

Installation de YEARN

```
PS C:\WINDOWS\system32> mkdir hadoop-yarn


    Répertoire : C:\WINDOWS\system32


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        01/04/2025     12:39                hadoop-yarn


PS C:\WINDOWS\system32> cd .\hadoop-yarn
PS C:\WINDOWS\system32\hadoop-yarn> New-Item -Path . -Name "docker-compose.yml" -ItemType "file"


    Répertoire : C:\WINDOWS\system32\hadoop-yarn


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        01/04/2025     12:42              0 docker-compose.yml


PS C:\WINDOWS\system32\hadoop-yarn> notepad .\docker-compose.yml$
PS C:\WINDOWS\system32\hadoop-yarn> notepad .\docker-compose.yml
PS C:\WINDOWS\system32\hadoop-yarn>
```