مــكيــن
Makeen

سلطنة عُمان
وزارة النقل والاتصالات وتقنية المعلومات
Sultanate of Oman
Ministry of Transport, Communications and Information Technology

Code Academy

Apache Airflow Technology: ETL Pipeline Implementation

Trainee: Raunaq Al Harthi

Instructor: Kulthoom Shoukat

Table of Contents

Introduction 3

What is Airflow? 3

Project Objective 4

Tools and Technologies Used 4

ETL VS ELT 4

Simplified View of Apache Airflow 5

Features of Apache Airflow 5

Principles of Apache Airflow 5

Whay is DAG in Airflow? 6

Linking Apache Airflow with MySQL 6

Conclusion: 7

Figure 1 3
Figure 2 4
Figure 3 4
Figure 4 5
Figure 5 6

Introduction

Apache Airflow is an open-source platform used to programmatically author, schedule, and monitor workflows. This document outlines the implementation of an **ETL (Extract, Transform, Load)** pipeline using Airflow, focusing on sales data processing. The pipeline automates data extraction from a CSV file, transformation (cleaning, encoding, scaling), and loading into a MySQL database.

What is Airflow?

Apache Airflow is an open-source workflow automation platform that orchestrates complex data pipelines using Python. It structures workflows as **Directed Acyclic Graphs (DAGs)**, where tasks and dependencies are defined programmatically. Key features include:

- **Scheduling & Automation**: Run workflows at set intervals (e.g., hourly/daily).
- **Extensibility**: Custom operators integrate with databases, APIs, and cloud services.
- **Scalability**: Distributes tasks via executors like Celery or Kubernetes.
- **Monitoring**: Built-in UI for tracking pipeline status and debugging.

Airflow is widely used for ETL, ML pipelines, and data processing automation.

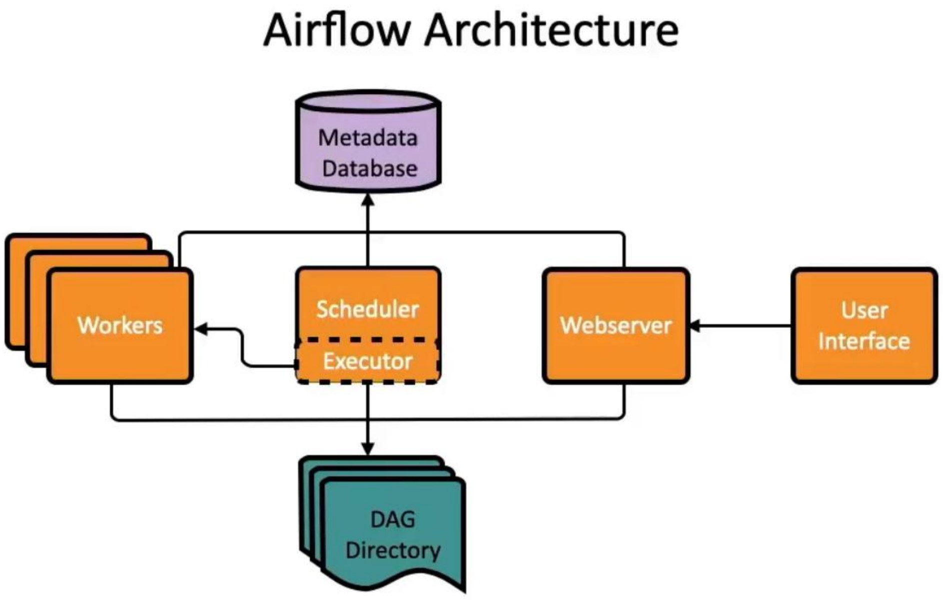


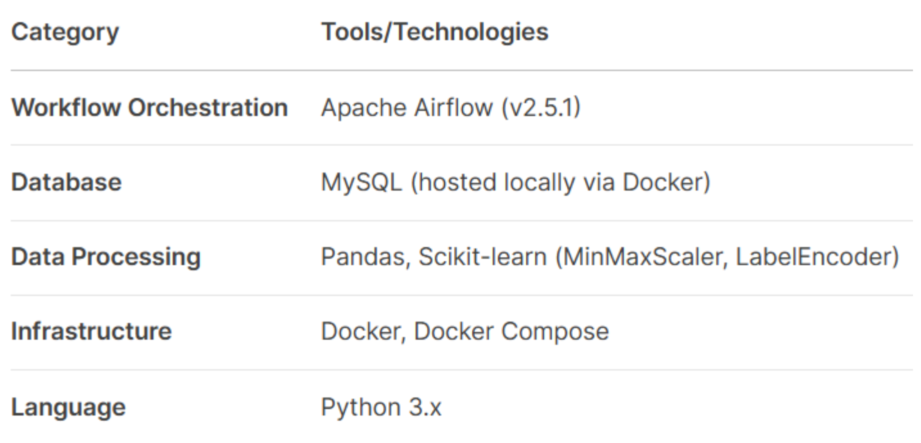
Figure 1

Project Objective

The goal of this project is to:

- ✓ **Extract** sales data from a CSV file.
- ✓ **Transform** the data (handle missing values, encode categorical features, normalize numerical data).
- ✓ **Load** the processed data into a MySQL database (sales3.sales_data).
- ✓ **Automate** the pipeline using Airflow's PythonOperator for seamless ETL execution.

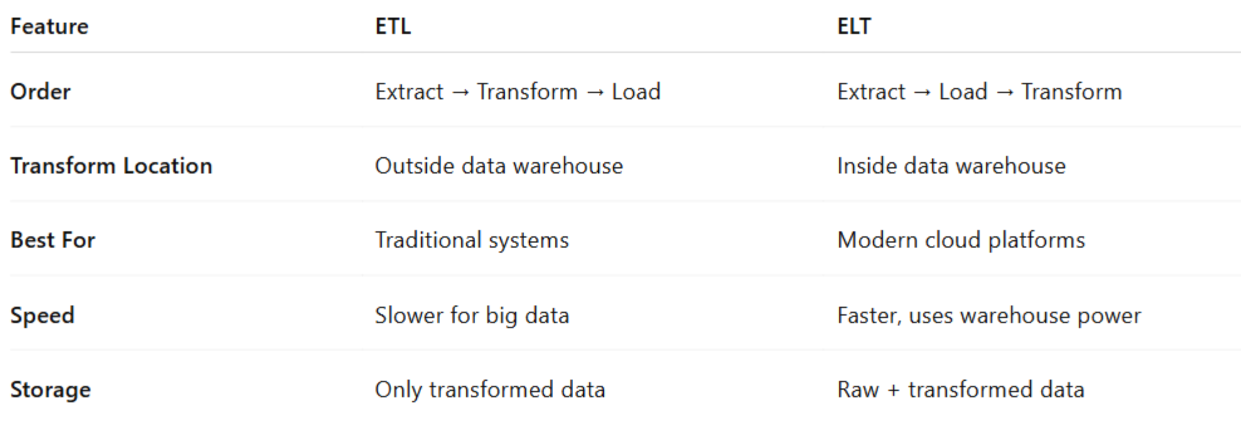
Tools and Technologies Used



Category	Tools/Technologies
Workflow Orchestration	Apache Airflow (v2.5.1)
Database	MySQL (hosted locally via Docker)
Data Processing	Pandas, Scikit-learn (MinMaxScaler, LabelEncoder)
Infrastructure	Docker, Docker Compose
Language	Python 3.x

Figure 2

ETL VS ELT



Feature	ETL	ELT
Order	Extract → Transform → Load	Extract → Load → Transform
Transform Location	Outside data warehouse	Inside data warehouse
Best For	Traditional systems	Modern cloud platforms
Speed	Slower for big data	Faster, uses warehouse power
Storage	Only transformed data	Raw + transformed data

Figure 3

Simplified View of Apache Airflow

Apache Airflow has 3 main components:

- Scheduler: Triggers tasks based on time or conditions
- Executor: Runs the tasks (can be local, Celery, Kubernetes, etc.)
- Web UI: Monitor and manage your workflows visually.

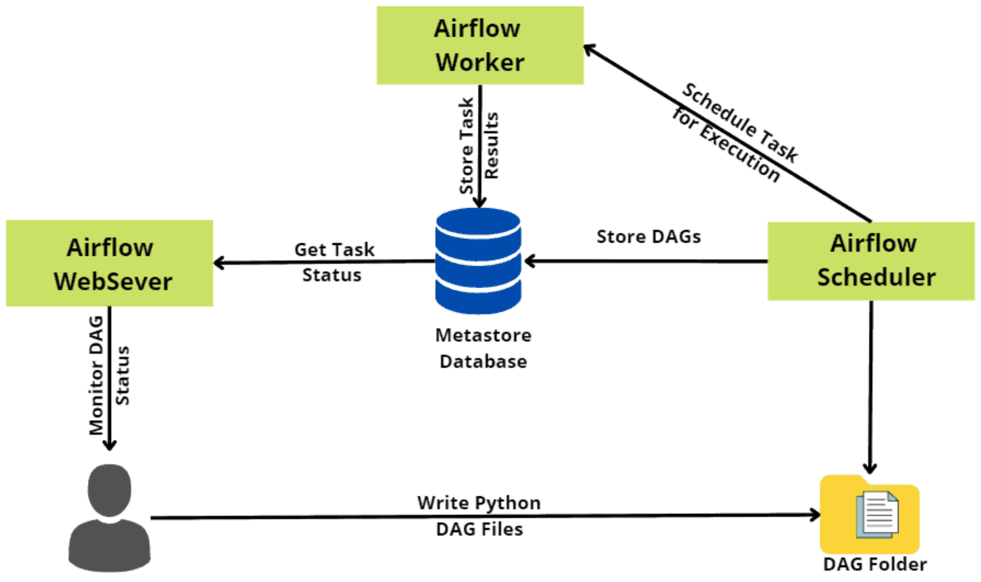


Figure 4

Features of Apache Airflow

- Dynamic pipeline generation using Python
- Web UI for managing workflows
- Scheduler and monitoring tools
- Plug-in support for various services (e.g., MySQL, AWS, etc.)
- Task dependencies and retries

Principles of Apache Airflow

1. Dynamic: Workflows are defined as Python code
2. Scalable: Supports distributed execution
3. Extensible: Easy to write plugins and operators
4. Elegant: Simple yet powerful user interface

Whay is DAG in Airflow?

In **Apache Airflow**, a **DAG** (Directed Acyclic Graph) is a collection of **tasks** organized in a **specific order** to run a workflow.

In Simple Terms:

A DAG is like a **recipe**:

- Steps (tasks) must be done in a certain order.
- Airflow uses the DAG to **schedule, run, and monitor** these steps.

Linking Apache Airflow with MySQL

To connect Airflow with MySQL:

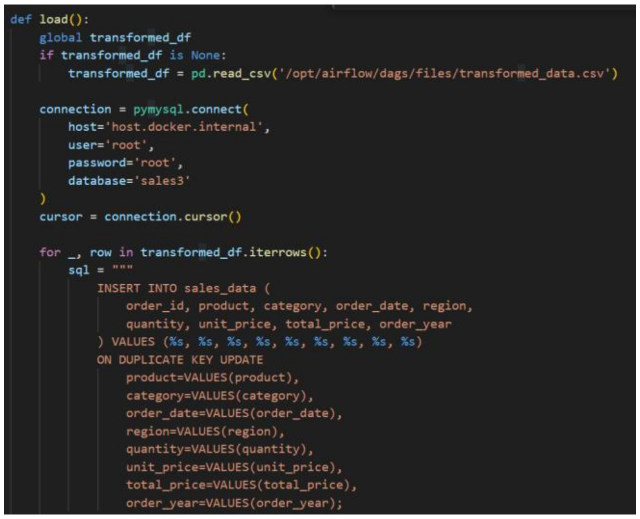
1. Install MySQL client inside the Docker container:

- pip install apache-airflow-providers-mysql

2. Set up MySQL connection in the Airflow UI:

- Go to Admin > Connections > Create - - -
- Connection ID: mysql_conn
- Connection Type: MySQL
- Host, Schema, User, Password, Port

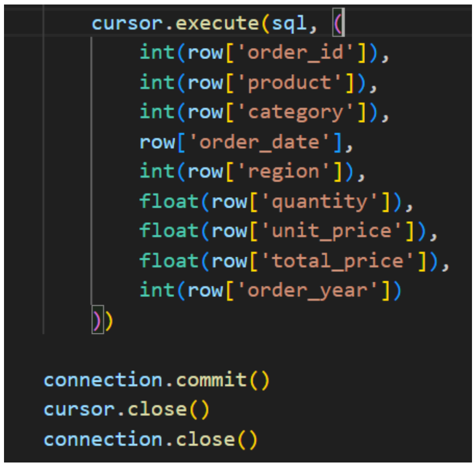
3. Use the connection in a task:



```python
def load():
    global transformed_df
    if transformed_df is None:
        transformed_df = pd.read_csv('/opt/airflow/dags/files/transformed_data.csv')

    connection = pymysql.connect(
        host='host.docker.internal',
        user='root',
        password='root',
        database='sales3'
    )
    cursor = connection.cursor()

    for _, row in transformed_df.iterrows():
        sql = """
            INSERT INTO sales_data (
                order_id, product, category, order_date, region,
                quantity, unit_price, total_price, order_year
            ) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)
            ON DUPLICATE KEY UPDATE
                product=VALUES(product),
                category=VALUES(category),
                order_date=VALUES(order_date),
                region=VALUES(region),
                quantity=VALUES(quantity),
                unit_price=VALUES(unit_price),
                total_price=VALUES(total_price),
                order_year=VALUES(order_year);
```



```python
        cursor.execute(sql, (
            int(row['order_id']),
            int(row['product']),
            int(row['category']),
            row['order_date'],
            int(row['region']),
            float(row['quantity']),
            float(row['unit_price']),
            float(row['total_price']),
            int(row['order_year'])
        ))

    connection.commit()
    cursor.close()
    connection.close()
```

Figure 5

Conclusion:

Apache Airflow is a powerful and flexible tool for managing data workflows. Its use of DAGs provides a clear, structured way to define and monitor tasks. When integrated with Docker and MySQL, Airflow becomes even more scalable, portable, and reliable. This makes it a modern solution well-suited for building and automating complex data pipelines.