Apache Airflow

Airflow

- Apache Airflow is an open-source platform designed to programmatically author, schedule, and monitor workflows as Directed Acyclic Graphs (DAGs).
- It is a workflow orchestration tool that allows you to automate and manage complex workflows and data pipelines.

Key Features

- **DAG Representation**: Workflows are defined as Directed Acyclic Graphs, enabling visualization of task dependencies and execution flow.
- **Task Scheduling**: Airflow provides a scheduler to execute tasks based on defined time intervals or external triggers.
- **Extensibility**: It supports plugins, custom operators, and a wide range of integrations.
- **Monitoring**: You can monitor task execution, retry failed tasks, and track logs using its user-friendly web interface.
- **Scalability**: It can be scaled horizontally to manage large-scale workflows.

How Airflow Helps in Machine Learning?

- **Data Ingestion and Preprocessing**
- Automates the collection of raw data from multiple sources.
- Schedules preprocessing tasks (e.g., cleaning, feature engineering) in a reproducible way.
- Ensures tasks are run in the correct sequence using task dependencies.
- **Model Training**
- Orchestrates training jobs, including handling distributed training setups.
- Enables retraining workflows triggered by events (e.g., new data arrival).
- Tracks performance metrics and logs for auditability.

How Airflow Helps in Machine Learning?

- **Model Evaluation**
- Automates evaluation processes, including cross-validation and performance comparisons.
- Integrates with tools to visualize metrics and results.
- **Model Deployment**
- Manages pipelines for pushing trained models into production environments.
- Integrates with tools like Docker, Kubernetes, and cloud services to handle deployments.

How Airflow Helps in Machine Learning?

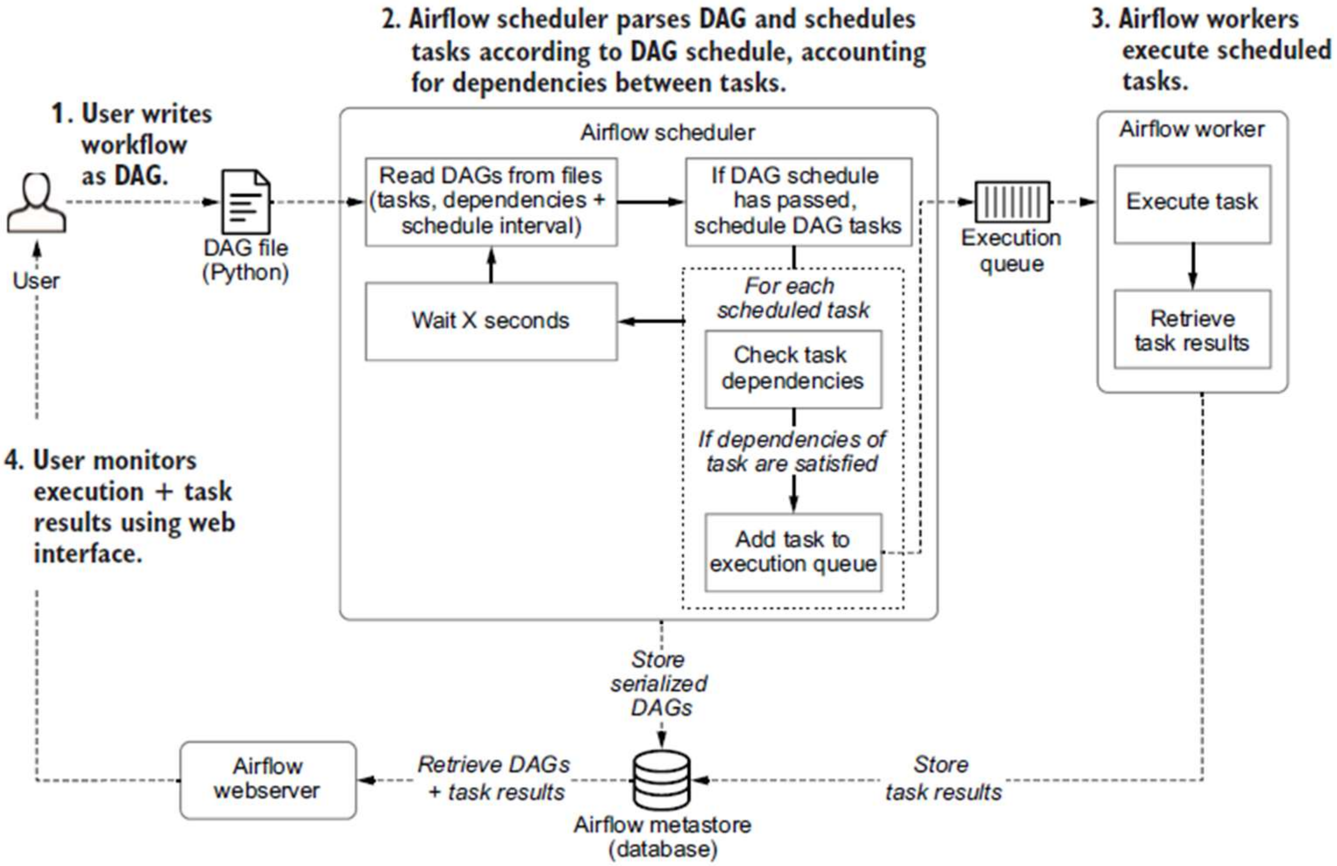
- **Pipeline Monitoring and Maintenance**
- Provides a central dashboard for monitoring task status.
- Enables retries, failure notifications, and pipeline re-execution with ease.
- **Collaboration and Version Control**
- Workflows are defined in Python scripts, which can be version-controlled using Git.
- Encourages collaborative development and debugging of workflows.

Data Pipelines

- Data pipelines generally consist of several tasks or actions that need to be executed to achieve the desired result.

Data Pipeline as Graphs

- One way to make dependencies between tasks more explicit is to draw the data pipeline as a graph.
- In this graph-based representation, tasks are represented as nodes in the graph, while dependencies between tasks are represented by directed edges between the task nodes.
- The direction of the edge indicates the direction of the dependency, with an edge pointing from task A to task B, indicating that task A needs to be completed before task B can start.
- Note that this type of graph is generally called a directed graph, due to the directions in the graph edges.



1. User writes workflow as DAG.

2. Airflow scheduler parses DAG and schedules tasks according to DAG schedule, accounting for dependencies between tasks.

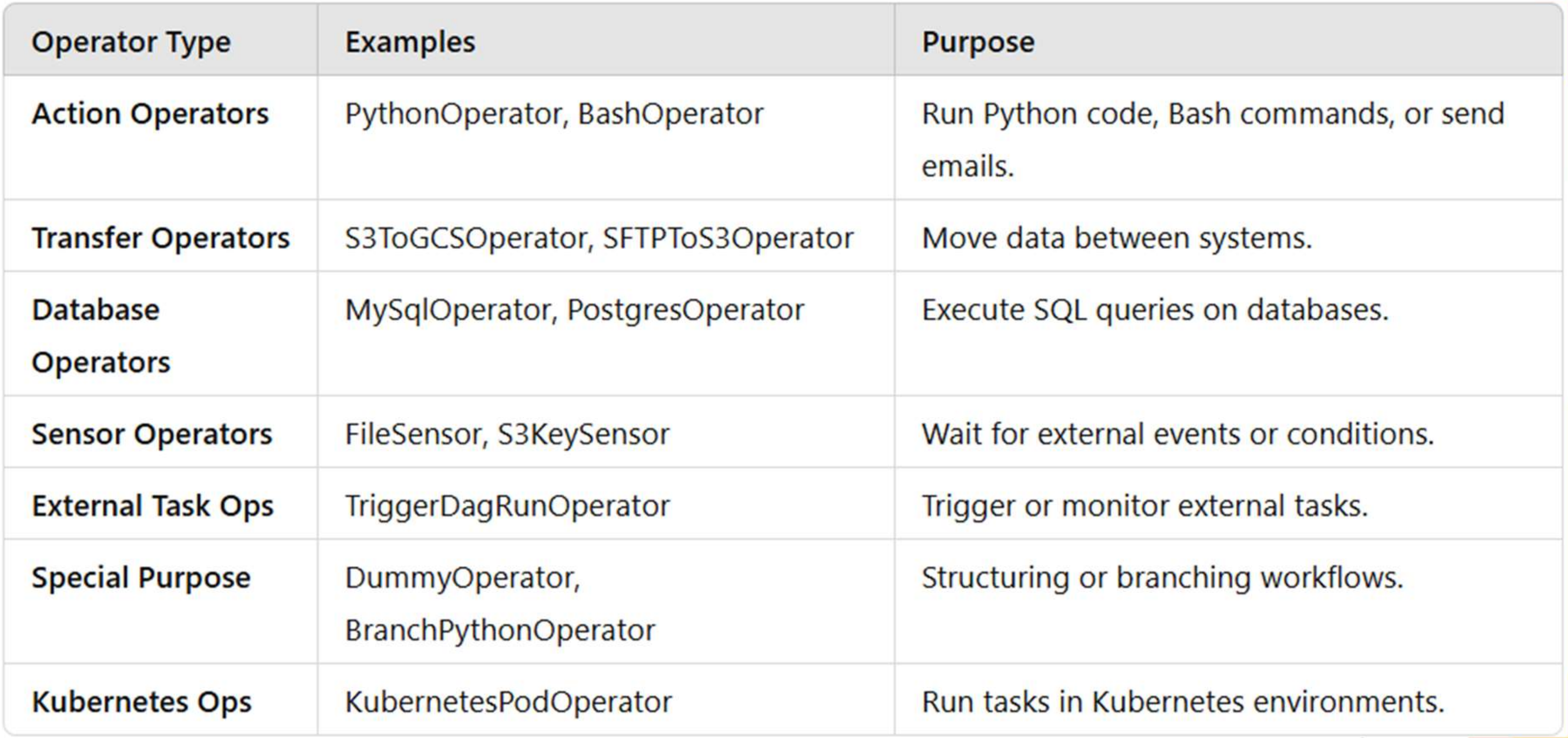
3. Airflow workers execute scheduled tasks.

4. User monitors execution + task results using web interface.

Operators

- Airflow workflow script consists of one or more operators, which perform the actual work.
- Each operator performs a single unit of work, and multiple operators together form a workflow or DAG in Airflow.
- Operators run independently of each other, although you can define the order of execution, which we call dependencies in Airflow.

Operators



Operator Type	Examples	Purpose
Action Operators	PythonOperator, BashOperator	Run Python code, Bash commands, or send emails.
Transfer Operators	S3ToGCSOperator, SFTPToS3Operator	Move data between systems.
Database Operators	MySqlOperator, PostgresOperator	Execute SQL queries on databases.
Sensor Operators	FileSensor, S3KeySensor	Wait for external events or conditions.
External Task Ops	TriggerDagRunOperator	Trigger or monitor external tasks.
Special Purpose	DummyOperator, BranchPythonOperator	Structuring or branching workflows.
Kubernetes Ops	KubernetesPodOperator	Run tasks in Kubernetes environments.

DAG Object

```python
from airflow import DAG
from airflow.operators.dummy import DummyOperator
from datetime import datetime

# Define the DAG
dag = DAG(
    dag_id='example_dag',
    start_date=datetime(2023, 1, 1),
    schedule_interval='@daily',
    catchup=False
)

# Define tasks
start = DummyOperator(task_id='start', dag=dag)
end = DummyOperator(task_id='end', dag=dag)

# Set task dependencies
start >> end
```

The DAG runs daily starting from January 1, 2023.
Two tasks (start and end) are executed in sequence.