الجامعة الدولية للرباط
ⵜⴰⵙⴷⴰⵡⵉⵜ ⵜⴰⵅⴰⵡⴰⵢⵜ ⵏ ⵕⴱⴰⵟ
Université Internationale de Rabat
L'UNIVERSITÉ INNOVANTE

UIR ÉCOLE SUPÉRIEURE D'INFORMATIQUE ET DU NUMÉRIQUE

Météo-Stream : Suivi Intelligent en Temps Réel

Réalisé par:

Meryem Ben-aammi,

Sara Malki,

Aicha Alaoui Hanafi,

Mohammed Houssam Najah

Encadré par:

Mme. Nada Sbihi

M. Hamza Gamouh

2024-2025

Table des matières

Introduction Générale

Dans un monde où les conditions météorologiques influencent de nombreux aspects de la vie quotidienne — de la mobilité urbaine à l'agriculture, en passant par la sécurité publique — disposer d'un système capable de suivre en temps réel l'évolution du climat devient une nécessité stratégique. Le projet **"Suivi en temps réel de la météo"** vise à répondre à ce besoin en s'appuyant sur une architecture moderne orientée Big Data.

En exploitant l'API OpenWeatherMap, les données météo (température, humidité, pression, etc.) sont récupérées périodiquement pour plusieurs villes. Ces flux d'information sont ensuite traités en temps réel à l'aide d'Apache Kafka, filtrés et nettoyés via Kafka Streams, puis stockés dans Elasticsearch. La visualisation des données est assurée par Kibana, permettant de construire des Dashboard interactifs pour suivre l'évolution du climat ville par ville.

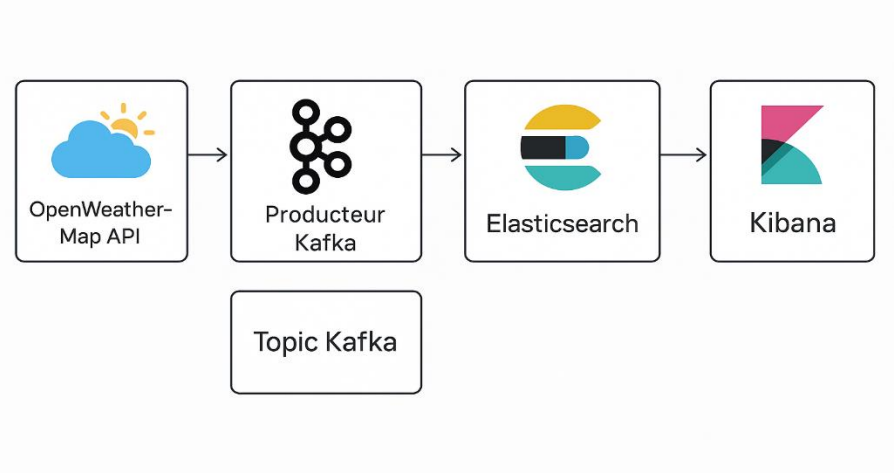
Ce projet démontre l'intérêt de l'intégration de technologies distribuées et scalables pour la collecte, le traitement et l'analyse en continu de données météorologiques, tout en rendant l'information accessible de manière claire, rapide et dynamique.

Objectif principal du projet :

Mettre en place une **infrastructure Big Data temps réel** permettant de collecter, traiter, stocker et visualiser les données météorologiques (température, humidité, pression, etc.) de plusieurs villes via **l'API OpenWeatherMap**, en utilisant les technologies suivantes :

- **Kafka** pour l'ingestion et le streaming des données.
- **Elasticsearch** pour le stockage et l'indexation des données.
- **Kibana** pour la visualisation et le dashboarding en temps réel.

I. Architecture détaillée :



1. Collecte des Données

- **Source :** L'API OpenWeatherMap (appels HTTP périodiques pour récupérer les données météo).
- **Méthode :**
  - Un **script Python** interroge l'API à intervalles réguliers.
  - Les données sont envoyées vers un **topic Kafka** (weather-data).

2. Streaming et Traitement (Kafka)

- **Kafka Producer** : Envoie les données météo vers un topic dédié.
- **Kafka Consumer** : Récupère les données pour traitement ultérieur (filtrage, agrégation, etc.).
- **Kafka Connect** : Utilisé pour exporter les données vers Elasticsearch automatiquement.

3. Stockage et Indexation (Elasticsearch)

Les données sont stockées dans **Elasticsearch** pour une recherche et une analyse efficaces.

4. Visualisation (Kibana)

Création de **Dashboard dynamiques** pour suivre :

- **La température en temps réel par ville.**
- **L'humidité et la pression atmosphérique.**
- **Des cartes géographiques avec des marqueurs météo.**

- **Alertes Kibana : Notifications si seuils critiques (ex : température trop élevée).**

II. Technologies utilisées :

Technologie	Rôle
OpenWeatherMap API	Source des données météo.
Python (Requests, Kafka-Python)	Script de collecte et envoi vers Kafka.
Apache Kafka	Middleware de streaming temps réel.
Elasticsearch	Base de données NoSQL pour stockage et recherche.
Kibana	Outil de visualisation et monitoring.

1. OpenWeatherMap API

OpenWeatherMap est une plateforme en ligne qui fournit des données météorologiques accessibles via une API REST. Elle permet de récupérer, en temps réel, des informations climatiques sur n'importe quelle ville du monde.

- **Utilisation** : Il permet d'obtenir des informations comme la température, l'humidité, la pression ou les conditions climatiques d'une ville à partir d'une simple requête HTTP.

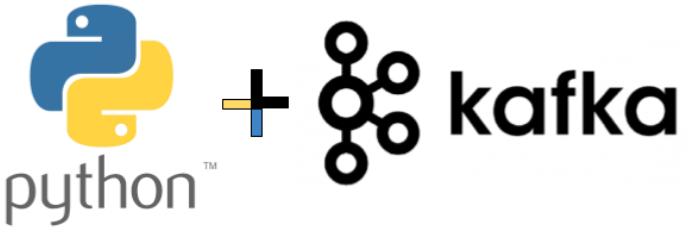
2. Python (Requests, Kafka-Python)

Python est un langage de programmation simple et puissant.

Requests est une bibliothèque permettant de faire des requêtes HTTP de manière intuitive.

Kafka-Python est une bibliothèque qui permet à un script Python de se connecter à Kafka pour envoyer ou consommer des messages.

- **Utilisation :** Python est utilisé ici pour écrire un producteur Kafka qui récupère les données météo via API, puis les publie sur un topic Kafka.



3. Apache Kafka

Apache Kafka est une plateforme de streaming distribuée conçue pour gérer des flux de données en temps réel.
Elle permet de publier, stocker, et traiter de grandes quantités de données en continu.

- **Utilisation** : Kafka agit comme un bus central qui transmet les données météo depuis le producteur vers les autres composants du système.

4. Elastic Search

Elasticsearch est un moteur de recherche et d'analyse distribué basé sur le modèle NoSQL. Il permet d'indexer, stocker et interroger rapidement de grandes quantités de données structurées ou semi-structurées.

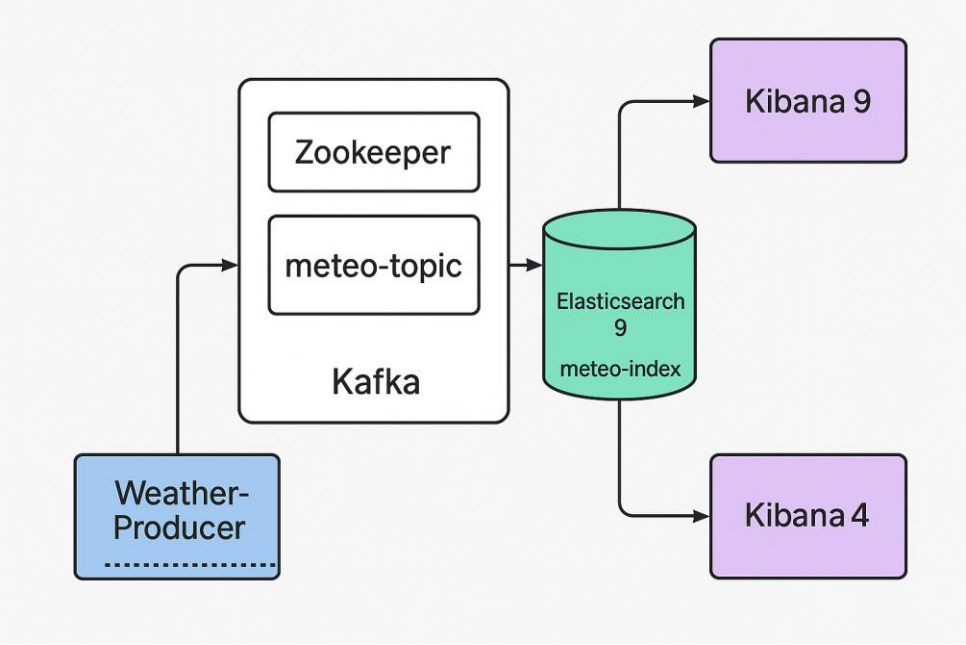
- **Utilisation** : Les données météo filtrées sont stockées dans Elasticsearch pour permettre des recherches et visualisations rapides.

5. Kibana

Kibana est un outil de visualisation de données open-source conçu pour fonctionner avec Elasticsearch.
Il permet de créer des Dashboards interactifs pour analyser les données indexées.

- **Utilisation** : Il sert ici à afficher les températures, humidités et autres mesures météo sous forme de graphiques, cartes et courbes.

III. Étapes de Configuration (Elasticsearch 9 + Kibana 9)



1. Démarrer Kafka

Avant tout, Kafka et Zookeeper doivent être en marche.

```
C:\Users\ab>cd C:\kafka\kafka_2.13-3.6.1

C:\kafka\kafka_2.13-3.6.1>bin\windows\zookeeper-server-start.bat config\zookeeper.properties
[2025-04-30 17:21:53,962] INFO Reading configuration from: config\zookeeper.properties (org.apache.zookeeper.server.quor
um.QuorumPeerConfig)
[2025-04-30 17:21:53,964] WARN config\zookeeper.properties is relative. Prepend .\ to indicate that you're sure! (org.ap
ache.zookeeper.server.quorum.QuorumPeerConfig)
[2025-04-30 17:21:54,000] WARN \tmp\zookeeper is relative. Prepend .\ to indicate that you're sure! (org.apache.zookeepe
r.server.quorum.QuorumPeerConfig)
[2025-04-30 17:21:54,027] INFO clientPortAddress is 0.0.0.0:2181 (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2025-04-30 17:21:54,028] INFO secureClientPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2025-04-30 17:21:54,029] INFO observerMasterPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2025-04-30 17:21:54,030] INFO metricsProvider.className is org.apache.zookeeper.metrics.impl.DefaultMetricsProvider (or
g.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2025-04-30 17:21:54,041] INFO autopurge.snapRetainCount set to 3 (org.apache.zookeeper.server.DatadirCleanupManager)
[2025-04-30 17:21:54,042] INFO autopurge.purgeInterval set to 0 (org.apache.zookeeper.server.DatadirCleanupManager)
[2025-04-30 17:21:54,042] INFO Purge task is not scheduled. (org.apache.zookeeper.server.DatadirCleanupManager)
[2025-04-30 17:21:54,043] WARN Either no config or no quorum defined in config, running in standalone mode (org.apache.z
ookeeper.server.quorum.QuorumPeerMain)
[2025-04-30 17:21:54,049] INFO Log4j 1.2 jmx support not found; jmx disabled. (org.apache.zookeeper.jmx.ManagedUtil)
[2025-04-30 17:21:54,051] INFO Reading configuration from: config\zookeeper.properties (org.apache.zookeeper.server.quor
um.QuorumPeerConfig)
[2025-04-30 17:21:54,052] WARN config\zookeeper.properties is relative. Prepend .\ to indicate that you're sure! (org.ap
ache.zookeeper.server.quorum.QuorumPeerConfig)
[2025-04-30 17:21:54,056] WARN \tmp\zookeeper is relative. Prepend .\ to indicate that you're sure! (org.apache.zookeepe
r.server.quorum.QuorumPeerConfig)
[2025-04-30 17:21:54,066] INFO clientPortAddress is 0.0.0.0:2181 (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
```

Dans un terminal, lancer :

```
C:\Users\ab>cd C:\kafka\kafka_2.13-3.6.1

C:\kafka\kafka_2.13-3.6.1>bin\windows\kafka-server-start.bat config\server.properties
[2025-04-30 17:26:44,543] INFO Registered kafka:type=kafka.Log4jController MBean (kafka.utils.Log4jControllerRegistratio
n$)
[2025-04-30 17:26:45,271] INFO Setting -D jdk.tls.rejectClientInitiatedRenegotiation=true to disable client-initiated TL
S renegotiation (org.apache.zookeeper.common.X509Util)
[2025-04-30 17:26:45,416] INFO starting (kafka.server.KafkaServer)
[2025-04-30 17:26:45,417] INFO Connecting to zookeeper on localhost:2181 (kafka.server.KafkaServer)
[2025-04-30 17:26:45,456] INFO [ZooKeeperClient Kafka server] Initializing a new session to localhost:2181. (kafka.zooke
eper.ZooKeeperClient)
[2025-04-30 17:26:45,467] INFO Client environment:zookeeper.version=3.8.3-6ad6d364c7c0bcf0de452d54ebefa3058098ab56, buil
t on 2023-10-05 10:34 UTC (org.apache.zookeeper.ZooKeeper)
[2025-04-30 17:26:45,467] INFO Client environment:host.name=DESKTOP-FR2HK4S (org.apache.zookeeper.ZooKeeper)
[2025-04-30 17:26:45,468] INFO Client environment:java.version=11.0.20 (org.apache.zookeeper.ZooKeeper)
[2025-04-30 17:26:45,468] INFO Client environment:java.vendor=Oracle Corporation (org.apache.zookeeper.ZooKeeper)
[2025-04-30 17:26:45,469] INFO Client environment:java.home=C:\Users\ab\Desktop\jdk-11 (org.apache.zookeeper.ZooKeeper)
[2025-04-30 17:26:45,469] INFO Client environment:java.class.path=C:\kafka\kafka_2.13-3.6.1\libs\activation-1.1.1.jar;C:
\kafka\kafka_2.13-3.6.1\libs\aopalliance-repackaged-2.6.1.jar;C:\kafka\kafka_2.13-3.6.1\libs\argparse4j-0.7.0.jar;C:\kaf
ka\kafka_2.13-3.6.1\libs\audience-annotations-0.12.0.jar;C:\kafka\kafka_2.13-3.6.1\libs\caffeine-2.9.3.jar;C:\kafka\kafk
a_2.13-3.6.1\libs\checker-qual-3.19.0.jar;C:\kafka\kafka_2.13-3.6.1\libs\commons-beanutils-1.9.4.jar;C:\kafka\kafka_2.13
-3.6.1\libs\commons-cli-1.4.jar;C:\kafka\kafka_2.13-3.6.1\libs\commons-collections-3.2.2.jar;C:\kafka\kafka_2.13-3.6.1\l
ibs\commons-digester-2.1.jar;C:\kafka\kafka_2.13-3.6.1\libs\commons-io-2.11.0.jar;C:\kafka\kafka_2.13-3.6.1\libs\commons
-lang3-3.8.1.jar;C:\kafka\kafka_2.13-3.6.1\libs\commons-logging-1.2.jar;C:\kafka\kafka_2.13-3.6.1\libs\commons-validator
```

Création d'un topic Kafka nommé **meteo-topic**.

```
C:\Users\ab>cd C:\kafka\kafka_2.13-3.6.1

C:\kafka\kafka_2.13-3.6.1>.\bin\windows\kafka-topics.bat --create --topic meteo-topic --bootstrap-server localhost:9092
--partitions 1 --replication-factor 1
Created topic meteo-topic.

C:\kafka\kafka_2.13-3.6.1>
```

2. Lancer Elasticsearch :

En utilisant la commande : **.\bin\elasticsearch.bat**

Et pour y accéder :
http://localhost:9200

Cela confirme qu'Elasticsearch fonctionne correctement.

3. Lancer Kibana après lancement d'Elasticsearch :

Lancer Kibana après avoir vérifié qu'Elasticsearch est bien actif :

En utilisant la commande : **PS C:\Users\ab\meteo-realtime-project\kibana\kibana-9.0.0> .\bin\kibana.bat**
Et pour accéder à KIBANA :

http://localhost:5601

4. Création du topic Kafka :

Ce topic servira de **canal de transfert des données météorologiques**.

```
C:\kafka\kafka_2.13-3.6.1>.\bin\windows\kafka-topics.bat --list --bootstrap-server localhost:9092
__consumer_offsets
meteo-topic
```

- **Installation des dépendances :**

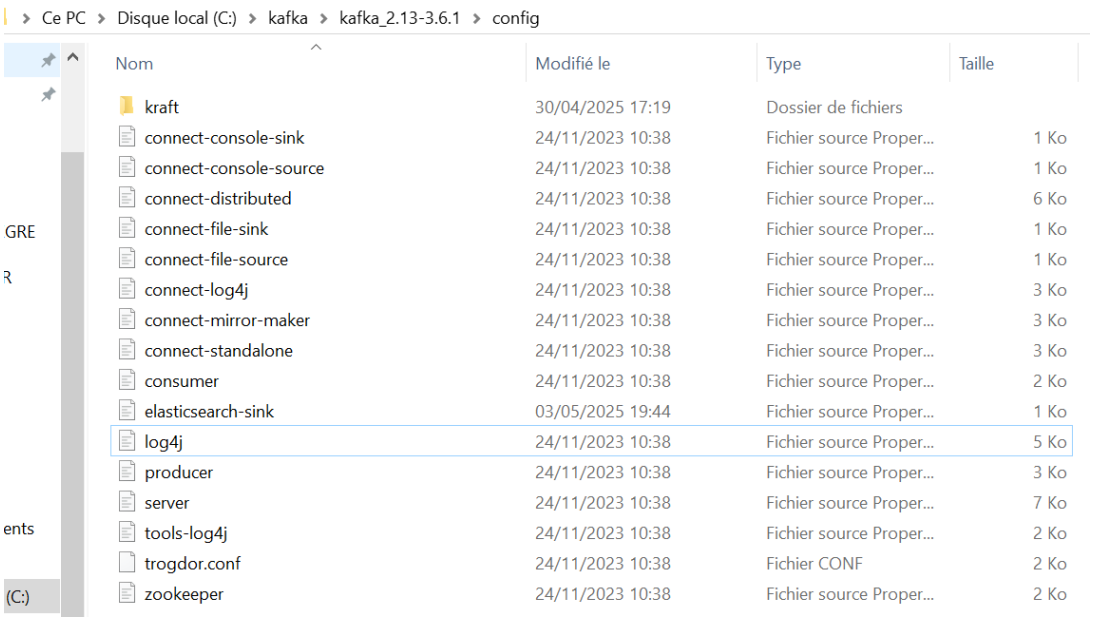
```
PS C:\kafka\kafka_2.13-3.6.1> pip install kafka-python requests
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: kafka-python in c:\users\ab\appdata\roaming\python\python313\site-packages (2.2.1)
Requirement already satisfied: requests in c:\users\ab\appdata\roaming\python\python313\site-packages (2.32.3)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\ab\appdata\roaming\python\python313\site-packages (from requests) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in c:\users\ab\appdata\roaming\python\python313\site-packages (from requests) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\ab\appdata\roaming\python\python313\site-packages (from requests) (2.4.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\ab\appdata\roaming\python\python313\site-packages (from requests) (2025.4.26)
PS C:\kafka\kafka_2.13-3.6.1>
```

5. Étapes pour connecter Kafka à Elasticsearch :

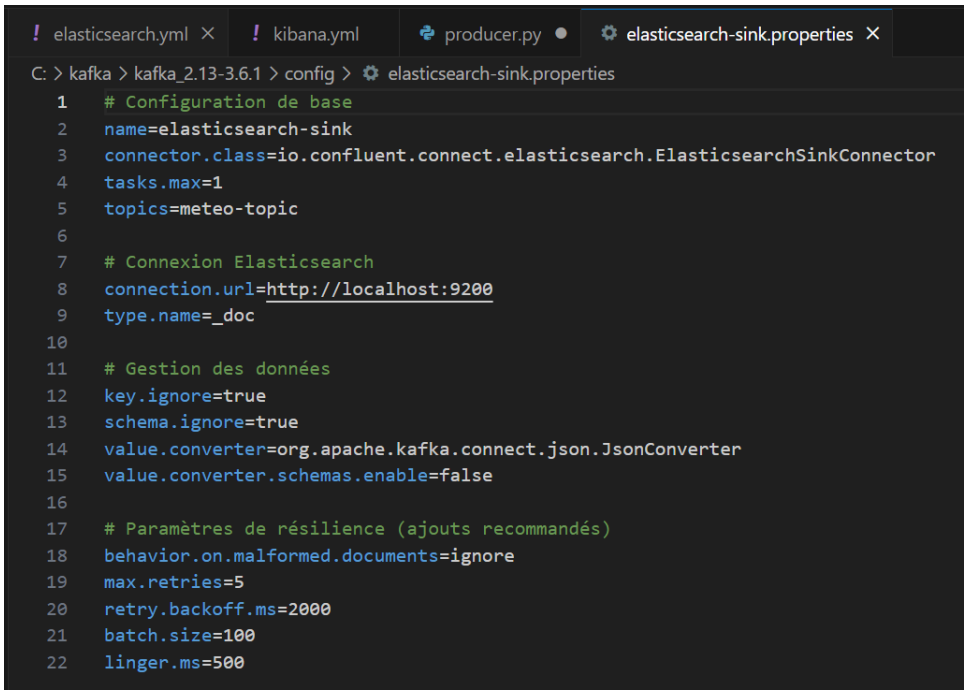
Installation et extraction du Connecteur Elasticsearch :

```
PS C:\Users\ab> Expand-Archive -Path "$env:USERPROFILE\Downloads\confluentinc-kafka-connect-elasticsearch-15.0.0.zip" -DestinationPath "C:\kafka\kafka_2.13-3
.6.1\libs\"
```

On avons créé Elasticsearch-sink.properties :




```
PS C:\kafka\kafka_2.13-3.6.1\config> code C:\kafka\kafka_2.13-3.6.1\config\elasticsearch-sink.properties
```

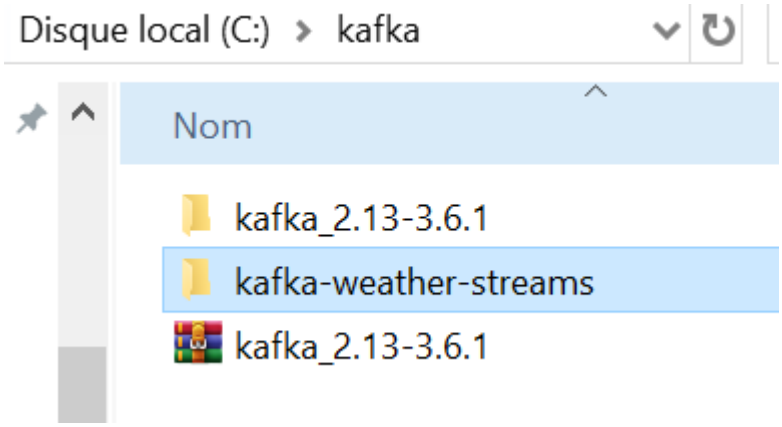


Pour exécuter le connecteur Kafka → Elasticsearch :

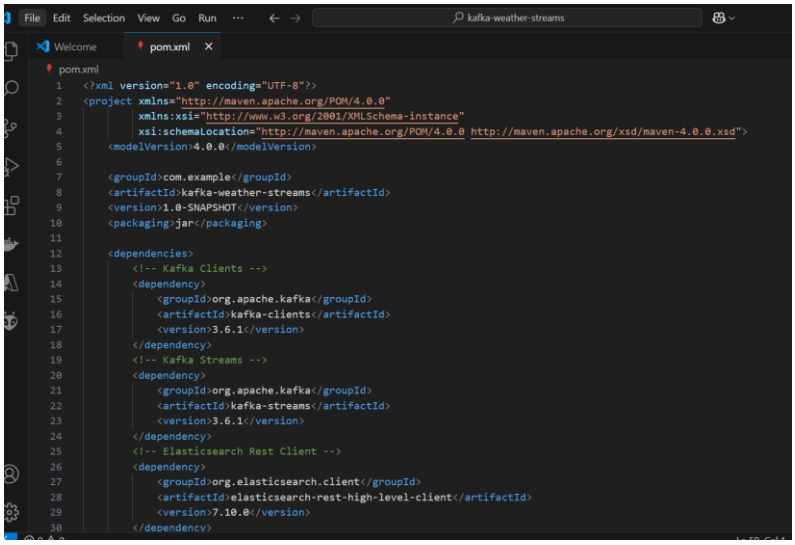
- **Installer GIT**
- **Une fois installé, fais un clic droit dans le dossier C:\kafka\kafka_2.13-3.6.1 et choisis "Git Bash Here".**
- **.\bin\windows\connect-standalone.bat .\config\connect-standalone.properties .\config\elasticsearch-sink.properties**

6. Création du projet Java avec Maven (Visual Studio Code)

Créer un dossier de projet :



Créer un fichier pom.xml dans ce dossier avec les dépendances nécessaires :



On utilise la commande suivante pour compiler le projet : **mvn clean install**

- **Ajouter d'un fichier WeatherProducer.java (Code du producteur) :** Ce fichier contient le **code Java qui génère et envoie les données météo** à Kafka.

7. Démarrer Zookeeper:

Commande utilisé:

PS C:\Users\ab\meteo-realtime-project\kafka\kafka_2.13-3.6.1> bin\windows\connect-standalone.bat config\connect-standalone.properties config\elasticsearch-sink.properties

8. Lancer le serveur Kafka

Dans un nouveau terminal :

PS C:\kafka\kafka_2.13-3.6.1> .\bin\windows\kafka-server-start.bat .\config\server.properties

Ensuite : nous avons exécuté le script Python du producteur de données météo :

- **C:\Users\ab>cd "C:\Users\ab\Desktop\4eme annee\S8\bigdata\PROJET"**
  C:\Users\ab\Desktop\4eme annee\S8\bigdata\PROJET>python producer.py

Puis le consommateur Kafka :

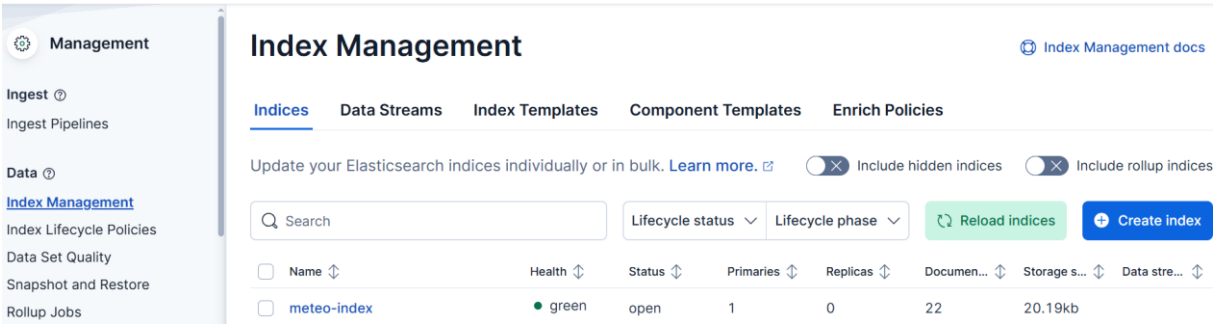
- **C:\Users\ab>cd C:\Users\ab\Desktop\4eme annee\S8\bigdata\PROJET**
  C:\Users\ab\Desktop\4eme annee\S8\bigdata\PROJET>python consumer.py

9. Créer index Elasticsearch météo-index :

Commande curl pour créer un index avec mapping personnalisé :

- C:\Users\ab\Desktop\4eme annee\S8\bigdata\PROJET>curl -X PUT "http://localhost:9200/meteo-index" -H "Content-Type: application/json" -d "{\"mappings\":{\"properties\":{\"ville\":{\"type\":\"keyword\"},\"temp\":{\"type\":\"float\"},\"humidity\":{\"type\":\"integer\"},\"description\":{\"type\":\"text\"},\"timestamp\":{\"type\":\"date\"}}}}"{"acknowledged":true,"shards_acknowledged":true,"index":"meteo-index"}

- C:\Users\ab\Desktop\4eme annee\S8\bigdata\PROJET>



Cette interface **Index Management** de **Kibana 9**, utilisée pour gérer les indices Elasticsearch.

Elle permet de :

- Vérifier que l'ingestion de données via Kafka → Elasticsearch est bien configurée,
- Monitorer l'état et le contenu des indices (taille, statut, documents),
- Créer ou supprimer des indices manuellement.

10. Start Kafka Connect:

- producer.py → pour envoyer des données
- consumer.py → pour les afficher

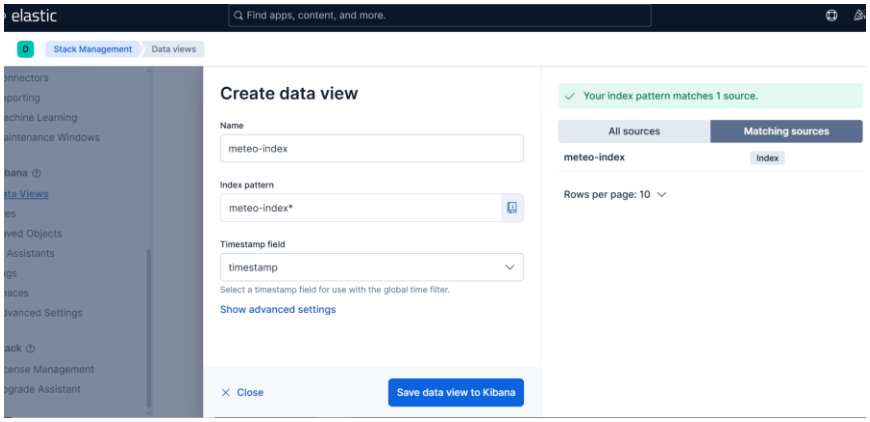
En utilisant les commandes suivantes :

- **C:\kafka\kafka_2.13-3.6.1>.\bin\windows\connect-standalone.bat .\config\connect-standalone.properties**
- Dans un terminal :
  C:\Users\ab\Desktop\4eme année\S8\bigdata\PROJET>python producer.py

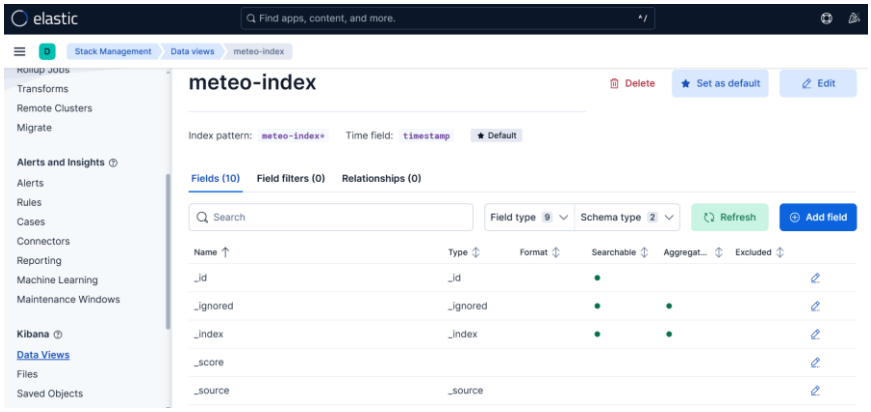
- Dans un autre terminal :
  C:\Users\ab\Desktop\4eme annee\S8\bigdata\PROJET>python consumer.py

11.Creating data view :

On clique sur Management (ou Stack Management)/ Kibana > Data Views



On crée une data view liée à l'index meteo-index.

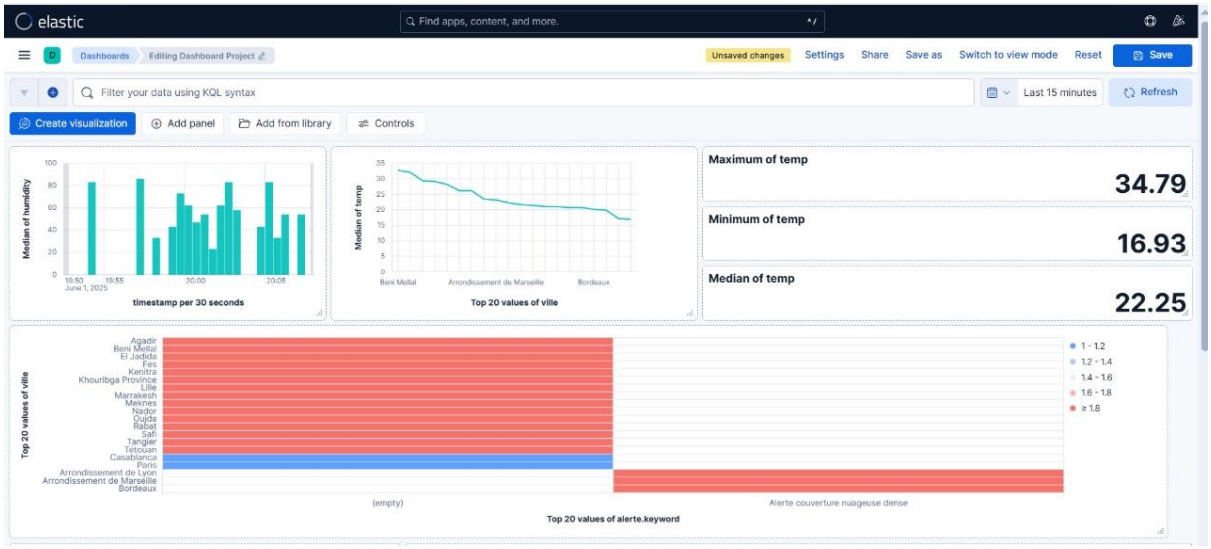


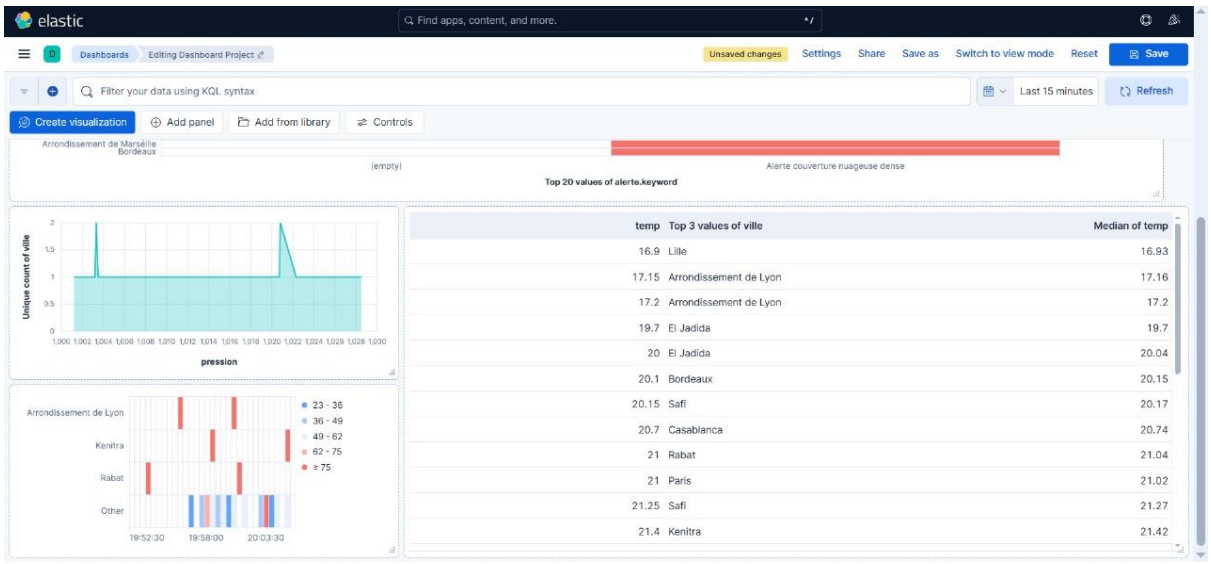
Cela permettra d'explorer visuellement les données météo dans **Discover** ou via des **Dashboard** personnalisés.

IV. Création du Dashboard :

Ce Dashboard fournit une **analyse en temps réel des données météorologiques** avec :

- Surveillance des **températures**, **pressions**, **alertes météorologiques**.
- Classement des villes par différents indicateurs.
- Utile pour des **décisions prédictives**, **alertes climatiques** ou **analyse environnementale**.





Visualisations principales :

1. **Histogramme de l'humidité moyenne par timestamp (30 sec)**
   - Affiche l'évolution de l'humidité sur une période glissante.
   - Utile pour surveiller les variations rapides.
2. **Courbe du Médian de température par ville**
   - Représente la température médiane pour plusieurs villes comme "Béni Mellal", "Marseille", "Bordeaux", etc.
   - On y observe une tendance décroissante.
3. **Barres horizontales – Alertes météorologiques par ville**
   - "Top 20 values of alerte.keyword" : ici, on voit la couverture d'alertes par ville.
   - Les couleurs indiquent les niveaux de sévérité (par ex. rouge = fort).
   - Beaucoup d'alertes de type Alerte couverture nuageuse dense.
4. **Cartes de valeurs numériques :**
   - **Maximum of temp** : 34.79°C
   - **Minimum of temp** : 16.93°C
   - **Median of temp** : 22.25°C

5. **Barre horizontale des alertes météo**

    - Même type d'alerte qu'avant (Alerte couverture nuageuse dense) centrée sur Marseille et Bordeaux.

6. **Graphique des villes uniques par pression**

    - Affiche les occurrences de valeurs uniques de ville en fonction de la pression atmosphérique.
    - Pic régulier suggérant des mesures cycliques.

7. **Heatmap des valeurs de pression par ville dans le temps**

- Donne un aperçu de la répartition des pressions-zones de chaleur (par ex Lyon, Kenitra, Rabat…)

8. **Tableau – Température des villes**

- Liste les **valeurs de température** par ville.
- Montre aussi la **médiane** par ville.

- Exemple :
    - **Lille** : Temp = 16.93 (le plus bas)
    - **Kenitra** : Temp = 21.4
    - **Casablanca** : Temp = 20.7

V. Conclusion :

Ce projet illustre parfaitement la puissance des technologies Big Data dans la collecte, le traitement et la visualisation de données dynamiques. En combinant OpenWeatherMap, Kafka, Elasticsearch et Kibana

Cette solution peut être étendue à d'autres cas d'usage (surveillance IoT, analyse de logs, détection d'anomalies, etc.), démontrant ainsi la flexibilité des outils choisis.

Enfin, ce projet renforce les compétences en intégration de données, streaming et analytique temps réel, ouvrant la voie à des applications plus avancées comme le Machine Learning prédictif ou les systèmes d'alertes automatisés.