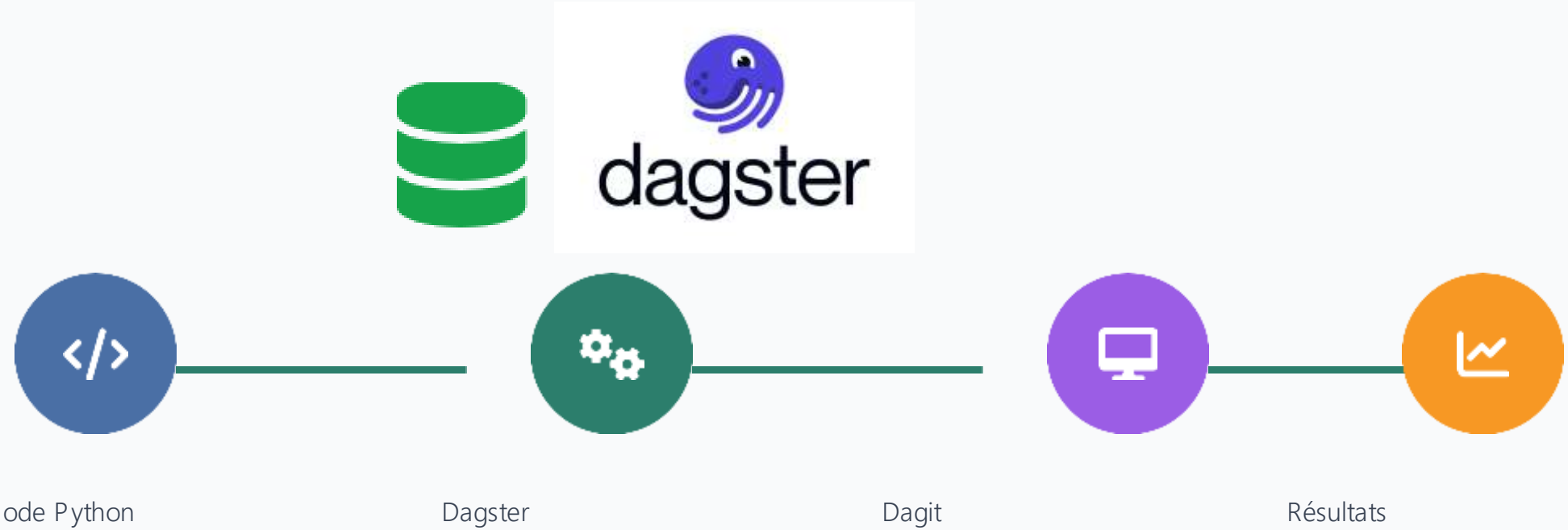
Dagster – Orchestration des pipelines Big Data



Réalisé par :
Hamza Elmourabit

Encadré par :
Pr.Lamia Karim

16 décembre 2025

Contexte Big Data

Croissance rapide des données

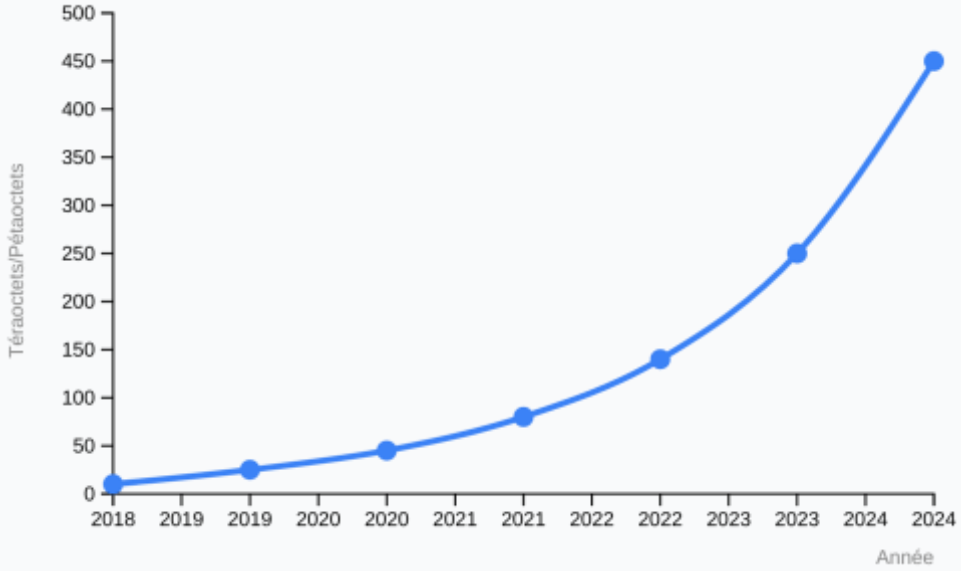
Volume de données générées et collectées augmente de manière exponentielle, rendant leur gestion de plus en plus complexe.

Automatisation des traitements

Nécessité d'automatiser les processus de collecte, de transformation et d'analyse des données pour garantir l'efficacité et la réactivité.

Besoin d'un orchestrateur

Face à la complexité croissante des pipelines de données, un outil d'orchestration est indispensable pour coordonner les différentes étapes.



Croissance exponentielle des données au fil des années

Qu'est-ce que Dagster ?

Dagster est un outil d'orchestration de pipelines basé sur Python

Gère des pipelines de données complexes de manière fiable, du développement à la production.

Définition de pipeline

Les pipelines sont définis en Python, utilisant toute la puissance du langage pour orchestrer les données.

Gestion des dépendances

Dagster gère automatiquement les dépendances entre les composants du pipeline.

Fiabilité

Conçu pour la robustesse avec des mécanismes de gestion des erreurs et de re-tentative.

Centre de données

Dagster se concentre sur la gestion des "assets" plutôt que sur les tâches individuelles, offrant une approche centrée sur les données.

Pourquoi Dagster ?

Fiabilité

Conçu pour la robustesse avec des mécanismes de gestion des dépendances, de re-tentative automatique en cas d'échec et de gestion des erreurs. L'approche centrée sur les "assets" garantit que les données sont à jour et cohérentes.

Observabilité

Fournit une visibilité en temps réel sur l'état des workflows, permettant d'identifier rapidement les goulots d'étranglement. Intègre un système de journalisation robuste, un catalogue de données intégré, la lignée des assets et des contrôles de qualité.

Modularité

Encourage une architecture modulaire et réutilisable. Les pipelines peuvent être décomposés en tâches ou composants plus petits et réutilisables, ce qui favorise l'efficacité et la cohérence. Cette modularité accélère le développement et simplifie la maintenance.

Facilité d'intégration

Bibliothèque d'intégrations avec les outils de données les plus populaires, y compris les principaux fournisseurs de cloud (AWS, GCP, Azure), les outils ETL (Fivetran, Airbyte, dlt, Sling) et les outils de BI (Tableau, Power BI, Looker, Sigma).

Concepts clés

Assets

Objets de données produits, consommés ou mis à jour par vos pipelines. Peuvent être des tables, des fichiers, des modèles ML ou des rapports.

→ Gestion de la lignée des données

Jobs

Représente une exécution de pipeline. Unité principale d'exécution et de surveillance des définitions d'actifs dans Dagster. Peut cibler des actifs spécifiques.

→ Coordination des dépendances

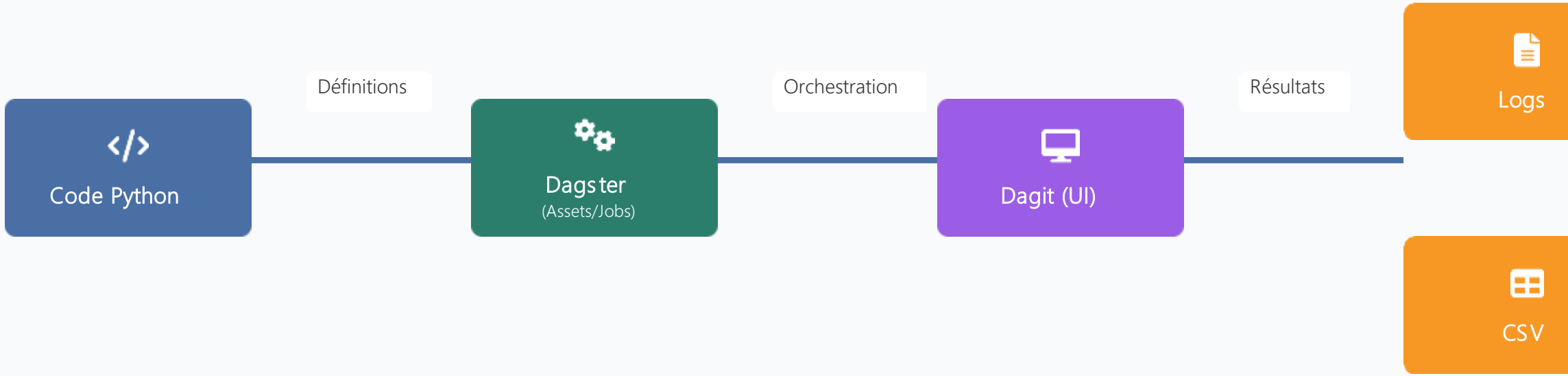
Dagit

Interface utilisateur web de Dagster. fournit une vue interactive des pipelines, permettant de visualiser les actifs, de surveiller les exécutions et de gérer les logs.

→ Observabilité et gestion

Diagramme de workflow

Architecture Dagster



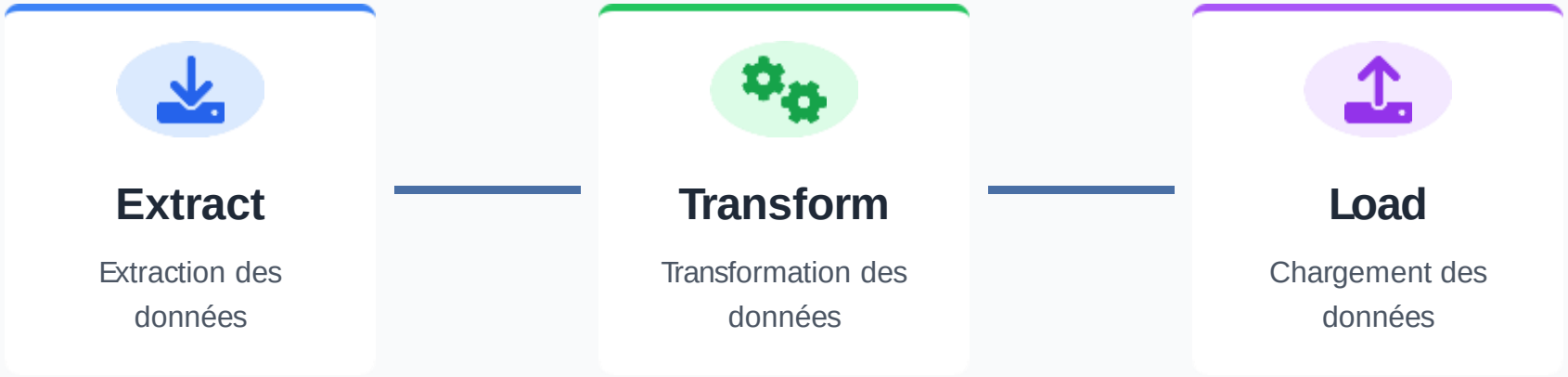
Code Python
Où les ingénieurs de données définissent leurs Assets et Jobs.

Dagit UI
Interface pour visualiser et gérer les pipelines.

Dagster Engine
Gère les dépendances et l'exécution des pipelines.

Artefacts
Résultats : Logs et fichiers de données.

Exemple de pipeline



Extract (Extraction)

Collecte les données brutes à partir de diverses sources : bases de données, fichiers CSV, APIs, systèmes de streaming, etc.

Transform (Transformation)

Nettoyage, enrichissement, agrégation et restructurazione des données pour répondre aux exigences du modèle cible.

Load (Chargement)

Chargement des données transformées dans un système de destination : entrepôt de données, lac de données ou base de données analytique.

Code du projet

Extrait de code Python illustrant l'approche **asset-centric** de Dagster.

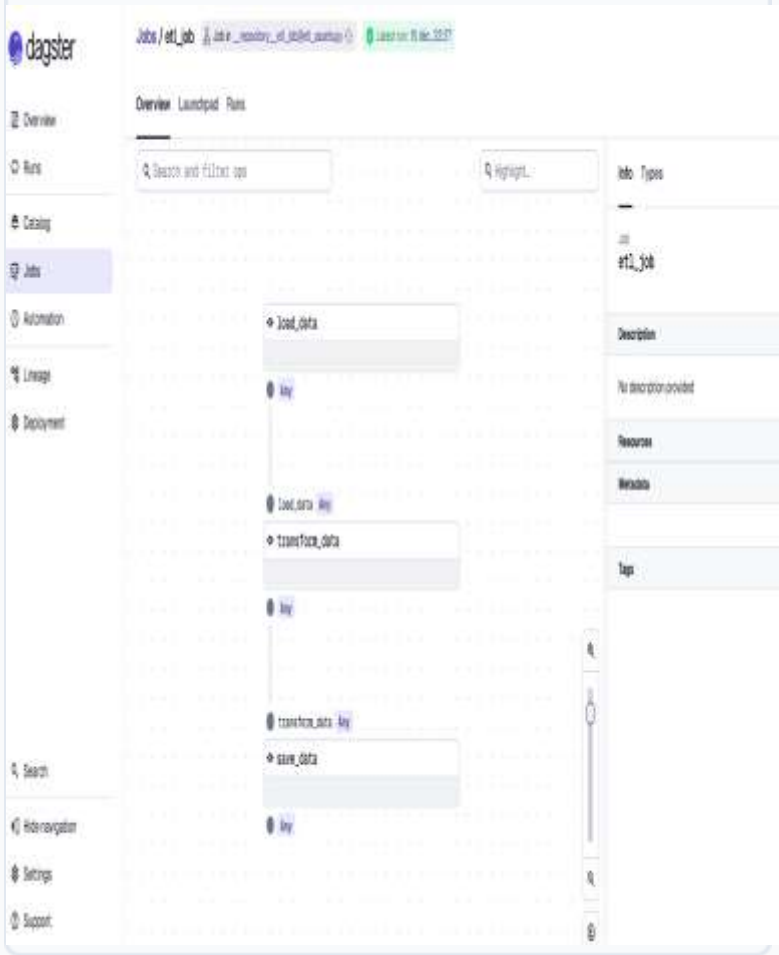
Chaque fonction décorée par **@asset** représente un objet de données persistant.

Les dépendances sont explicitement définies via **deps**.

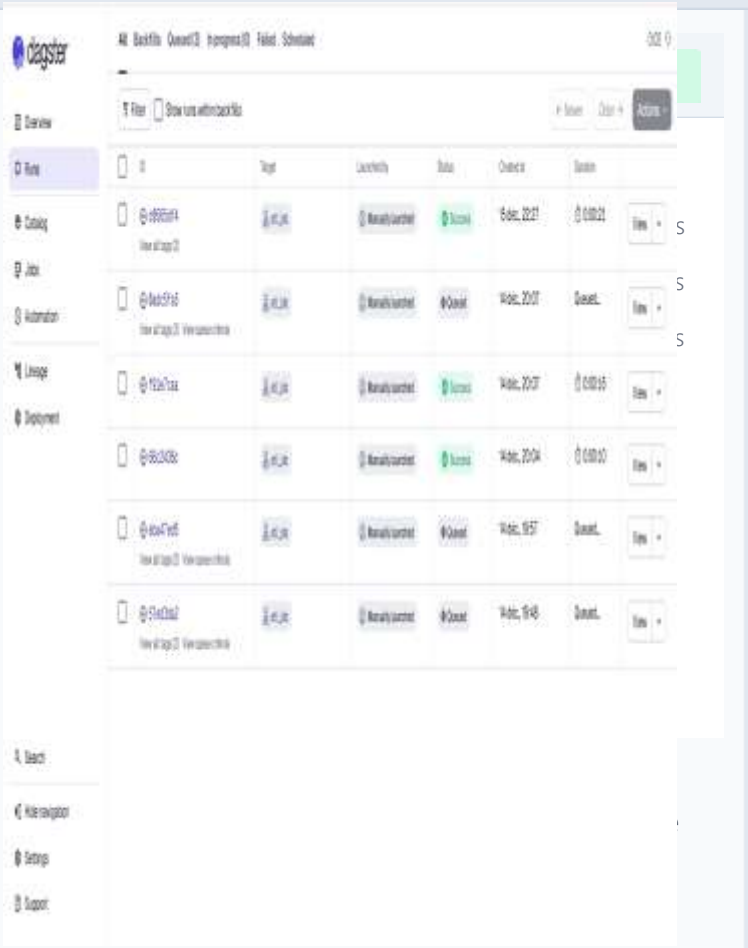
```python
import pandas as pd
from dagster import asset


@asset
def load_data():
    df = pd.read_csv("input.csv")
    print("Data loaded:\n", df.head())
    return df


@asset
def transform_data(load_data):
    df = load_data.copy()
    df["total"] = df["quantity"] * df["price"]
    print("Data transformed:\n", df.head())
    return df


@asset
def save_data(transform_data):
    df = transform_data
    df.to_csv("output.csv", index=False)
    print("Data saved to output.csv")
    return "Success"
```

Démonstration

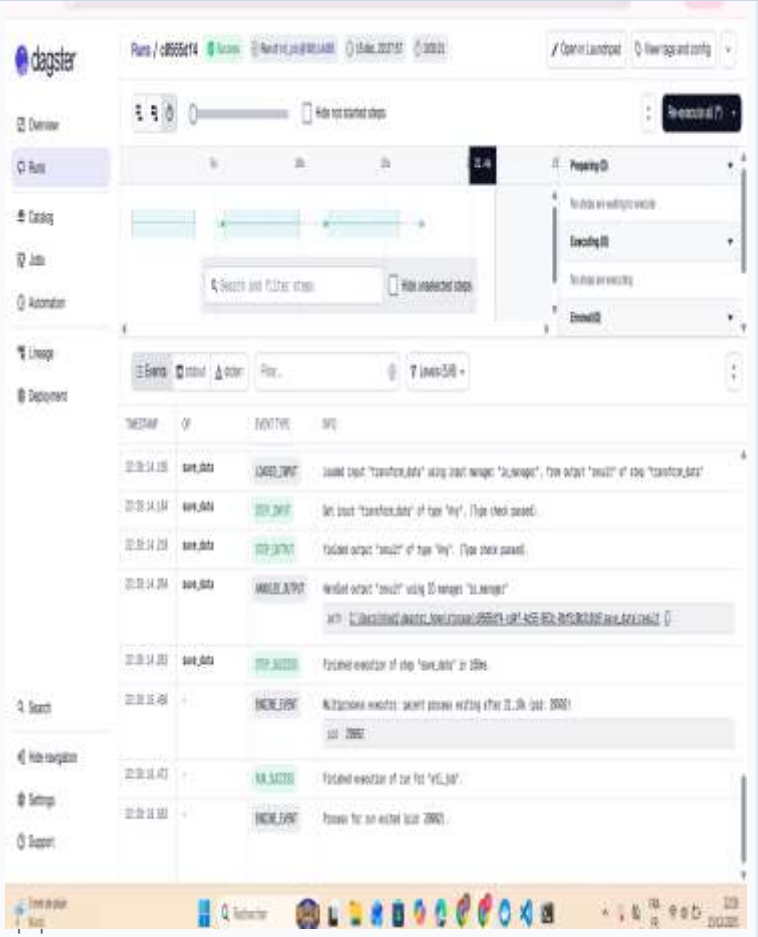
Vue du pipeline (Lineage)



Exécution réussie (Run Success)



Logs d'exécution



Cas d'usage Big Data

Orchestration de jobs Spark

Dagster peut orchestrer des traitements de données massives exécutés sur Apache Spark, gérant les dépendances, surveillant les exécutions et visualisant le lignage des données produites.

Intégration avec DBT

Dagster s'intègre parfaitement avec dbt (data build tool) pour la transformation de données, en modélisant les modèles dbt comme des "assets" et en assurant une gestion cohérente du cycle de vie des données.

Préparation de données ML

Dagster facilite la création de pipelines de préparation de données pour les modèles de Machine Learning, en gérant l'extraction, la transformation et le chargement des données nécessaires à l'entraînement et à l'inférence.

Avantages & Limites

Dagster présente des avantages significatifs pour l'orchestration des pipelines Big Data, mais présente également certaines limites par rapport à des outils plus établis.

Avantages

Fiabilité
Mécanismes de gestion des dépendances, de re-tentative automatique en cas d'échec et de gestion des erreurs.

Observabilité
Visibilité en temps réel sur l'état des workflows, journalisation robuste et lignée des assets.

Modularité
Architecture modulaire et réutilisable qui accélère le développement et simplifie la maintenance.

Facilité d'intégration
Vaste bibliothèque d'intégrations avec les outils de données les plus populaires et les fournisseurs de cloud.

Limites

Complexité initiale
Apprentissage requis pour configurer correctement les pipelines pour des cas d'usage complexes.

Temps d'exécution
Peut augmenter le temps total d'exécution des pipelines par rapport à des solutions plus simples.

Surcharge de concept
Les concepts de "assets" et "jobs" peuvent sembler superflus pour des pipelines simples.

Dépendances externes
Nécessite une gestion proactive des versions pour maintenir la compatibilité avec les outils intégrés.

Conclusion

Dagster est un outil clé pour l'orchestration de pipelines Big Data fiables et automatisés

Répondant aux exigences des plateformes de données modernes

Fiabilité

Gestion robuste des dépendances et des erreurs

Observabilité

Vue d'ensemble intuitive de vos workflows de données

Modularité

Architecture réutilisable qui accélère le développement

Intégration

Compatibilité avec vos outils Big Data existants

Merci pour votre attention