USN NUMBER:1RVU23CSE153
NAME:Ekshu DP

Ex No: 6 Date:	Apache Airflow

Objective:

The objective of this lab is to familiarize students with the fundamental concepts and practical implementation of **Apache Airflow** for workflow automation and data pipeline orchestration. Through this exercise, students will learn to understand Airflow's core components, including DAGs, tasks, operators, the scheduler, executor, and metadata database. They will gain hands-on experience in setting up and running Airflow using Docker, creating and deploying a simple Python-based DAG, and managing task dependencies and retries. Additionally, students will explore how to trigger, monitor, and analyze workflow runs through the Airflow Web UI, thereby developing the skills needed to design, schedule, and monitor data pipelines in a real-world environment.

Outcomes:

1. Explain core Airflow components: DAG, Task/Operator, Scheduler, Executor, Worker, Web UI and Metadata DB.
2. Deploy Airflow locally in Docker and access the Airflow web UI.
3. Write a valid Airflow DAG in Python containing at least three tasks (e.g., extract → transform → load pattern).
4. Configure task dependencies and basic retry behavior, and check logs for debugging.
5. Trigger DAGs manually and interpret run history and task states in the UI.

Materials

Computer with Docker & Docker Compose installed (Windows / Mac / Linux).

The `docker-compose.yml` and a sample `Dockerfile` (or use official `apache/airflow` image).

Text editor / IDE (VS Code, PyCharm, etc.).

Python 3.9+ (for authoring DAGs locally).

Example DAG file (provided below).

Browser to access Airflow UI (e.g., `http://localhost:9099/home` if using your compose).

(Optional) PostgreSQL or MySQL if configuring an external metadata DB — otherwise use default SQLite for local experiments.

USN NUMBER:1RVU23CSE153
NAME:Ekshu DP

Lab Procedure

A. Start Airflow with Docker Compose

1. From the lab folder with the docker-compose.yml, run:
    - docker-compose build (if you have a Dockerfile)
    - docker-compose up -d

2. Confirm containers are running: docker-compose ps or docker ps.
3. Open the Airflow UI in your browser (e.g., http://localhost:9099/home).

B. Deploy the DAG

1. Place example_lab6_dag.py into the host dags/ folder which is mounted to the Airflow dags directory.
2. In the Airflow UI, go to DAGs and refresh if needed. You should see lab6_etl_example.

C. Trigger & Monitor

1. Turn the DAG ON (toggle switch) and click Trigger DAG (or trigger a DagRun manually).
2. Open the Graph View to see task nodes and dependencies.
3. Click a task to view logs; examine stdout printed by the PythonOperator.
4. Note task states: queued → running → success (or failed if there is an error).
5. Review retry behaviour by intentionally causing an error (optional): change transform to raise an exception and watch retry.

D. Inspect Metadata DB (optional)

1. If using PostgreSQL/MySQL, connect to the DB and inspect the dag_run and task_instance tables to see run metadata. Your slides show how a metadata DB stores run information.

E. Cleanup

1. Stop containers: docker-compose down (use -v to remove volumes if desired).
2. Remove temporary files if created.

USN NUMBER:1RVU23CSE153
NAME:Ekshu DP

Results:

6. lab6_etl_example DAG appears in the Airflow UI.
7. The three tasks run in order: extract_task → transform_task → load_task.
8. Console/log output displays the simulated extract/transform/load prints.
9. Task logs and run history visible via the UI.
10. If an intentional error is introduced, the task will retry according to retries and retry_delay and the UI shows attempts.