Atelier 01 : La Pratique de l'ETL avec Apache Airflow

1. Objectif :

L'objectif de cet atelier est de vous familiariser avec les concepts de l'ETL (Extract, Transform, Load) en utilisant Apache Airflow, une plateforme open-source pour développer, planifier et surveiller des workflows par lots.

2. Introduction :

Les données sont essentielles à toute application et sont utilisées dans la conception d'un pipeline efficace pour la livraison et la gestion de l'information au sein d'une organisation. En général, on définit un pipeline de données lorsqu'il est nécessaire de traiter les données tout au long de leur cycle de vie. Le pipeline peut commencer là où les données sont générées et stockées dans n'importe quel format. Le pipeline peut se terminer par l'analyse des données, leur utilisation en tant qu'informations commerciales, leur stockage dans un entrepôt de données ou leur traitement dans un modèle d'apprentissage automatique (figure 1).

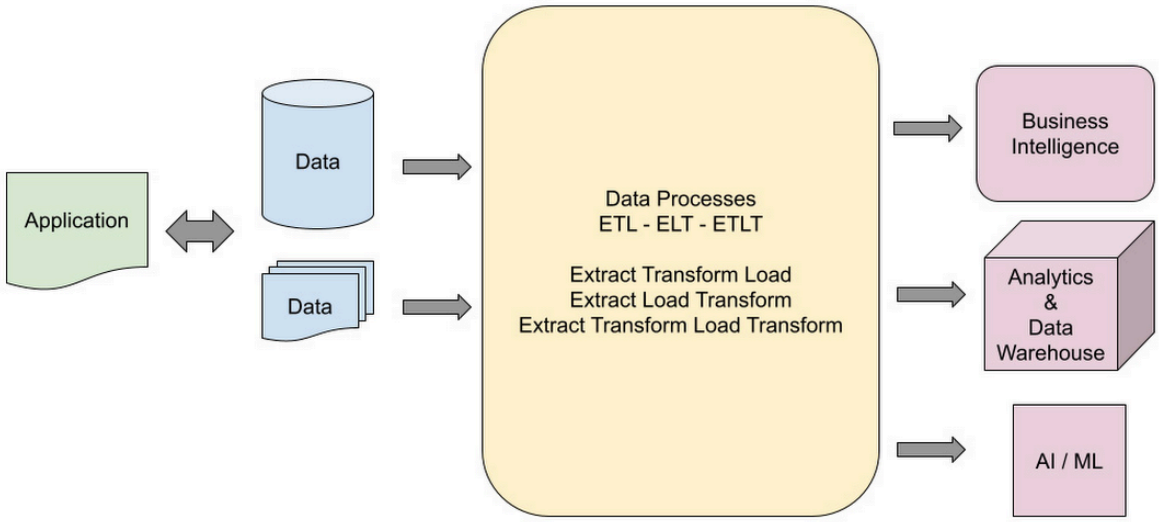


Figure 1 : le processus ETL

Les données sont extraites, traitées et transformées en plusieurs étapes en fonction des besoins des systèmes en aval. Les étapes de traitement et de transformation sont définies dans un pipeline de données. Selon les exigences, les pipelines peuvent être aussi simples qu'une seule étape ou aussi complexes que plusieurs étapes de transformation et de traitement.

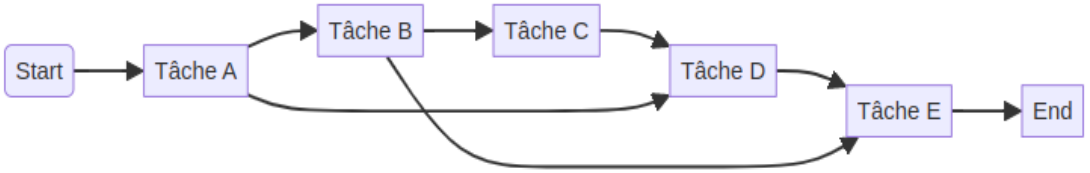
Les processus ETL garantissent que les données sont préparées et disponibles pour les workflows de traitement **en temps réel** et **par lots**. Le traitement en temps réel gère les flux de données au fur et à mesure qu'ils se produisent, tandis que le traitement par lots gère les données à des intervalles prédéterminés et de grande taille.

Aspect	Traitement par Lots	Traitement en Temps Réel
Latence	Haute (retardée)	Basse (instantanée)
Complexité	Moins complexe	Plus complexe
Coût	Moins coûteux	Plus coûteux
Volume de Données	Large volume traité à intervalles	Données continues et flux de données
Réactivité	Retardée	Immédiate
Cas d'Utilisation	Rapports de fin de journée, agrégations	Surveillance en temps réel, alertes

3. Concepts Clés d'Apache Airflow :

Apache Airflow repose sur plusieurs concepts fondamentaux qui définissent son fonctionnement et son approche en matière de gestion de workflows. Comprendre ces concepts est essentiel pour tirer le meilleur parti d'Airflow.

- Directed Acyclic Graphs (DAGs) : Le cœur d'Airflow est le DAG, un graphe acyclique dirigé. Un DAG dans Airflow représente l'ensemble des tâches à exécuter, organisées de manière à refléter leurs dépendances et leur ordre d'exécution. Chaque nœud du DAG représente une tâche et les arêtes indiquent les dépendances entre ces tâches. Cette structure assure qu'une tâche ne s'exécute que lorsque ses dépendances sont satisfaites, apportant clarté et efficacité dans la gestion des workflows.



- Les Operators sont les blocs de construction d'un DAG. Ils définissent ce qu'une tâche doit faire. Airflow propose une variété d'Operators intégrés pour des tâches courantes comme exécuter une commande Python, un script Bash, ou interagir avec des bases de données et des systèmes de fichiers. La personnalisation et la création d'Operators spécifiques permettent d'étendre les fonctionnalités d'Airflow pour répondre aux besoins spécifiques d'un workflow.
- Une **tâche** (Task) dans **Airflow** est une instance d'un Operator. Lorsqu'un DAG est exécuté, **Airflow** crée des instances de tâches (Task Instances) pour chaque tâche définie dans le DAG. Ces instances de tâches sont les éléments qui sont réellement exécutés par les **workers** d'**Airflow**. Le suivi de l'état de chaque instance de tâche (comme réussi, échoué, en cours etc.) est essentiel pour la gestion et le dépannage des workflows.
- **Workflow Scheduling** : Airflow n'est pas seulement un système d'exécution de tâches, mais aussi un planificateur de tâches. Il permet de définir des plannings complexes pour l'exécution des workflows, en utilisant des expressions cron ou des intervalles personnalisés. Cette fonctionnalité rend

Airflow particulièrement puissant pour les ETL, où les données doivent être traitées et déplacées à des intervalles réguliers.

4. Les « workflows en tant que code » de Airflow :

- Dynamique : les pipelines Airflow sont configurés sous forme de code Python, ce qui permet la génération dynamique de pipelines.
- Extensible : le framework Airflow® contient des opérateurs permettant de se connecter à de nombreuses technologies. Tous les composants Airflow sont extensibles pour s'adapter facilement à votre environnement.
- Flexible : le paramétrage du workflow est intégré en exploitant le moteur de création de modèles Jinja.

5. Installation d'Airflow :

- Pour installer Apache Airflow en utilisant pip, suivez les étapes ci-dessous. Notez qu'Apache Airflow nécessite un environnement Python et quelques dépendances spécifiques. Nous allons installer Airflow dans un environnement virtuel pour éviter les conflits avec d'autres packages Python. Assurez-vous d'avoir Python 3.6+ et pip installés sur votre système:

- Il est recommandé de créer un environnement virtuel pour l'installation d'Airflow. Cela permet de gérer les dépendances de manière isolée :

  (virtualenv airflow_env, source airflow_env/bin/activate)

- Install : pip install apache-airflow
- définir la variable d'environnement AIRFLOW_HOME :

  export AIRFLOW_HOME=${PWD}

- créer le fichier de configuration :

  airflow config list --defaults > "${AIRFLOW_HOME}/airflow.cfg"

- Dans le fichier décommenter les lignes suivantes :

```
7 dags_folder = /home/msakkari/dags

124 load_examples = True

638 base_log_folder = /home/msakkari/logs
```

- mkdir dags/logs

6. Création de la base de données

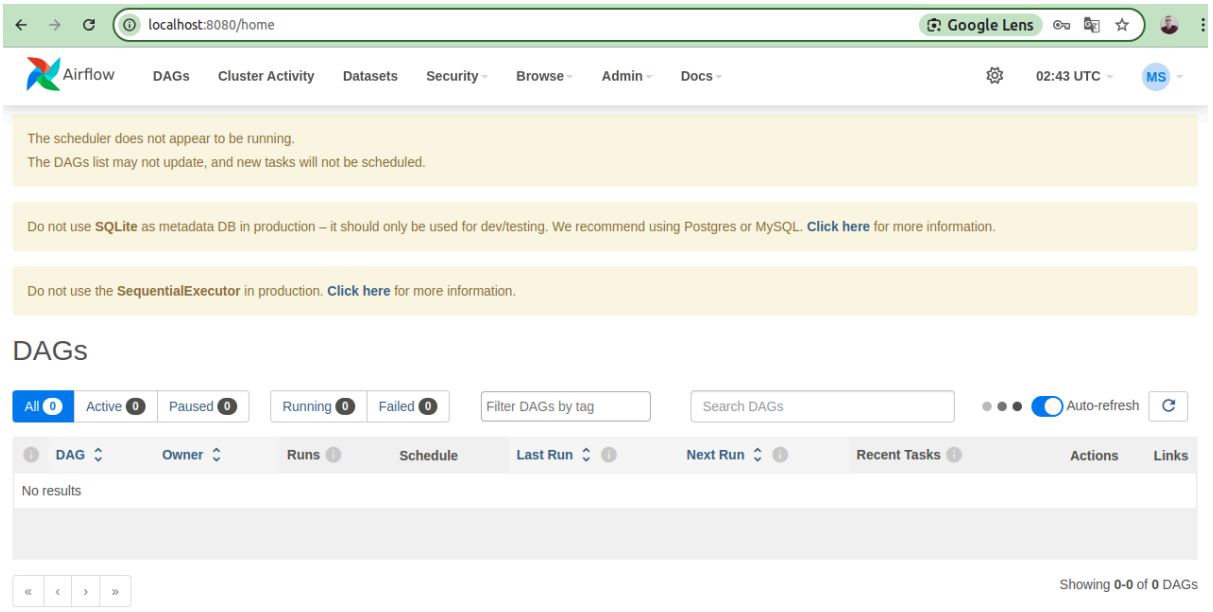
- migrer la base de données : airflow db migrate
- créer un utilisateur pour accéder à l'interface Web d'Airflow

  airflow users create --username admin --password admin --firstname FIRST_NAME --lastname LAST_NAME --role Admin --email EMAIL_ADDRESS

  example : airflow users create --username admin --password admin --firstname med --lastname sakkari --role Admin --email sakkari.tn@gmail.com

7. Interface Utilisateur d'Airflow

- démarrer le serveur Web et le scheduler avec : airflow webserver
- L'interface utilisateur web d'Apache Airflow offre une visibilité et un contrôle complets sur les workflows. Dès que vous avez configuré et lancé le serveur web Airflow, vous pouvez accéder à cette interface via votre navigateur à l'adresse http://localhost:8080. Connectez-vous avec le user admin créé précédemment.
- Le tableau de bord principal affiche une liste de tous les DAGs disponibles. Pour chaque DAG, vous pouvez voir son état (actif ou non), la fréquence d'exécution, la dernière exécution et d'autres détails utiles. C'est le point de départ pour surveiller et gérer vos workflows.



8. Création d'un Workflow Simple :

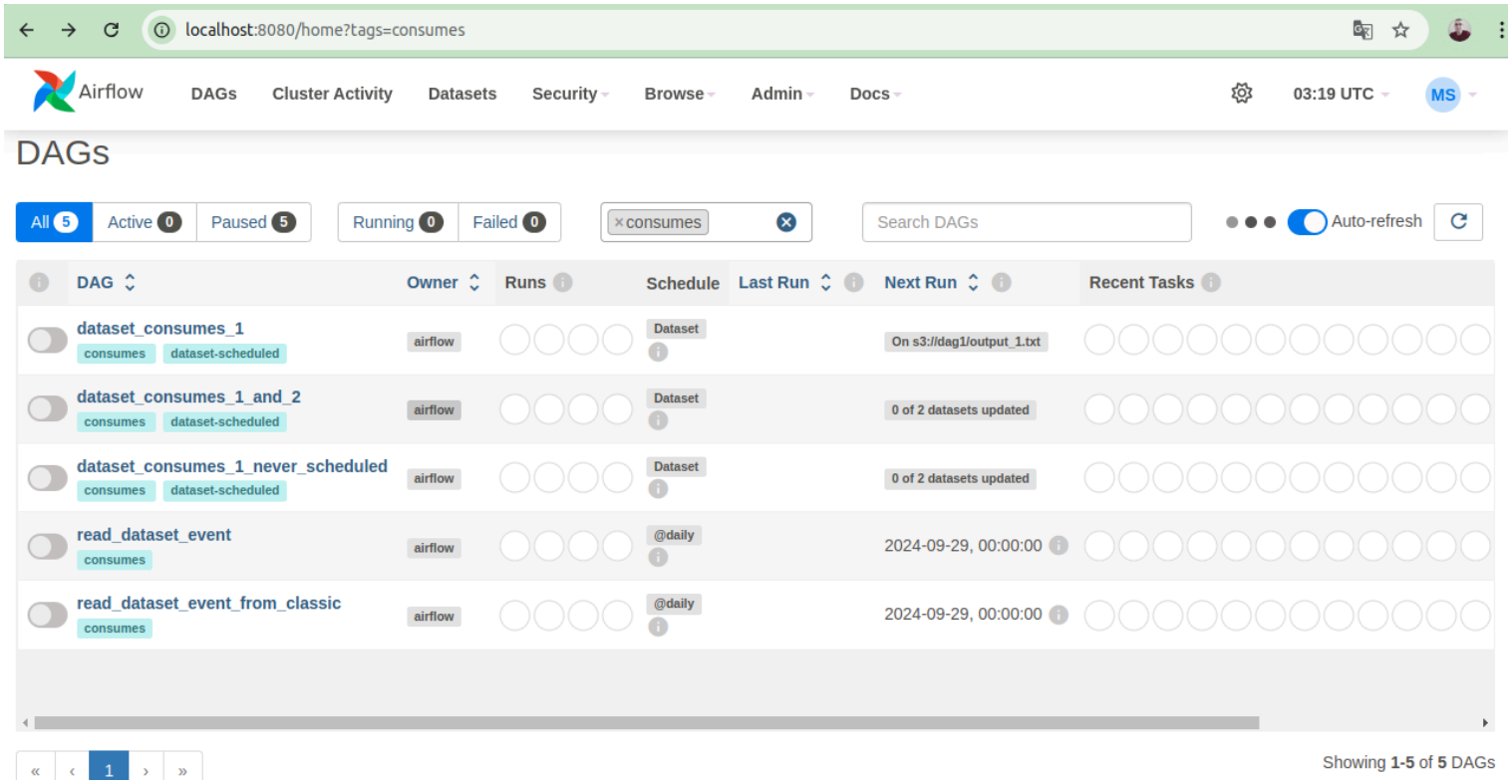
Supposons que nous voulions créer un workflow quotidien qui exécute une tâche de script Python. Ce workflow peut être défini comme suit :

A. Définition du DAG : Tout d'abord, nous devons définir le DAG dans Airflow. Créez un fichier Python dans le répertoire dags d'Airflow. Nommons ce fichier data5_atelier01_exemple1_dag.py
B. Importation des Modules
C. Définition de la Fonction de la Tâche
D. Paramètres du DAG
E. Création de l'Operator

instance connection | airflow.cfg | *data5_atelier01_exemple1_dag.py

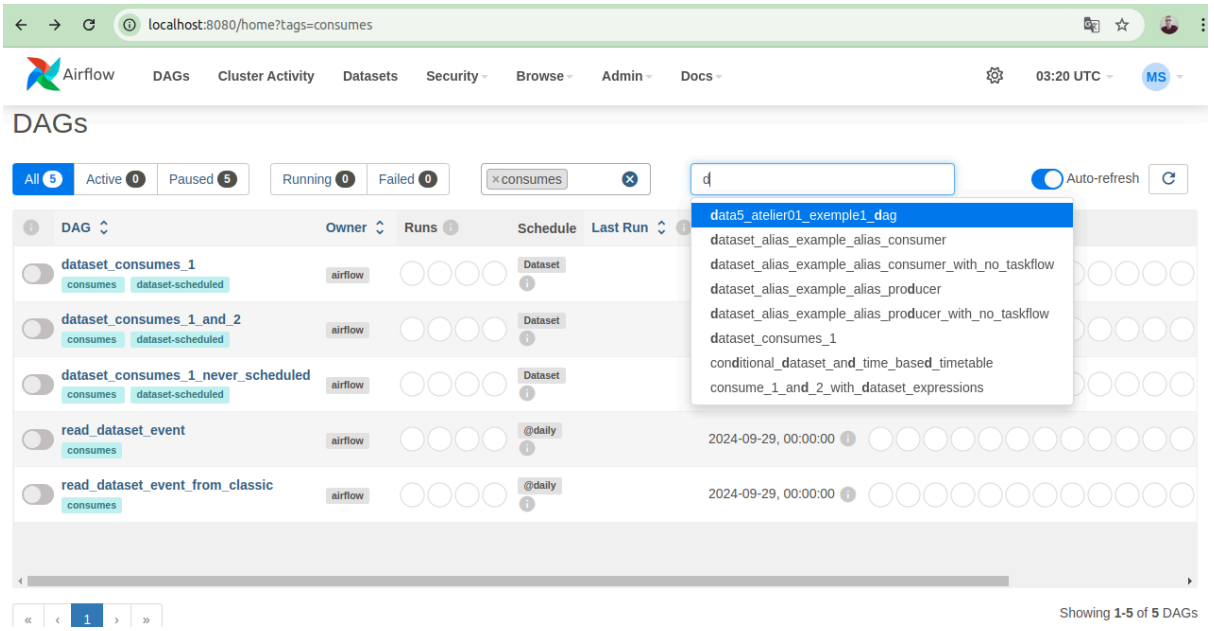
```python
# B. Importation des Modules
from datetime import timedelta
from airflow import DAG
from airflow.operators.python_operator import PythonOperator
from airflow.utils.dates import days_ago

# C. Définition de la Fonction de la Tâche : Définissez une fonction Python pour la tâche que vous voulez exécuter. Par exemple :
def my_simple_task():
    print("Exécution de ma tâche simple")

# D. Paramètres du DAG : Définissez les paramètres de votre DAG, comme l'identifiant du DAG, le calendrier et les paramètres par défaut :
default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'start_date': days_ago(1),
    'email_on_failure': False,
    'email_on_retry': False,
    'retries': 1,
    'retry_delay': timedelta(minutes=5),
}

dag = DAG(
    'data5_atelier01_exemple1_dag',
    default_args=default_args,
    description='Un exemple de DAG simple',
    schedule_interval=timedelta(days=1),
)
# E. Création de l'Operator : Créez un operator pour exécuter la fonction de tâche. Dans cet exemple, nous utilisons un PythonOperator :
t1 = PythonOperator(
    task_id='simple_task',
    python_callable=my_simple_task,
    dag=dag,
)
```

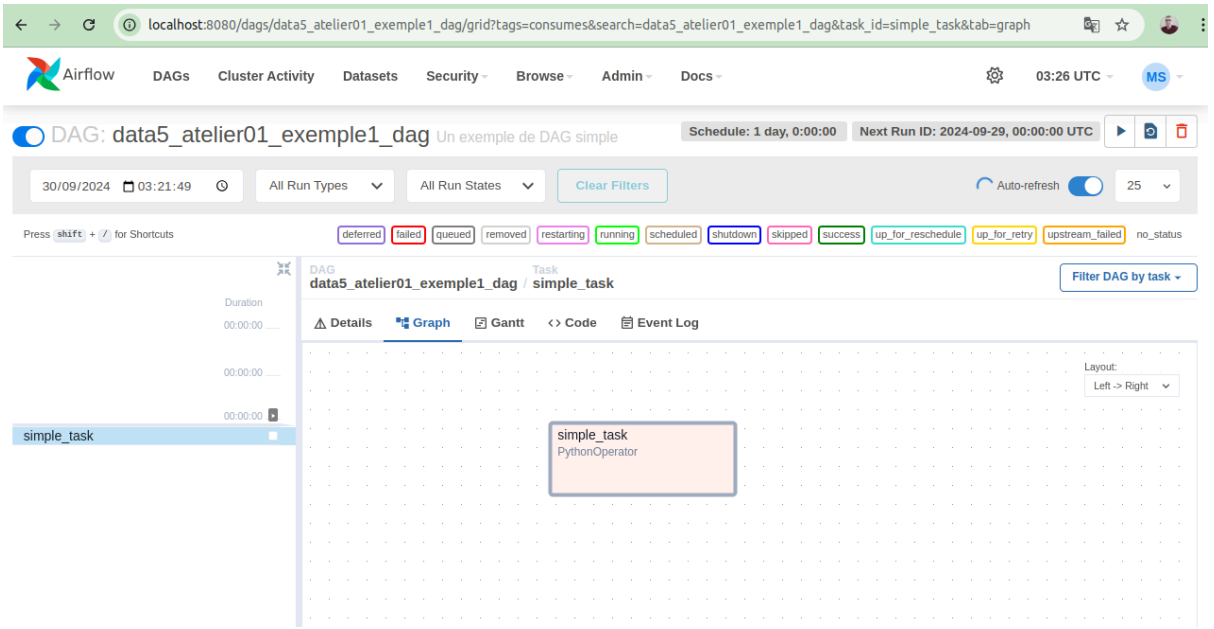
F. **Démarrage du scheduler :** Lancez le scheduler d'**Airflow**, qui surveille les DAGs et exécute les tâches : Airflow scheduler,



G. **Exécution du Workflow** : Une fois le fichier enregistré dans le répertoire `dags`, Airflow détectera automatiquement le nouveau DAG. Vous pouvez alors l'activer et le surveiller via l'interface utilisateur web:

```
airflow webserver
```





Après avoir exécuté votre premier workflow dans Apache Airflow, explorez l'interface web et répondez aux questions suivantes pour mieux comprendre les différentes fonctionnalités :

1. Tableau de Bord (Dashboard) :
    - Que pouvez-vous voir sur le tableau de bord d'Airflow ? (Décrivez chaque onglet avec une capture d'écran)
    - Comment pouvez-vous accéder à la liste de tous les DAGs disponibles ?
2. Visualisation des DAGs :
    - Lorsque vous sélectionnez un DAG, quelles informations sont affichées concernant ses tâches ?
    - Comment pouvez-vous identifier l'état d'exécution de chaque tâche dans le DAG ?
3. Historique des Exécutions :
    - Où pouvez-vous trouver l'historique des exécutions d'un DAG spécifique ?
    - Quelles informations sont fournies pour chaque exécution dans l'historique ?
4. Accès aux Journaux (Logs) :
    - Comment pouvez-vous accéder aux journaux d'une tâche spécifique ?
    - Pourquoi les journaux sont-ils importants pour le suivi et le débogage des workflows ?
5. Configuration et Paramètres :
    - Quels types de paramètres ou configurations pouvez-vous ajuster via l'interface web ?
    - Comment ces paramètres peuvent-ils affecter l'exécution des workflows ?

Bon travail