DIC2 GIT / EPT

DONNÉES MÉTÉO EN TEMPS RÉEL ET EN PRÉDICTION

Big data et applications

Présenté par :
Ameth Ba
Thierno Saydou Talla

PROFESSEUR : M GUÈYE

PLAN

- Introduction
- Problématique et Objectifs
- Workflow du projet
- Visuels du Dashboard
- Conclusion

Problématique et Objectif

Les données météo évoluent rapidement et nécessitent des traitements temps réel pour être utiles aux utilisateurs. Il faut une architecture scalable, capable d'ingérer des flux, de les analyser et de restituer les informations pertinentes.

Objectif

L’objectif était de concevoir un système capable de collecter, traiter et analyser en temps réel des données météorologiques, tout en proposant des prédictions pour les heures à venir.

Ingestion des Données

Dans notre projet, l'ingestion est assurée par Apache Kafka, qui reçoit les données de deux producteurs (des scripts python) :

Producteur 1 – API OpenWeather

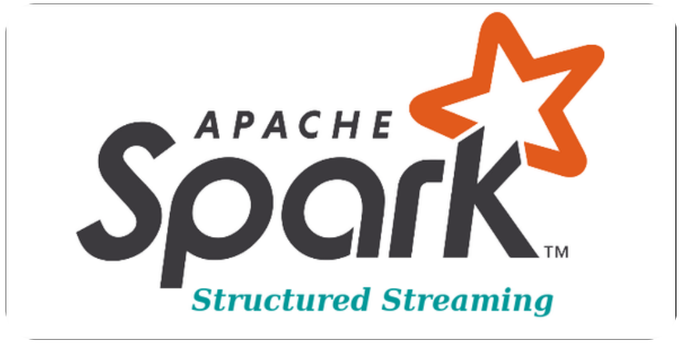
- Script Python (weather_api.py)
- Collecte toutes les 10 minutes : température, humidité, vent, etc.
- Données envoyées au topic Kafka weather-api.

Producteur 2 – Capteur IoT simulé

- Script Python (iot_device.py)
- Génère des données locales toutes les 10 minutes.
- Données envoyées au topic capteur-iot

- Kafka est déployé via Docker (docker-compose) avec Kafka-UI pour visualiser les messages en temps réel.
- Ce choix permet une ingestion fiable, évolutive, et en temps réel, avec des topics séparés pour chaque source.

Traitement avec Spark



Nous utilisons Spark Structured Streaming pour traiter les données météo en temps réel depuis Kafka.

Spark lit deux flux Kafka : weather-api (données réelles) et capteur-iot (simulées).

Les données sont nettoyées, fusionnées, puis agrégées :

- **10 minutes → weather_raw_10min**
- **1 heure → weather_hourly**
- **1 jour → weather_daily**

Les résultats et les alertes sont enregistrés dans PostgreSQL.

Priorité capteur local > OpenWeather en cas de conflit

Stockage des données

Les données sont centralisées dans PostgreSQL, via un schéma structuré :

weather_raw_10min

données brutes
toutes les 10 min

weather_hourly

weather_daily

agrégats temporels
(horaires et journalières)

weather_prediction

sorties du
modèle

weather_alerts

Les alerts

Modèle de prédiction

Un modèle simple basé sur les 6 dernières heures est utilisé pour prédire la température pour les 4 prochaines heures.

Méthode

moyenne pondérée avec corrections basées sur l'évolution récente

- Implémenté dans un fichier python, appelé dans Spark via udf().
- Résultat stocké dans la table weather_predictions.

Logique du modèle

- calcule une moyenne pondérée des 6 dernières températures.
- On détecte la tendance thermique (hausse ou baisse) via une régression linéaire sur les 6 points précédents

La tendance est multipliée par un facteur (0.7) pour pondérer son effet sur les heures suivantes.

- Correction par l'humidité

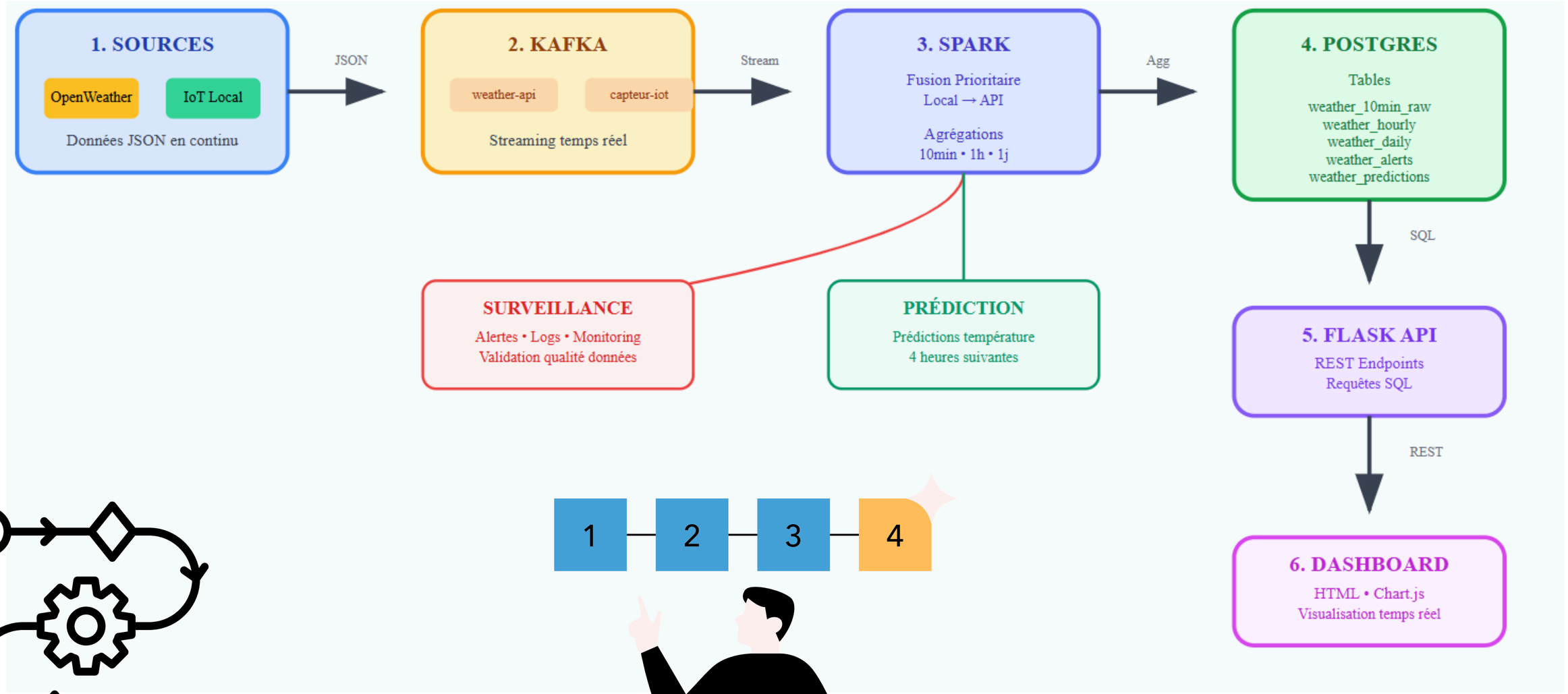
- Si humidité > 80%, effet refroidissant : -0.5
- Si humidité < 30%, effet réchauffant : +0.3
- Sinon : aucun effet

- Correction par le vent

- Si vent > 20 km/h, refroidissement fort : -0.8
- Si vent > 10, refroidissement modéré : -0.3
- Sinon : aucun effet

L'idée est de simuler le refroidissement par convection

Workflow


1. SOURCES
OpenWeather
IoT Local
Données JSON en continu
JSON
2. KAFKA
weather-api
capteur-iot
Streaming temps réel
Stream
3. SPARK
Fusion Prioritaire
Local → API
Agrégations
10min • 1h • 1j
Agg
4. POSTGRES
Tables
weather_10min_raw
weather_hourly
weather_daily
weather_alerts
weather_predictions
SQL
SURVEILLANCE
Alertes • Logs • Monitoring
Validation qualité données
PRÉDICTION
Prédictions température
4 heures suivantes
5. FLASK API
REST Endpoints
Requêtes SQL
REST
6. DASHBOARD
HTML • Chart.js
Visualisation temps réel

1
2
3
4

Dashboard de visualisation

Les données sont centralisées dans PostgreSQL, via un schéma structuré :

Objectif

- Visualiser les données météo en temps réel + prédictions sur 4h.
- Alertes pour anomalies (retards de données, pics de température).

Technos

- Frontend : HTML/CSS + Chart.js pour les graphiques.
- Backend : Flask , PostgreSQL (données agrégées par Spark).

Fonctionnalités Clés

Onglet "Vue d'ensemble

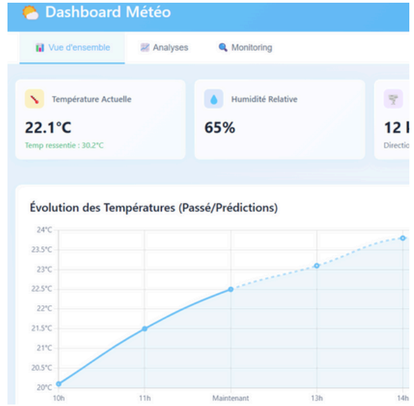
- Metrics temps réel : Température, humidité, vent, conditions.
- Graphique combiné : Historique + Prédictions.

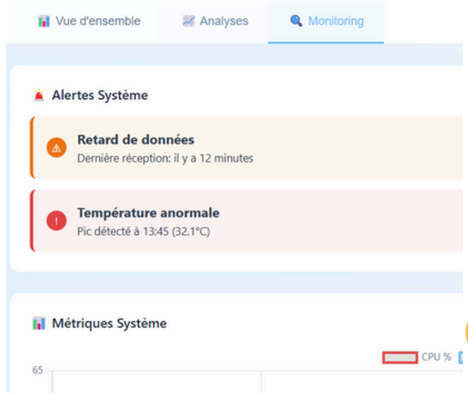
Onglet "Analyses"

- Min/Max hebdo : Barres juxtaposées pour amplitudes thermiques.
- Rose des vents : Radar chart pour directions dominantes.

Onglet "Monitoring"

- Alertes : Détection automatique des anomalies.
- logs





Dashboard de visualisation

Vue d’ensemble

Dashboard de visualisation

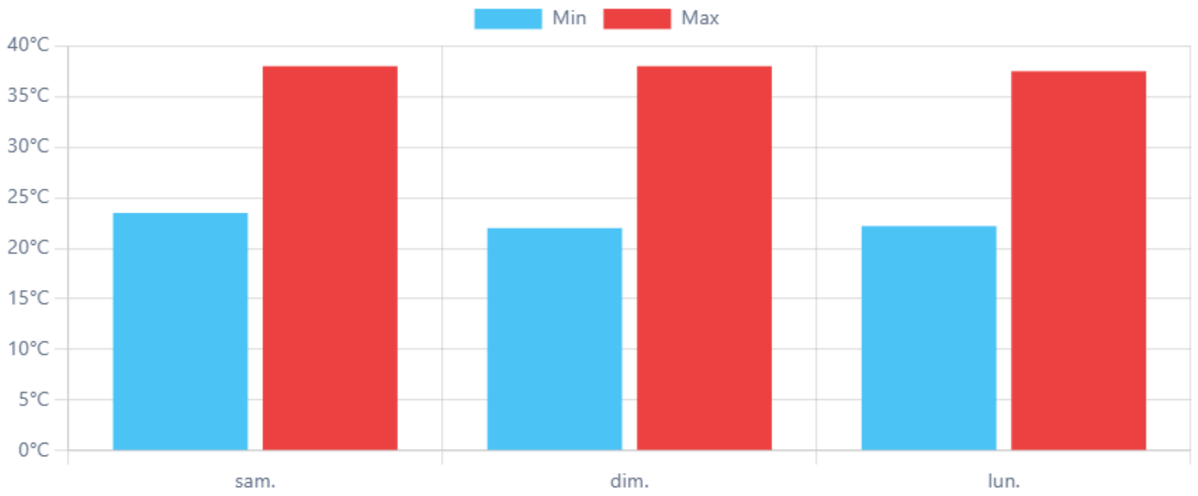
Analyses

Dashboard Météo

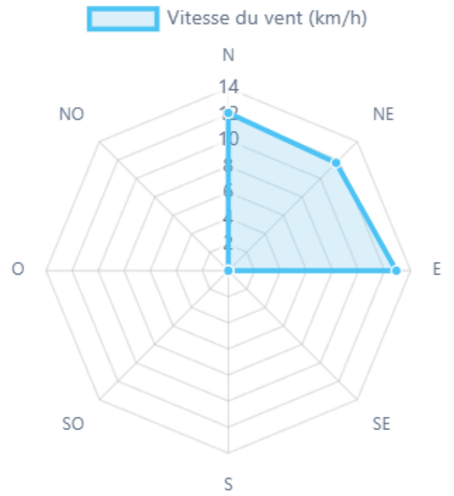
EPT

Vue d'ensemble | Analyses | Monitoring

Températures Extrêmes (7j)



Rose des Vents



Dashboard de visualisation

Monitoring

Vue d'ensemble | Analyses | Monitoring

Alertes Système

Pipeline fermé proprement
system • 3min • system_health

Erreur validation batch 1 (daily_aggregation): Arrêt inattendu de Spark
daily_aggregation • 3min • processing_error

Batch 1 vide reçu de hourly_aggregation
hourly_aggregation • 4min • data_quality

Batch 1 vide recu de raw_aggregation

Statistiques Alertes

1	0
Critiques	Élevées
5	**2**
Moyennes	Informations

Journal des Erreurs

```
[9:24:38 AM] CRITICAL - daily_aggregation: Erreur validation batch 1 (daily_aggregation): Arrêt inattendu de Spark
```

État des Composants

Pipeline Spark	Avertissement
API Météo	Opérationnel
Capteur Local	Opérationnel

Conclusion

Ce projet nous a permis de mettre en œuvre une architecture Big Data complète et fonctionnelle, autour d'un cas pratique concret : le traitement et l'analyse des données météorologiques en temps réel.

Grâce à l'intégration de Kafka, Spark Structured Streaming, PostgreSQL et d'un dashboard web, nous avons pu :

- Collecter et traiter un flux continu de données issues de plusieurs sources (API et capteur simulé),
- Appliquer des opérations de nettoyage, d'agrégation et de prévision,
- Stocker les résultats dans une base de données relationnelle,
- Et proposer une visualisation dynamique, claire et interactive.

Ce travail nous a permis de mieux comprendre les enjeux du traitement de flux de données en temps réel, ainsi que l'interconnexion des technologies dans un écosystème Big Data moderne.

MERCI !

Smart Data for Clever Solutions !