



Apache  
**Airflow**

**Orchestration**

**Automation**

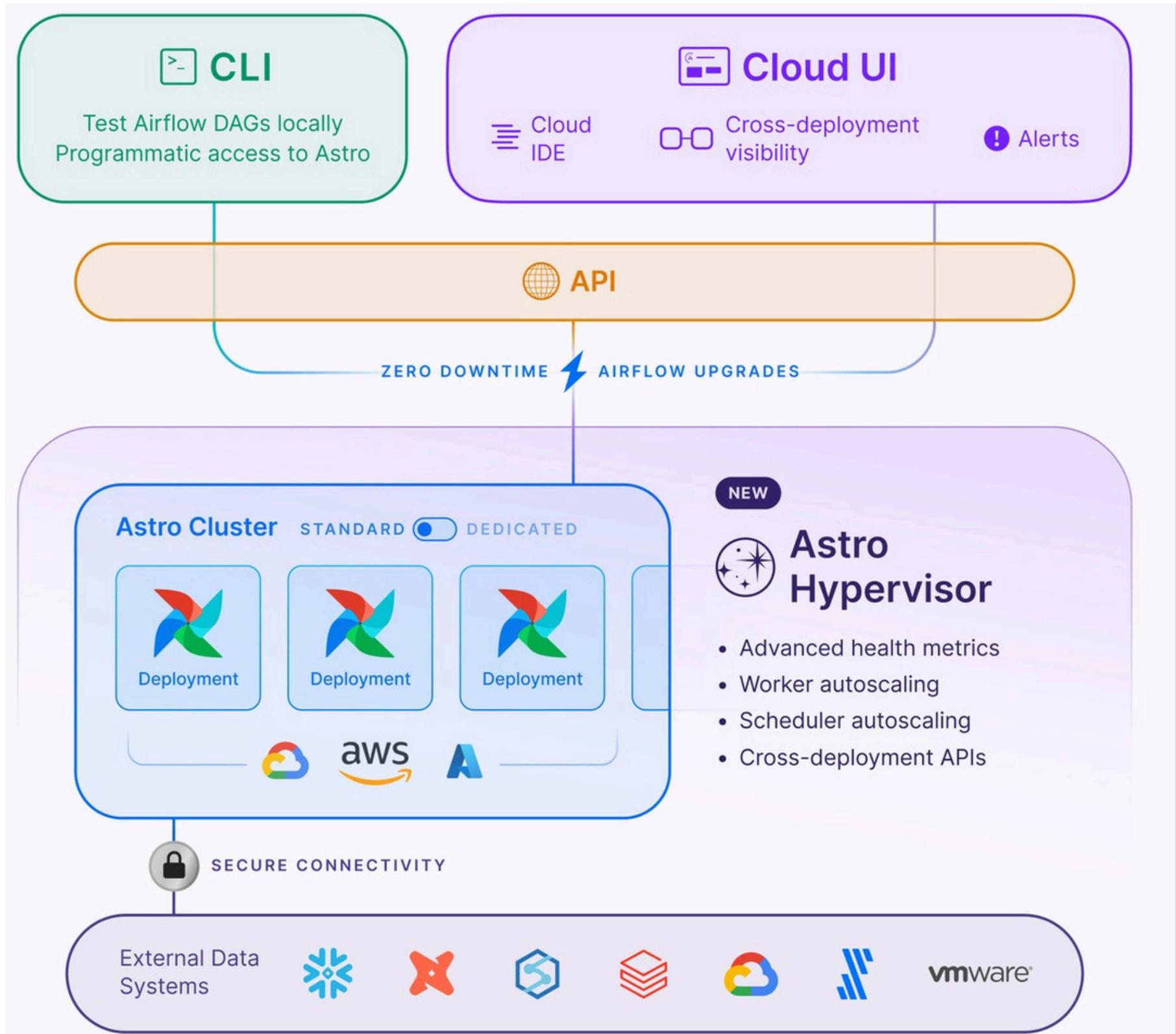
**Monitoring**

**Scheduling**

**Pipeline**

**Python**

# Airflow Architecture (Astro)



# Apache Airflow

Apache Airflow is an open-source tool for creating, scheduling, and monitoring workflows as code, using DAGs (Directed Acyclic Graphs) to define task dependencies and manage execution, retries, and failures.

## Airflow Web UI (Enable/Disable DAGS ,Monitor,Refresh)

The screenshot shows the Apache Airflow Web UI interface. At the top, there is a navigation bar with links for Airflow, DAGs, Cluster Activity, Datasets, Security, Browse, Admin, and Docs. On the far right, it shows the time as 14:17 UTC and a user icon labeled AU. Below the navigation bar, the title "DAGs" is displayed. The main content area is titled "1 Permanent Table" and "5 Hybrid Tables" and "7 Dynamic tables". The "Permanent Table" section has a heading "dataset\_consumes\_1" and shows a table with columns: DAG, Owner, Runs, Schedule, Last Run, Next Run, Recent Tasks, and Actions. There are 53 DAGs listed in total, with 9 Active, 44 Paused, 7 Running, and 1 Failed. A search bar and an "Auto-refresh" button are also present. The table lists various DAG names along with their owner (airflow), run status, schedule (e.g., @daily, None), last run date, next run date, recent tasks count, and actions like refresh and delete.

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions
dataset_consumes_1	airflow	0 0 0 0 1	2024-03-21, 14:15:47	C:\.../dag1/output_1.txt		0	refresh delete
dataset_consumes_1_and_2	airflow	0 0 0 0 1	Daily	2024-03-21, 14:15:47	2024-03-22, 00:00:00	2	refresh delete
dataset_consumes_1_never_scheduled	airflow	0 0 0 0 1	Never		2024-03-22, 00:00:00	0	refresh delete
dataset_consumes_unknown_never_scheduled	airflow	0 0 0 0 1	Java	2024-03-21, 14:15:47	2024-03-22, 00:00:00	2	refresh delete
dataset_produces_1	airflow	0 0 0 0 1	@daily	2024-03-20, 00:00:00	2024-03-21, 00:00:00	1	refresh delete
dataset_produces_2	airflow	0 0 0 0 1	None		2024-03-21, 00:00:00	1	refresh delete
example_bash_operator	airflow	0 0 1 0 1	0 0 * * 1	2024-03-20, 00:00:00	2024-03-21, 00:00:00	1	refresh delete
example_branch_datetime_operator	airflow	0 1 0 0 1	@daily	2024-03-20, 00:00:00	2024-03-21, 00:00:00	1	refresh delete
example_branch_datetime_operator_2	airflow	0 1 0 0 1	@daily	2024-03-20, 00:00:00	2024-03-21, 00:00:00	1	refresh delete
example_branch_datetime_operator_3	airflow	0 1 0 0 1	@daily	2024-03-20, 00:00:00	2024-03-21, 00:00:00	1	refresh delete
example_branch_dop_operator_v3	airflow	0 1 2 0 1	*/* * * * 1	2024-03-21, 14:16:00	2024-03-21, 14:15:00	2	refresh delete
example_branch_labels	airflow	0 0 1 0 1	@daily	2024-03-20, 00:00:00	2024-03-21, 00:00:00	4	refresh delete

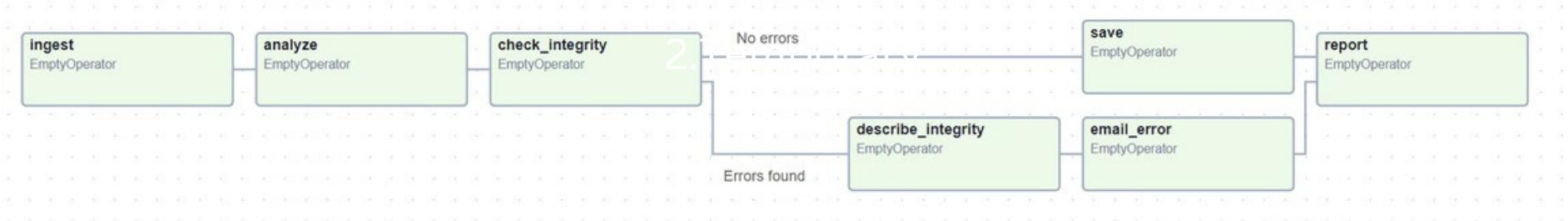
# Airflow DAG

## DAGs (Directed Acyclic Graphs) -

A DAG represents the overall structure of a workflow in Airflow

tasks in a workflow, with specified execution order and An Airflow DAG (Directed Acyclic Graph) is a collection of dependencies. Written in Python, it ensures tasks run in

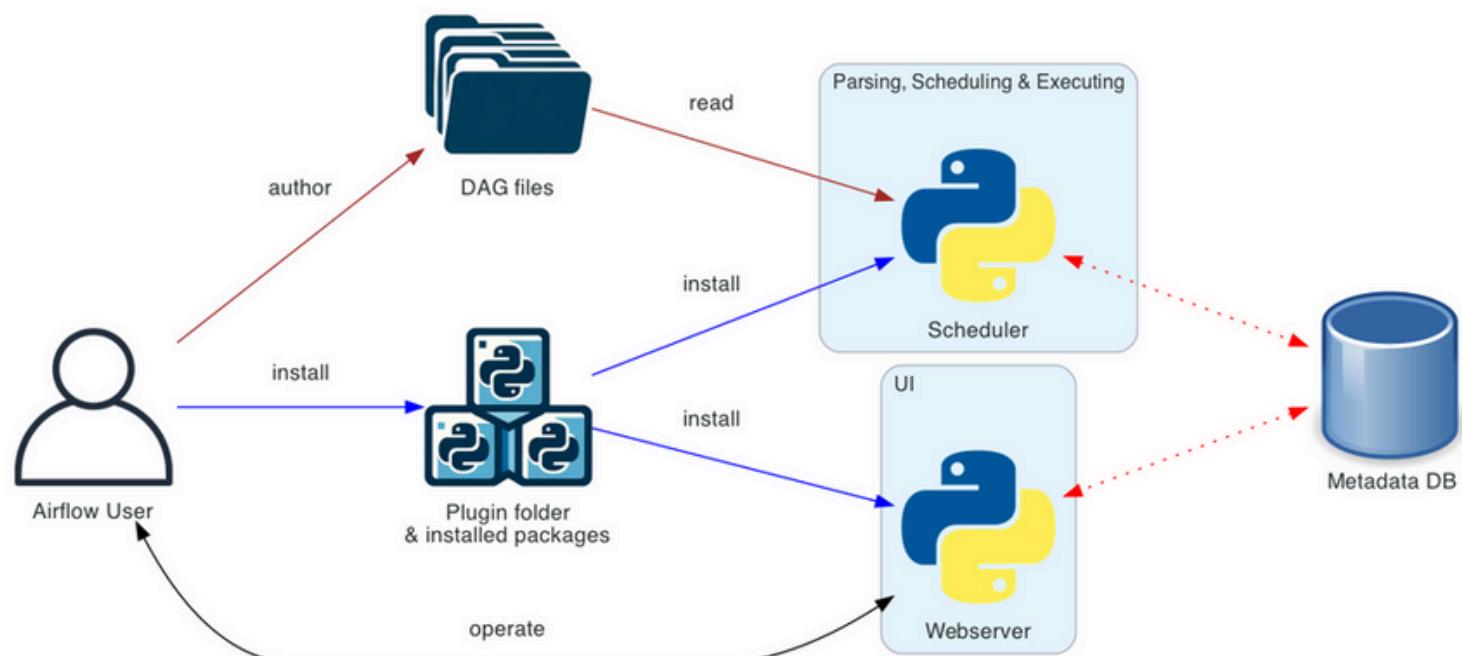
workflows ,including scheduling and logic. sequence without loops, allowing users to define complex



A DAG defines task dependencies, determining their execution order, while tasks specify the actions, such as fetching data, running analyses, or triggering other systems.

# Airflow Architecture

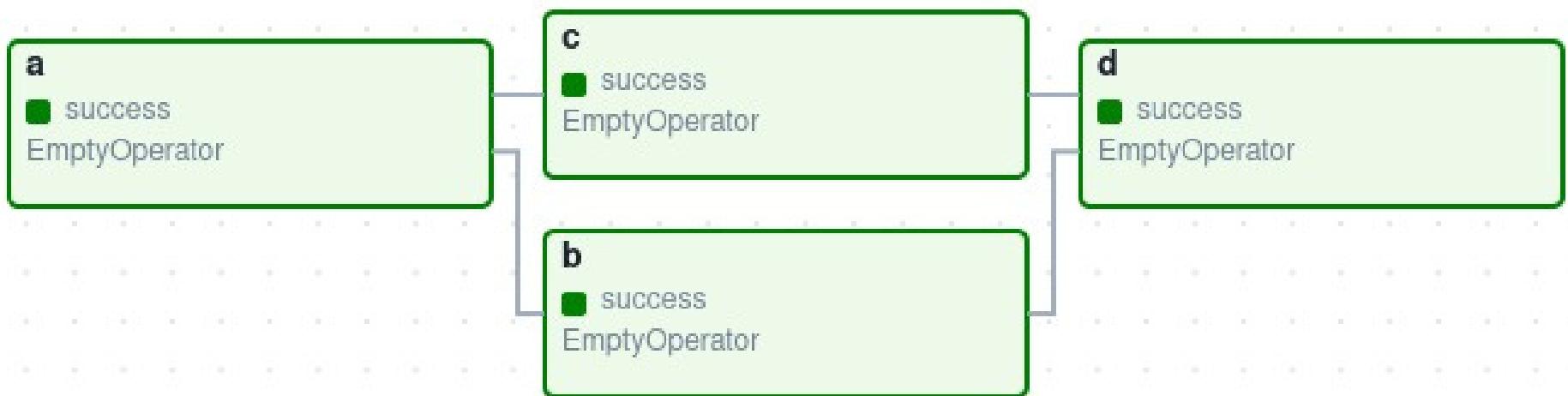
The diagrams below show different ways to deploy Airflow – gradually from the simple “one machine” and single person deployment, to a more complex deployment with separate components, separate user roles and finally with more isolated security perimeters.



# Create a DAG

## Design a Basic DAG –

It defines four Tasks – A, B, C, and D – and dictates the order in which they have to run, and which tasks depend on what others. It will also say how often to run the DAG – maybe “every 5 minutes starting tomorrow”, or “every day since January 1st, 2020”.



# Python program DAG

## Part 1/2

```
# Import the necessary modules from the Airflow library
from airflow import DAG
from airflow.operators.python_operator import PythonOperator
from datetime import datetime, timedelta
import time

# Define a Python function to print the current date
def print_current_date():
    print(f"Current date and time is: {datetime.now()}")

# Define a Python function to simulate sleeping for 5 seconds
def sleep_for_seconds():
    print("Sleeping for 5 seconds...")
    time.sleep(5)
    print("Woke up after 5 seconds")

# Default arguments that will be passed to each task in the DAG
# These settings are applied to the entire DAG and can be overridden for individual tasks
default_args = {
    'owner': 'airflow', # Owner of the DAG
    'depends_on_past': False, # Don't depend on previous runs
    'start_date': datetime(2024, 10, 18), # Start date of the DAG
    'email_on_failure': False, # Do not send an email when a task fails
    'email_on_retry': False, # Do not send an email when a task retries
    'retries': 1, # Number of retry attempts in case of failure
    'retry_delay': timedelta(minutes=5), # Wait 5 minutes between retries
}
```

# Python program DAG

## Part 2//2

```
# Initialize the DAG (Directed Acyclic Graph)
# The DAG object is created with a unique name, default arguments, and a schedule interval
with DAG(
    'simple_dag', # Name of the DAG
    default_args=default_args, # Default arguments for the DAG
    description='A simple tutorial DAG', # Description of the DAG
    schedule_interval=timedelta(days=1), # The DAG will run once every day
) as dag:

    # Define the first task, which prints the current date
    # We use PythonOperator to run a Python function as a task
    task1 = PythonOperator(
        task_id='print_date', # Unique task identifier
        python_callable=print_current_date, # The function to be executed
    )

    # Define the second task, which will sleep for 5 seconds
    task2 = PythonOperator(
        task_id='sleep_task', # Unique task identifier
        python_callable=sleep_for_seconds, # The function to be executed
    )

    # Set task dependencies (task1 will run first, then task2)
    task1 >> task2 # This means task1 must complete before task2 starts
```

Then Deploy python code in GIT/AWS S3 based on  
repo (it's very simple)

DAG will display in Airflow UI once you place DAG  
code

# Ways to View and Manage DAGs in Airflow

## Airflow DAGs View

The screenshot shows the Airflow web interface under the 'DAGs' tab. At the top, there are filters for 'All', 'Active', and 'Paused' DAGs, along with search bars for 'Filter DAGs by tag' and 'Search DAGs'. A 'Recent Tasks' section is also present. The main table lists ten DAGs:

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions	Links
data_quality_example_dag	airflow	8 / 9	1 day, 0:00:00	2022-08-18, 10:46:37	2022-08-27, 19:58:32	1 / 1 / 1 / 1		
dynamic_test	airflow	14 / 15	1 day, 0:00:00	2022-08-19, 13:56:04	2022-08-27, 19:58:32	3 / 4 / 4 / 4		
example-dag-complex	airflow	7 / 7	1 day, 0:00:00	2022-08-28, 19:53:40	2022-08-28, 19:52:52	22 / 22 / 22 / 22		
load_connections_dag	airflow	1 / 1	1 day, 0:00:00	2022-08-17, 19:55:18	2022-08-27, 19:58:04	1 / 1 / 1 / 1		
mapping_xcoms_dag	airflow	2 / 12	1 day, 0:00:00	2022-08-27, 19:44:42	2022-08-28, 19:44:42	4 / 12 / 12 / 12		
multiple_parameters_example	airflow	4 / 9	1 day, 0:00:00	2022-08-27, 19:44:42	2022-08-28, 19:44:42	2 / 10 / 10 / 10		
snowflake_to_slack_dag	airflow	11 / 3	1 day, 0:00:00	2022-08-18, 10:39:30	2022-08-27, 19:58:04	2 / 2 / 2 / 2		
taskflow_example	airflow	0 / 0	None			0 / 0 / 0 / 0		
zipping_examples	airflow	7 / 24	1 day, 0:00:00	2022-08-27, 19:44:42	2022-08-28, 19:44:42	1 / 4 / 4 / 4		

## Airflow Graph View

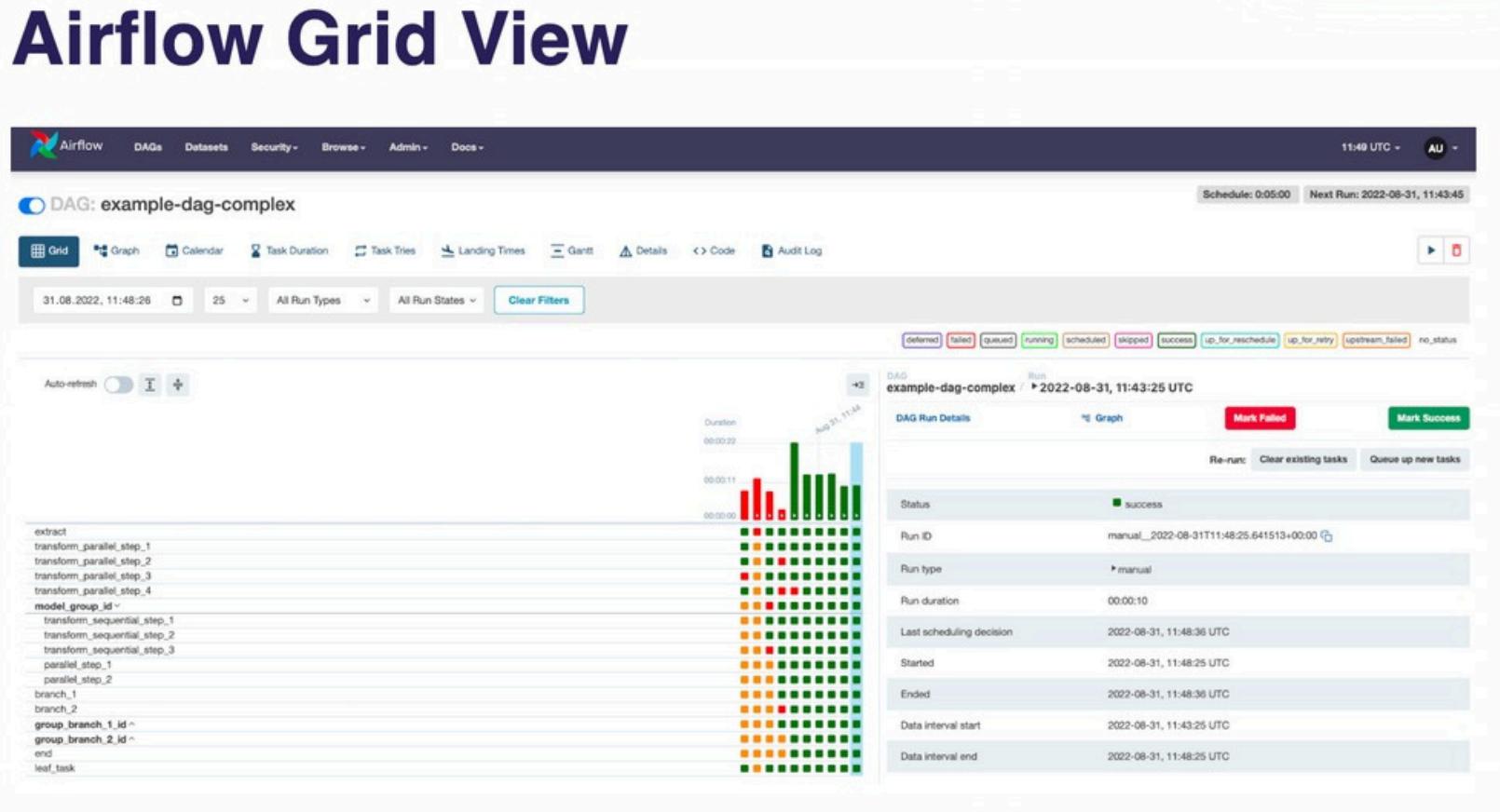
The screenshot shows the Airflow web interface under the 'Graph' tab for the 'example-dag-complex' DAG. The top bar includes a success status, schedule information, and a next run timestamp. The main area displays the task dependency graph:

```
graph TD; extract --> transform_parallel_step_1; extract --> transform_parallel_step_2; extract --> transform_parallel_step_3; extract --> transform_parallel_step_4; transform_parallel_step_1 --> model_group_id; transform_parallel_step_2 --> model_group_id; transform_parallel_step_3 --> model_group_id; transform_parallel_step_4 --> model_group_id; model_group_id --> leaf_task; model_group_id --> branch_1; model_group_id --> branch_2; branch_1 --> group_branch_1_id; branch_2 --> group_branch_2_id; group_branch_1_id --> end; group_branch_2_id --> end;
```

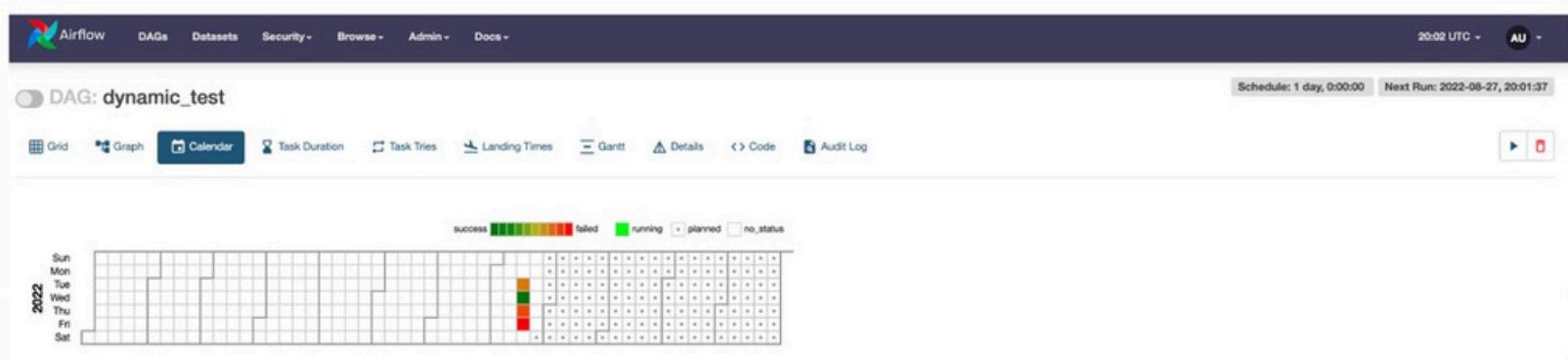
Task states are indicated by colors: green for running, blue for succeeded, and red for failed. A legend at the bottom shows state abbreviations: deferred, failed, queued, running, scheduled, skipped, success, up\_for\_reschedule, up\_for\_retry, upstream\_failed, and no\_status.

# Ways to View and Manage DAGs in Airflow

## Airflow Grid View



## Airflow Calendar View



**Anil Patel**  
@aniltppatel



Architect-Data  
Engineering & Analytics  
Career Transition Coach

# Ways to View and Manage DAGs in Airflow

## Airflow Browse Tab

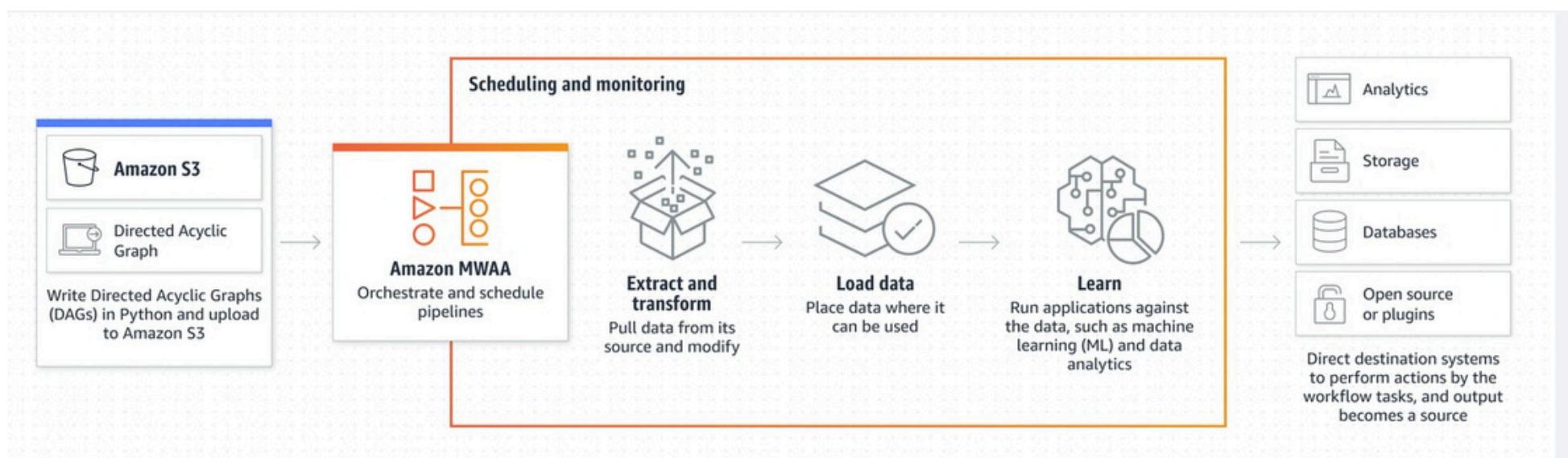
The screenshot shows the Airflow web interface with the 'Browse' tab selected. The main area displays a table of DAGs. On the left, a sidebar lists navigation options: DAGs, Datasets, Security, Browse, Admin, and Docs. Under 'DAGs', there are filters for All (9), Active (4), and Paused (5). A dropdown menu for 'DAG' is open, showing 'data\_quality\_example\_dag'. The main table has columns for 'Runs' (with 8 green circles), 'Schedule' (1 day, 0:00:00), 'Last Run' (2022-08-18, 10:46:37), 'Next Run' (2022-08-27, 20:05:12), and 'Recent Tasks' (10 tasks with various status icons). Action and Link buttons are available for each row.

# cloud providers that offer Airflow services



## Amazon Web Services (AWS) –

Amazon Managed Workflows for Apache Airflow (MWAA) is a fully managed Airflow service on AWS



## **cloud providers that offer Airflow services**



### **Microsoft Azure –**

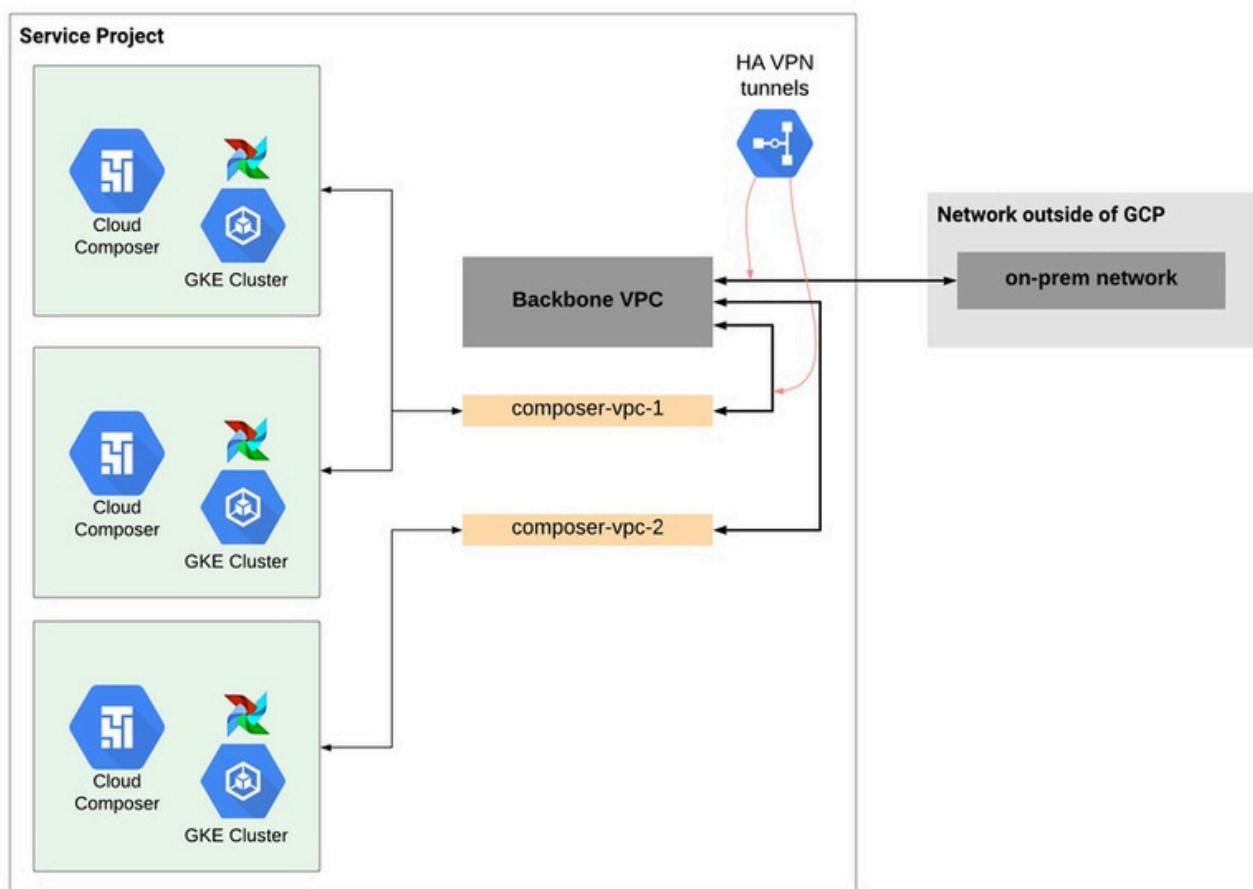
Apache Airflow integrates with Microsoft Azure services through Microsoft. azure provider. You can install any provider package by editing the airflow environment from the Azure Data Factory UI

Apache Airflow™ on Astro – An Azure Native ISV Service

# cloud providers that offer Airflow services

## Google Cloud Platform (GCP)-

Cloud Composer is a fully managed workflow orchestration service that runs on GCP. It's built on the Apache Airflow open source project



## **cloud providers that offer Airflow services**



### **Astronomer –**

A managed Airflow service that allows data teams to build, run, and manage data pipelines as code. Astronomer can run Airflow on AWS, GCP, Azure, or on-premise

# Introduction to Apache Airflow

INTRODUCTION TO APACHE AIRFLOW IN PYTHON



# What is data engineering?

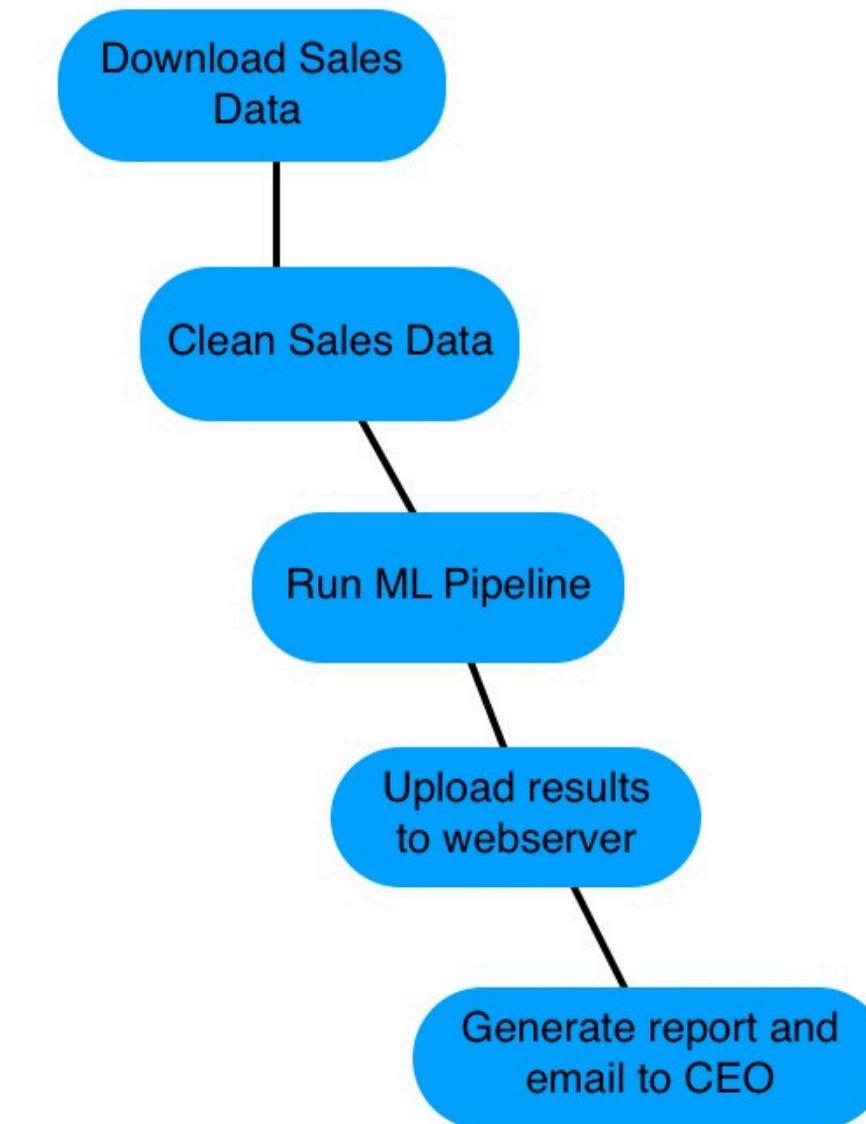
*Data engineering* is:

- Taking any action involving data and turning it into a reliable, repeatable, and maintainable process.

# What is a workflow?

A *workflow* is:

- A set of steps to accomplish a given data engineering task
  - Such as: downloading files, copying data, filtering information, writing to a database, etc
- Of varying levels of complexity
- A term with various meaning depending on context



# What is Airflow?

*Airflow* is a platform to program workflows, including:

- Creation
- Scheduling
- Monitoring



Apache  
**Airflow**

# Airflow continued...

- Can implement programs from any language, but workflows are written in Python
- Implements workflows as DAGs: Directed Acyclic Graphs
- Accessed via code, command-line, or via web interface / REST API



<sup>1</sup> <https://airflow.apache.org/docs/stable/>

# Other workflow tools

Other tools:

- ADF
- SSIS
- Dagster

Prefect

Mage

Apache Oozie

Informatica

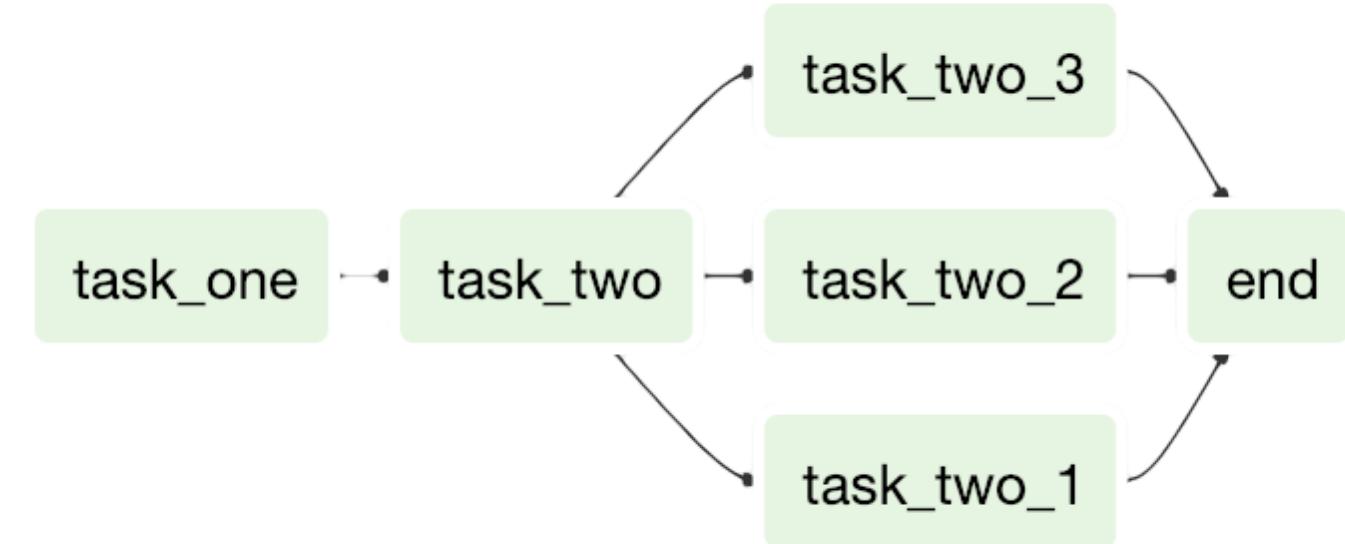
# Quick introduction to DAGs

A *DAG* stands for *Directed Acyclic Graph*

- In Airflow, this represents the set of tasks that make up your workflow.
- Consists of the tasks and the dependencies between tasks.
- Created with various details about the
- DAG, including the name, start date, owner, etc.

Further depth in the next lesson.

- 



# DAG code example

Simple DAG definition:

```
etl_dag = DAG(  
    dag_id='etl_pipeline',  
    default_args={"start_date": "2024-01-08"}  
)
```

# Running a workflow in Airflow

Running a simple Airflow task

```
airflow tasks test <dag_id> <task_id> [execution_date]
```

Using a DAG named *example-etl* , a task named *download-file* on 2024-01-10:

```
airflow tasks test example-etl download-file 2024-01-10
```

# **Let's practice!**

**INTRODUCTION TO APACHE AIRFLOW IN PYTHON**

# Airflow DAGs

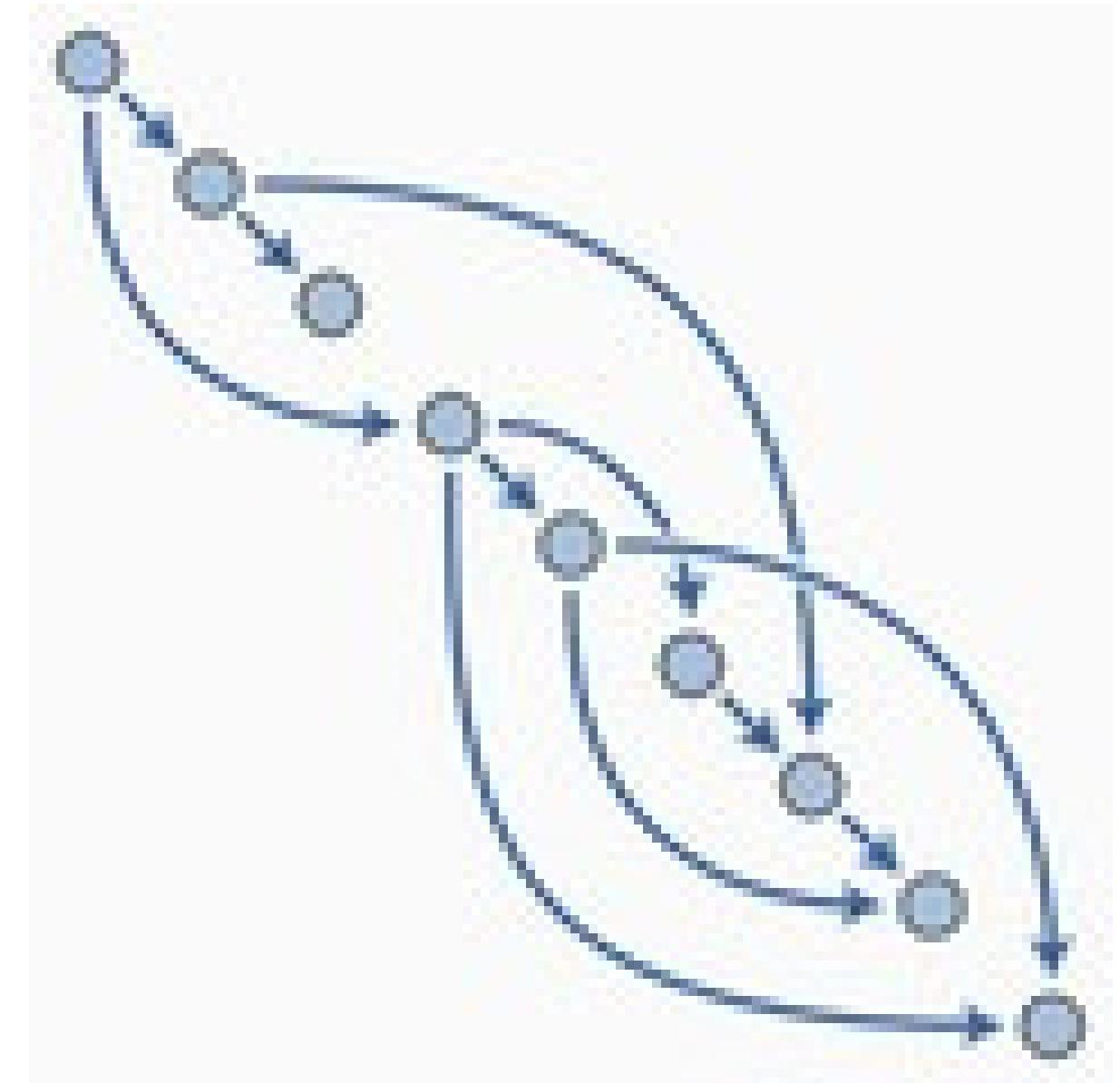
INTRODUCTION TO APACHE AIRFLOW IN PYTHON



# What is a DAG?

DAG, or *Directed Acyclic Graph*:

- *Directed*, there is an inherent flow representing dependencies between components.
- *Acyclic*, does not loop / cycle / repeat.
- *Graph*, the actual set of components.
- Seen in Airflow, Apache Spark, dbt



# DAG in Airflow

Within Airflow, DAGs:

- Are written in Python (but can use components written in other languages).
- Are made up of components (typically *tasks*) to be executed, such as operators, sensors, etc.
- Contain dependencies defined explicitly or implicitly.
  - ie, Copy the file to the server before trying to import it to the database service.

# Define a DAG

Example DAG:

```
from airflow import DAG

from datetime import datetime
default_arguments = {
    'owner': 'jdoe',
    'email': 'jdoe@datacamp.com',
    'start_date': datetime(2020, 1, 20)
}

with DAG('etl_workflow', default_args=default_arguments ) as etl_dag:
```

# Define a DAG (before Airflow 2.x)

Example DAG:

```
from airflow import DAG

from datetime import datetime
default_arguments = {
    'owner': 'jdoe',
    'email': 'jdoe@datacamp.com',
    'start_date': datetime(2020, 1, 20)
}

etl_dag = DAG('etl_workflow', default_args=default_arguments )
```

# DAGs on the command line

Using `airflow` :

- The `airflow` command line program contains many subcommands.
- `airflow -h` for descriptions.
- Many are related to DAGs.
- `airflow dags list` to show all recognized DAGs.

# Command line vs Python

Use the command line tool to:

- Start Airflow processes
- Manually run DAGs / Tasks
- Get logging information from Airflow

Use Python to:

- Create a DAG
- Edit the individual properties of a DAG

# **Let's practice!**

**INTRODUCTION TO APACHE AIRFLOW IN PYTHON**

# Airflow web interface

INTRODUCTION TO APACHE AIRFLOW IN PYTHON



# DAGs view

Airflow DAGs Cluster Activity Datasets Security Browse Admin Docs 22:46 UTC Log In

## DAGs

All 2 Active 0 Paused 2 Running 0 Failed 0 Filter DAGs by tag Search DAGs

Auto-refresh

i	DAG ▾	Owner	Runs i	Schedule	Last Run ▾ i	Next Run ▾ i	Recent Tasks i
	<a href="#">example_dag</a>	airflow	 1 day, 0:00:00		2024-01-10, 00:00:00	i	
	<a href="#">update_state</a>	airflow	 1 day, 0:00:00		2024-01-10, 00:00:00	i	

# DAGs view DAGs

The screenshot shows the Apache Airflow interface for viewing DAGs. At the top, there is a navigation bar with links for Airflow, DAGs, Cluster Activity, Datasets, Security, Browse, Admin, Docs, the current time (22:46 UTC), and a Log In button. Below the navigation bar, the title "DAGs" is displayed. Underneath the title, there are several filter and search options: "All 2" (selected), "Active 0", "Paused 2", "Running 0", "Failed 0", "Filter DAGs by tag", and "Search DAGs". There is also an "Auto-refresh" toggle switch and a refresh button. The main content area displays a table of DAGs. The first row of the table has a red border around the "DAG" column header and the first data cell. The table columns include: DAG, Owner, Runs, Schedule, Last Run, Next Run, and Recent Tasks. Two DAG entries are listed: "example\_dag" and "update\_state", both owned by "airflow". Both DAGs have a schedule of "1 day, 0:00:00" and their last run and next run are both at "2024-01-10, 00:00:00".

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks
example_dag	airflow	4	1 day, 0:00:00	2024-01-10, 00:00:00	2024-01-10, 00:00:00	10
update_state	airflow	4	1 day, 0:00:00	2024-01-10, 00:00:00	2024-01-10, 00:00:00	10

# DAGs view owner

The screenshot shows the Apache Airflow web interface under the 'DAGs' tab. At the top, there is a navigation bar with links for 'DAGs', 'Cluster Activity', 'Datasets', 'Security', 'Browse', 'Admin', 'Docs', the current time '22:46 UTC', and a 'Log In' button. Below the navigation bar, the page title is 'DAGs'. There are several filter and search options: 'All 2' (selected), 'Active 0', 'Paused 2', 'Running 0', 'Failed 0', 'Filter DAGs by tag', and 'Search DAGs'. A 'Auto-refresh' toggle switch is also present.

The main content area displays a table of DAGs. The columns are: DAG (dropdown), Owner (dropdown), Runs (info icon), Schedule, Last Run (info icon), Next Run (info icon), and Recent Tasks (info icon). Two DAGs are listed:

- example\_dag**: Owner is airflow. The 'Owner' column for this row is highlighted with a red box.
- update\_state**: Owner is airflow.

Each DAG row includes a toggle switch, the DAG name, and a series of four small circles representing recent task states.

# DAGs view runs

The screenshot shows the Apache Airflow interface for viewing DAG runs. At the top, there is a navigation bar with links for Airflow, DAGs, Cluster Activity, Datasets, Security, Browse, Admin, Docs, the current time (22:46 UTC), and a Log In button. Below the navigation bar, the title "DAGs" is displayed. Underneath the title, there are several filter and search options: "All 2" (selected), "Active 0", "Paused 2", "Running 0", "Failed 0", "Filter DAGs by tag", and "Search DAGs". There is also an "Auto-refresh" toggle switch with a refresh icon.

The main content area displays a table of DAG runs. The columns are labeled: DAG, Owner, Runs, Schedule, Last Run, Next Run, and Recent Tasks. Two DAGs are listed:

- example\_dag** (Owner: airflow):
  - Runs: 4 (highlighted with a red box)
  - Schedule: 1 day, 0:00:00
  - Last Run: 2024-01-10, 00:00:00
  - Next Run: 2024-01-10, 00:00:00
  - Recent Tasks: 10
- update\_state** (Owner: airflow):
  - Runs: 4
  - Schedule: 1 day, 0:00:00
  - Last Run: 2024-01-10, 00:00:00
  - Next Run: 2024-01-10, 00:00:00
  - Recent Tasks: 10

# DAGs view schedule

The screenshot shows the Apache Airflow web interface for viewing DAG schedules. At the top, there is a navigation bar with links for Airflow, DAGs, Cluster Activity, Datasets, Security, Browse, Admin, Docs, the current time (22:46 UTC), and a Log In button. Below the navigation bar, the title "DAGs" is displayed.

Below the title, there are several filter and search options:

- Status filters: All (2), Active (0), Paused (2), Running (0), Failed (0).
- Filter DAGs by tag.
- Search DAGs.

There is also an "Auto-refresh" toggle switch and a refresh button.

The main content area displays a table of DAGs with the following columns:

	DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks
<input type="checkbox"/>	example_dag	airflow	○ ○ ○ ○	1 day, 0:00:00	2024-01-10, 00:00:00	2024-01-10, 00:00:00	○ ○ ○ ○ ○ ○ ○ ○
<input type="checkbox"/>	update_state	airflow	○ ○ ○ ○	1 day, 0:00:00	2024-01-10, 00:00:00	2024-01-10, 00:00:00	○ ○ ○ ○ ○ ○ ○ ○

The "Schedule" column for both DAGs shows the value "1 day, 0:00:00". The "example\_dag" row is highlighted with a red box around its "Schedule" cell.

# DAGs view last run

The screenshot shows the Apache Airflow web interface for viewing DAGs. At the top, there is a navigation bar with links for Airflow, DAGs, Cluster Activity, Datasets, Security, Browse, Admin, Docs, the current time (22:46 UTC), and a Log In button. Below the navigation bar, the title "DAGs" is displayed. Underneath the title, there are several filter buttons: "All 2" (highlighted in blue), "Active 0", "Paused 2", "Running 0", and "Failed 0". There are also buttons for "Filter DAGs by tag" and "Search DAGs". A toggle switch for "Auto-refresh" is shown next to a refresh icon. The main content area displays two DAG entries: "example\_dag" and "update\_state". Each entry includes columns for "Owner" (airflow), "Runs" (with four circular icons), "Schedule" (1 day, 0:00:00), and "Last Run" (which is highlighted with a red box). The "Last Run" column for "example\_dag" is empty, while for "update\_state", it shows the date "2024-01-10, 00:00:00". The "Next Run" and "Recent Tasks" columns are also present for each DAG.

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks
example_dag	airflow	4	1 day, 0:00:00		2024-01-10, 00:00:00	6
update_state	airflow	4	1 day, 0:00:00		2024-01-10, 00:00:00	6

# DAGs view next run

The screenshot shows the Apache Airflow web interface for viewing DAGs. At the top, there is a navigation bar with links for Airflow, DAGs, Cluster Activity, Datasets, Security, Browse, Admin, Docs, the current time (22:46 UTC), and a Log In button. Below the navigation bar, the title "DAGs" is displayed.

Below the title, there are several filter and search options:

- Buttons for filtering DAGs: All (2), Active (0), Paused (2), Running (0), Failed (0).
- A "Filter DAGs by tag" input field.
- A "Search DAGs" input field.
- An "Auto-refresh" toggle switch, which is turned on, indicated by a blue circle.

The main content area displays a table of DAGs. The columns are:

- DAG (dropdown menu)
- Owner (dropdown menu)
- Runs (button)
- Schedule (text)
- Last Run (button)
- Next Run (button)
- Recent Tasks (button)

Two DAGs are listed:

- example\_dag**: Owner airflow. Schedule: 1 day, 0:00:00. Last Run: N/A. Next Run: 2024-01-10, 00:00:00.
- update\_state**: Owner airflow. Schedule: 1 day, 0:00:00. Last Run: N/A. Next Run: 2024-01-10, 00:00:00.

The "Next Run" column for both DAGs is highlighted with a red box.

# DAGs view recent tasks

The screenshot shows the Apache Airflow interface for viewing DAGs. At the top, there is a navigation bar with links for Airflow, DAGs, Cluster Activity, Datasets, Security, Browse, Admin, Docs, the current time (22:46 UTC), and a Log In button. Below the navigation bar, the title "DAGs" is displayed. Underneath the title, there are several filter and search options: "All 2" (selected), "Active 0", "Paused 2", "Running 0", "Failed 0", "Filter DAGs by tag", and "Search DAGs". There is also an "Auto-refresh" toggle switch and a refresh button.

The main content area displays two DAG entries:

	DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks
<input type="checkbox"/>	example_dag	airflow	○ ○ ○ ○	1 day, 0:00:00	2024-01-10, 00:00:00	<a href="#">i</a>	○ ○ ○ ○ ○ ○
<input type="checkbox"/>	update_state	airflow	○ ○ ○ ○	1 day, 0:00:00	2024-01-10, 00:00:00	<a href="#">i</a>	○ ○ ○ ○ ○ ○

A red box highlights the "Recent Tasks" column for both DAGs, which shows six small circles representing task instances. The "Recent Tasks" column header is also highlighted with a red box.

# DAGs view example\_dag

Airflow    DAGs    Cluster Activity    Datasets    Security    Browse    Admin    Docs    22:46 UTC    Log In

## DAGs

All 2    Active 0    Paused 2    Running 0    Failed 0    Filter DAGs by tag    Search DAGs

Auto-refresh

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks
example_dag	airflow	4	1 day, 0:00:00	2024-01-10, 00:00:00	2024-01-10, 00:00:00	10
update_state	airflow	4	1 day, 0:00:00	2024-01-10, 00:00:00	2024-01-10, 00:00:00	10

# DAG detail view

The screenshot shows the Apache Airflow web interface for the DAG `example_dag`. The top navigation bar includes links for Airflow, DAGs, Cluster Activity, Datasets, Security, Browse, Admin, Docs, and Log In. The current time is 21:30 UTC. The main header displays the DAG name `example_dag`, its schedule (1 day, 0:00:00), and the next run time (2023-02-01, 00:00:00). Below the header are several tabs: Grid (selected), Graph, Calendar, Task Duration, Task Tries, Landing Times, and Gantt. There are also links for Details, Code, and Audit Log. A toolbar at the top provides filtering options for date (01/28/2024, 09:29:58 PM), tasks (25), run types, run states, and a Clear Filters button. An Auto-refresh toggle is also present. A keyboard shortcut keytip is shown above the filter buttons. A legend below the toolbar lists various task states: deferred, failed (highlighted in red), queued, removed, restarting, running, scheduled, shutdown, skipped, success, up\_for\_reschedule, up\_for\_retry, upstream\_failed, and no\_status. The main content area shows the DAG summary for `example_dag`, which has 1 total task, 1 BashOperator task, and no owners. It also indicates that there are no tags and no schedule interval.

Schedule: 1 day, 0:00:00 | Next Run: 2023-02-01, 00:00:00

DAG: example\_dag

Grid Graph Calendar Task Duration Task Tries Landing Times Gantt

Details Code Audit Log

01/28/2024, 09:29:58 PM 25 All Run Types All Run States Clear Filters Auto-refresh

Press `shift + /` for Shortcuts

deferred failed queued removed restarting running scheduled shutdown skipped success up\_for\_reschedule up\_for\_retry upstream\_failed no\_status

« » DAG example\_dag

Details Graph Gantt Code

DAG Summary

Total Tasks	1
BashOperator	1

DAG Details

Owners	
Tags	No tags
Schedule interval	

# DAG graph view

The screenshot shows the Apache Airflow web interface for the DAG `example_dag`. The top navigation bar includes links for Airflow, DAGs, Cluster Activity, Datasets, Security, Browse, Admin, Docs, the current time (21:30 UTC), and Log In. Below the navigation is a header bar showing the DAG name `example_dag`, its schedule (1 day, 0:00:00), and next run (2023-02-01, 00:00:00). A toolbar below the header provides navigation between Grid, Graph, Calendar, Task Duration, Task Tries, Landing Times, and Gantt views, along with a refresh button and a trash icon. Below the toolbar are links for Details, Code, and Audit Log. The main area features a search bar with date filters (01/28/2024, 09:30:30 PM) and dropdowns for Run Types (25) and Run States (All). It also includes a "Clear Filters" button and an "Auto-refresh" toggle. A keyboard shortcut keytip for "shift + /" is displayed above a list of run states: deferred, failed, queued, removed, restarting, running, scheduled, shutdown, skipped, success, up\_for\_reschedule, up\_for\_retry, upstream\_failed, and no\_status. The central part of the screen displays the DAG `example_dag` with a grid layout. A specific task, `generate_random_number` (BashOperator), is highlighted with a callout box. The right side of the grid has a "Layout:" dropdown set to "Left -> Right".

# DAG code view

The screenshot shows the Airflow web interface for the DAG `example_dag`. At the top, there's a navigation bar with links for Airflow, DAGs, Cluster Activity, Datasets, Security, Browse, Admin, Docs, the current time (22:14 UTC), and a Log In button. Below the navigation bar, the title "DAG: example\_dag" is displayed, along with the schedule "1 day, 0:00:00" and the next run time "2023-02-01, 00:00:00". A toolbar below the title includes buttons for Grid, Graph, Calendar, Task Duration, Task Tries, Landing Times, Gantt, Details, Code (which is selected), and Audit Log. There are also forward and delete icons. A search bar at the bottom of the toolbar allows filtering by date (01/28/2024, 10:14:28 PM), number of runs (25), run types (All Run Types), run states (All Run States), and provides a "Clear Filters" button and an "Auto-refresh" toggle. Below the search bar is a legend for run statuses: deferred (purple), failed (red), queued (orange), removed (yellow), restarting (pink), running (green), scheduled (light orange), shutdown (blue), skipped (magenta), success (dark green), up\_for\_reschedule (cyan), up\_for\_retry (yellow-orange), upstream\_failed (light blue), and no\_status (grey). The main content area shows the DAG structure with a tree view and a code editor. The code editor displays the Python code for the `generate_random_number` task:

```
1 from airflow import DAG
2 from airflow.operators.bash import BashOperator
3
4 with DAG(
5     'example_dag',
6     default_args={"start_date": "2023-02-01"}
7 ):
8
9     part1 = BashOperator(
10         task_id='generate_random_number',
11         bash_command='echo $RANDOM'
```

A "Toggle Wrap" button is located in the top right corner of the code editor.

# Audit logs

The screenshot shows the Apache Airflow web interface with the 'Audit logs' page selected. The top navigation bar includes links for Airflow, DAGs, Cluster Activity, Datasets, Security, Browse (which is highlighted with a red box), Admin, Docs, the current time (04:14 UTC), and Log In.

The 'Audit Logs' section is open, showing a list of audit events. The table has the following columns: Id, Dttm, Dag Id, Task Id, Event, Log, and xtra. The data is as follows:

Id	Dttm	Dag Id	Task Id	Event	Log	xtra
3	2024-01-29, 04:13:04	None		cli_dag_reserialize	repl	{"host_name": "4d0ff600-a020-4f46-9989-ace648ef13e4", "full_command": "['/usr/local/bin/airflow', 'dags', 'reserialize']"}
2	2024-01-29, 04:12:44	None		cli_scheduler	repl	{"host_name": "4d0ff600-a020-4f46-9989-ace648ef13e4", "full_command": "['/usr/local/bin/airflow', 'scheduler']"}
						{"host_name": "4d0ff600-a020-4f46-9989-ace648ef13e4", "full_command": "['/usr/local/bin/airflow', 'scheduler']"}

# Web UI vs command line

In most cases:

- Equally powerful depending on needs
- Web UI is easier
- Command line tool may be easier to access depending on settings

# **Let's practice!**

**INTRODUCTION TO APACHE AIRFLOW IN PYTHON**

# *Thanks for reading !*

*Follow my profile for more coding related contents*

Like



Comment



Share



*@rganesh0203*