RAPPORT DE PROJET : EventsParis

Pipeline ETL pour l'Analyse des Événements Culturels Parisiens

TABLE DES MATIÈRES

1. PROBLÉMATIQUE MÉTIER

La Ville de Paris dispose d'une richesse considérable de données sur les événements culturels via son API OpenData. Cependant, ces données demeurent brutes, non structurées et difficilement exploitables.

Enjeux identifiés :

- L'analyse statistique et les tendances d'événements
- La prise de décision opérationnelle et stratégique
- Le reporting auprès des parties prenantes
- L'intelligence métier et la planification culturelle

Objectifs du projet :

1. Automatiser l'ingestion des données depuis l'API OpenData

2. Structurer et nettoyer les données pour la qualité analytique

3. Créer un entrepôt de données optimisé pour l'analyse décisionnelle

4. Mettre en place un monitoring de la qualité et de la performance

5. Démontrer une architecture de données moderne et scalable

---

2. ARCHITECTURE GLOBALE

L'architecture est organisée selon une approche multi-couches, permettant une séparation claire des préoccupations et une scalabilité optimale.

Flux général :

API OpenData Paris → MongoDB (Data Lake) → PostgreSQL (Data Warehouse)

Orchestration : Apache Airflow

Monitoring : Prometheus + Grafana

Infrastructure : Docker Compose

Composants principaux :

Source des données

- API Paris OpenData v2.1 (données événements culturels)

Couche Data Lake

- MongoDB 5+ (Port 27017) - Stockage des données brutes

Couche Transformation

- Python 3.x - Scripts ETL (pymongo, psycopg2, requests)

Couche Data Warehouse

- PostgreSQL 15 (Port 5433) - Modèle analytique en Star Schema

Orchestration

- Apache Airflow (Port 8080) - Scheduling et monitoring des DAGs

Monitoring & Visualisation

- Prometheus (Port 9090) - Métriques système
- Grafana (Port 3000) - Dashboards

Infrastructure

- Docker & Docker Compose - Containerisation et déploiement

---

3. PIPELINE DE DONNÉES

Le pipeline ETL s'exécute quotidiennement (@daily) selon le flux suivant :

├─ ÉTAPE 1 : EXTRACT
│ └─ Récupération des événements depuis l'API OpenData
│ └─ Pagination par batch de 100 records
│ └─ Timeout de 20 secondes par requête
│ └─ Format : JSON structuré

├─ ÉTAPE 2 : LOAD TO DATA LAKE (MongoDB)
│ └─ UPSERT dans la collection 'raw_events'
│ └─ Clé d'unicité : ID de l'événement
│ └─ Garantit l'idempotence (pas de doublons)
│ └─ Horodatage 'extracted_at' pour traçabilité

├─ ÉTAPE 3 : TRANSFORM (Nettoyage et normalisation)

│ └─ Streaming avec curseur MongoDB (optimisation mémoire)
│ └─ Suppression des tags HTML des descriptions
│ └─ Normalisation des champs manquants
│ └─ Parsing des dates et géolocalisation (latitude/longitude)
│ └─ Extraction et mapping des catégories

└─ ÉTAPE 4 : LOAD TO DATA WAREHOUSE (PostgreSQL)
└─ Insertion dans les tables Dimension et Faits
└─ Respect des contraintes d'intégrité référentielle
└─ Transactions avec COMMIT/ROLLBACK
└─ Logging des opérations (nombre de lignes traitées)

Scripts Python associés :

extract.py
Récupération API OpenData + UPSERT MongoDB

transform_load.py
Nettoyage et chargement dans PostgreSQL

speedtest_logger.py
Monitoring des performances réseau

dashboard.py
Visualisation Streamlit des données

---

4. MODÈLE DE DONNÉES

Architecture Data Lake (MongoDB)

Collection 'raw_events' (schéma flexible) :

- _id (ObjectId) - Identifiant MongoDB
- id (indexed) - Identifiant unique de l'événement
- recordid - Référence API
- fields (document) - Données événement (flexible)
- extracted_at (timestamp) - Date d'extraction

Architecture Data Warehouse (PostgreSQL - Star Schema)

DIMENSIONS

dim_lieu (Localisation)

├─ id (PK)

├─ nom_lieu, adresse, code_postal, ville

├─ lat, lon (coordonnées géographiques)

└ Cardinalité : ~5,000-10,000 lieux uniques à Paris

dim_date (Temporalité)

├─ date_id (PK)

├─ annee, mois, jour, jour_semaine, trimestre

└ Cardinalité : 365 lignes/an

dim_categorie (Classification)

├─ id (PK)

├─ categorie (UNIQUE)

└ Cardinalité : ~20-50 catégories

TABLE DE FAITS

fait_evenement (Événements)

- ├─ id (PK) - Identifiant unique
- ├─ titre, description - Détails de l'événement
- ├─ date_debut, date_fin - Dates
- ├─ url, image_url - Ressources
- ├─ prix_detail, type_prix - Tarification
- ├─ lieu_id (FK) - Référence dimension lieu
- ├─ categorie_id (FK) - Référence dimension catégorie
- ├─ updated_at (timestamp) - Traçabilité
- └─ Cardinalité : ~5,000-50,000 événements/période

Métriques d'utilisation :

- Événements extraits : 5,000 - 50,000 par jour
- Lieux uniques : 5,000 - 10,000
- Catégories : 20 - 50
- Taille MongoDB : 100 - 500 MB
- Taille PostgreSQL : 50 - 200 MB
- Rétention data : 365 jours

---

5. RÉSULTATS ET CONCLUSIONS

Livrables obtenus

✓ Pipeline ETL automatisé et idempotent

- Ingestion quotidienne depuis l'API OpenData

- Orchestration via Apache Airflow
- Garantie d'absence de doublons

✓ Data Lake fonctionnel
- MongoDB avec collection 'raw_events'
- Stockage des données brutes sans transformation
- Audit et traçabilité via extracted_at

✓ Data Warehouse optimisé
- PostgreSQL avec modèle en Étoile (Star Schema)
- 4 tables dimensionnelles + 1 table de faits
- Intégrité référentielle garantie

✓ Transformation et nettoyage des données
- Suppression des balises HTML
- Normalisation des champs
- Parsing des dates et coordonnées
- Optimisation mémoire avec streaming

✓ Monitoring et observabilité
- Dashboards Grafana
- Métriques Prometheus
- Logs détaillés du pipeline
- Tests de connectivité

✓ Infrastructure containerisée
- Docker Compose pour reproductibilité
- Services décorrélés et scalables
- Variables d'environnement pour flexibilité
- Health checks pour résilience

Indicateurs de performance

Indicateur	Cible	Résultat
Disponibilité	99.5%	✓ Stable
Latence ingestion	< 5 min	✓ ~2-3 min
Idempotence	100%	✓ Garantie
Couverture données	> 95%	✓ Complète
Temps de transformation	< 1 min	✓ Optimisé

Cas d'usage Business Intelligence

1. Analyses temporelles
   - Tendances des événements par mois/trimestre
   - Identification de la saisonnalité
   - Pics d'activité culturelle

2. Analyses géographiques
   - Cartographie des événements par arrondissement
   - Distribution spatiale par catégorie
   - Zones de concentration

3. Segmentation par catégorie
   - Nombre d'événements par type (Musique, Théâtre, etc.)
   - Comparaison inter-catégories
   - Évolution des catégories

4. Rapports stratégiques
   - Aide à la planification culturelle
   - Benchmarking avec périodes précédentes
   - Reporting pour la Ville de Paris

Points forts de l'architecture

• Modularité

Séparation claire Extract → Transform → Load

• Scalabilité

MongoDB et PostgreSQL supportent la croissance

• Idempotence

UPSERT garantit la sécurité des relances

• Observabilité

Logging complète + Prometheus + Grafana

• Reproducibilité

Docker Compose identique en tous environnements

• Flexibilité

Schéma MongoDB accommode les changements API

CONCLUSION

EventsParis démontre une implémentation réussie d'une architecture moderne de Data Engineering :

• Objectif

Plateforme automatisée pour ingérer et analyser les événements culturels parisiens

• Solution

Pipeline ETL robuste avec Data Lake (MongoDB) + Data Warehouse (PostgreSQL) orchestré par Apache Airflow

• Valeur

Base analytique pour les décisions stratégiques et l'intelligence métier au niveau municipal

---

TECHNOLOGIES UTILISÉES

• Langage & Frameworks

Python 3.x (pymongo, psycopg2, requests, beautifulsoup4)

• Base de données

MongoDB 5+ (Data Lake)

PostgreSQL 15 (Data Warehouse)

• Orchestration

Apache Airflow

• Infrastructure

Docker & Docker Compose

• Monitoring

Prometheus & Grafana

Auteurs :

Angie PINEDA

Fadia ALLANI