**PROJET : STREAMING AVEC KAFKA**

**Nom des membres :**

* **Florian FOUALEM**
* **Ronel NJOGANG**
* **Abiola David DJAKOPO**
* **Ines Armelle MATCHEU**
* **Junior KAMSAP**

***Création d’un Pipeline de collecte automatisé en temps réel des données météorologiques de la France à partir d’une API open data.***

**Étape 1 : Recherche et Sélection d'API**

+ Choix de l'API : on a choisi d'utiliser OpenWeatherMap, qui est une API bien établie pour les données météorologiques. Pour ce projet, nous utiliserons l'API gratuite "Current Weather Data" d'OpenWeatherMap, qui fournit des informations météorologiques en temps réel.

+ Accéder à l'API : on s'inscrie sur OpenWeatherMap et on obtient une clé API (API key).

+ On télécharge le fichier "*city.list.json.gz*", et on extrait son contenu pour obtenir la ***liste des villes françaises*** (le script python ***import\_data.py*** nous montre qu’on a 19965 villes francaise identifiées).

Le fichier contient les informations suivantes : ID de la ville, nom de la ville, latitude, longitude, etc. Vous utiliserez l'ID pour interroger l'API OpenWeatherMap.

**Étape 2 : Configuration de l'environnement de développement**

Docker : Installer Docker

Docker Compose : Installer Docker Compose

Installer Python et les bibliothèques associées

**Étape 3 : Développement du Producteur Python**

Nous allons développer un producteur Kafka en Python qui interrogera l'API OpenWeatherMap et enverra les données à un cluster Kafka.

On crèe un fichier qu'on nomme "***producer.py***" ; à l'intérieur duquel on écrit le script du **producer**. Pour tester ce producteur, on aura besoin d’un cluster Kafka en cours d’exécution localement

**Étape 4 : Conception de la base de données Cassandra/MongoDB**

On a décidé de réaliser notre projet avec Cassandra :

+ On crèe un schéma de base de données Cassandra :

+ Le schéma doit contenir les colonnes pertinentes pour stocker les données météorologiques, comme :

city\_id, city\_name, temperature, humidity, weather\_description, timestamp, etc.

+ On crèe un Dockerfile pour Cassandra : \*\*\****FROM cassandra:latest***\*\*\*\*\*\*

**Étape 5 : Développement de Consumer Kafka avec Cassandra en Python**

On crèe un fichier nommé "***consumer.py***" ; dans lequel on écrira le code pour un **consumer** Kafka qui insère des données dans Cassandra

**Étape 6 : Développement d’un Dashboard Temps Réel avec Streamlit**

On install Streamlit :

On crèe un fichier "***dashboard.py***".

**Étape 7 à 9 : Conteneurisation et Orchestration avec Docker et Docker Compose**

+ Écrire les Dockerfiles pour chaque service : ***dashboard, consumer, et producer***

+ On écrit le fichier **docker-compose.yml**

**Étape 10 à 11 : Test, Optimisation et Documentation**

Lancez l'ensemble de l'infrastructure avec Docker Compose : \*\*\*\*\*\*\*sudo **docker-compose up**\*\*\*\*\*\*\*\*\*\*\*

Vérifiez que toutes les parties du pipeline fonctionnent correctement : Producteur, Kafka, Cassandra, Consumer, et le tableau de bord Streamlit.

**On accède au tableau de bord à l'adresse** ***« http://localhost : 8501 »***