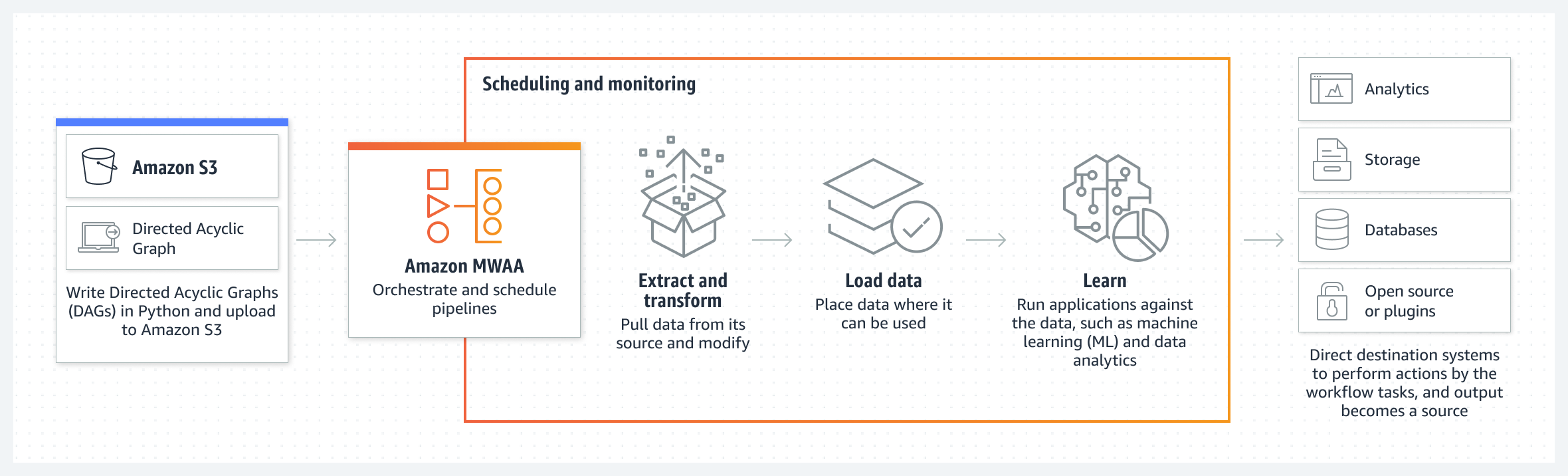
Aws Apache airflow services

Amazon MWAA is a managed service for Apache Airflow that lets you use your current, familiar Apache Airflow platform to orchestrate your workflows. You gain improved scalability, availability, and security without the operational burden of managing underlying infrastructure.

How it works

Amazon Managed Workflows for Apache Airflow (Amazon MWAA) orchestrates your workflows using Directed Acyclic Graphs (DAGs) written in Python. You provide MWAA an Amazon Simple Storage Service (S3) bucket where your DAGs, plugins, and Python requirements reside. Then run and monitor your DAGs from the AWS Management Console, a command line interface (CLI), a software development kit (SDK), or the Apache Airflow user interface (UI).



What Apache Airflow is used for

Apache Airflow is used for the scheduling and orchestration of data pipelines or workflows. Orchestration of data pipelines refers to the sequencing, coordination, scheduling, and managing of complex data pipelines from diverse sources.

What Is Apache Airflow

Apache Airflow is an open-source platform for authoring, scheduling and monitoring data and computing workflows. First developed by Airbnb, it is now under the Apache Software Foundation. Airflow uses Python to create workflows that can be easily scheduled and monitored. Airflow can run anything—it is completely agnostic to what you are running.

Benefits of Apache Airflow include:

* **Ease of use**—you only need a little python knowledge to get started.
* **Open-source community**—Airflow is free and has a large community of active users.
* **Integrations**—ready-to-use operators allow you to integrate Airflow with cloud platforms (Google, AWS, Azure, etc).
* **Coding with standard Python—**you can create flexible workflows using Python with no knowledge of additional technologies or frameworks.**‍**
* **Graphical UI**—monitor and manage workflows, check the status of ongoing and completed tasks.

This is part of our series of articles about [machine learning operations](https://www.run.ai/guides/machine-learning-operations).

**In this article, you will learn:**

* [Airflow Use Cases](https://www.run.ai/guides/machine-learning-operations/apache-airflow#Airflow-Use-Cases)
* [Workloads](https://www.run.ai/guides/machine-learning-operations/apache-airflow#Workloads)
* [Airflow Architecture](https://www.run.ai/guides/machine-learning-operations/apache-airflow#Airflow-Architecture)
* [Getting Started with Apache Airflow](https://www.run.ai/guides/machine-learning-operations/apache-airflow#Tutorial)
* [Airflow Best Practices](https://www.run.ai/guides/machine-learning-operations/apache-airflow#Airflow-Best-Practices)

Why Use Airflow? Key Use Cases

Apache Airflow's versatility allows you to set up any type of workflow. Airflow can run ad hoc workloads not related to any interval or schedule. However, it is most suitable for pipelines that change slowly, are related to a specific time interval, or are pre-scheduled.

In this context, slow change means that once the pipeline is deployed, it is expected to change from time to time (once every several days or weeks, not hours or minutes). This has to do with the lack of versioning for Airflow pipelines.

Airflow is best at handling workflows that run at a specified time or every specified time interval. You can trigger the pipeline manually or using an external trigger (e.g. via REST API).

You can use Apache Airflow to schedule the following:

* ETL pipelines that extract data from multiple sources, and run Spark jobs or other data transformations
* Machine learning model training
* Automated generation of reports
* Backups and other DevOps tasks

Airflow is commonly used to automate machine learning tasks. To understand machine learning automation in more depth, **read our guides** to:

* [Machine learning workflow](https://www.run.ai/guides/machine-learning-engineering/machine-learning-workflow)
* [Machine learning automation](https://www.run.ai/guides/machine-learning-engineering/machine-learning-automation)

Workloads

The DAG runs through a series of Tasks, which may be subclasses of Airflow's BaseOperator, including:

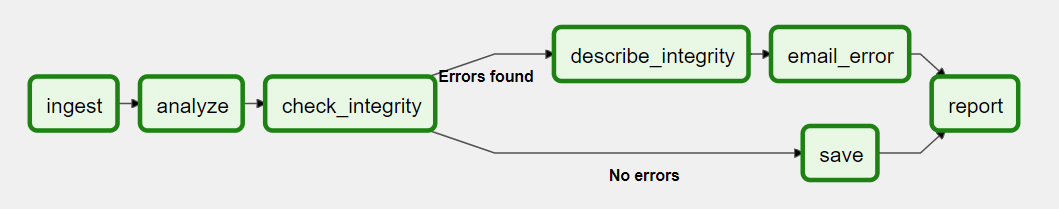
* **Operators**—predefined tasks that can be strung together quickly
* **Sensors**—a type of Operator that waits for external events to occur
* **TaskFlow—**a custom Python function packaged as a task, which is decorated with @tasks

Operators are the building blocks of Apache Airflow, as they define how the Tasks run and what they do. The terms Task and Operator are sometimes used interchangeably, but they should be considered separate concepts, with Operators and Sensors serving as templates for creating Tasks.

‍

Airflow Architecture

The Airflow platform lets you build and run workflows, which are represented as Directed Acyclic Graphs (DAGs). A sample DAG is shown in the diagram below.



A DAG contains Tasks (action items) and specifies the dependencies between them and the order in which they are executed. A Scheduler handles scheduled workflows and submits Tasks to the Executor, which runs them. The Executor pushes tasks to workers.

Other typical components of an Airflow architecture include a database to store state metadata, a web server used to inspect and debug Tasks and DAGs, and a folder containing the DAG files.

