# Introduction to Apache Airflow and Data Pipelines

Understanding Workflow Automation in Data Engineering

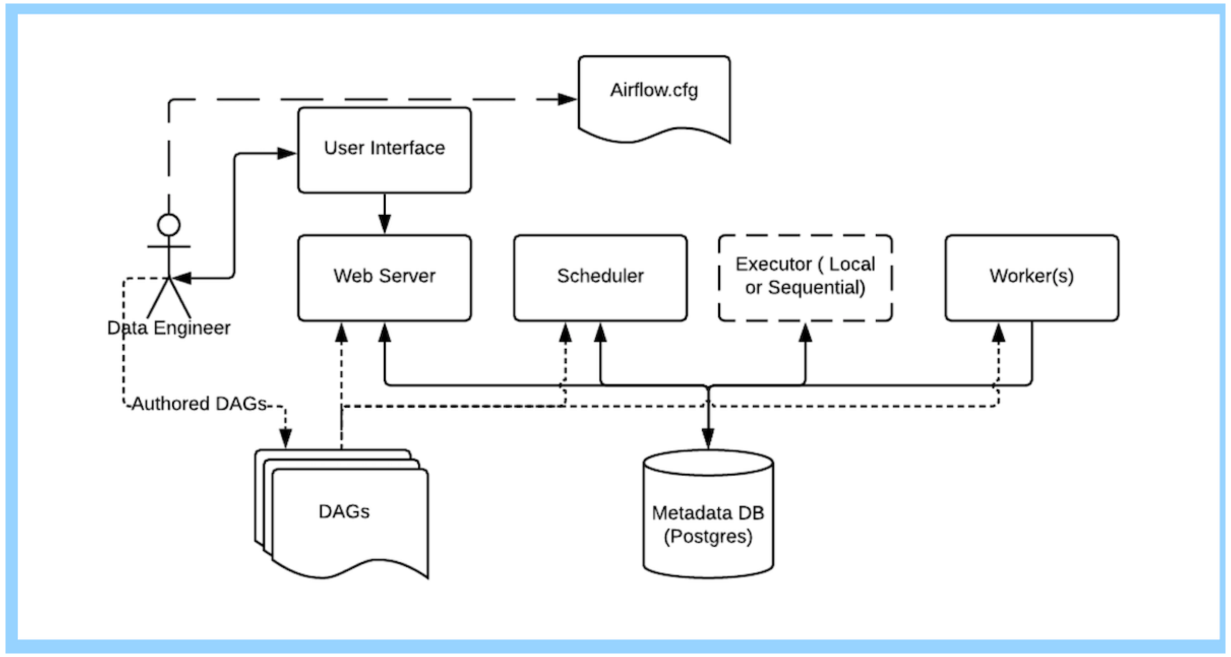
# Introducing Apache Airflow

- Apache Airflow is an open-source tool for orchestrating complex workflows and data processing pipelines. It is a platform to programmatically schedule, and monitor workflows for scheduled jobs

## Airflow Architecture

Airflow is a distributed system that consists of the following components:

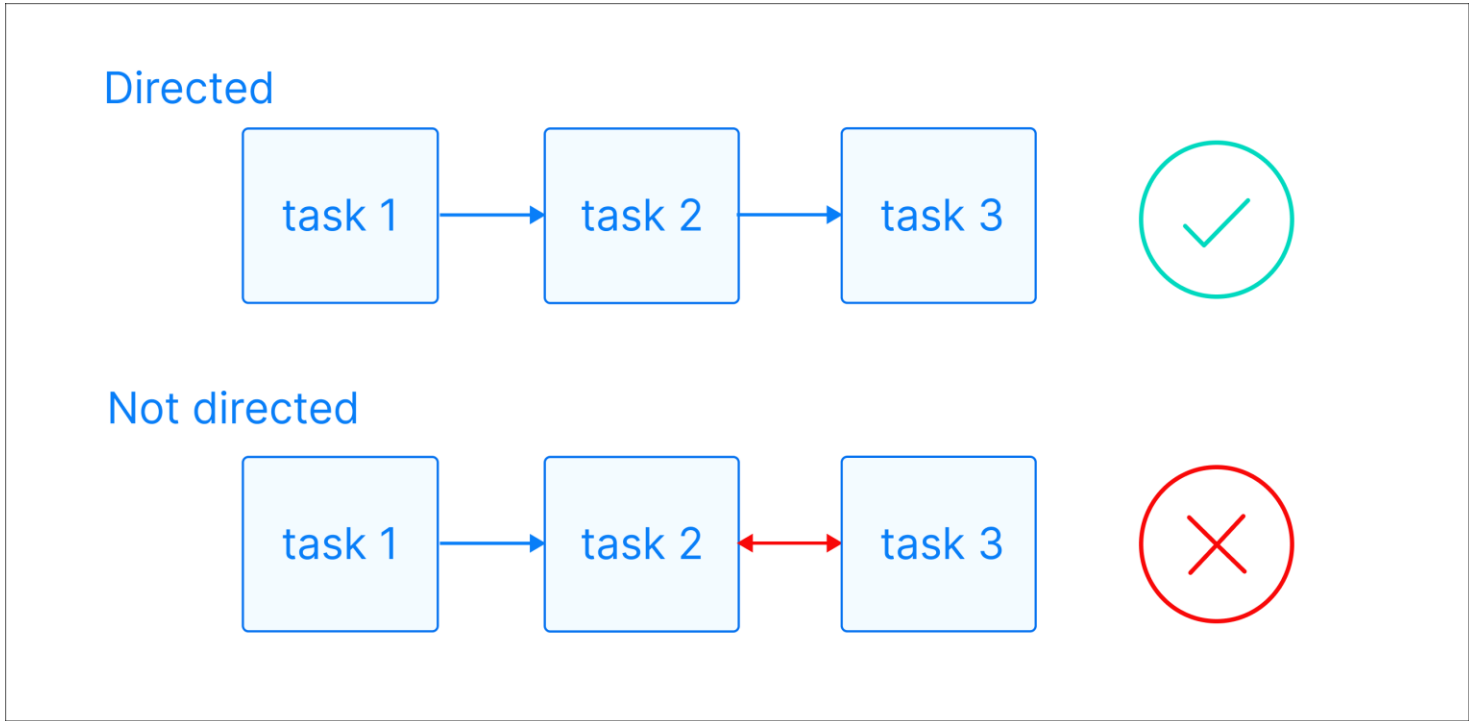
- **Webserver:** The webserver provides a user interface for managing Airflow workflows.
- **Scheduler:** The scheduler is responsible for scheduling and running Airflow tasks.
- **Worker:** The worker nodes execute Airflow tasks.
- **Executor:** The executor is responsible for running tasks on worker nodes.
- **Metadata database:** The metadata database stores information about Airflow workflows, tasks, and other entities.



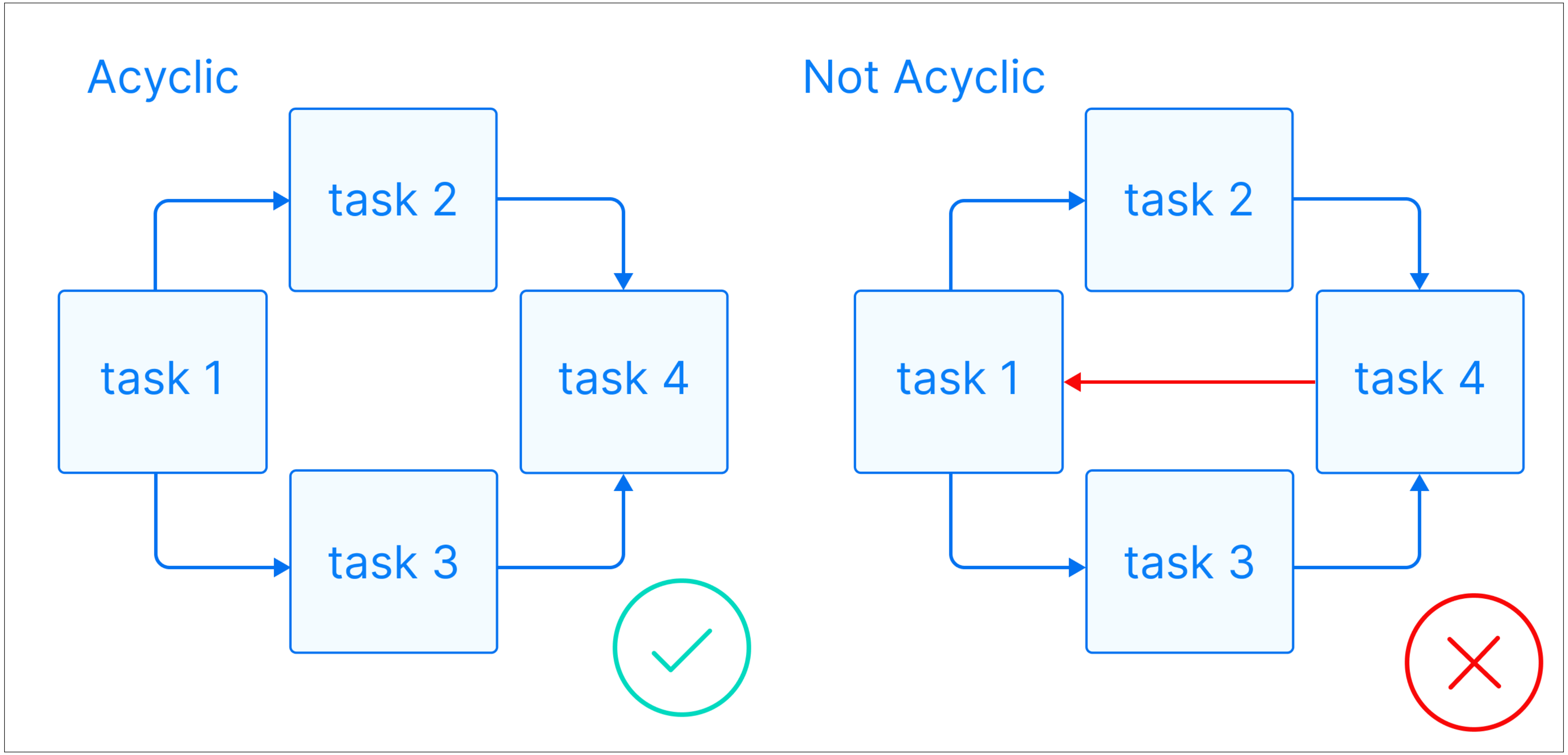
# What is a DAG?

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.

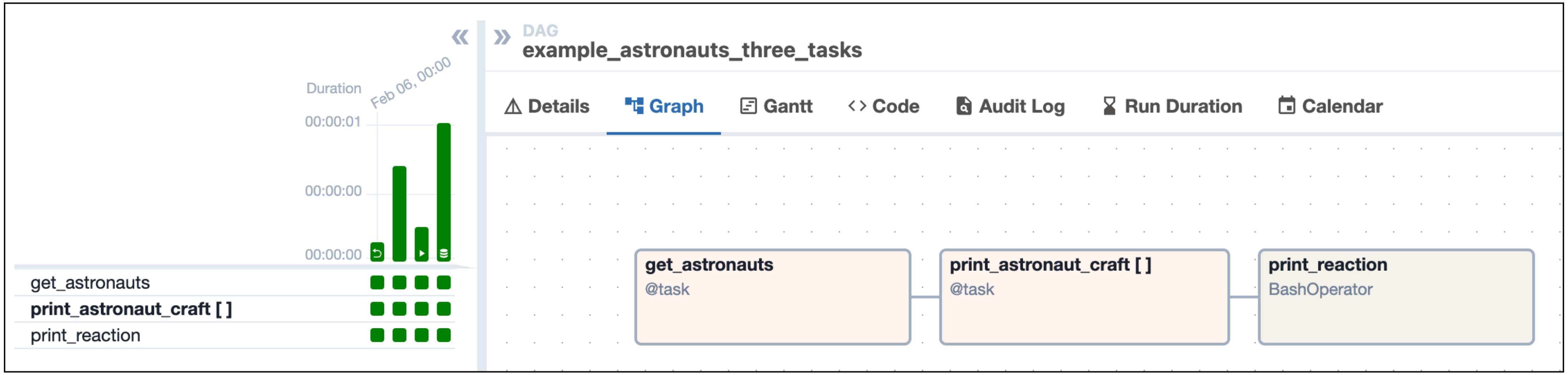
**Directed**: There is a clear direction of flow between tasks. A task can be either upstream, downstream, or parallel to another task.



**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.

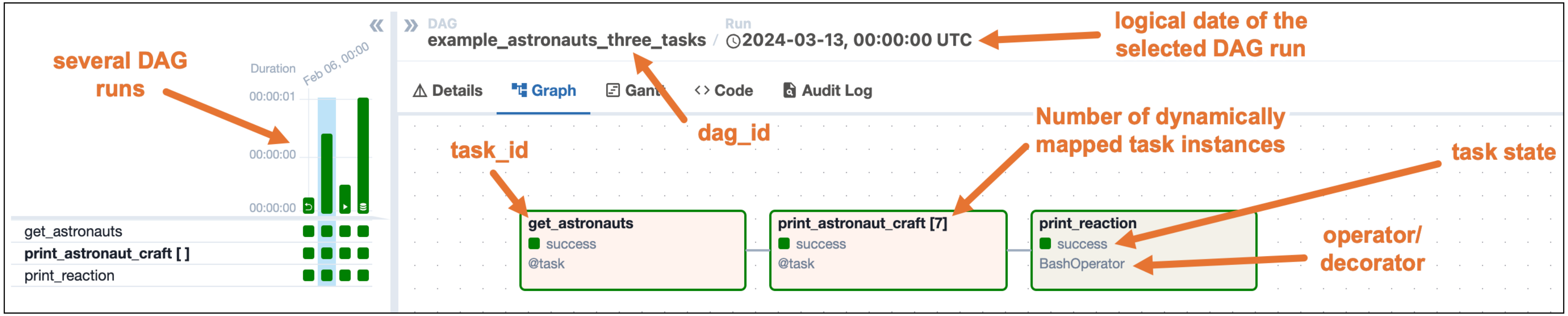


**Graph**: A DAG is a graph, which is a structure consisting of nodes and edges. In Airflow, nodes are tasks and edges are dependencies between tasks. Defining workflows as graphs helps you visualize the entire workflow in a way that's easy to navigate and conceptualize.



# DAG run properties

A DAG run graph in the Airflow UI contains information about the DAG run, as well as the status of each task instance in the DAG run. The following screenshot shows the same DAG as in the previous section, but with annotations explaining the different elements of the graph.



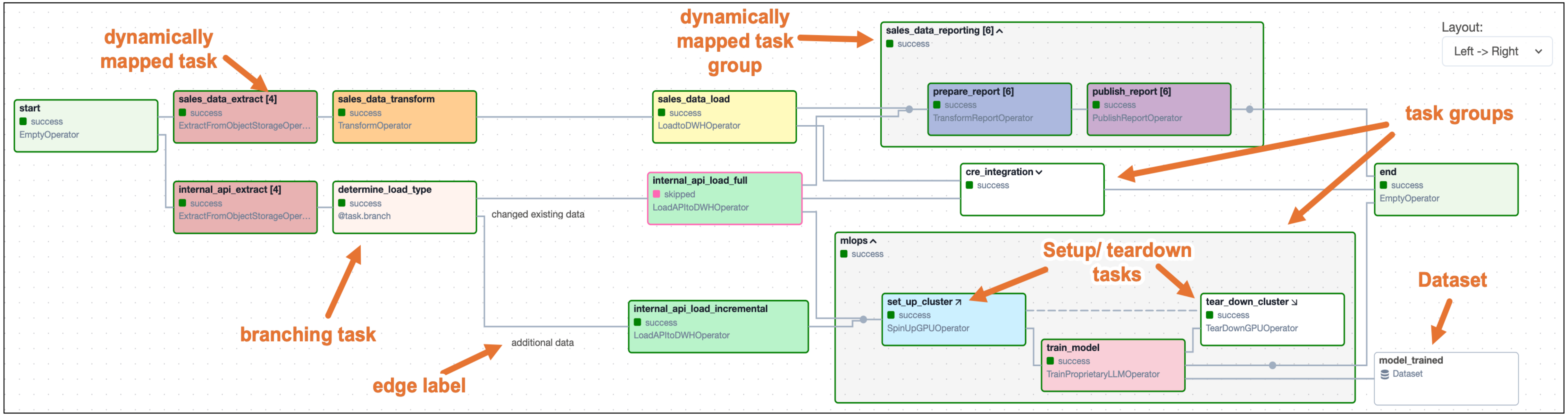
## DAG-level parameters

- **dag_id:** The name of the DAG. This must be unique for each DAG in the Airflow environment. When using the @dag decorator and not providing the dag_id parameter name, the function name is used as the dag_id.
- **start_date:** The date and time after which the DAG starts being scheduled.
- **schedule:** The schedule for the DAG. There are many different ways to define a schedule, see Scheduling in Airflow for more information.
- **catchup:** Whether the scheduler should backfill all missed DAG runs between the current date and the start date when the DAG is unpaused. It is a best practice to always set it to False unless you specifically want to backfill missed DAG runs.

## DAG statuses

- **Queued**: The time after which the DAG run can be created has passed but the scheduler has not created task instances for it yet.
- **Running**: The DAG run is eligible to have task instances scheduled.
- **Success**: All task instances are in a terminal state (success, skipped, failed or upstream_failed) and all leaf tasks (tasks with no downstream tasks) are either in the state success or skipped.
- **Failed**: All task instances are in a terminal state and at least one leaf task is in the state failed or upstream_failed.

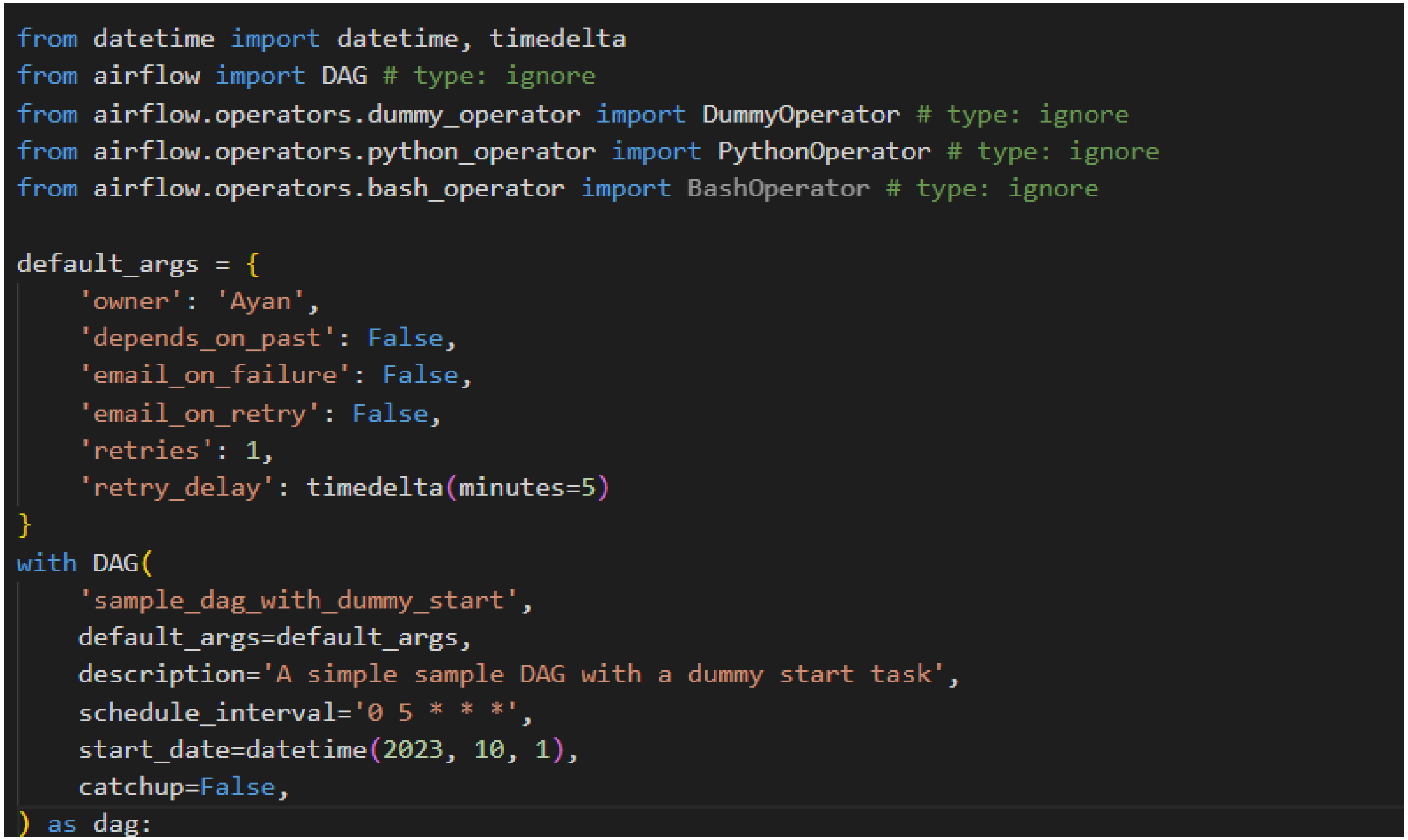
# Complex DAG runs



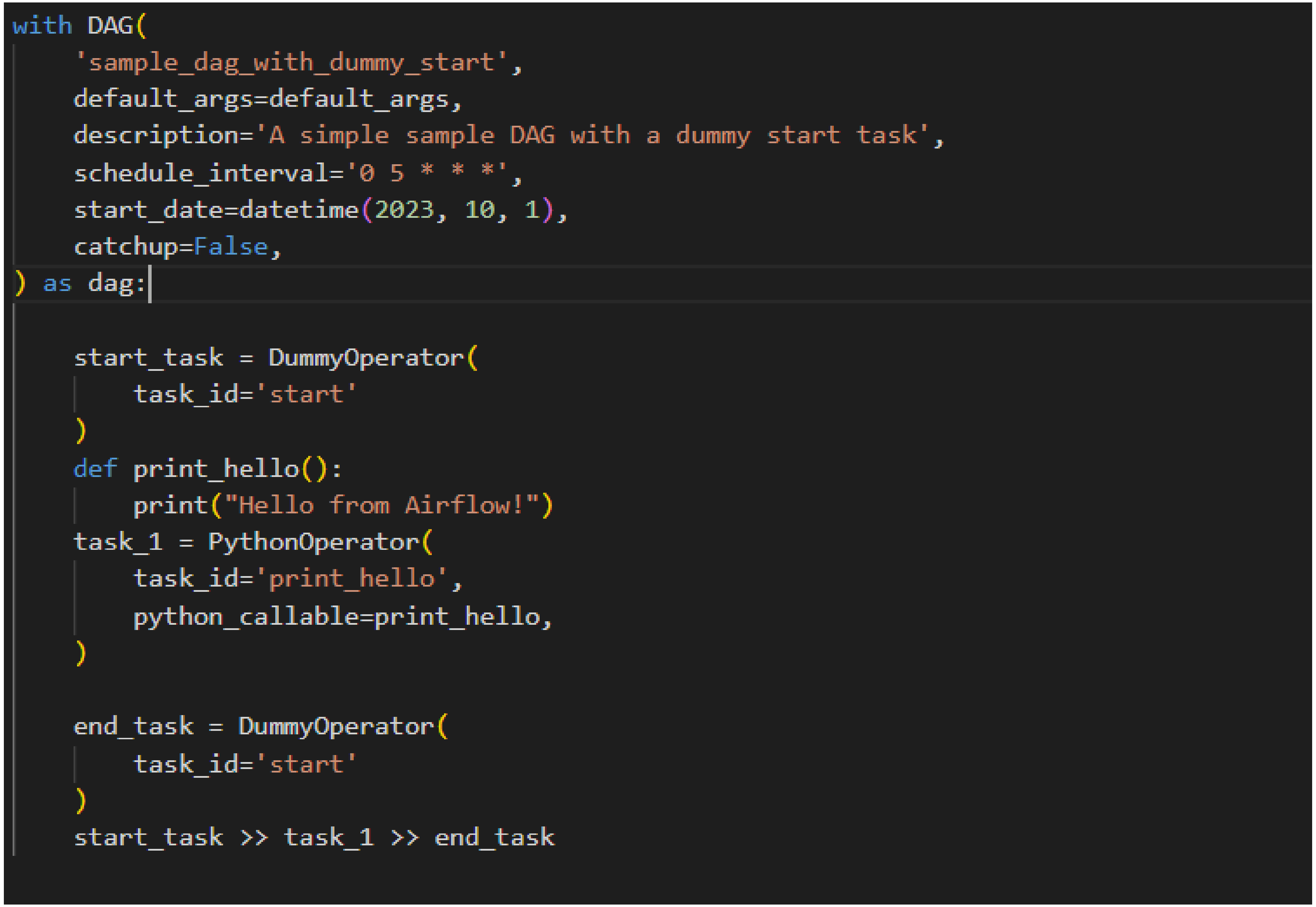
## Complex DAG runs

- **Dynamically mapped tasks**: A dynamically mapped task is created dynamically at runtime based on user-defined input. The number of dynamically mapped task instances is shown in brackets ([]) behind the task ID.
- **Branching tasks**: A branching task creates a conditional branch in the DAG.
- **Edge labels**: Edge labels appear on the edge between two tasks. These labels are often helpful to annotate branch decisions in a DAG graph.
- **Task groups**: A task group is a tool to logically and visually group tasks in an Airflow DAG.
- **Setup/teardown tasks**: When using Airflow to manage infrastructure, it can be helpful to define tasks as setup and teardown tasks to take advantage of additional intelligent dependency behavior. Setup and teardown tasks appear with diagonal arrows next to their task IDs and are connected with a dotted line.
- **Datasets**: Datasets are shown in the DAG graph. If a DAG is scheduled on a dataset, it is shown upstream of the first task of the DAG.

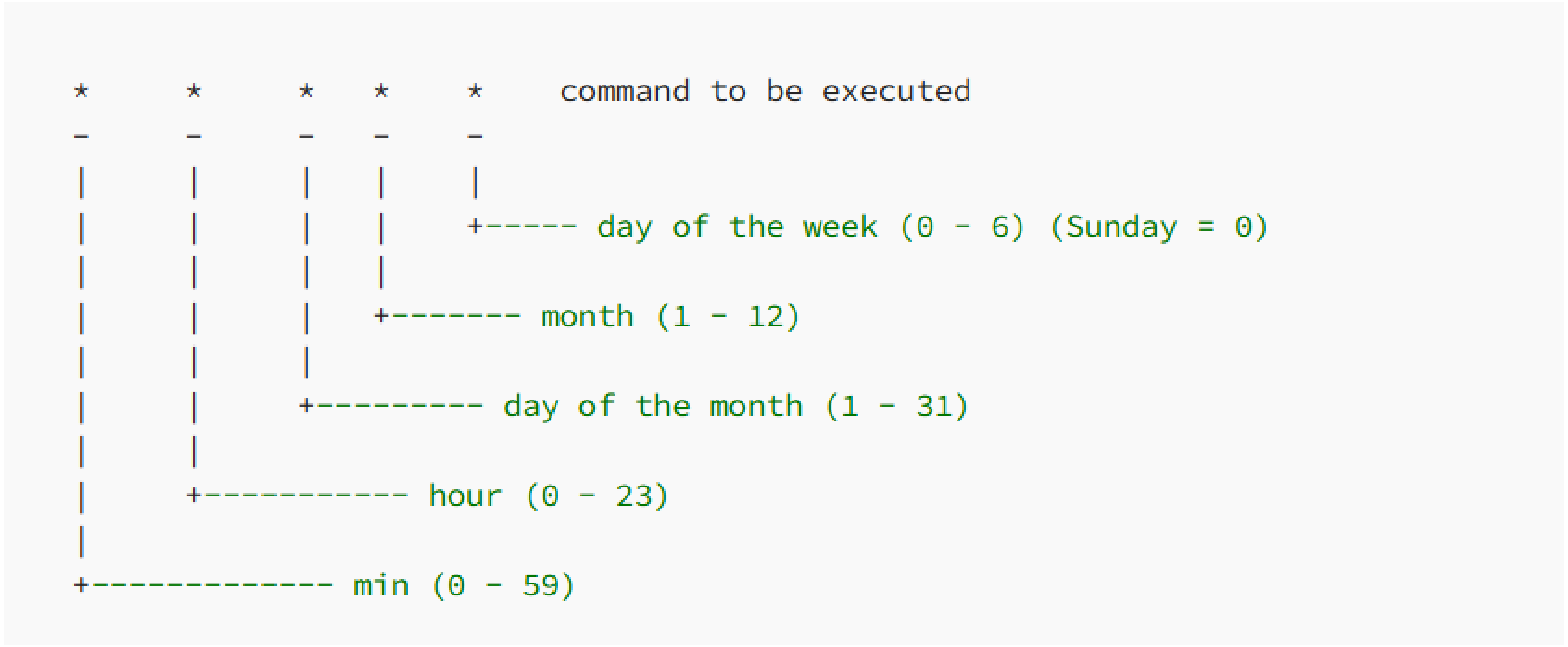
# Write a DAG



```python
from datetime import datetime, timedelta
from airflow import DAG # type: ignore
from airflow.operators.dummy_operator import DummyOperator # type: ignore
from airflow.operators.python_operator import PythonOperator # type: ignore
from airflow.operators.bash_operator import BashOperator # type: ignore

default_args = {
    'owner': 'Ayan',
    'depends_on_past': False,
    'email_on_failure': False,
    'email_on_retry': False,
    'retries': 1,
    'retry_delay': timedelta(minutes=5)
}
with DAG(
    'sample_dag_with_dummy_start',
    default_args=default_args,
    description='A simple sample DAG with a dummy start task',
    schedule_interval='0 5 * * *',
    start_date=datetime(2023, 10, 1),
    catchup=False,
) as dag:
```

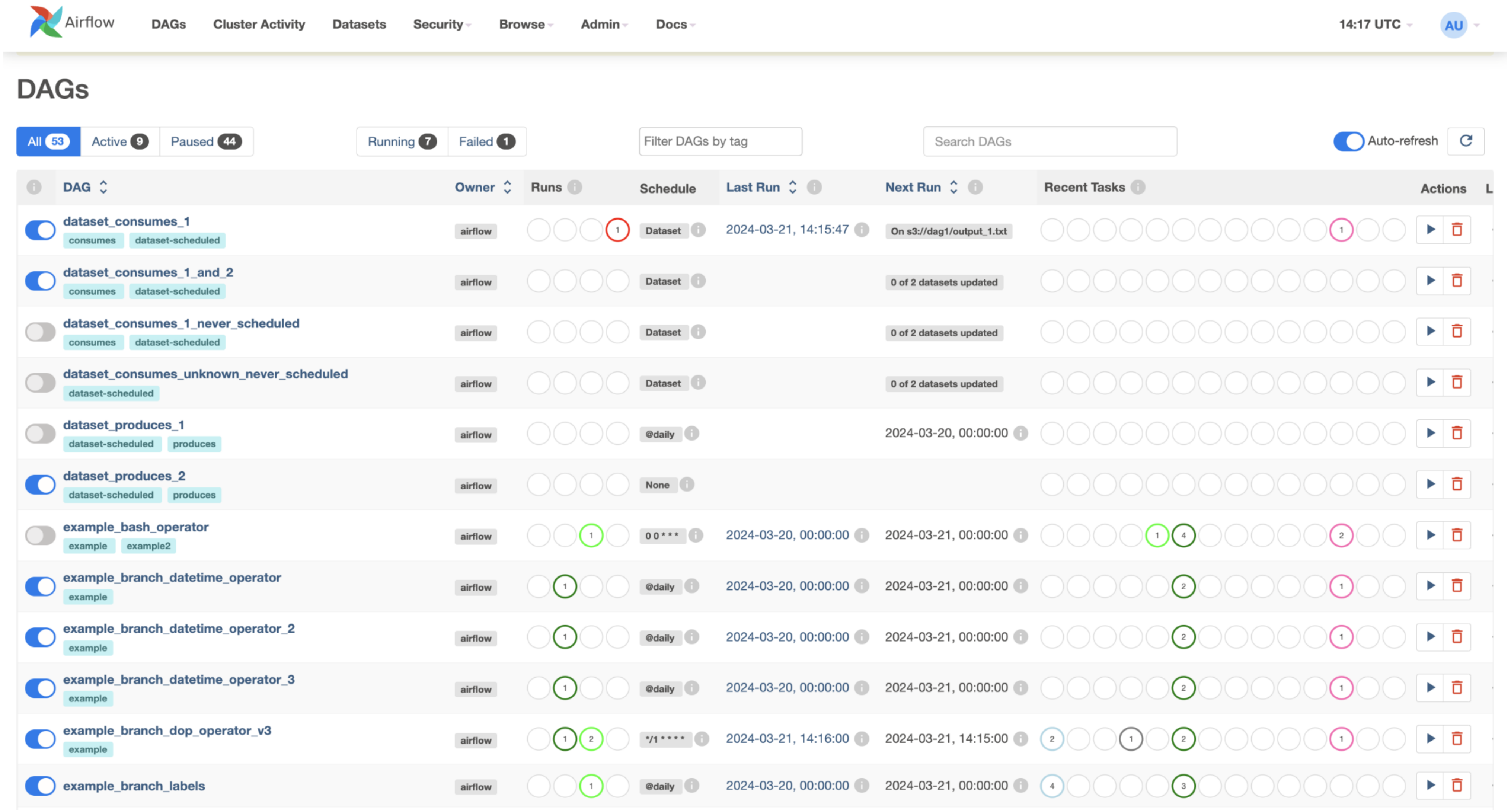

```python
with DAG(
    'sample_dag_with_dummy_start',
    default_args=default_args,
    description='A simple sample DAG with a dummy start task',
    schedule_interval='0 5 * * *',
    start_date=datetime(2023, 10, 1),
    catchup=False,
) as dag:

    start_task = DummyOperator(
        task_id='start'
    )
    def print_hello():
        print("Hello from Airflow!")
    task_1 = PythonOperator(
        task_id='print_hello',
        python_callable=print_hello,
    )

    end_task = DummyOperator(
        task_id='start'
    )
    start_task >> task_1 >> end_task
```

## Cron Schedule



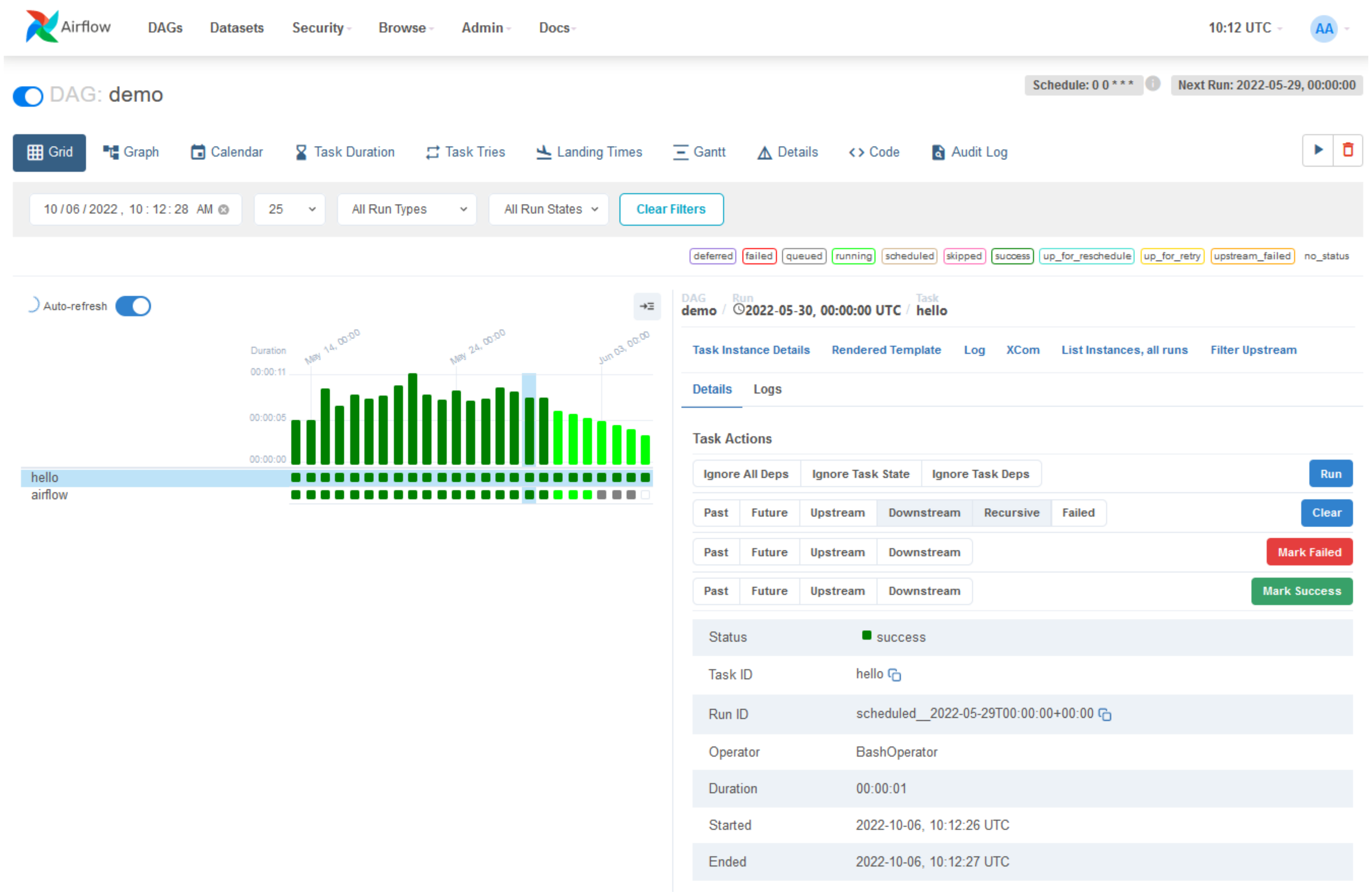
```
*     *     *   *    *    command to be executed
-     -     -   -    -
|     |     |   |    |
|     |     |   |    +----- day of the week (0 - 6) (Sunday = 0)
|     |     |   |
|     |     |   +------- month (1 - 12)
|     |     |
|     |     +--------- day of the month (1 - 31)
|     |
|     +----------- hour (0 - 23)
|
+------------- min (0 - 59)
```

# DAGs



DAGs

| DAG | Owner | Runs | Schedule | Last Run | Next Run | Recent Tasks | Actions |
|---|---|---|---|---|---|---|---|
| dataset_consumes_1<br>consumes dataset-scheduled | airflow | 1 | Dataset | 2024-03-21, 14:15:47 | On s3://dag1/output_1.txt | 1 | |
| dataset_consumes_1_and_2<br>consumes dataset-scheduled | airflow | | Dataset | | 0 of 2 datasets updated | | |
| dataset_consumes_1_never_scheduled<br>consumes dataset-scheduled | airflow | | Dataset | | 0 of 2 datasets updated | | |
| dataset_consumes_unknown_never_scheduled<br>dataset-scheduled | airflow | | Dataset | | 0 of 2 datasets updated | | |
| dataset_produces_1<br>dataset-scheduled produces | airflow | | @daily | | 2024-03-20, 00:00:00 | | |
| dataset_produces_2<br>dataset-scheduled produces | airflow | | None | | | | |
| example_bash_operator<br>example example2 | airflow | 1 | 0 0 * * * | 2024-03-20, 00:00:00 | 2024-03-21, 00:00:00 | 1 4 2 | |
| example_branch_datetime_operator<br>example | airflow | 1 | @daily | 2024-03-20, 00:00:00 | 2024-03-21, 00:00:00 | 2 1 | |
| example_branch_datetime_operator_2<br>example | airflow | 1 | @daily | 2024-03-20, 00:00:00 | 2024-03-21, 00:00:00 | 2 1 | |
| example_branch_datetime_operator_3<br>example | airflow | 1 | @daily | 2024-03-20, 00:00:00 | 2024-03-21, 00:00:00 | 2 1 | |
| example_branch_dop_operator_v3<br>example | airflow | 1 2 | */1 * * * * | 2024-03-21, 14:16:00 | 2024-03-21, 14:15:00 | 2 1 2 1 | |
| example_branch_labels | airflow | 1 | @daily | 2024-03-20, 00:00:00 | 2024-03-21, 00:00:00 | 4 3 | |



DAG: demo

Schedule: 0 0 * * *

Next Run: 2022-05-29, 00:00:00

10 / 06 / 2022 , 10 : 12 : 28 AM | 25 | All Run Types | All Run States

DAG demo / Run 2022-05-30, 00:00:00 UTC / Task hello

Task Actions

| | |
|---|---|
| Status | 🟩 success |
| Task ID | hello |
| Run ID | scheduled__2022-05-29T00:00:00+00:00 |
| Operator | BashOperator |
| Duration | 00:00:01 |
| Started | 2022-10-06, 10:12:26 UTC |
| Ended | 2022-10-06, 10:12:27 UTC |