

Pipeline d'Analytics Météo Cloud-Native

Architecture End-to-End : De l'API OpenWeather aux Dashboards Temps Réel

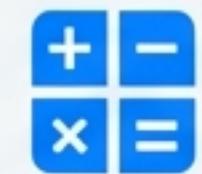
Projet Semestriel
Janvier 2026

Le Défi : Collecte Temps Réel vs Contraintes API



Le Problème (Contraintes)

- Quota Gratuit OpenWeather : Limité à 1000 appels/jour
- Objectif : Surveillance de 5 villes stratégiques



La Solution (Mathématique)

Calcul de fréquence :

$$86400 \text{ sec/jour} \div 600 \text{ sec (10 min)} = \mathbf{144 \text{ appels/ville/jour}}$$

Consommation totale :

$$144 \times 5 \text{ villes} = \mathbf{720 \text{ appels/jour}}$$



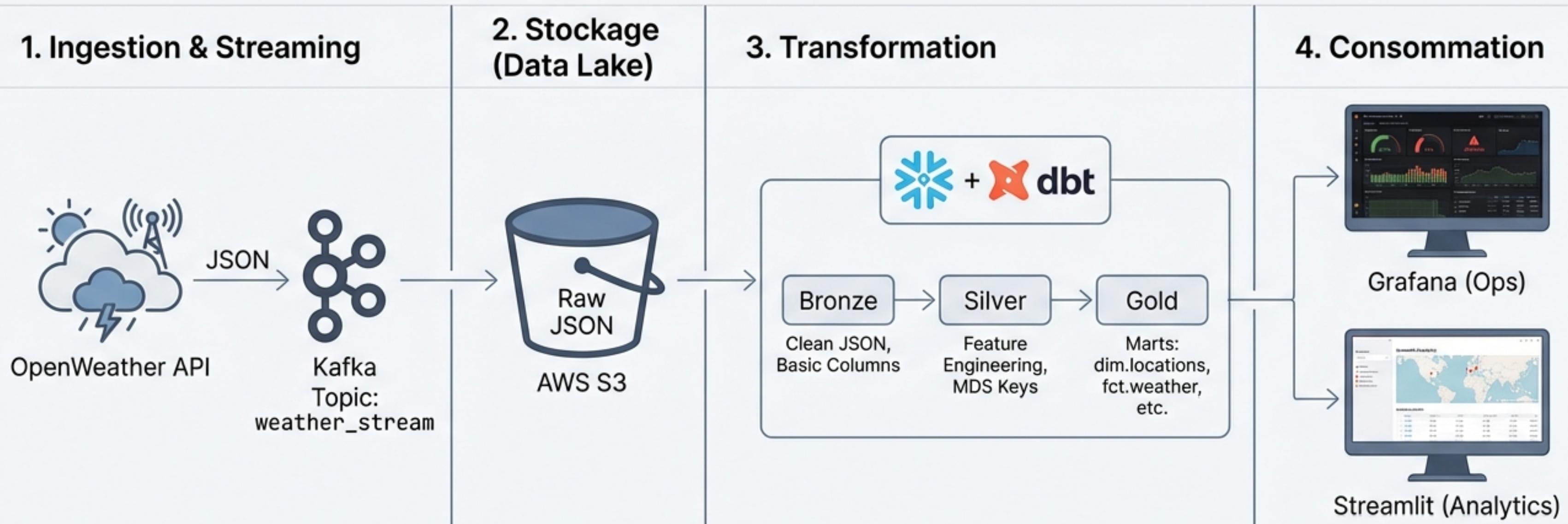
Résultat :

$$720 < 1000$$

(Marge de sécurité de 28% respectée)

Insight Clé : Une fréquence de 10 minutes est le point d'équilibre optimal pour maximiser la fraîcheur des données sans briser le quota gratuit.

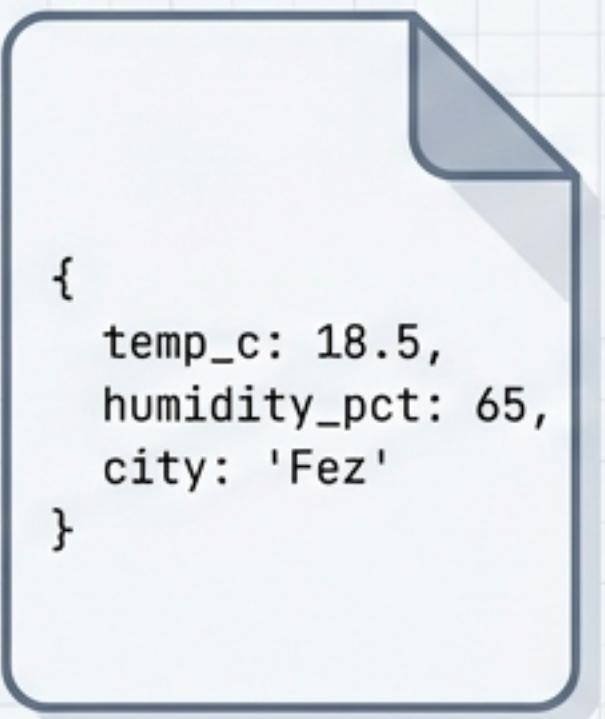
Architecture Globale : Flux de Données End-to-End



Une approche ELT (Extract, Load, Transform) découplant le stockage du calcul pour une flexibilité maximale.

Ingestion et Streaming : La Vélocité via Kafka

Source : API OpenWeather

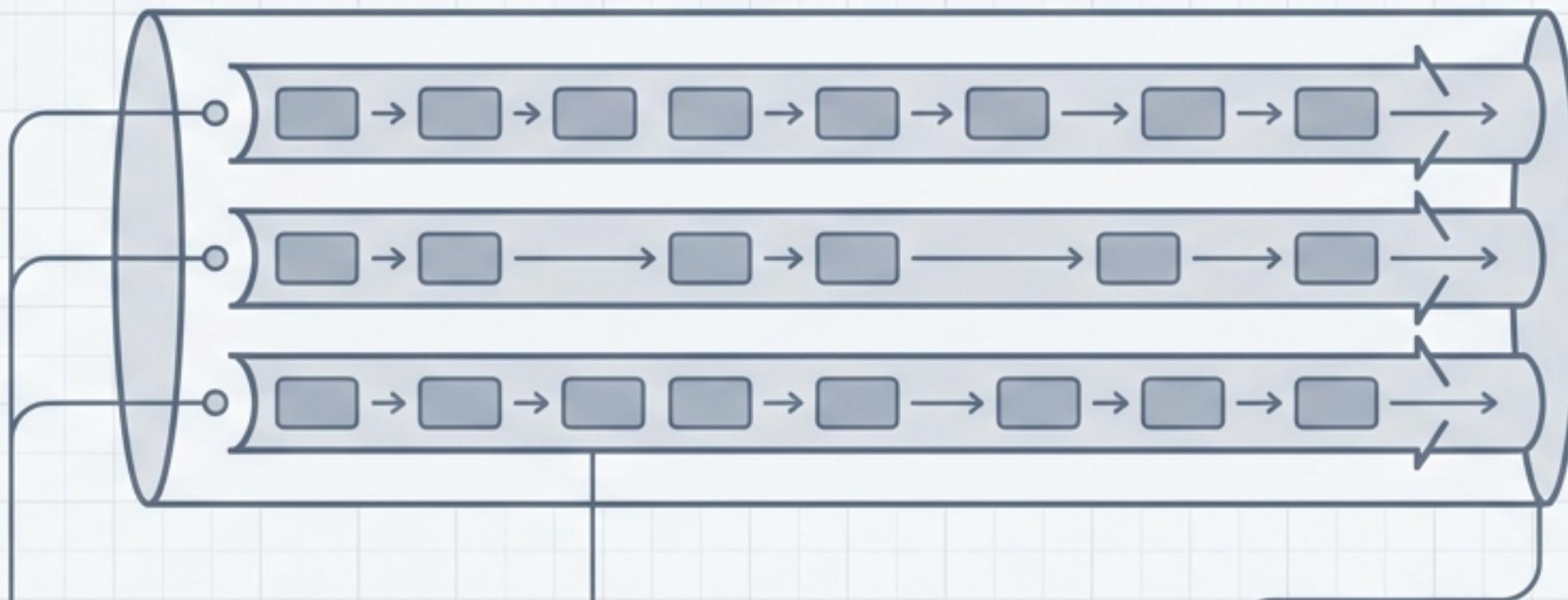


Payload JSON riche

- (Température, Humidité, Vent, Coordonnées)

Transport : Apache Kafka

Topic: weather_stream



Partitions : 3
(Pour paralléliser le traitement par ville)

Replication Factor : 2
(Haute disponibilité)

Rétention : 7 jours
(Buffer de sécurité)

Data Lake S3 : Stockage Brut et Optimisation des Coûts

```
s3://weather-bucket/weather/raw/  
└── year=2026  
    └── month=01  
        └── day=20  
            └── hour=14  
                ├── city=Fez  
                └── city=Paris
```

Stratégie de Partitionnement

- Structure : Année / Mois / Jour / Heure / Ville
- Bénéfice : "Partition Pruning" pour accélérer les requêtes et réduire les coûts de lecture.



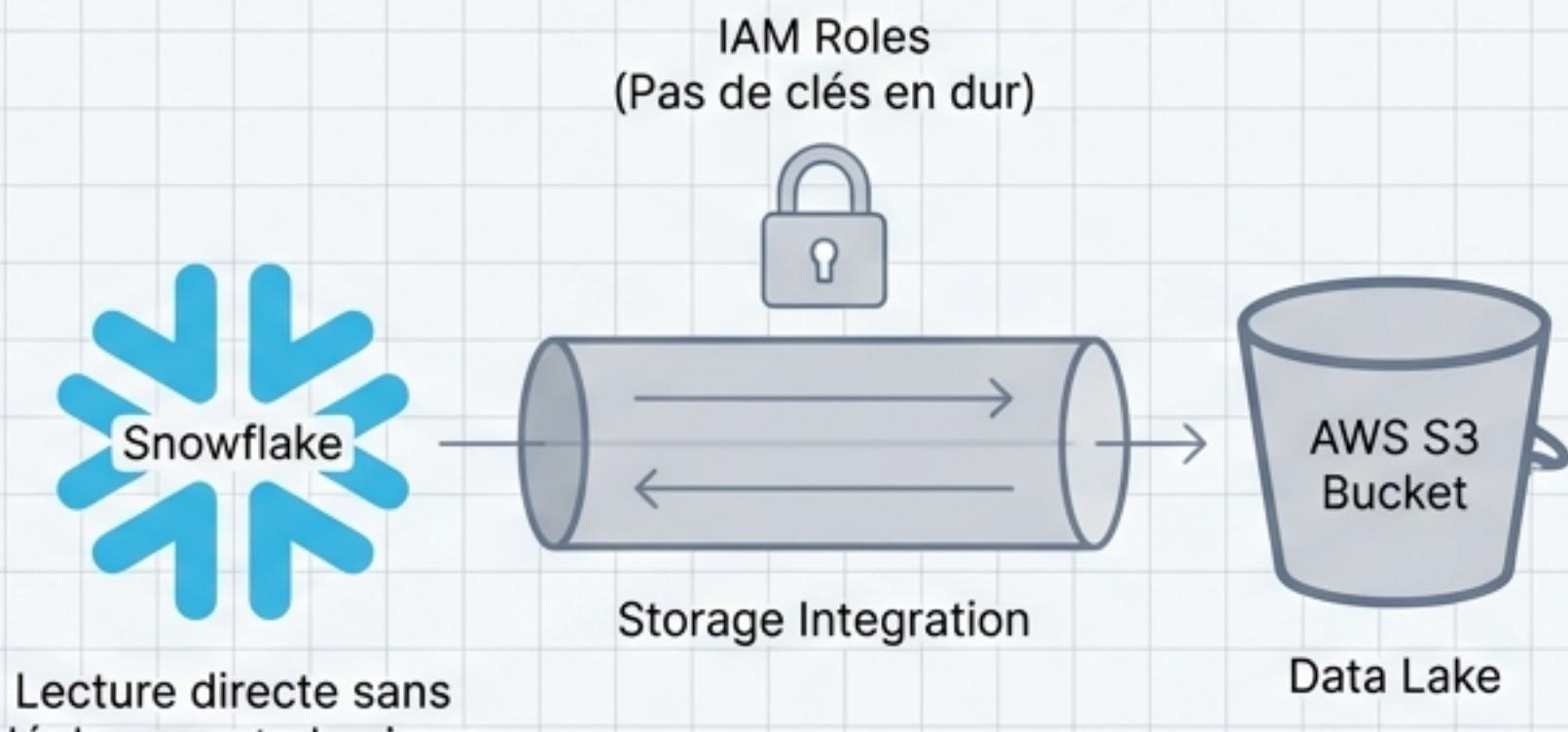
Politique de Cycle de Vie (Lifecycle Policy)

Règle : weather-retention-30days
Action : Suppression automatique après 30 jours.
Objectif : Maintenir le Data Lake léger (éphémérité utile).

Le Pont Sécurisé : Intégration S3 vers Snowflake

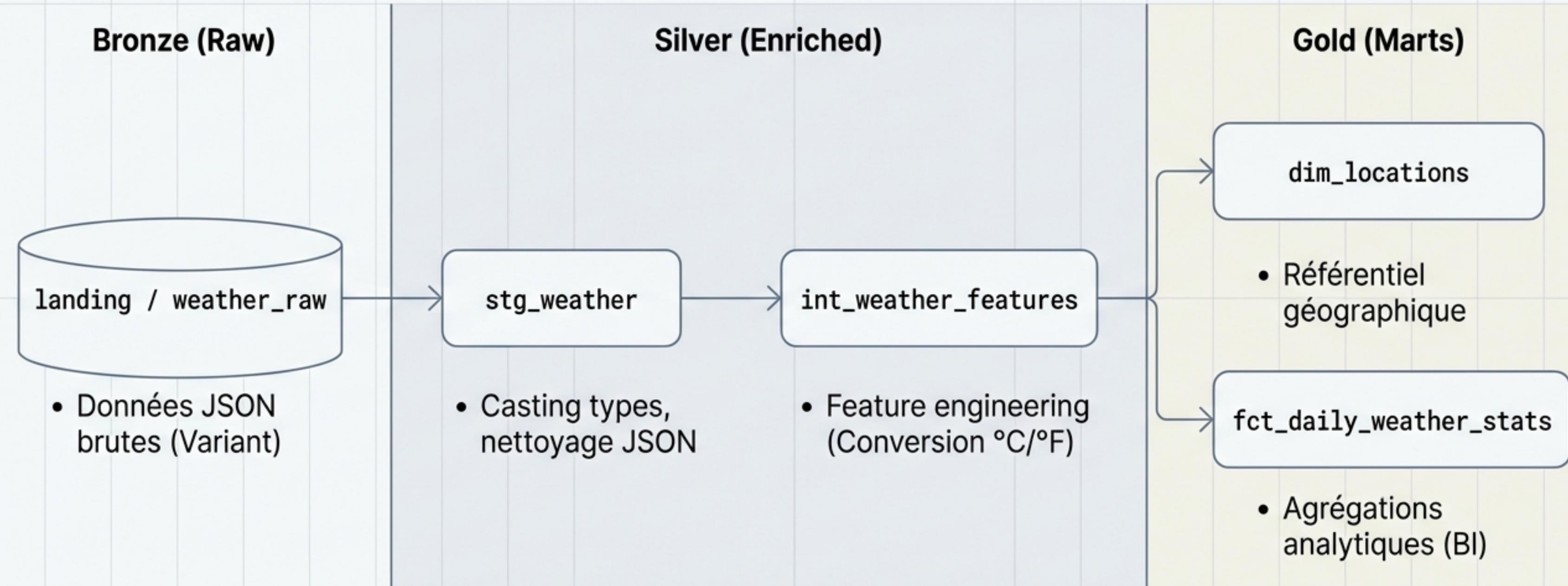
```
CREATE STORAGE INTEGRATION s3_kafka_integration  
TYPE = EXTERNAL_STAGE  
STORAGE_PROVIDER = 'S3'  
STORAGE_AWS_ROLE_ARN = 'arn:aws:iam::.../role/aya-role'
```

Mécanisme : External Stage



Avantage : Sécurité et séparation nette entre Compute (Snowflake) et Storage (AWS S3).

Transformation et Qualité : Architecture en Médaillo



Tests automatisés (Unique, Not-Null) intégrés au pipeline.

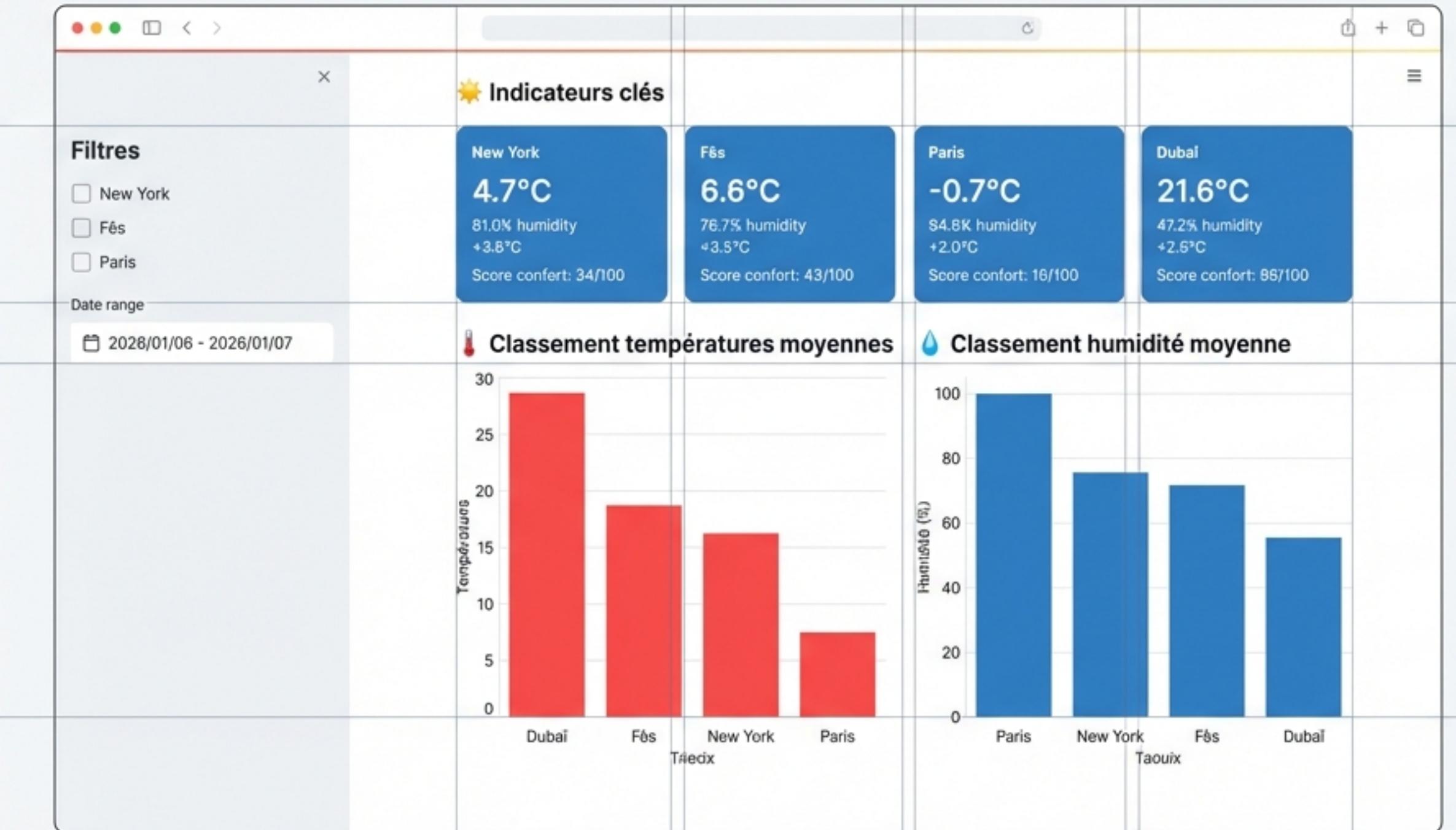
Monitoring Opérationnel : Le Cockpit Grafana



Rôle : Surveillance Temps Réel

- **Actualisation :** Quasi-instantanée (Connecteur Snowflake)
- **Alerting :** Seuils critiques (Canicule > 40°C, Vent > 50km/h)
- **Use Case :** L'écran de contrôle pour la réaction immédiate.

Exploration Analytique : Le Labo Streamlit



Rôle : Data Science & Exploration

- **Expérience "Low Code"** : Filtres dynamiques et sliders.
- **Indicateurs Métier** : Calcul du "Score de Confort".
- **Use Case** : L'outil pour découvrir des tendances et exporter des rapports.

Synthèse : Un Pipeline Résilient et Optimisé

Fiabilité



Architecture distribuée (Kafka) et tolérante aux pannes. Reprise automatique via Checkpointing.

Coût



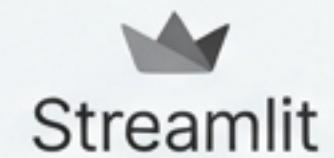
Zéro dépassement de quota API. Nettoyage S3 automatique via Lifecycle Policies.

Valeur



Double interface de restitution (Ops vs Analytics) couvrant tous les besoins métier.

Tech Stack



Une fondation solide prête pour l'intégration de modèles de Machine Learning prédictifs.