Mise en place d'une Architecture Distribuée pour l'Analyse de Logs Web

Objectifs :

- Analyser un fichier de logs web pour en extraire des informations pertinentes
- Mettre en place une architecture distribuée Big Data pour le traitement des données

Description du Dataset (web_server.log) :

Le fichier « **web_server.log** » contient les logs enregistrés par un serveur web pour un site e-commerce spécialisé dans la vente de produits cosmétiques. Chaque ligne du fichier correspond à une requête HTTP effectuée par un utilisateur, capturant des informations essentielles telles que l'adresse IP, la méthode HTTP utilisée, l'URL demandée, le code de réponse du serveur, et la taille des données envoyées en réponse.

Les logs suivent le format standard des serveurs web, avec les champs suivants :

- **Adresse IP** : Identifie le client ou utilisateur effectuant la requête.
- **Timestamp** : Indique la date et l'heure précises de la requête.
- **Méthode HTTP** : Spécifie le type d'action demandée (ex. : GET, POST).

- **URL** : Représente la ressource demandée, incluant les pages produits et autres fonctionnalités du site. Les URLs des produits incluent un **ID unique** pour chaque produit. Les catégories couvertes sont :
  - Maquillage : /products/lipstick, /products/foundation, /products/mascara.
  - Soins de la peau : /products/skincare/cream, /products/skincare/sunscreen.
  - Soins capillaires : /products/hair/shampoo, /products/hair/conditioner.
  - Exemple d'URL produit : /products/eyeliner?id=4562.

  Les logs incluent également des URLs pour des fonctionnalités générales comme:

  - `/cart` : Page du panier.
  - `/checkout` : Validation de commande.
  - `/user/login` : Connexion utilisateur.

- **Code de réponse HTTP** : Indique le statut de la requête :

  - `200` : Succès de la requête.
  - `404` : Ressource non trouvée.
  - `301` : Redirection.

- `500` : Erreur interne du serveur.
- `403` : Accès refusé.

- **Taille de la réponse** : Spécifie la quantité de données renvoyées au client en octets.

```
192.168.4.78 - - [28/Jan/2025:15:40:01 +0000] "GET /products/skincare/sunscreen?id=8 HTTP/1.1" 200 1450
192.168.1.100 - - [28/Jan/2025:15:40:02 +0000] "GET /products/lipstick?id=105 HTTP/1.1" 301 512
192.168.3.56 - - [28/Jan/2025:15:40:02 +0000] "POST /cart HTTP/1.1" 403 843
192.168.7.89 - - [28/Jan/2025:15:40:03 +0000] "GET /products/hair/shampoo?id=4821 HTTP/1.1" 200 7543
192.168.2.11 - - [28/Jan/2025:15:40:04 +0000] "GET /products/mascara?id=1 HTTP/1.1" 404 1200
```

Exemple de Données

1. **Travail demandé :**

Vous devez choisir **au moins 2 analyses de données**, en combinant traitements batch et stream. La liste des traitements ci-dessus n'est **pas exhaustive**. Ce sont des **exemples représentatifs** de ce que vous pouvez réaliser à partir des logs. Cependant, vous êtes également **libres de proposer et d'implémenter d'autres types de traitements** qui vous paraissent pertinents pour analyser les données ou répondre à des besoins spécifiques.

Exemples des analyses en Batch (traitements sur données statiques) :

- **Produits les plus consultés** :
  - Identifier les produits (par leur ID) ayant reçu le plus de requêtes sur une période donnée.
- **Répartition des codes HTTP** :
  - Analyser la fréquence des codes HTTP (200, 404, 500, etc.) sur une journée pour évaluer les performances du site.
- **Adresses IP les plus actives** :
  - Identifier les 10 adresses IP qui ont généré le plus de requêtes.
- **Temps de réponse moyen par catégorie de produit** :
  - Calculer la taille moyenne des réponses pour chaque catégorie de produit (ex. : maquillage, soins de la peau).

Exemples des analyses en Stream (traitements sur données en temps réel) :

- **Détection des erreurs en temps réel** :

    - Surveiller les logs pour détecter des pics d'erreurs (codes 404 ou 500) sur un intervalle de temps (e.g 5 minutes).
- **Produits en tendance** :
    - Identifier les produits les plus consultés en temps réel (IDs populaires dans un intervalle de temps donné, e.g consulté plus de 20 fois en une minute).
- **Surveillance de l'activité utilisateur (par adresse IP)**
    - Identifier les adresses IP effectuant un volume anormal de requêtes (potentiellement des bots ou des attaques DDoS).

Mise en place de l'architecture Big Data

Dans le cadre du cours, vous avez étudié plusieurs frameworks et outils pour mettre en place une architecture distribuée permettant :

- **Le traitement parallèle** de gros volumes de données.
- **Le stockage distribué** pour des données non structurées ou semi-structurées.

Pour ce projet, vous allez mobiliser ces compétences afin de construire une architecture adaptée à vos traitements (batch et/ou stream) en utilisant **Docker**.

En fonction des traitements que vous choisirez (**batch** et **stream**), mettez en place la bonne architecture distribuée :

- **HDFS (Hadoop Distributed File System)** : Pour stocker vos logs de manière distribuée.
- **Apache Spark** : Pour effectuer les traitements (batch et/ou stream).

Utilisez **Docker** pour déployer ces composants :

- Créez un fichier **docker-compose.yml** pour gérer l'orchestration des différents conteneurs (Hadoop, Spark, kafka, MongoDB).
- Vérifiez la communication entre les différents services.

Une fois l'architecture configurée, implémentez les **jobs Spark** pour effectuer les traitements nécessaires :

- **Batch** : Lire les données statiques depuis HDFS.
- **Stream** : Consommer les données qui arrivent en flux et les traiter en temps réel.

Livrable :

- **Le code source des traitements Spark (batch et/ou stream)** : Implémenté pour répondre aux scénarios choisis.
- Un **fichier docker-compose.yml** pour le déploiement des composants distribués.