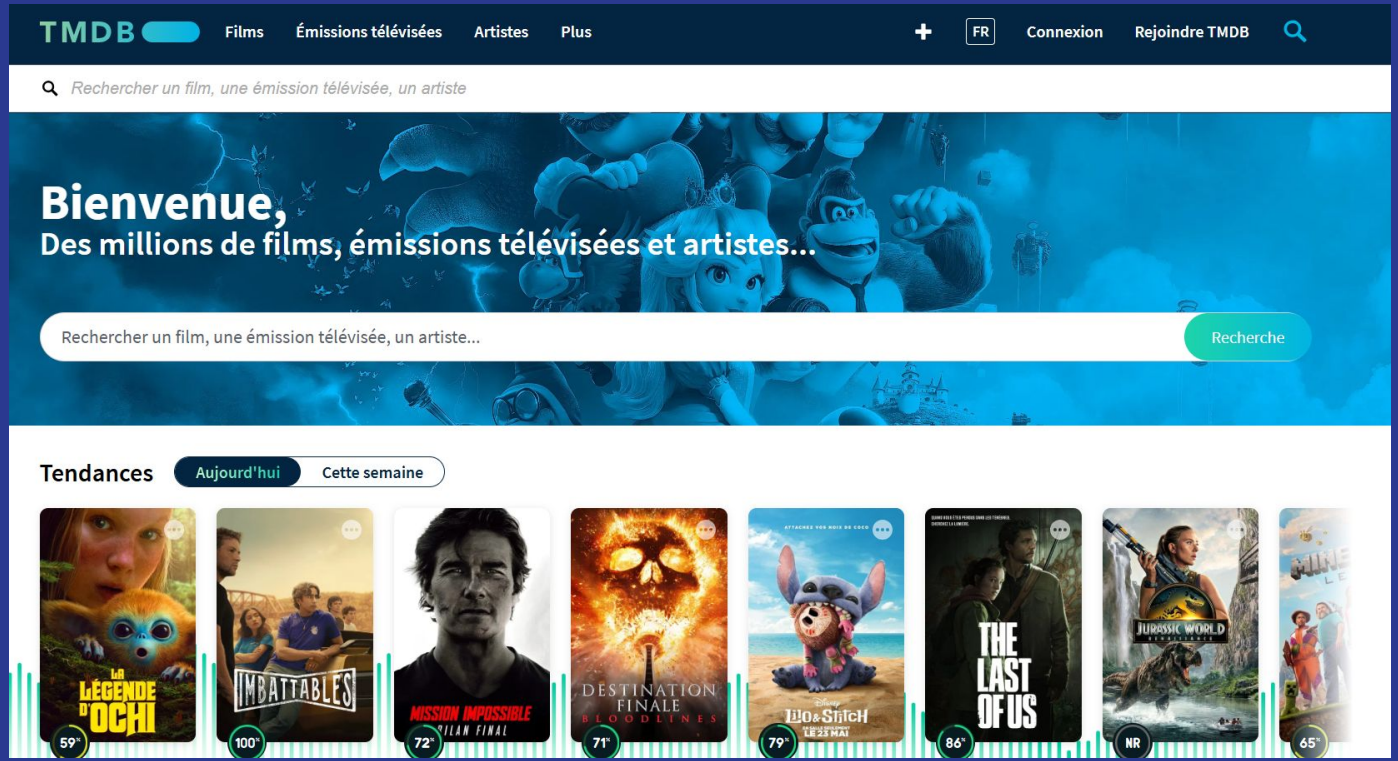
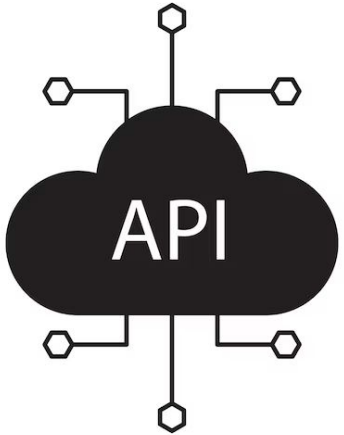
TMDB ELT Pipeline



Aymen Djerad et Yanis Zedira DIA2

Pourquoi avoir choisi l’API TMDB ?

- Données publiques, accessibles via API
- Sujet grand public : films, genres, popularité
- Données régulièrement mises à jour avec les derniers films sortis
- Richesse de données : votes, langues, dates, affiches, etc.
- Permet une dataviz dynamique et visuelle



Une API, c’est comme un serveur dans un restaurant. Vous êtes le client, vous passez commande. Mais vous ne parlez pas directement au cuisinier.

C’est le serveur – l’API – qui va transmettre votre commande à la cuisine, puis revenir avec ce que vous avez demandé.

Dans notre projet, on a “commandé” des films à l’API TMDB, et elle nous a répondu avec des données structurées, qu’on a ensuite utilisées dans notre pipeline. C’est un moyen simple, rapide, et standard d’échanger des données entre deux systèmes.

Objectifs du projet

- Construire un pipeline ELT complet (Extract, Load, Transform)
- Travailler à partir d'une source API réelle
- Ingestion, transformation SQL, orchestration
- Automatiser le pipeline via Cloud Function + Scheduler
- Visualiser les données avec Looker Studio

Présentation des données extraites

- API utilisée : `/discover/movie` pour les films, `/genre/movie/list` pour les genres
- Champs collectés : titre, genre id, id du film, description du film, nombre de votes, note moyenne, date de sortie, popularité, genres, langue, poster
- Format brut : JSON transformé en CSV
- Extraction sur 5 pages (100 films) par défaut

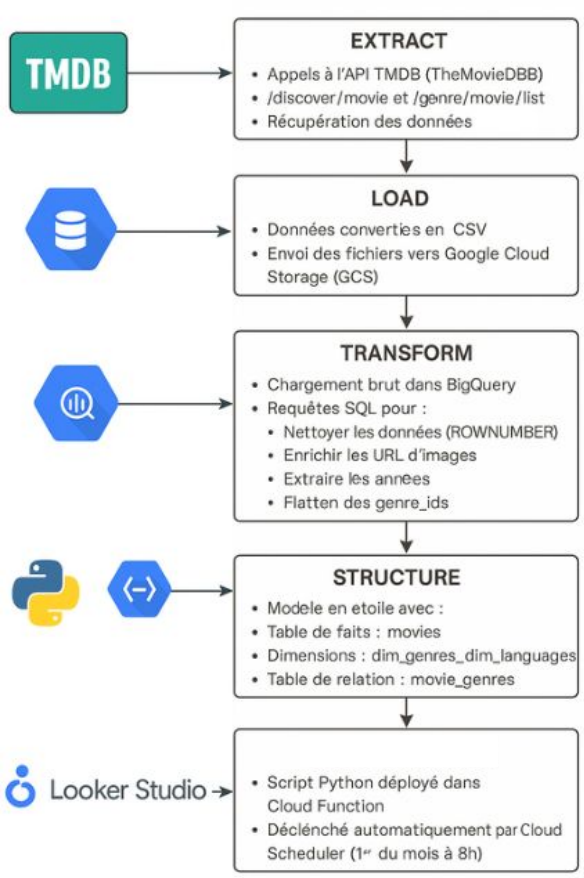
Technologies utilisées

- Google Cloud Platform : BigQuery, GCS, Cloud Function, Scheduler
- Python : orchestration, extraction API, traitement CSV
- SQL : nettoyage, enrichissement, transformation
- Looker Studio : visualisation interactive
- GitHub : versionning, doc

Sécurité & bonnes pratiques

- Clé API stockée via variable d'environnement (`TMDB_API_KEY`)
- Aucune info sensible stockée en dur
- `.env` exclu du repo avec `.gitignore`
- Code organisé : fonction Python

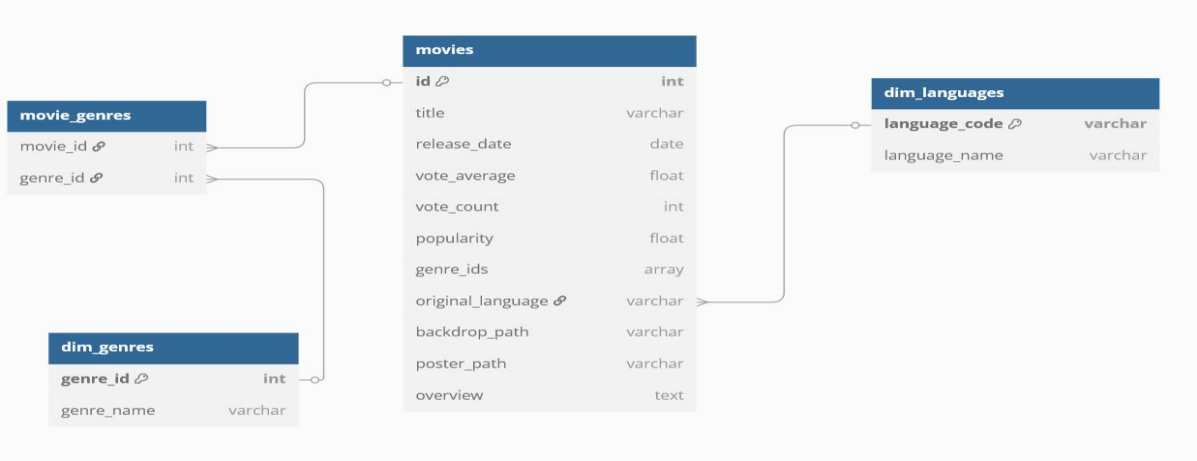
Pipeline ELT



- EXTRACT : appels API TMDB (films, genres)
- LOAD : fichiers CSV envoyés sur GCS
- TRANSFORM : requêtes SQL sur BigQuery (nettoyage, enrichissement, suppression doublons)
- STRUCTURE : création du schéma en étoile
- Fichier orchestré via Cloud Function

Schéma en étoile

- Table de faits : `movies`
- Dimensions : `dim_genres`, `dim_languages`
- Table de relation N\:N : `movie_genres`
- Relations : `movie_id` ↔ `genre_id` et `original_language` ↔ `language_code`



Automatisation

- Pour automatiser notre pipeline, on a utilisé **Cloud Function** pour héberger le script complet en Python, et **Cloud Scheduler** pour le déclencher automatiquement.
- La fonction s'exécute **tous les 1er du mois à 8h**, ce qui permet d'avoir des données toujours fraîches sans intervention manuelle. (on ajoute seulement des données pas encore présente pour optimiser le stockage)
- On peut aussi la lancer à la demande grâce à une URL HTTP.
- Toute l'orchestration se fait dans le cloud, ce qui garantit une exécution fiable, sécurisée et indépendante de notre poste local.

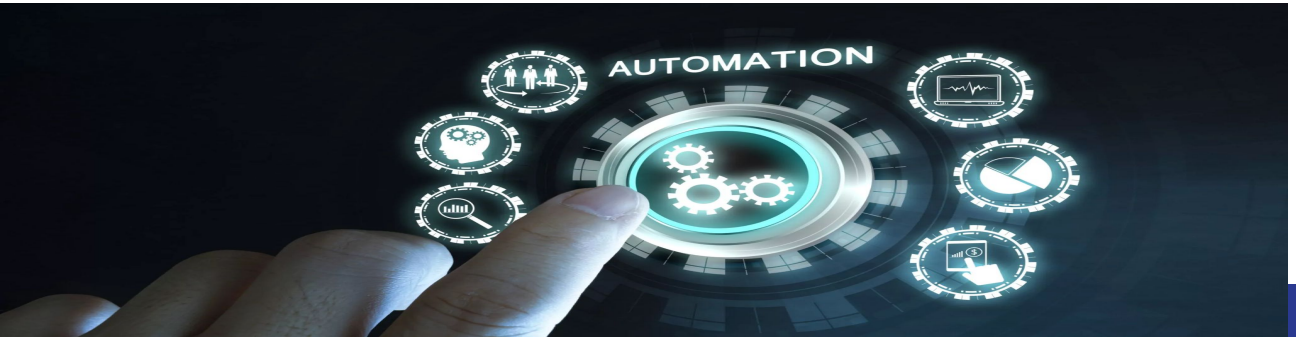
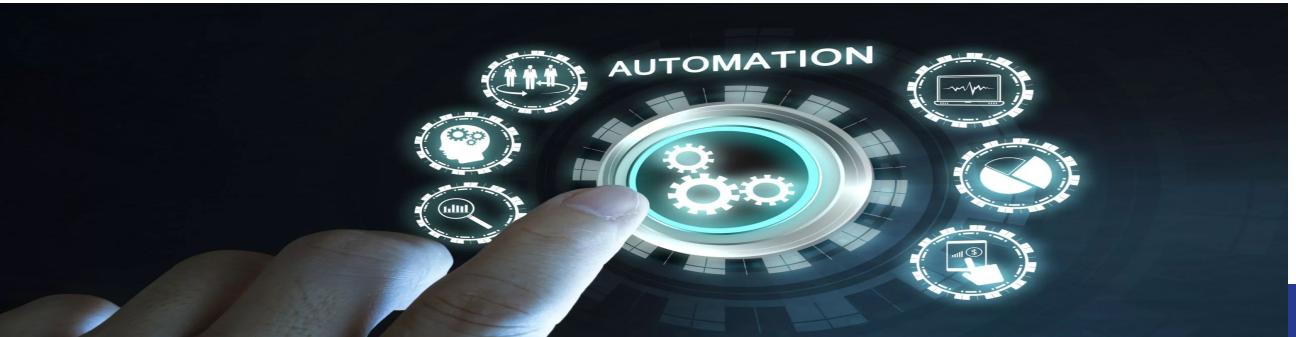


Table de logs

Pour suivre l’exécution du pipeline, nous avons mis en place une **table de log** dans BigQuery.

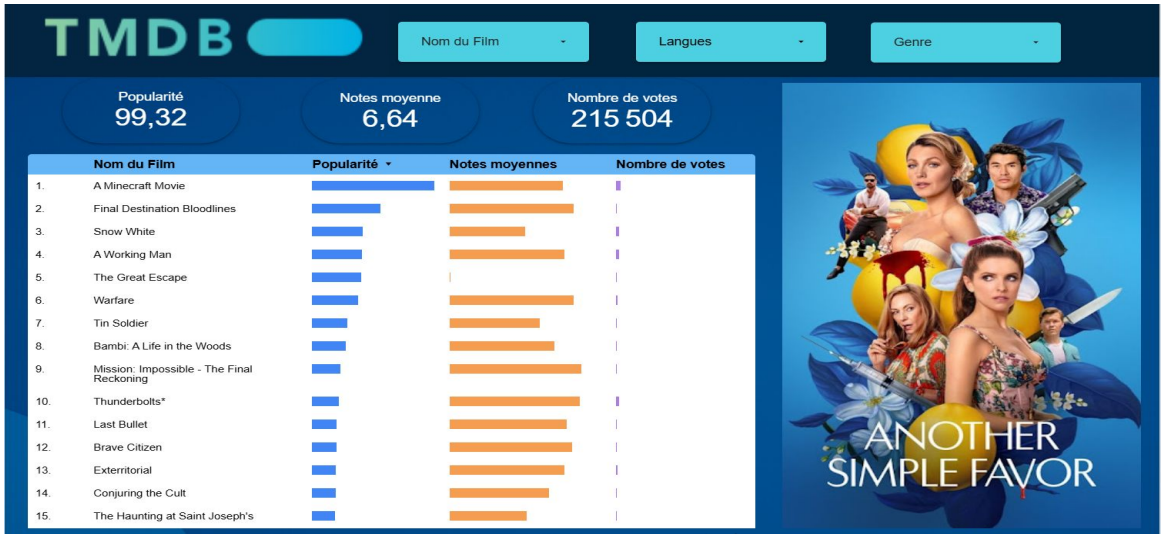
Elle enregistre à chaque passage la date, l’heure, le statut d’exécution, et éventuellement le nombre de lignes traitées.

Ce type de suivi est utile en production pour monitorer la pipeline, détecter les erreurs, et garder une trace des traitements.



Dashboard Looker Studio

- Connexion directe à BigQuery (table `movies` enrichie)
- Jointure avec `dim_languages` pour afficher les langues complètes
- Filtres dynamiques : par film, genre, langue
- Graphiques : top films par vote, par popularité, affiches affichées
- Fond design personnalisé (background stylisé)



Conclusion

- Projet fonctionnel, automatisé, sécurisé
- Pipeline complet de bout en bout
- Dataviz claire, enrichie, exploitable
- Prochaine étape possible : + de pages API , + de KPI dans le dashboard
- Projet réutilisable dans tout contexte GCP/ELT

Lien vers le rapport looker

Lien vers le github