# Étape 1 : Sélection d'une source de données ouverte

OpenWeatherMap

# Étape 2 : Mise en place du pipeline d'ingestion de données

mettre en place un pipeline d'ingestion de données localement en utilisant Python et Apache Airflow pour l'orchestration.

Configurer Apache Airflow localement :

Installez Airflow sur votre machine locale (via Docker, ou pip de Python).

Écrire un DAG (Directed Acyclic Graph) pour automatiser l'extraction de données à partir des API ou des sources sélectionnées.

|  |
| --- |
| from airflow import DAG  from airflow.operators.python\_operator import PythonOperator  from datetime import datetime, timedelta  import requests  def fetch\_data():  response = requests.get("<API URL>")  with open('/path/to/data.csv', 'w') as file:  file.write(response.text)  default\_args = {  'owner': 'student',  'depends\_on\_past': False,  'start\_date': datetime(2023, 1, 1),  'retries': 1,  'retry\_delay': timedelta(minutes=5),  }  dag = DAG('data\_ingestion', default\_args=default\_args, schedule\_interval='@hourly')  fetch\_data\_task = PythonOperator(  task\_id='fetch\_data',  python\_callable=fetch\_data,  dag=dag,  ) |

Ingérer les données :

Programmer le DAG pour qu'il s'exécute périodiquement et récupère les données (par exemple, toutes les heures ou tous les jours).

Stockez les données ingérées localement au format CSV ou SQLite.

Résultat : Un pipeline d'ingestion de données géré localement à l'aide d'Airflow, qui collecte les données de la source ouverte en temps réel.

# Étape 3 : Traitement des données en temps réel avec Synapse Analytics (utiliser Python)

Maintenant que vous disposez des données, l'étape suivante consiste à les traiter localement à l'aide de Python et de la bibliothèque Pandas.

Installez Pandas :

Installez Pandas via pip (pip install pandas).

Écrire le code de transformation :

Chargez les données ingérées (à partir d'un fichier CSV ou d'une base de données) et effectuez des transformations à l'aide de Pandas.

Exemple de script de transformation :

|  |
| --- |
| import pandas as pd  # Load data from local CSV  df = pd.read\_csv('/path/to/data.csv')  # Perform data cleaning or filtering  df\_cleaned = df[df['value'] > 100] # Example filtering based on some column  # Save the transformed data  df\_cleaned.to\_csv('/path/to/cleaned\_data.csv', index=False) |

Résultat : Pipeline de traitement et de transformation des données à l'aide de Python Pandas, prêt pour la visualisation.

# Étape 4 : Visualisation des données avec Power BI

Utilisez Power BI Desktop pour créer des rapports et des tableaux de bord interactifs qui visualisent les indicateurs clés à partir des données.

Il peut s'agir, par exemple, de statistiques sur des événements en direct, de l'évolution du cours des actions ou de la visualisation de modèles météorologiques.

Tableau de bord en temps réel :

Activez l'actualisation en temps réel dans Power BI pour que vos tableaux de bord soient mis à jour automatiquement au fur et à mesure que de nouvelles données sont traitées.

Sortie : Un tableau de bord Power BI interactif et en direct qui présente des informations en temps réel basées sur les données ingérées et traitées.

## SINON

Sinon, vous pouvez visualiser les données localement avec Tableau public ou jupyter notebooks.

Au lieu de Power BI, nous utiliserons Tableau Public ou Jupyter Notebooks pour la visualisation.

Carnets Jupyter :

Installez Jupyter à l'aide de pip (pip install jupyter).

Utilisez matplotlib ou plotly pour la visualisation des données dans les carnets.

Exemples :

|  |
| --- |
| import pandas as pd  import matplotlib.pyplot as plt  # Load transformed data  df\_cleaned = pd.read\_csv('/path/to/cleaned\_data.csv')  # Plot the data  df\_cleaned.plot(x='date', y='value', kind='line')  plt.show() |

Tableau Public :

Exportez les données nettoyées de Pandas au format CSV et importez-les dans Tableau Public.

Créez des tableaux de bord interactifs à l'aide des données fournies et publiez-les en ligne.

Résultat : Visualisations en temps réel à l'aide de Jupyter Notebook ou de Tableau Public avec la possibilité de rafraîchir les données fréquemment.

# Bonus : Infrastructure as Code (IaC) avec Terraform

Que si on fait avec Microsoft azure.