# **Rakshitha Devi J**

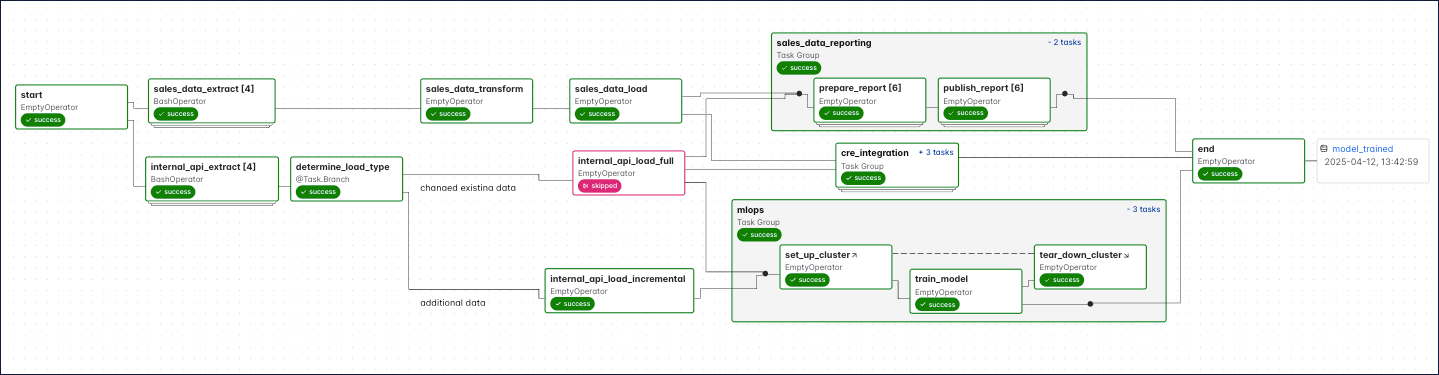
# **Apache Airflow DAGs**

# **Introduction:**

In [Apache Airflow](https://airflow.apache.org/), a DAG is a data pipeline or workflow. DAGs are the main organizational unit in Airflow; they contain a collection of tasks and dependencies that you want to execute on a schedule.

A DAG is defined in Python code and visualized in the Airflow UI. DAGs can be as simple as a single task or as complex as hundreds or thousands of tasks with complicated dependencies.

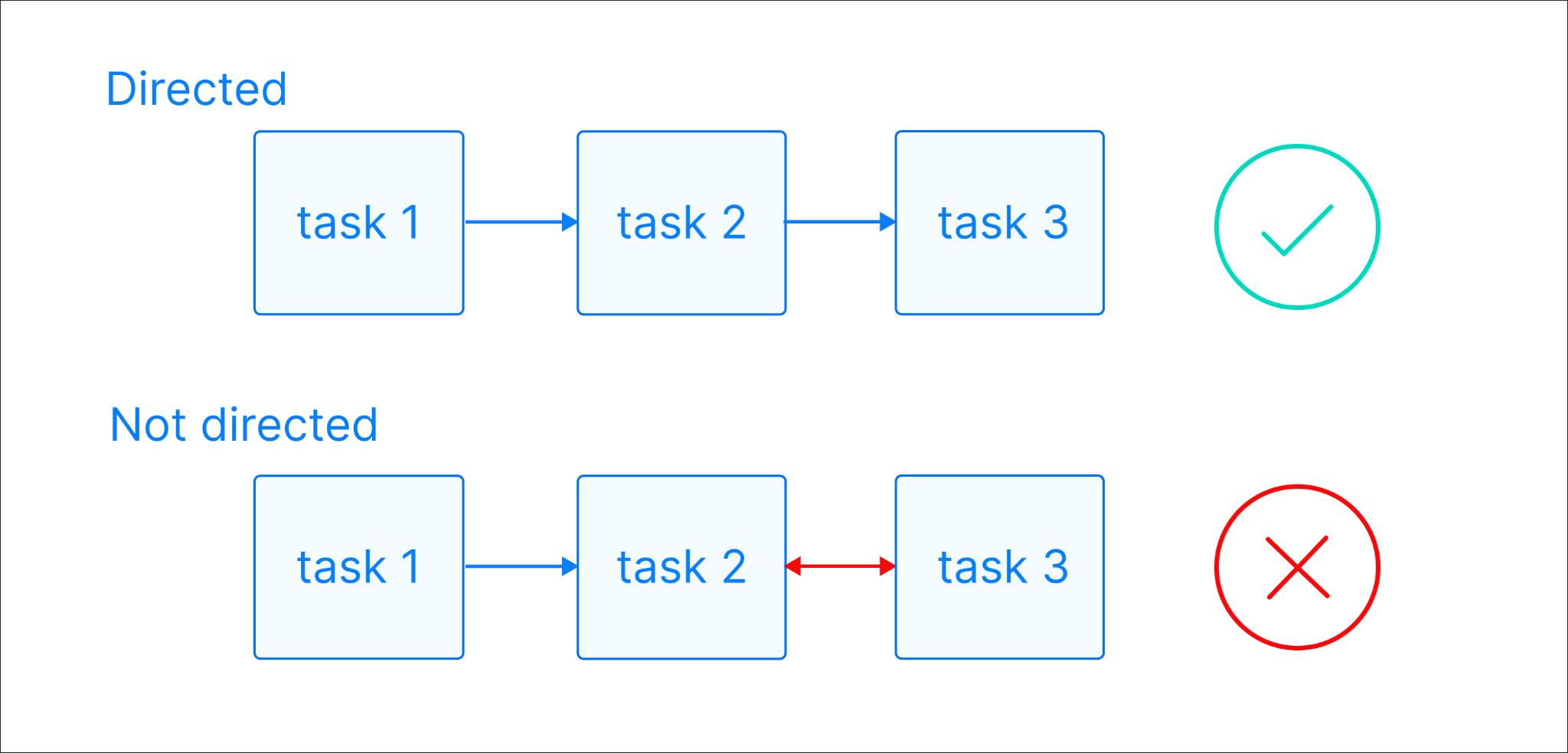
The following screenshot shows a [complex DAG run graph](https://www.astronomer.io/docs/learn/dags#complex-dag-run) in the Airflow UI. After reading this guide, you'll be able to understand the elements in this graph, as well as know how to define DAGs and use DAG parameters.



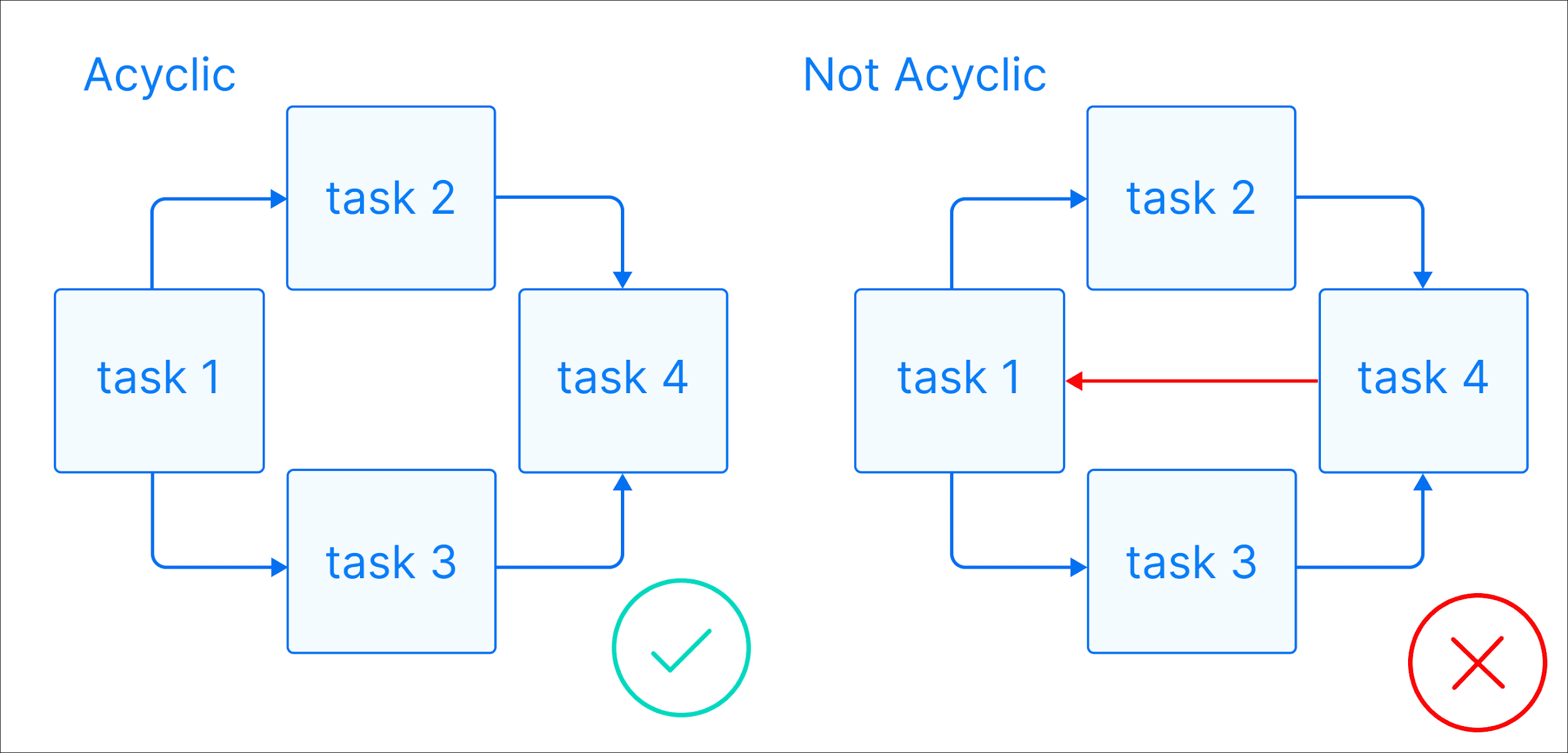
A *DAG* (directed acyclic graph) is a mathematical structure consisting of nodes and edges. In Airflow, a DAG represents a data pipeline or workflow with a start and an end.

The mathematical properties of DAGs make them useful for building data pipelines:

* Directed: There is a clear direction of flow between tasks. A task can be either upstream, downstream, or parallel to another task.



Acyclic: There are no circular dependencies in a DAG. This means that a task cannot depend on itself, nor can it depend on a task that ultimately depends on it.



* Graph: A DAG is a graph, which is a structure consisting of nodes and edges. In Airflow, nodes are tasks and edges are dependencies between tasks. Defining workflows as graphs helps you visualize the entire workflow in a way that's easy to navigate and conceptualize.

Core Components of Airflow:

**1. Scheduler**

* Watches DAG definitions and schedules tasks.
* Places runnable tasks into the **message queue (e.g., Celery or LocalExecutor)**.
* Handles triggers based on time or external events.Think of it as the **brain** that says: “It’s time to run this task!”

### **2. Web Server**

* A Flask-based **web UI** for:  
  + Browsing DAGs
  + Viewing logs
  + Triggering/running DAGs manually
  + Monitoring task status

Accessible at<http://localhost:8080> in the default setup.

### **3. Metadata Database (Metastore)**

* Stores all metadata:  
  + DAG run history
  + Task statuses
  + Schedules
  + Connections & variables

Typically powered by **PostgreSQL** or **MySQL**.

### **4. Executor**

* Executes the tasks scheduled by the Scheduler.
* Types of Executors:  
  + **SequentialExecutor**: Only for testing (single task at a time)
  + **LocalExecutor**: Parallel execution on one machine
  + **CeleryExecutor**: Distributed execution using Celery workers
  + **KubernetesExecutor**: Runs each task in a Kubernetes pod

### **5. Worker(s) (for Celery/KubernetesExecutors)**

* Pull tasks from the queue and run them.
* Enables **scalability** and **parallelism**.

### **6. Triggerer (optional, used for deferrable operators)**

* A lightweight process that listens asynchronously for external events (like sensors).
* Greatly reduces resource usage for long-waiting tasks.

### **7. CLI (Command Line Interface)**

* Used to interact with Airflow via terminal: