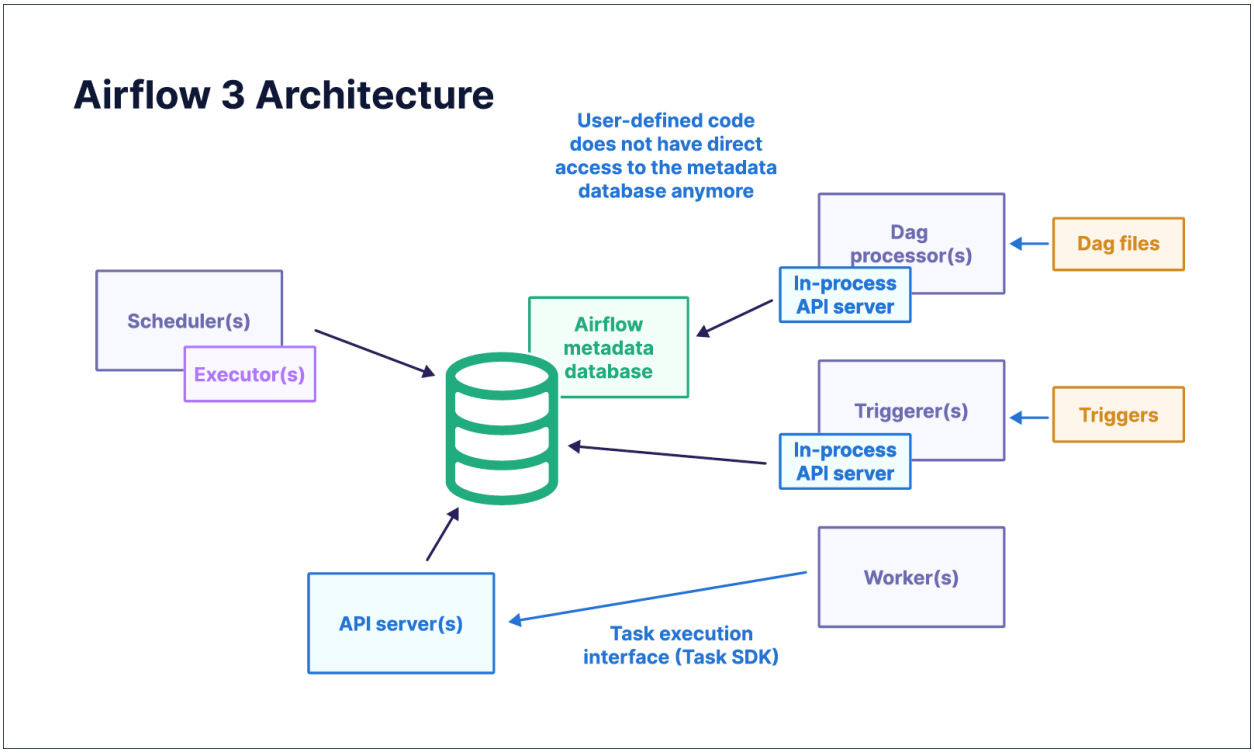
Apache Airflow and its components

Apache Airflow is an open-source platform responsible for authoring, scheduling, and monitoring workflows. Airflow workflows are defined as Directed Acyclic Graphs(DAGs), implying that the tasks in a workflow should be executed in a specific order to ensure that one task is completed before starting the execution of the next.

Apache Airflow is a **workflow orchestration tool**. It lets you programmatically author, schedule, and monitor **data pipelines**. Airflow helps automate sequences of tasks—like extracting data, transforming it, and loading it somewhere (ETL jobs)—or running machine learning pipelines.



Core components

The following are the core components of Airflow:

- **Scheduler:** The scheduler is the heart of Airflow. It monitors all tasks and DAGs and schedules task instances to run as soon as their dependencies are fulfilled. When creating a new DAG run, the scheduler always picks the latest version of

that DAG. When a task is ready to run, the scheduler uses its configured executor to run the task on a worker.

- **API server:** A FastAPI server that serves the Airflow UI, as well as three APIs:
  - An API for workers to interact with when running task instances.
  - An internal API for the Airflow UI that provides updates on dynamic UI components such as the state of task instances and DAG runs.
  - The public Airflow REST API that users can interact with.
- **DAG processor:** The DAG processor is responsible for retrieving and parsing the files from the location determined by the configured DAG bundle(s).
- **Metadata database:** The Airflow metadata database stores information vital to Airflow's functioning, such as Airflow connections, serialized DAGs, and XCom information. It also contains the history of previous DAG runs and task instances alongside metadata about their states. The most common backend used for the Airflow metadata database is PostgreSQL. See the Airflow documentation for supported versions.
- **Triggerer:** A separate process which supports running asynchronous Python functions as part of trigger classes. The triggerer is needed to use deferrable operators and event-driven scheduling.