BIG DATA ADVANCED

RESPONSABLE DE COURS

IGOR MARTY

ÉQUIPE

M'POUTOU BABINGUI Dixy

Manager

Anis SI BACHIR

Marketing Analyst

Ornella AKAKPO

Business Consultant

Basak Durgun Donmez

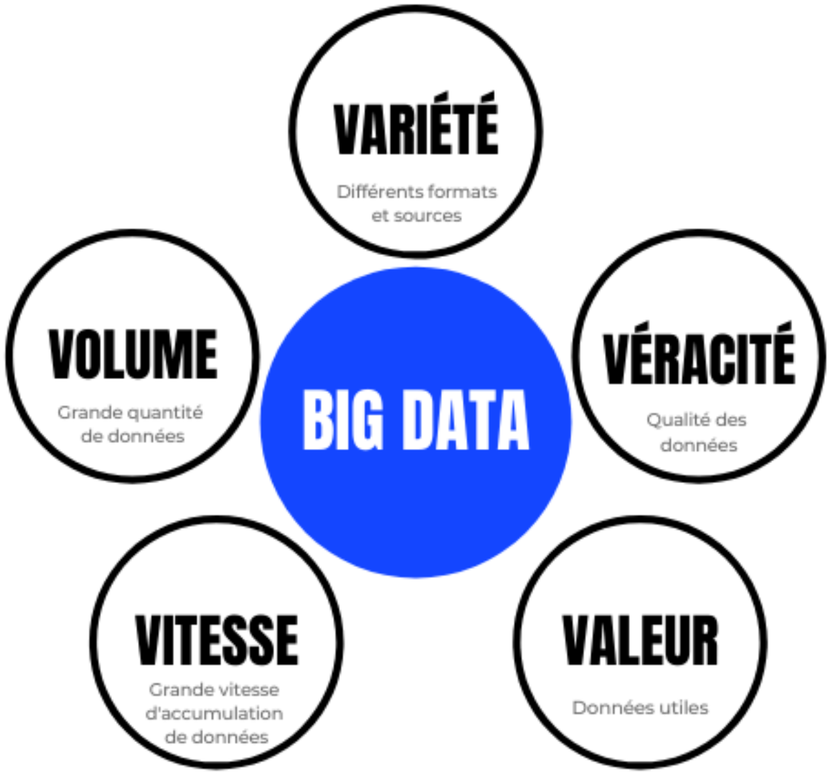
Business Consultant

PRÉSENTATION DU PROJET

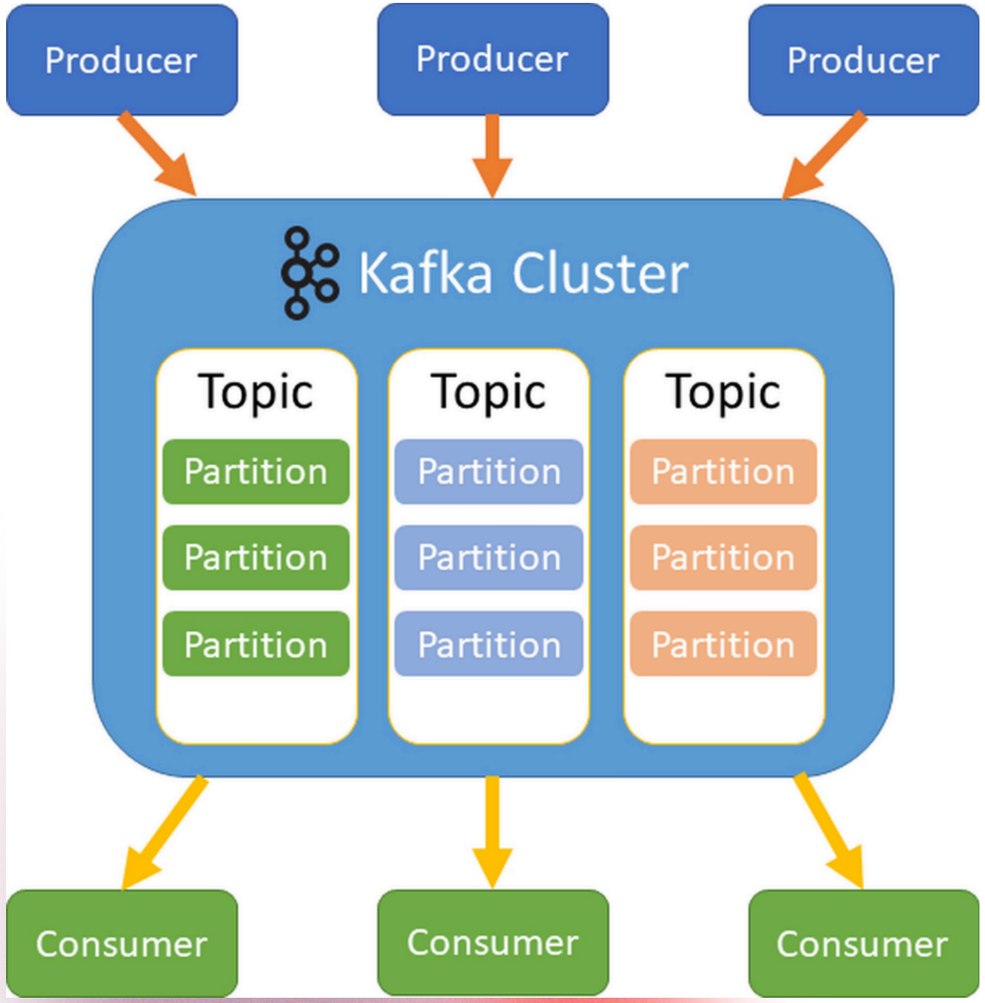
Notre projet consistait à mettre en place des environnements kafka et Hadoop en nous basant sur les 5 V.
Nous avons a utilisé l'API Velib qui fournit des données sur le réseau de vélos en libre-service Vélib' à Paris.

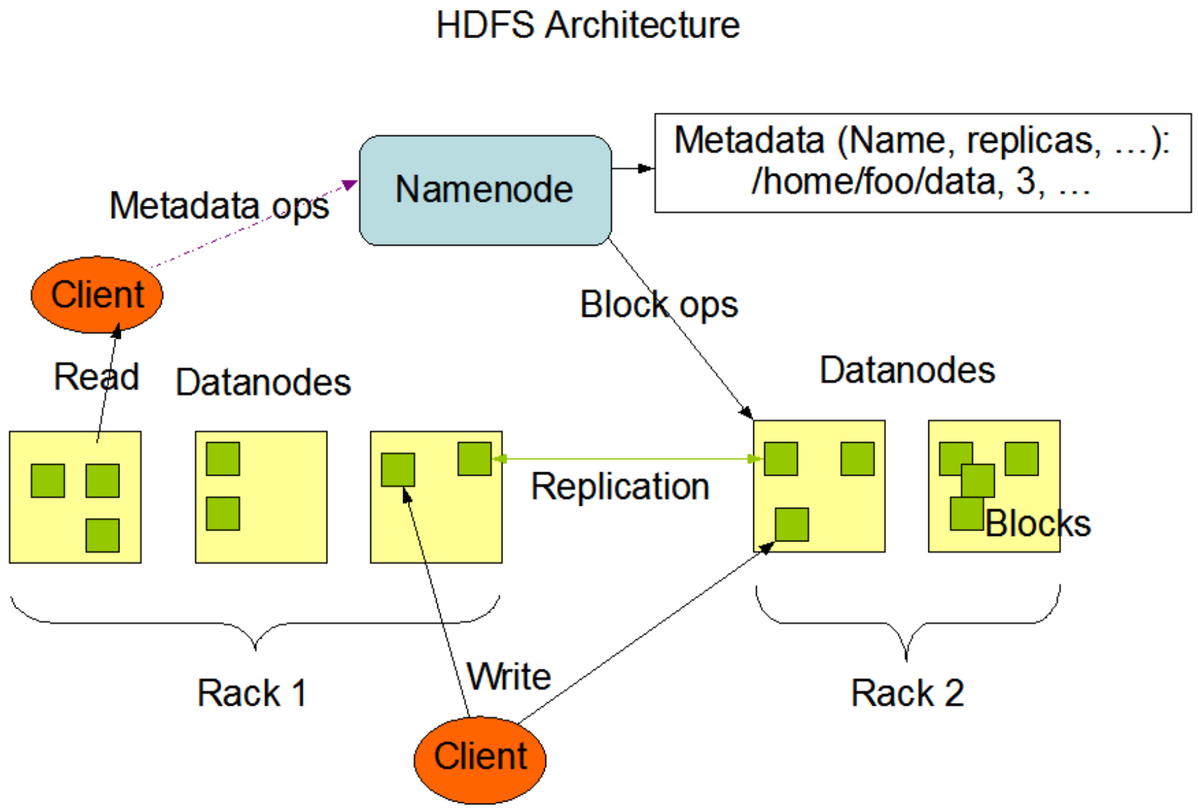
cliquez ici pour accéder au à l'API

C'EST QUOI LES 5 V?



PRÉSENTATION DES TECHNOLOGIES

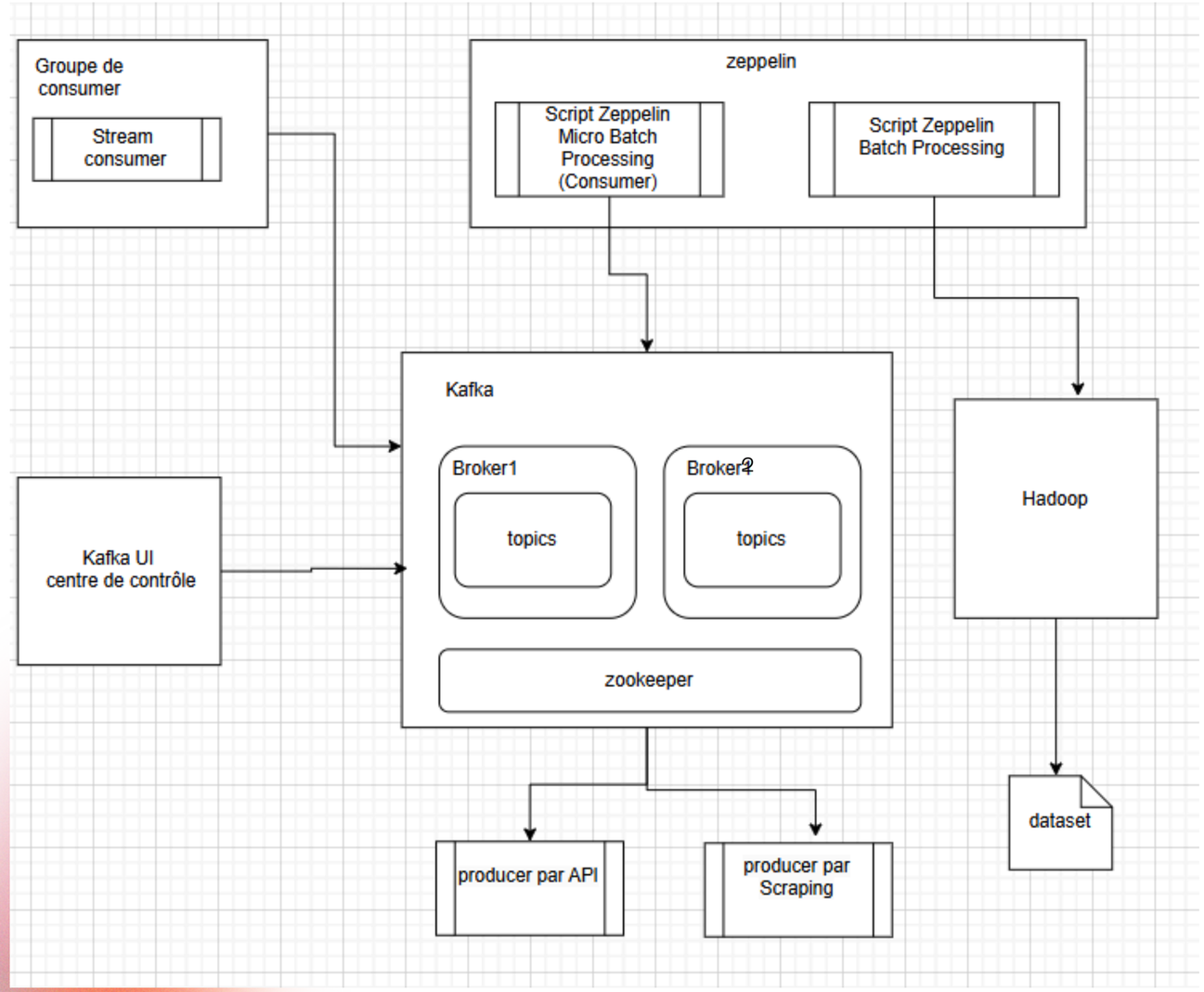




PRÉSENTATION DES TECHNOLOGIES

GitHub

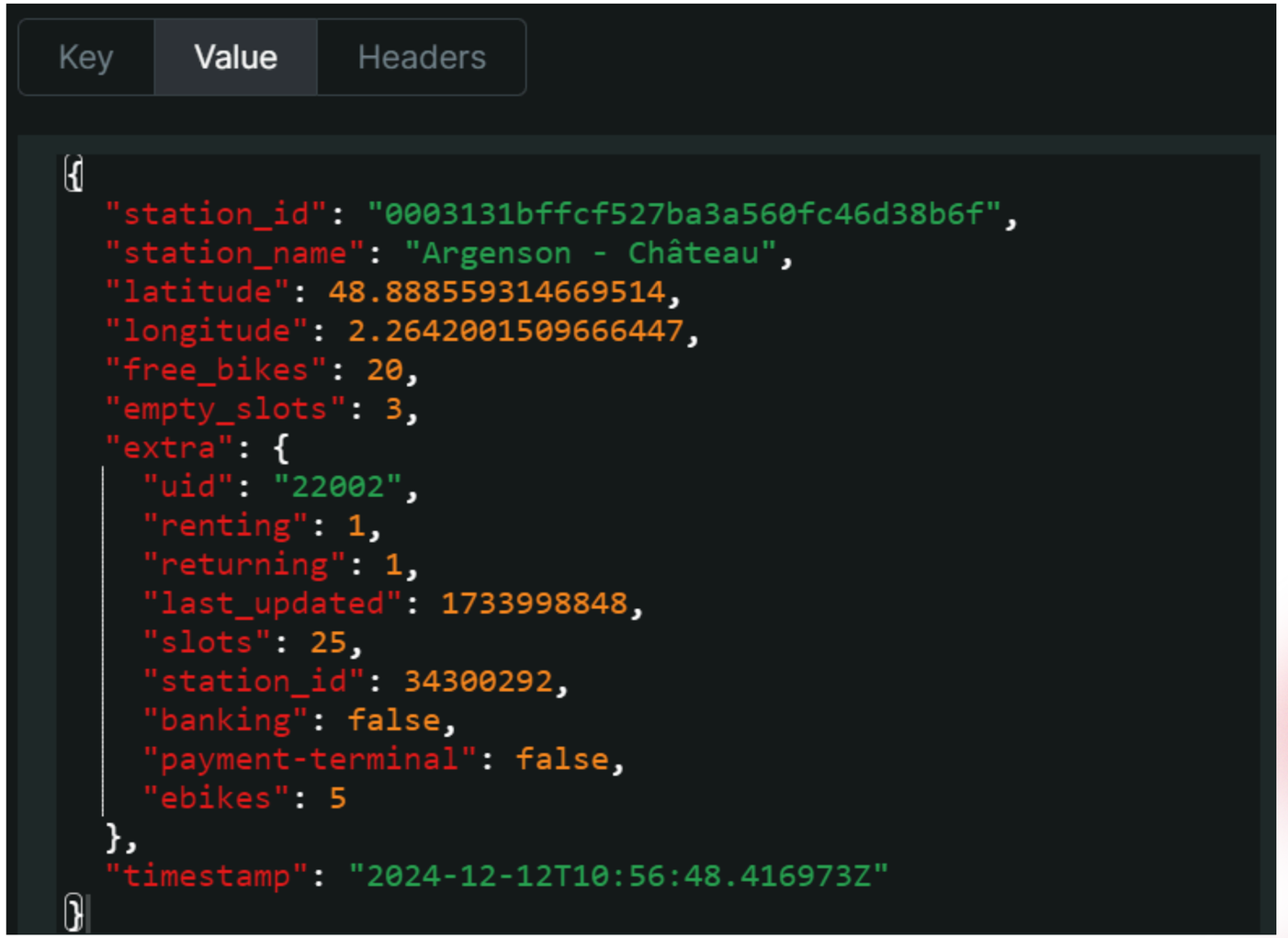
ARCHITECTURE DU PROJET



PRODUCER AVEC APPEL D'API

```
import json
import time
import requests

def json_serializer(data):
    return json.dumps(data).encode('utf-8')

# Vérifiez l'adresse et le port du serveur Kafka
producer = KafkaProducer(
    bootstrap_servers=['kafka1:9092', 'kafka2:9093'],  # Adresse de nos serveurs
    value_serializer=json_serializer
)

while True:
    url = "https://api.citybik.es/v2/networks/velib"
    try:
        response = requests.get(url)
        data = response.json()

        # Extraction des stations et leurs informations
        stations = data['network']['stations']
        for station in stations:
            station_info = {
                'station_id': station['id'],
                'station_name': station['name'],
                'latitude': station['latitude'],
                'longitude': station['longitude'],
                'free_bikes': station['free_bikes'],
                'empty_slots': station['empty_slots'],
                'extra': station['extra'],  # Récupération des informations supplémentaires
                'timestamp': station['timestamp']
            }
            print(f"Producing message: {station_info}")
            producer.send('velib_topic', station_info)  # Utilise le producer pour envoyer un objet
            producer.flush()  # Demande au producer d'attendre que le message soit bien envoyé

        time.sleep(60)  # Pause d'une minute entre les appels à l'API

    except Exception as e:
        print(f"Unexpected error: {e}")
```

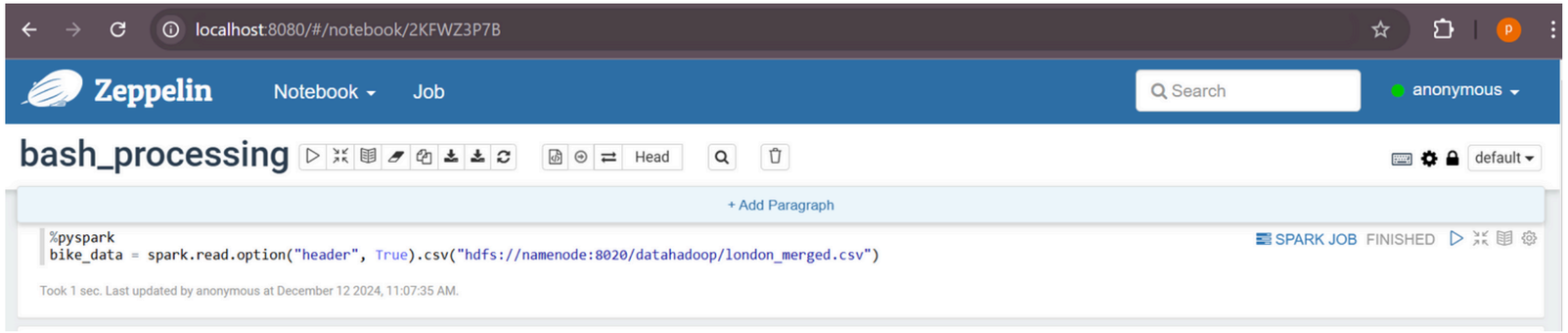
RÉSULTAT

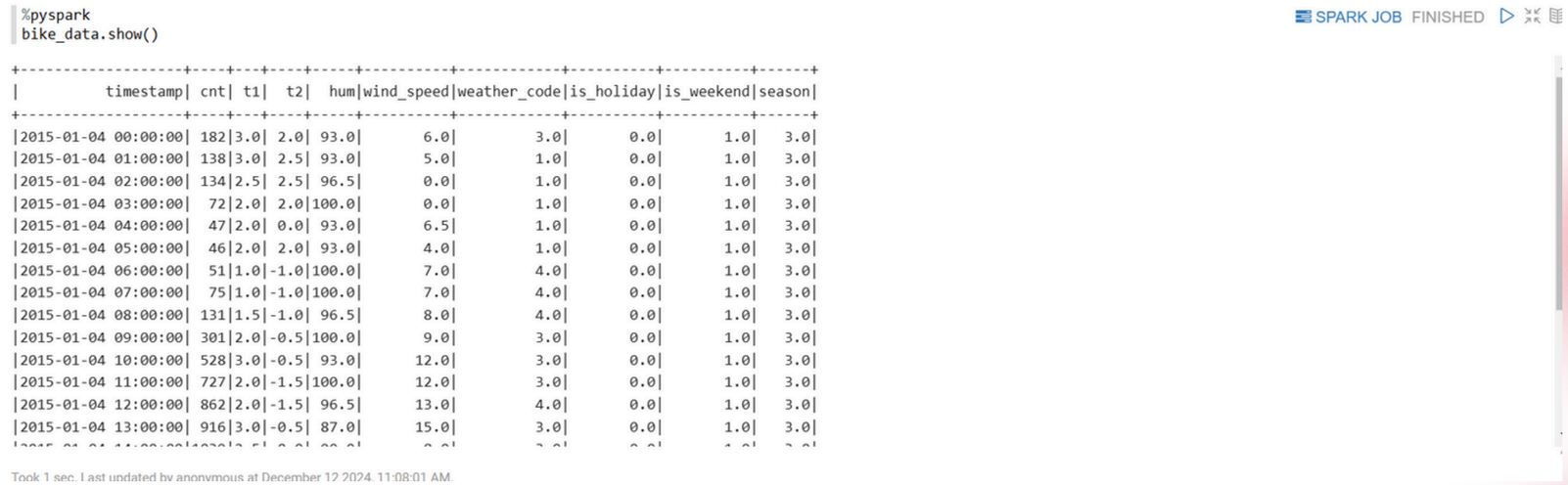

Key
Value
Headers
{
"station_id": "0003131bffcf527ba3a560fc46d38b6f",
"station_name": "Argenson - Château",
"latitude": 48.888559314669514,
"longitude": 2.26420015096666447,
"free_bikes": 20,
"empty_slots": 3,
"extra": {
"uid": "22002",
"renting": 1,
"returning": 1,
"last_updated": 1733998848,
"slots": 25,
"station_id": 34300292,
"banking": false,
"payment-terminal": false,
"ebikes": 5
},
"timestamp": "2024-12-12T10:56:48.416973Z"
}

PRODUCER WEB + SCRAPER

```python
from kafka import KafkaProducer
import json
import time
import requests
from bs4 import BeautifulSoup

def json_serializer(data):
    """
    Sérialise les données Python en format JSON pour l'envoi via Kafka.
    """
    return json.dumps(data).encode('utf-8')

# Création du producteur qui pointe vers notre cluster Kafka (localhost:9092)
producer = KafkaProducer(
    bootstrap_servers=['kafka:9092,kafka:9093'],
    value_serializer=json_serializer
)

while True:
    """
    Boucle infinie pour récupérer et envoyer les données en continu.
    """
    url = "https://api.citybik.es/v2/networks/velib"

    try:
        """
        Envoie une requête GET à l'API Velib et récupère les données.
        """
        response = requests.get(url)
        soup = BeautifulSoup(response.text, 'lxml')

        station_data = json.loads(response.text)  # Parse la réponse JSON

        # Extraction des données pertinentes de la station
        station_id = station_data['station_id']
        station_name = station_data['station_name']
        free_bikes = station_data['free_bikes']
        empty_slots = station_data['empty_slots']
        timestamp = station_data['timestamp']

        message = {
            'station_id': station_id,
            'station_name': station_name,
            'free_bikes': free_bikes,
            'empty_slots': empty_slots,
            'timestamp': timestamp
        }

        print(f"Envoi du message : {message}")
        producer.send('velib_topic', message)  # Envoie le message au topic spécifié
        producer.flush()  # Force l'envoi immédiat du message
        time.sleep(2)  # Pause de 2 secondes avant la prochaine requête

    except Exception as e:
        """
        Gestion des erreurs potentielles.
        """
        print(f"Erreur inattendue : {e}")
```

BATCH PROCESSING





BATCH PROCESSING

```
%pyspark
from pyspark.sql.functions import col

# Je convertis
bike_data = bike_data.withColumn("t1", col("t1").cast("float")) \
                     .withColumn("season", col("season").cast("int"))

# Vérifier si 't1' a des valeurs hors plage plausible (-50°C à 50°C)
invalid_t1_count = bike_data.filter((col("t1") < -50) | (col("t1") > 50)).count()
if invalid_t1_count > 0:
    print(f"Alerte : {invalid_t1_count} lignes ont des valeurs invalides pour 't1'.")
else:
    print("Les données de 't1' sont valides.")

# Calculate de la moyenne sur la  temperature (t1)
average_t1 = winter_data.selectExpr("AVG(t1) as average_t1").collect()[0]["average_t1"]

print(f"la température moyenne en hiver   (season 3) est : {average_t1}")
```

SPARK JOB FINISHED

```
Les données de 't1' sont valides.
la température moyenne en hiver   (season 3) est : 7.686951501154734
```

PROBLEMES RENCONTRÉS

On a un peu connu des soucis de communication dus peut être à la distance, et des difficultés de travail en équipe et finalement du mal à fournir un travail d'ensemble .

Mais aussi des soucis de virtualisation avec Docker. On s'est rendus compte qu'utiliser WSL 2 (Windows Linux Subsystem) fournissait une meilleure compatibilité car c'est l'approche préconisée par Docker.

PLACE À LA DEMO

CONCLUSION

Ce projet a été réalisé dans le but d'apprendre et d'appliquer les processus de traitement de données en temps réel. Dans ce cadre, une chaîne de traitement de données (data pipeline) a été créée en utilisant la plateforme Kafka. Ce système permet de collecter, traiter et analyser des données provenant de différentes sources grâce aux producteurs (Producers) et consommateurs (Consumers).

MERCI