**Key difference between Apache Airflow and Azure Data Factory:**  
**ETL Tool vs. Orchestration Tool:**

1. **ETL Tool**:
   * An **ETL tool** is software designed specifically to perform the tasks of **Extracting** data from a source, **Transforming** it (like cleaning, filtering, or aggregating data), and then **Loading** it into a target system (like a database or data warehouse).
   * **Example**: Suppose you want to move customer data from an Excel file to a database. An ETL tool would handle everything for you: it extracts data from Excel, transforms it (like removing duplicates), and loads it into the database.
2. **Orchestration Tool (like Airflow)**:
   * An **orchestration tool** coordinates different tasks, including ETL tasks, but it doesn’t do the extracting, transforming, or loading itself. Instead, it manages **when** and **how** those tasks should run, ensuring they happen in the correct order.
   * **Example**: Let’s say you have multiple tasks in a data workflow:
     + **Task 1**: Extract data from a website.
     + **Task 2**: Clean the data using a Python script.
     + **Task 3**: Load it into a database.

**Airflow** itself doesn’t handle the data extraction, transformation, or loading directly. Instead, it tells Task 1 to start at a specific time (or after another task finishes), waits until Task 1 is done, then triggers Task 2 to run, and finally Task 3.

1. You write **Python code** to define what each task does (like a script to extract or transform data), but Airflow’s job is to **orchestrate** these tasks, making sure they run in the correct sequence and handle dependencies (like retrying a task if it fails).

**Key Difference:**

* **ETL tool** (like ADF) is designed to **do** the actual extracting, transforming, and loading of data.
* **Airflow (orchestration tool)** coordinates and schedules when your ETL tasks (which could be Python scripts, SQL queries, or other commands) should run, but it doesn’t directly perform those ETL actions itself.

In simple terms, **Airflow is like the conductor of an orchestra** — it doesn't play the instruments (ETL tasks), but it tells them when and how to play together to make sure everything runs smoothly.

**Airflow is like ADF trigger?**

Yes, exactly! You can think of **Apache Airflow** as a much more advanced version of a **Data Factory trigger**, with additional features for managing dependencies and workflows.

Here’s a clearer comparison to solidify the idea:

**Data Factory Trigger:**

* **What it does**: In **Azure Data Factory (ADF)**, a **trigger** schedules when pipelines (which are ETL workflows) should run. It might start a pipeline based on a time schedule or an event (like a new file arriving in storage).
* **Simple Scheduling**: It just starts the pipeline, but doesn’t manage the detailed workflow of the pipeline itself.

**Apache Airflow (Orchestrator):**

* **What it does**: Airflow doesn’t just trigger a job or a script, it **manages the entire workflow** and all its dependencies. It tells each task (like extracting data, transforming it, or loading it) when to run, in what order, and what should happen if something fails.
* **Advanced Scheduling and Dependency Management**:
  + Airflow manages **complex task dependencies**. For example, Task B only runs after Task A is successfully completed. If Task A fails, Task B won't run.
  + It supports advanced scheduling and retries. You can say, "If Task A fails, try again after 5 minutes, and notify me if it still fails."

**Example Scenario:**

* **In ADF**:
  + You might use a trigger to run an ETL pipeline every day at midnight.
  + Inside ADF, the pipeline will run as a whole, but managing detailed dependencies or complex logic inside the pipeline could be limited or require a visual interface.
* **In Airflow**:
  + You could define a workflow (called a DAG) where:
    - **Step 1**: Extract data from a database.
    - **Step 2**: Clean and transform the data (this runs only after Step 1 completes successfully).
    - **Step 3**: Load the data into another system.
  + Airflow will manage the exact order of these steps and handle retries, failures, and notifications.
  + You can also add complex logic like running Step 2 and Step 3 in parallel or triggering extra tasks if needed.

**Summary:**

* **Data Factory trigger** starts pipelines or workflows but doesn't manage the details of the workflow’s internal dependencies in a complex way.
* **Airflow** manages everything from scheduling to dependency tracking between different tasks. It’s like a highly customizable conductor that can handle very complex workflows and dependencies, especially in multi-step ETL processes.

So yes, **Airflow** acts like a trigger **plus** a workflow manager that controls dependencies, retries, and task orchestration in a much more flexible way.

**Installing Airflow:**

Using docker for Airflow installation:

Following this video:  
<https://www.youtube.com/watch?v=2v9AKewyUEo>

For docker-compose.yml:  
<https://github.com/soumilshah1995/Learn-Apache-Airflow-in-easy-way-/blob/main/project/docker-compose.yml>

Docker-compose.yml is used for the configuration of the docker, first install docker desktop and then use IntelliJ with this project directory:  
A screenshot of a computer

Description automatically generated

Now copy the docker-compose file from the github and in Dockerfile type this:  
FROM puckel/docker-airflow:1.10.9

For setting up the airflow in docker, after above steps are done use this command:  


If ran successfully, it will host the airflow on 8080 port:  
A screenshot of a computer

Description automatically generated

If you want to install libraries in the code, then do it via Dockerfile-  
A screen shot of a computer

Description automatically generated: