Apache Airflow

APACHE-AIRFLOW

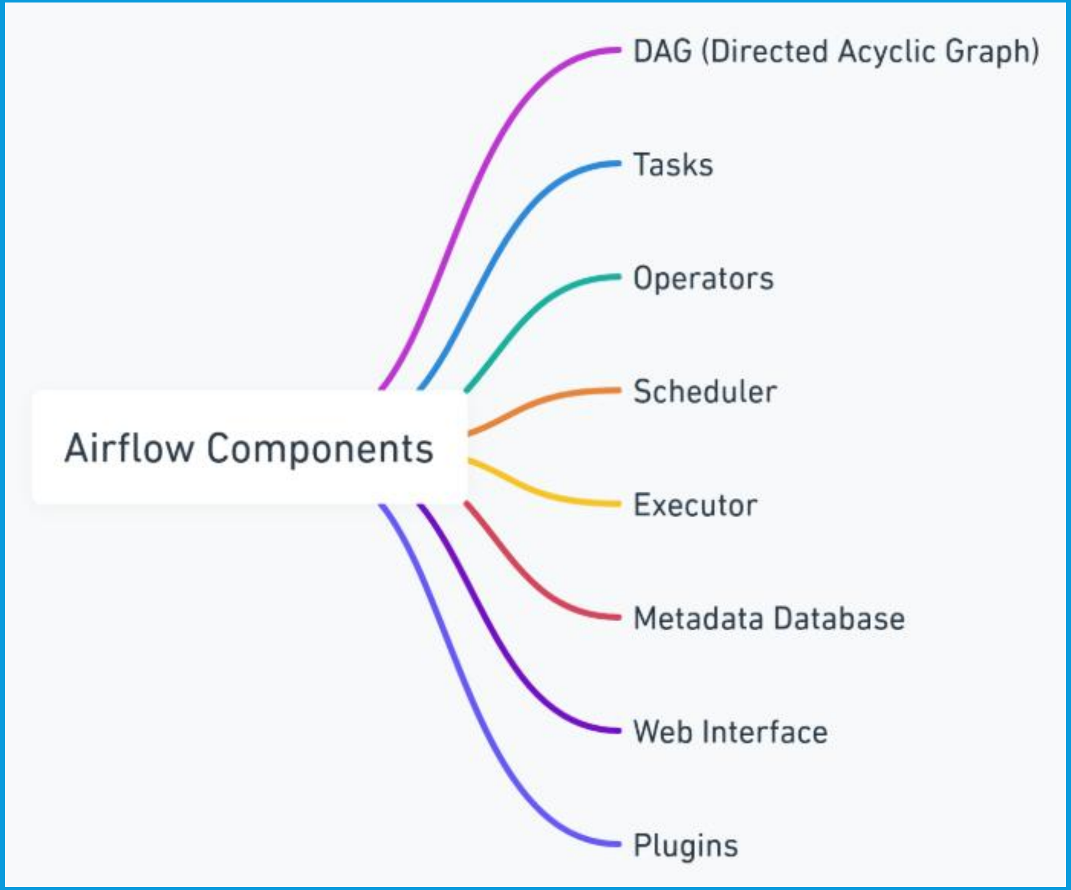
WHAT IS AIRFLOW

· Apache Airflow is an open-source platform used to programmatically create, schedule, and monitor workflows.

·Purpose: It automates complex workflows, making it easy to manage tasks that depend on each other.

·Example of a washing machine

COMPONENTS OF AIRFLOW

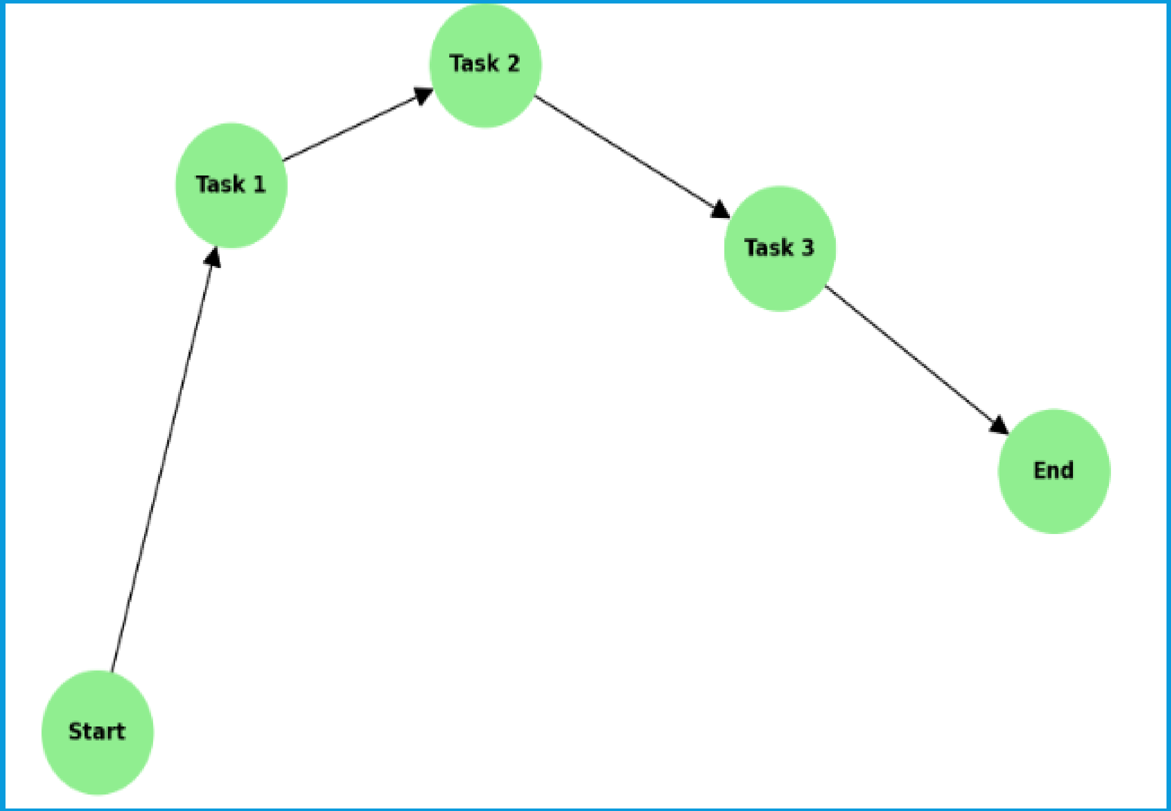


WHAT IS A WORKFLOW

- A workflow is a **sequence of tasks** or processes that are executed in a specific order to achieve a goal.
- **Steps or Tasks**: Individual components (e.g., downloading data, cleaning it, analyzing it) that make up the workflow.
- **Dependencies**: Tasks are executed based on their relationships or order of execution (e.g., Task B starts only after Task A finishes).
- **Automated Execution**: In Airflow, workflows are automated to save time and reduce manual effort.
- Example - Fetch data from an API → Clean the data → Load it into a database → Generate a report.

DAG (DIRECTED ACYCLIC GRAPH)

- A DAG is a structure that represents a workflow in Apache Airflow.
- It is a collection of tasks arranged in a specific order
- Tasks don’t loop back to previous tasks, meaning no cycles are allowed in the graph.



OPERATOR

- Each task in a DAG has an Operator that defines what the task needs to do.
- An Operator acts as an instruction for the task.
- It specifies the action the task will perform (e.g., running Python code, executing a SQL query, or waiting for a condition to be met).
- If a task is to run a Python function, you use the PythonOperator.
- PostgresOperator

SCHEDULER

- The scheduler decides when to start tasks based on their schedule intervals
- Continuously monitors all active DAGs to determine which tasks need to run.
- Ensures tasks are executed in the correct order by checking dependencies defined in the DAG.
- Reschedules tasks that fail, based on retry policies defined in the DAG.

EXECUTER

- The Executor in Apache Airflow is a component responsible for executing the tasks that the Scheduler schedules. It defines how and where the tasks will run (locally, in parallel, or distributed across multiple systems).
- Executes the tasks that are ready to run based on the scheduler's instructions.
- Manages the allocation of resources (CPU, memory) for task execution.

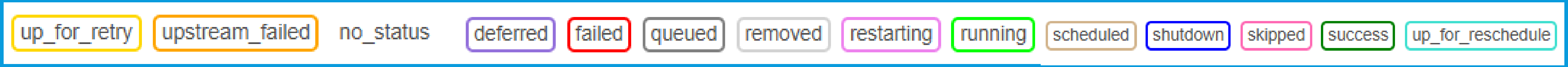
METADATA DATABASE

- The **Metadata Database** is a critical component of Apache Airflow. It stores all the information about workflows (DAGs), tasks, and their execution states. Without it, Airflow cannot track or manage workflows effectively.
- Details about DAGs, tasks, schedules, and dependencies.
- The **Scheduler**, **Executor**, and **Web Interface** interact with the metadata database to Read the current state of workflows, Update task statuses, Retrieve historical execution logs for monitoring.

KEY FEATURES OF APACHE AIRFLOW

- **Scalability: Can handle workflows of any size.**
- **Dynamic : Workflows are written in Python, allowing dynamic creation and modification.**
- **Extensibility: Provides many plugins and allows you to integrate with external systems like databases, cloud services, and APIs.**
- **Monitoring: Offers a web-based user interface to track progress, logs, and troubleshoot issues.**

TASK LIFE CYCLE



up_for_retry: The task failed and is waiting to retry based on the retry policy.
upstream_failed: The task didn't execute because an upstream task failed.
no_status: The task hasn't been assigned any status yet.
deferred: The task is waiting for an external trigger to proceed.
failed: The task execution failed.
queued: The task is waiting in the queue for a worker to pick it up.
removed: The task has been removed due to changes in the DAG structure.
restarting: The task is restarting after failure or manual intervention.
running: The task is currently being executed.
scheduled: The task is ready to be executed.
shutdown: The task failed and the DAG execution has stopped.
skipped: The task was intentionally skipped due to branching or conditions.
success: The task was successfully completed.
up_for_reschedule: The task is waiting for a specific interval to retry execution.

1. No Status:

- This is the initial state of a task. It doesn't yet have any defined state in the DAG lifecycle.

2. Scheduler:

- The **Scheduler** determines which tasks are ready to run based on their **schedule interval** and **dependencies**.
- The task transitions to **Schedule** if all its upstream tasks are successful or skipped.

3. Scheduled:

- The task is now ready to be executed and awaits assignment to an **Executor**.
- From this state, the task may:
  - Proceed to **Executor** to begin execution.
  - Be marked **Removed** if the DAG structure changes.
  - Be marked **Skipped** if a condition (like branching logic) skips the task.
  - Move to **Upstream Failed** if a dependency task fails.

4. Executor:

- The **Executor** is responsible for picking up the task and assigning it to a **Worker** for execution.
- The task transitions to **Queued** while it waits for a worker to become available.

5. Queued:

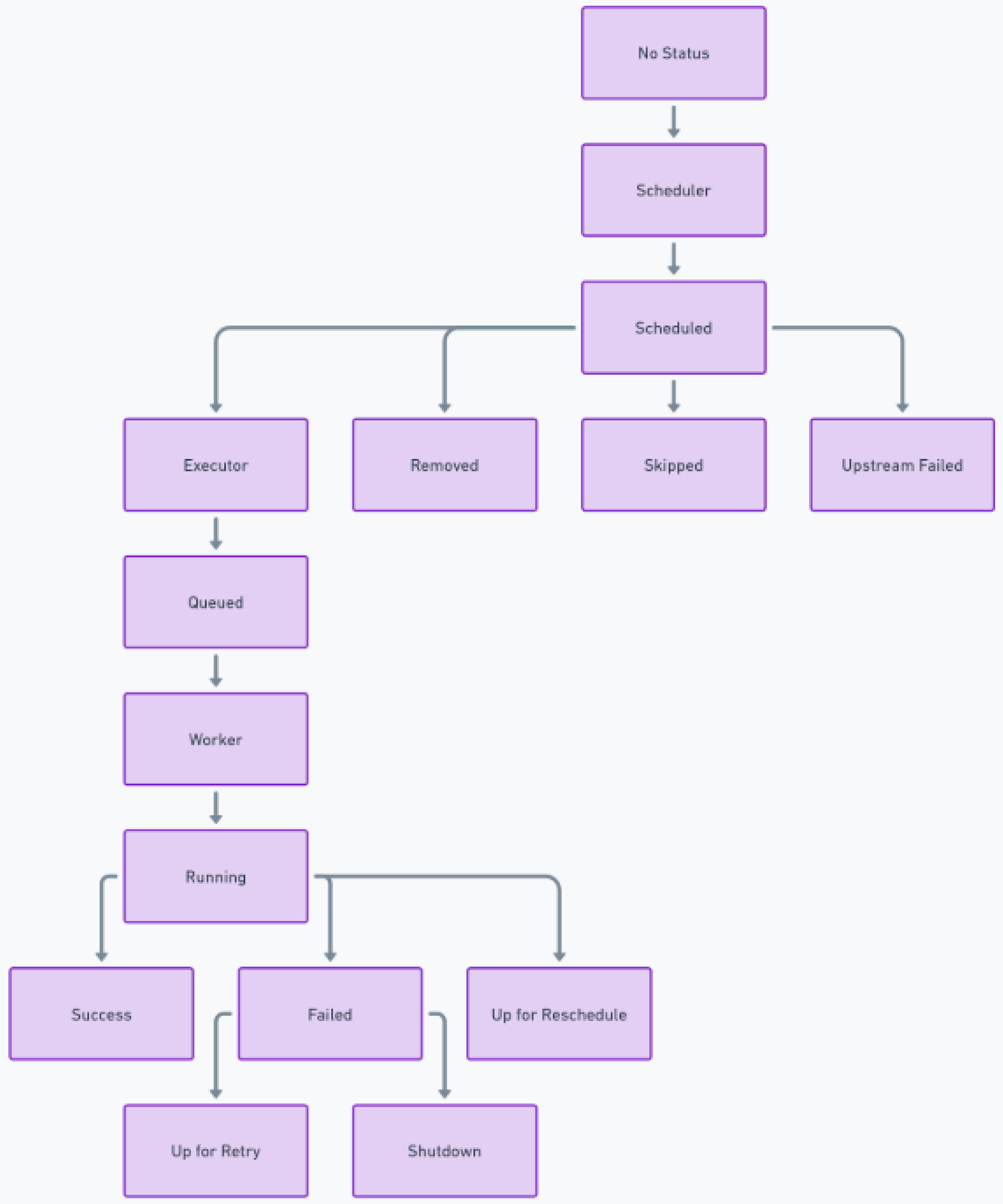
- The task waits in this state until a **Worker** can execute it.
- Once a worker is ready, the task moves to the **Running** state.

6. Worker:

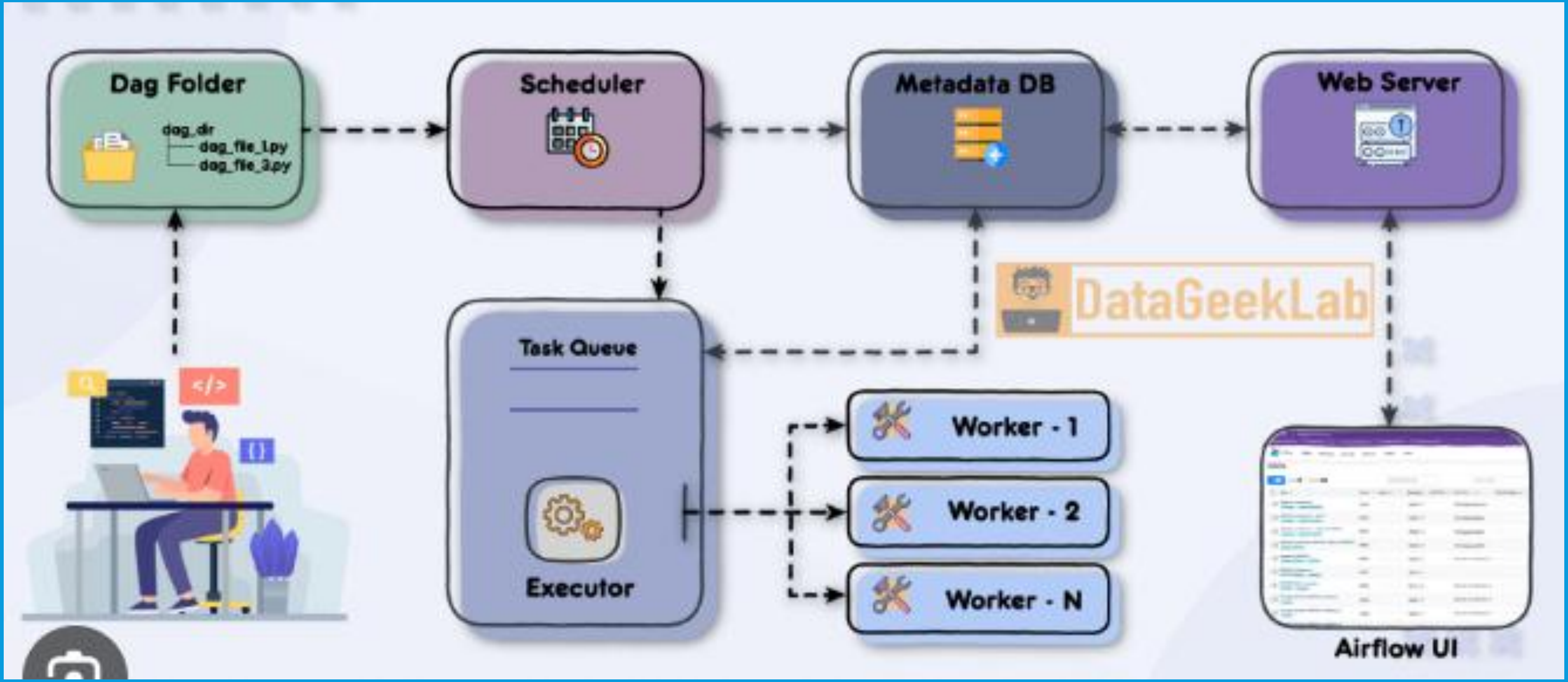
- A **Worker** picks up the task from the queue and begins execution.
- The task status changes to **Running** during this phase.

7. Running:

- The task is actively executing.
- After execution:
  - If the task completes successfully, it transitions to **Success**.
  - If the task fails, it transitions to **Failed**.
  - If retries are allowed, it moves to **Up for Retry**.
  - If it is a rescheduled task (e.g., sensors), it moves to **Up for Reschedule**.



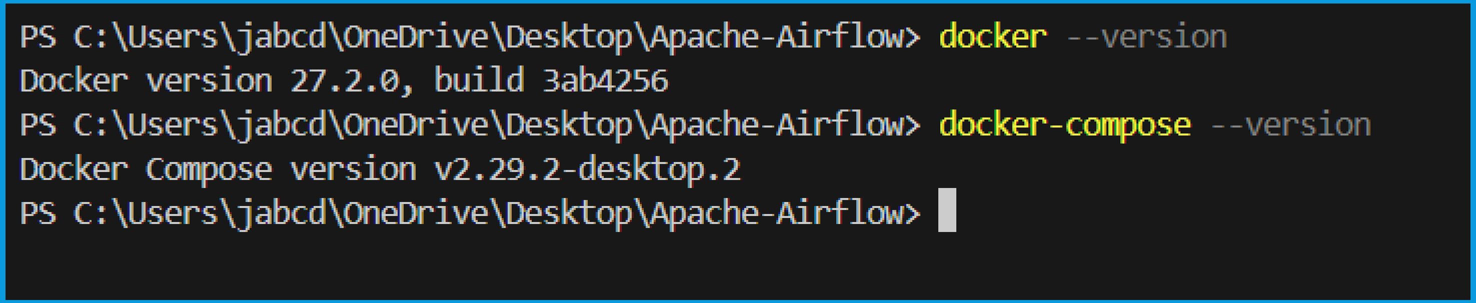
AIRFLOW ARCHITECTURE



Installing Apache-airflow in windows

INSTALLING APACHE INSIDE DOCKER

- Install docker and docker compose



```
PS C:\Users\jabcd\OneDrive\Desktop\Apache-Airflow> docker --version
Docker version 27.2.0, build 3ab4256
PS C:\Users\jabcd\OneDrive\Desktop\Apache-Airflow> docker-compose --version
Docker Compose version v2.29.2-desktop.2
PS C:\Users\jabcd\OneDrive\Desktop\Apache-Airflow>
```

- curl "https://airflow.apache.org/docs/apache-airflow/2.10.4/docker-compose.yaml" -o 'docker-compose.yaml'

mkdir ./dags

mkdir ./logs

mkdir ./plugins

mkdir ./config

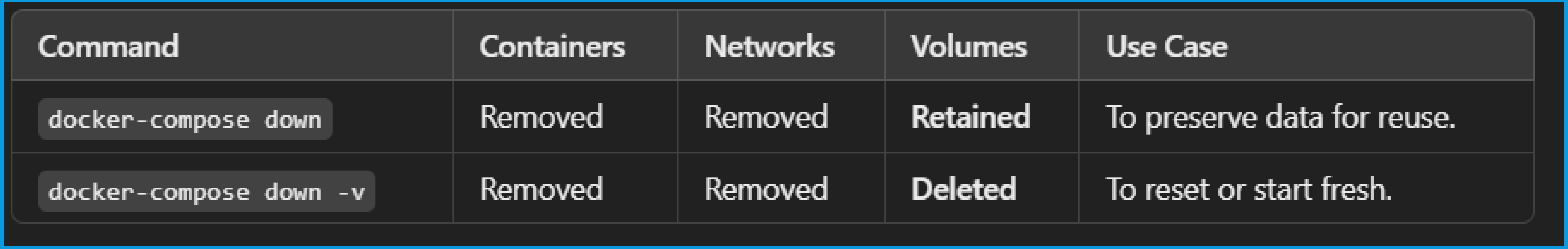
- docker compose up airflow-init

(The docker-compose up airflow-init command initializes Apache Airflow by setting up the **metadata database**, applying required **migrations**, and creating a default **admin user** for the web interface. It ensures the backend is ready for Airflow's scheduler and webserver to function properly. This step is typically required **once** during the initial setup or after resetting the database. You have to run this command only one time The airflow-init service needs to run **once** before starting Airflow for the first time. You only run this command again if, The database is deleted or reset. You make changes to the database configuration.)

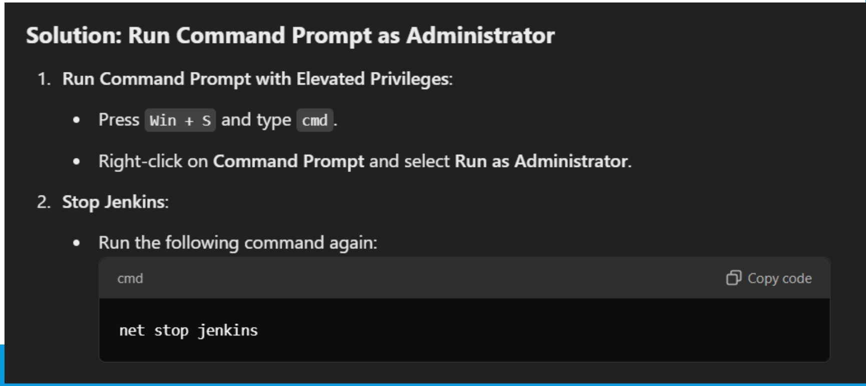
- (docker compose up) / (docker compose up –d)

- Without -d, you monitor services interactively but cannot use the same terminal for other commands. With -d, the services run in the background, freeing the terminal for other tasks

- (docker compose down)/ (docker-compose down –v)



Command	Containers	Networks	Volumes	Use Case
`docker-compose down`	Removed	Removed	**Retained**	To preserve data for reuse.
`docker-compose down -v`	Removed	Removed	**Deleted**	To reset or start fresh.



Solution: Run Command Prompt as Administrator

1. **Run Command Prompt with Elevated Privileges:**
   - Press `Win + S` and type `cmd`.
   - Right-click on **Command Prompt** and select **Run as Administrator**.
2. **Stop Jenkins:**
   - Run the following command again:

```
net stop jenkins
```

Airflow username and password is - airflow

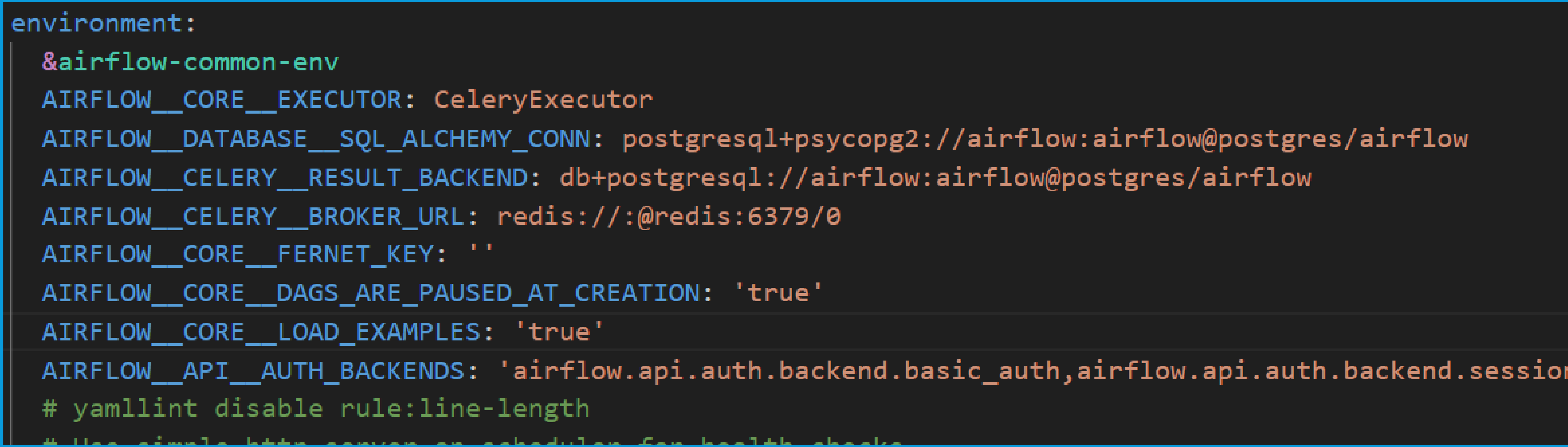
- **Without -v**: If you stop Airflow using docker-compose down, your **Metadata DB volume** persists, so you can restart later without losing data.
- **With -v**: If you use docker-compose down -v, the **Metadata DB volume** is deleted, and you'll need to reinitialize Airflow with docker-compose up airflow-init.

APACHE-AIRFLOW

- Remove all the Examples DAGS

REMOVE ALL THE EXAMPLES DAGS

- docker-compose down –v



```
environment:
  &airflow-common-env
  AIRFLOW__CORE__EXECUTOR: CeleryExecutor
  AIRFLOW__DATABASE__SQL_ALCHEMY_CONN: postgresql+psycopg2://airflow:airflow@postgres/airflow
  AIRFLOW__CELERY__RESULT_BACKEND: db+postgresql://airflow:airflow@postgres/airflow
  AIRFLOW__CELERY__BROKER_URL: redis://:@redis:6379/0
  AIRFLOW__CORE__FERNET_KEY: ''
  AIRFLOW__CORE__DAGS_ARE_PAUSED_AT_CREATION: 'true'
  AIRFLOW__CORE__LOAD_EXAMPLES: 'true'
  AIRFLOW__API__AUTH_BACKENDS: 'airflow.api.auth.backend.basic_auth,airflow.api.auth.backend.sessio
  # yamllint disable rule:line-length
```

docker-compose up -d

APACHE-AIRFLOW

- First DAG from Scratch

ML END TO END PIPELINE

1 – EXTRACTING THE DATA

2- TRANSFORMING THE DATA

3- TRAINING THE MODEL

4 – TESTING THE MODEL