

데이터 관리론

Data Engineering with Apache Airflow



Data Engineering with Apache Airflow

Unleash the Power of Automation → Apache Airflow

<https://www.youtube.com/watch?v=qNOGVu4v4xA&t=272s>



Airbnb

Google airbnb

전체 뉴스 이미지 쇼핑 동영상 더보기 도구 세아

검색결과 약 271,000,000개 (0.31초)

스폰서

33m2.co.kr https://www.33m2.co.kr 단기임대 No.1 삼삼엠투 — Airbnb보다 단기임대 원룸, 오피스텔, 아파트 단기임대에서 호텔 장기숙박까지. 필요한 만큼 주단위로 계약하세요! 오피스텔, 원룸, 투룸, 아파트에서 호텔 장기숙박 찾을 때, 50만명이 사용하는 단기임대 앱! 안심하고 단기임대 계약. 단기임대 계약 안전하게. 부동산 계약도...

강릉 한달살기 빠른 입주 가능한 강 찾을 때 1주부터 강릉 해변가에 살아보세

제주 한달살기 부담없이 1주부터 임대 가능 아파트, 원룸 등 다양하게 단기임대

에어비앤비 | 휴가지 숙소, 통나무집, 해변가 주택 등 에어비앤비. 뉴스룸 · 새로운 기능 · 채용정보 · 투자자 정보 · Airbnb 긴급 숙소. 바닥글 섹션. 언어 선택 한국어 (KR) 통화 선택 ₩KRW. © 2023 Airbnb, Inc. 개인정보 ...

에어비앤비 회사

airbnb

에어비앤비는 2008년 8월 시작된 세계 최대의 숙박 공유 서비스이다. 자신의 방이나 집, 별장 등 사람이 지낼 수 있는 모든 공간을 임대할 수 있다. 2013년 기준, 192 개국 3만 4800여 개 장소에 대한 숙박을 중개하고 있으며, 2초당 한 건 씩 예약이 이뤄지고 있다. 위키백과

주가: ABNB (NASDAQ) US\$136.14 -0.86 (-0.63%)
12월 29일 오후 4:00 GMT-5 - 면책조항

본사: 미국 캘리포니아 샌프란시스코

CEO: 브라이언 체스키 (2008년-)

창립자: 브라이언 체스키, 조 게비아, 네이선 블레차르 지크

Apache Airflow

Google airflow

전체 이미지 동영상 쇼핑 뉴스 더보기 도구 세이프

검색결과 약 166,000,000개 (0.24초)

 Apache Airflow
<https://airflow.apache.org> ::

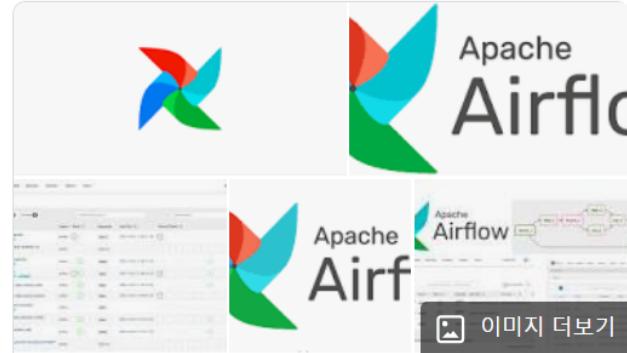
Apache Airflow
Airflow™ has a modular architecture and uses a message queue to orchestrate an arbitrary number of workers. Airflow™ is ready to scale to infinity.
[Documentation](#) · [Airflow.providers.google](#) · [Use cases](#) · [Community](#)

 꿈 많은 사람의 이야기
<https://lsjsj92.tistory.com> ::

Apache 에어플로우(Airflow) 시작하기 - 꿈 많은 사람의 이야기
2022. 2. 6. — Airflow를 설치하기에 앞서 가장 먼저 경로와 환경 변수 설정을 설정하고 가야합니다. Airflow 공식 문서에도 나와 있듯이 Airflow는 기본 경로를 '~' ...

 velog
<https://velog.io> › Airflow-기초-개념-및-장단점 ::

[Airflow] 에어플로우란? 기초 개념 및 장단점
2022. 11. 28. — 1. Airflow란? · Apache Airflow는 초기 에어비엔비(Airbnb) 엔지니어링 팀에서 개발한 오픈 소스 워크플로 관리 플랫폼이다.Apache Airflow는 초기 에어비엔비(Airbnb) 엔지니어링 팀에서 개발한 오픈 소스 워크플로 관리 플랫폼이다. 2014년 10월 에어비엔비에서 기업의 점차 복잡해지는 워크플로를 관리하기 위한 해결책으로서 시작하였다. 에어플로우는 파이썬으로 작성되어 있으며 워크플로는 파이썬 스크립트를 통해 만들어진다. 위키백과



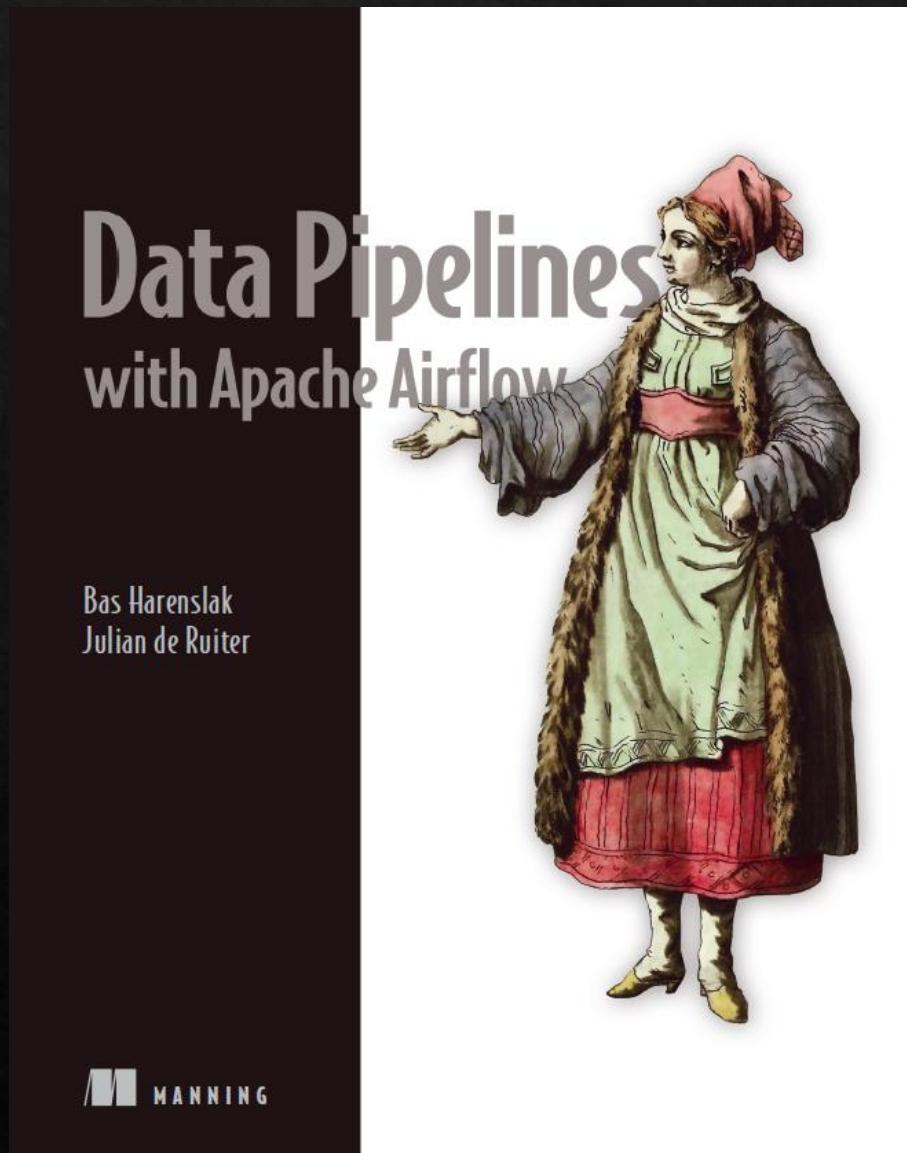
아파치 에어플로우

Apache Airflow

- ❖ Apache Airflow is an open-source workflow management platform for data engineering pipelines.
 - ❖ It started at Airbnb in October 2014[1] as a solution to manage the company's increasingly complex workflows.
- ❖ Apache Airflow, a **batch-oriented framework** for building data pipelines.
- ❖ Airflow is best thought of as a spider in a web: it sits in the middle of your data processes and **coordinates(orchestrates)** work happening across the different (distributed) systems.



The Book



brief contents

PART 1 GETTING STARTED 1

- 1 ■ Meet Apache Airflow 3
- 2 ■ Anatomy of an Airflow DAG 20
- 3 ■ Scheduling in Airflow 40
- 4 ■ Templating tasks using the Airflow context 60
- 5 ■ Defining dependencies between tasks 85

PART 2 BEYOND THE BASICS 113

- 6 ■ Triggering workflows 115
- 7 ■ Communicating with external systems 135
- 8 ■ Building custom components 157
- 9 ■ Testing 186
- 10 ■ Running tasks in containers 220

PART 3 AIRFLOW IN PRACTICE 253

- 11 ■ Best practices 255
- 12 ■ Operating Airflow in production 281
- 13 ■ Securing Airflow 322
- 14 ■ Project: Finding the fastest way to get around NYC 344

PART 4 IN THE CLOUDS 365

- 15 ■ Airflow in the clouds 367
- 16 ■ Airflow on AWS 375
- 17 ■ Airflow on Azure 394
- 18 ■ Airflow in GCP 412

Meet Apache Airflow

- ❖ Introducing Data Pipelines
- ❖ Introducing Apache Airflow



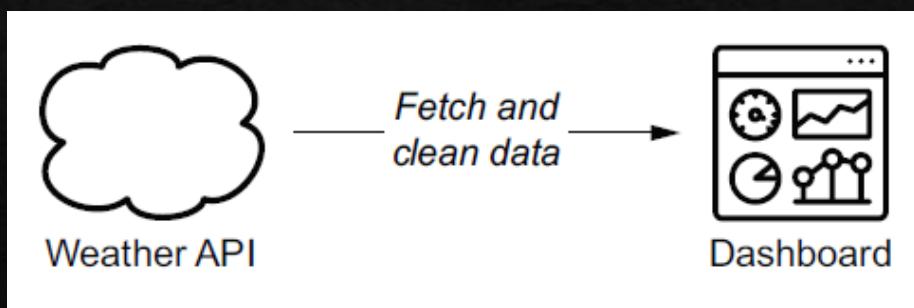
Meet Apache Airflow

- ❖ Introducing Data Pipelines
- ❖ Introducing Apache Airflow



Introducing data pipelines

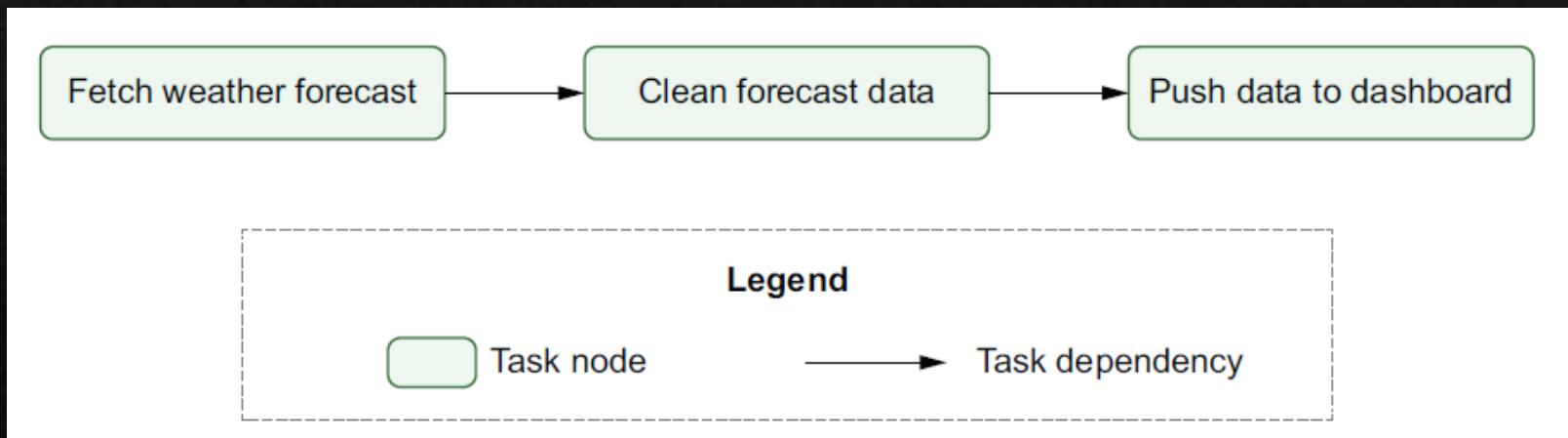
- ❖ To implement live weather dashboard, we need to perform something like the following steps:
 1. Fetch weather forecast data from a weather API.
 2. Clean or otherwise transform the fetched data (e.g., converting temperatures from Fahrenheit to Celsius or vice versa), so that the data suits our purpose.
 3. Push the transformed data to the weather dashboard.



- Overview of the weather dashboard use case, in which weather data is fetched from an external API and fed into a dynamic dashboard

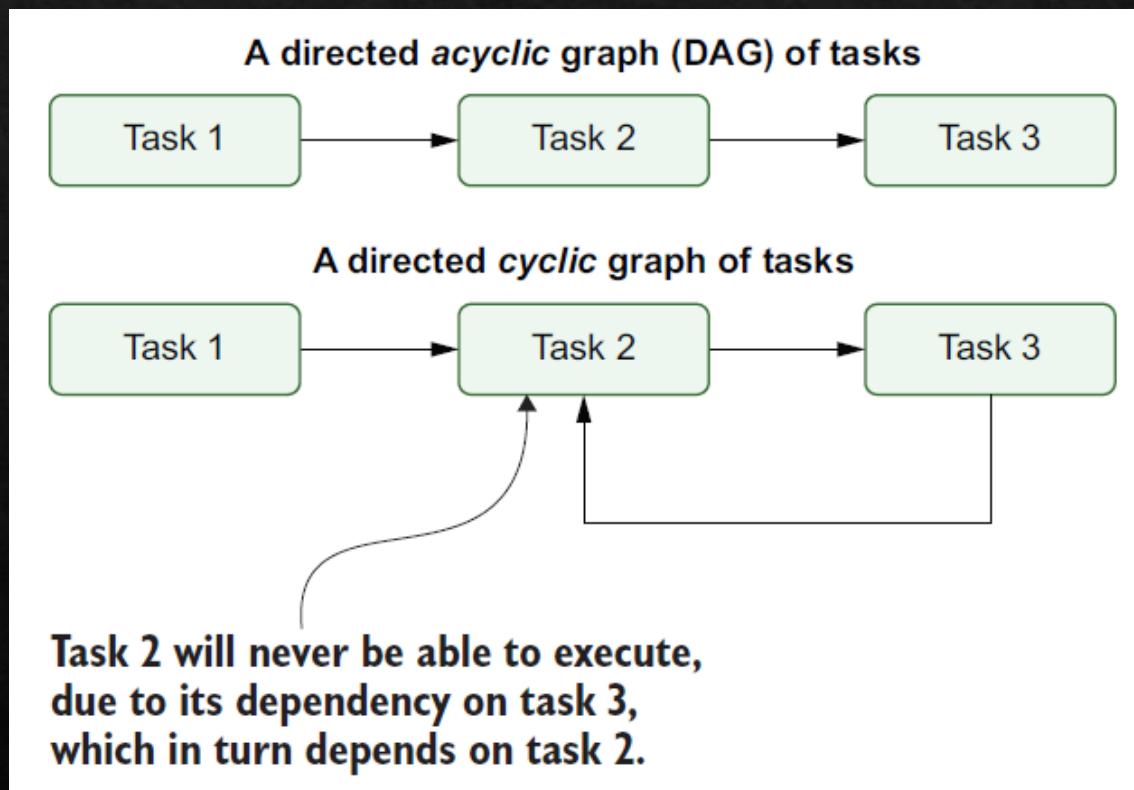
Data Pipelines as graphs

- ❖ Graph representation of the data pipeline for the weather dashboard.
- ❖ Nodes represent tasks and directed edges represent dependencies between tasks (with an edge pointing from task A to task B, indicating that task A needs to be run before task B).



Directed Acyclic Graph (DAG)

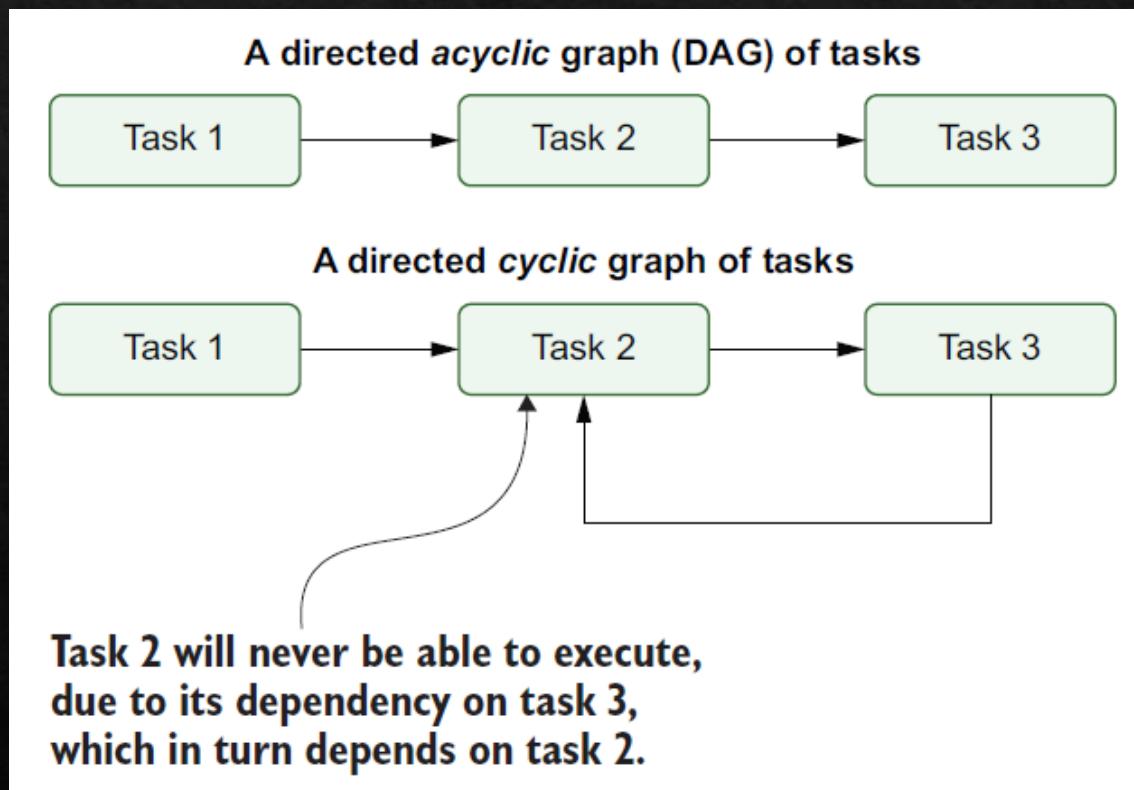
- ◊ The graph contains directed edges and does not contain any loops or cycles (acyclic).



- Cycles in graphs prevent task execution due to circular dependencies.
- In acyclic graphs (top), there is a clear path to execute the three different tasks.
- However, in cyclic graphs (bottom), there is no longer a clear execution path due to the interdependency between tasks 2 and 3.

Directed Acyclic Graph (DAG)

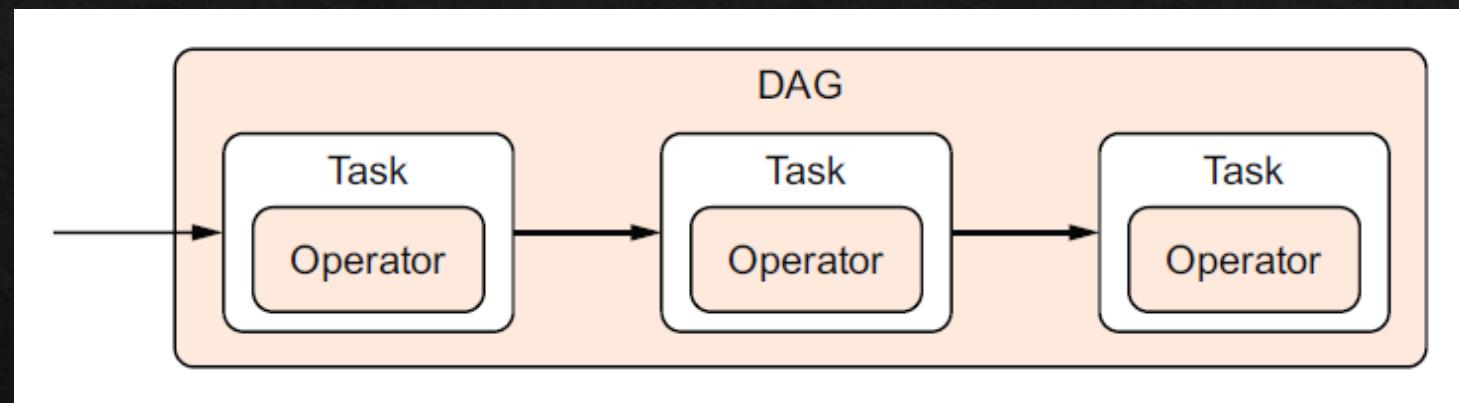
- ◊ The graph contains directed edges and does not contain any loops or cycles (acyclic).



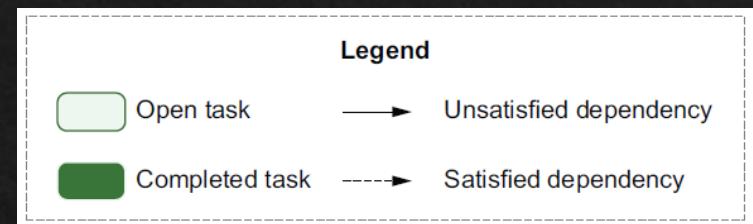
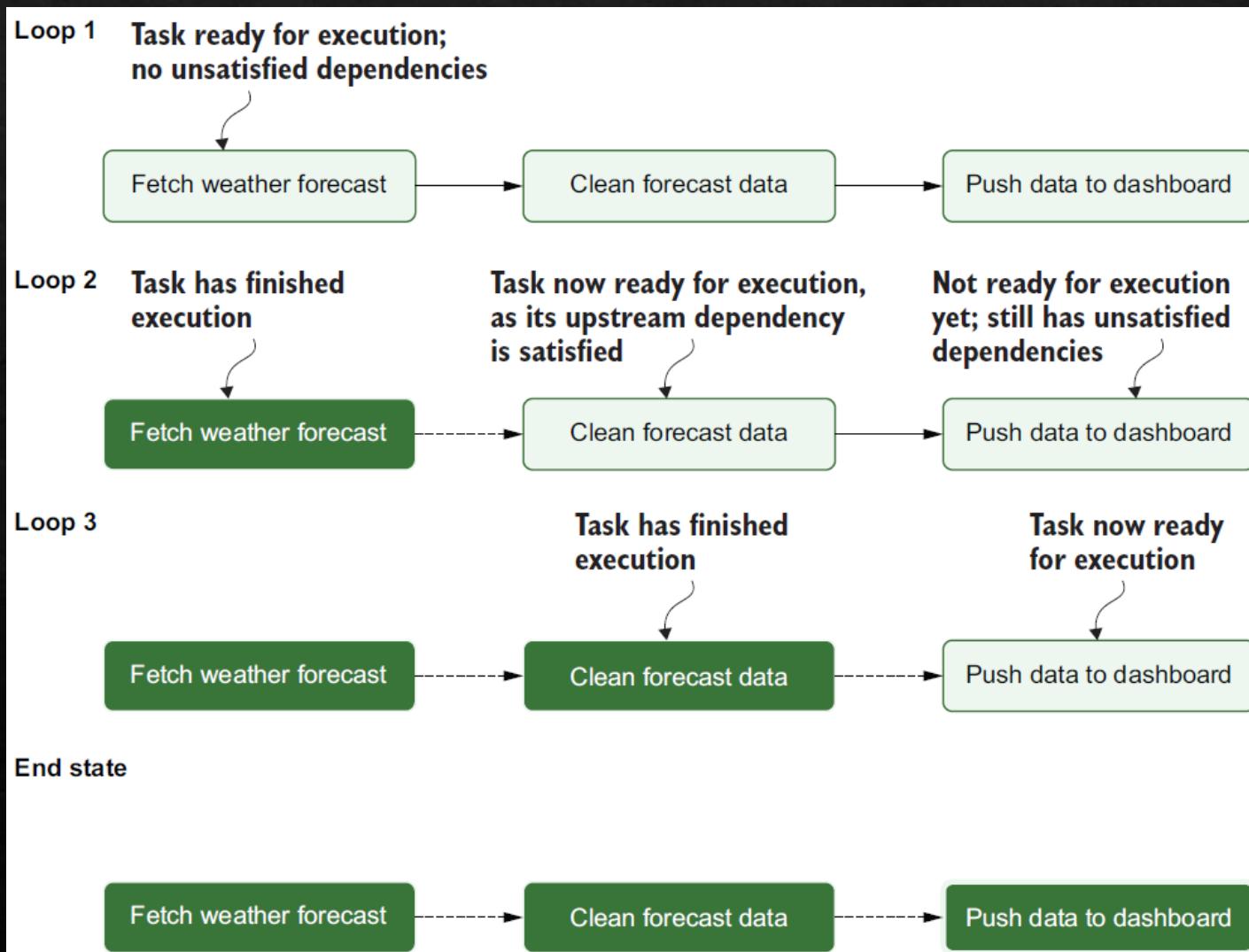
- Cycles in graphs prevent task execution due to circular dependencies.
- In acyclic graphs (top), there is a clear path to execute the three different tasks.
- However, in cyclic graphs (bottom), there is no longer a clear execution path due to the interdependency between tasks 2 and 3.

Task/Operator

- ❖ DAGs and operators are used by Airflow users. Tasks are internal components to manage operator state and display state changes (e.g., started/finished) to the user.



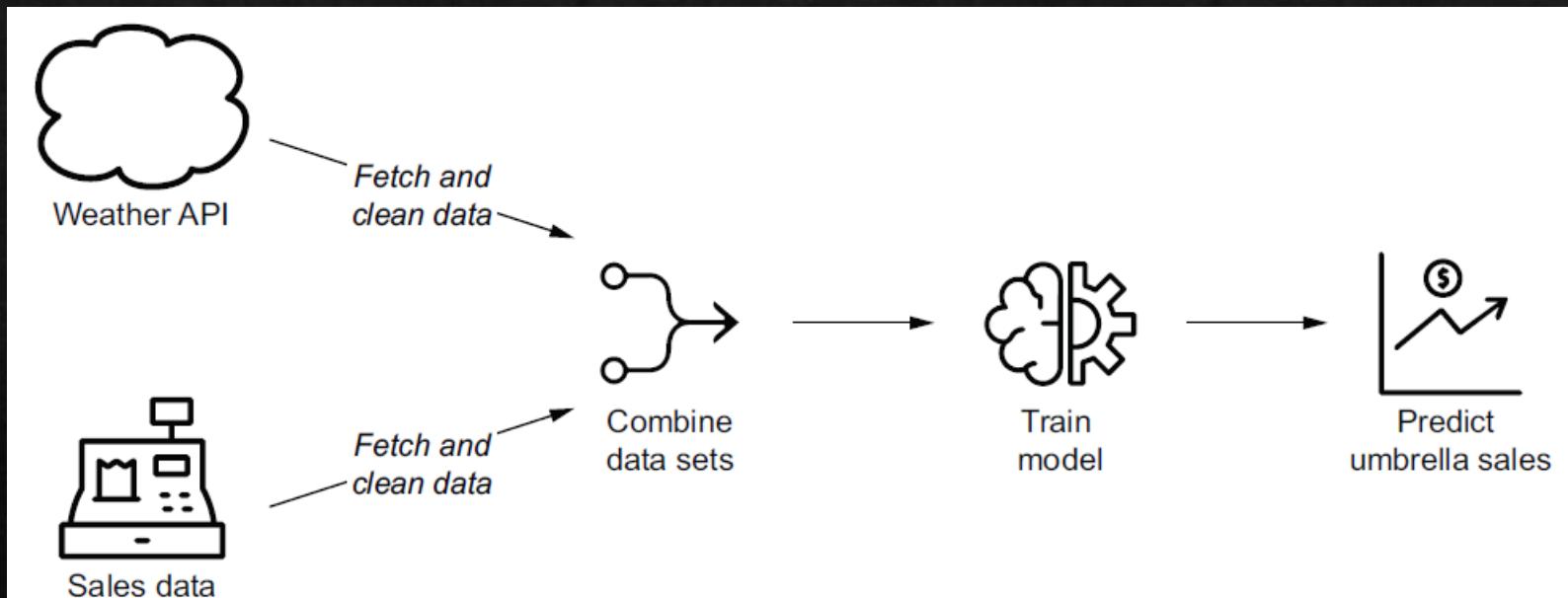
Pipeline graphs vs. sequential scripts



- Using the DAG structure to execute tasks in the data pipeline in the correct order: depicts each task's state during each of the loops through the algorithm, demonstrating how this leads to the completed execution of the pipeline (end state)

Pipeline graphs vs. sequential scripts

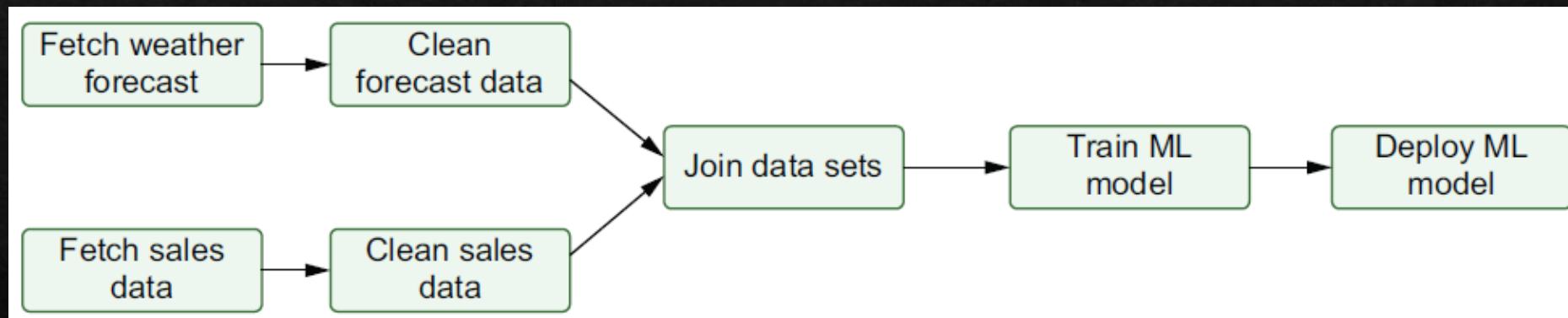
- ❖ To build a pipeline for training the ML model, we need to implement something like the following steps:



- Overview of the umbrella demand use case, in which historical weather and sales data are used to train a model that predicts future sales demands depending on weather forecasts

Divide-and-conquer algorithm

- ❖ A divide-and-conquer algorithm recursively breaks down a problem into two or more sub-problems of the same or related type, until these become simple enough to be solved directly.



- The useful property of the graph-based representation is that it clearly separates pipelines into small incremental tasks rather than having one monolithic script or process that does all the work.

Running pipeline using workflow managers

Name	Originated at ^a	Workflows defined in	Written in	Scheduling	Backfilling	User interface ^b	Installation platform	Horizontally scalable
Airflow	Airbnb	Python	Python	Yes	Yes	Yes	Anywhere	Yes
Argo	Applatix	YAML	Go	Third party ^c		Yes	Kubernetes	Yes
Azkaban	LinkedIn	YAML	Java	Yes	No	Yes	Anywhere	
Conductor	Netflix	JSON	Java	No		Yes	Anywhere	Yes
Luigi	Spotify	Python	Python	No	Yes	Yes	Anywhere	Yes
Make		Custom DSL	C	No	No	No	Anywhere	No
Metaflow	Netflix	Python	Python	No		No	Anywhere	Yes
Nifi	NSA	UI	Java	Yes	No	Yes	Anywhere	Yes
Oozie		XML	Java	Yes	Yes	Yes	Hadoop	Yes

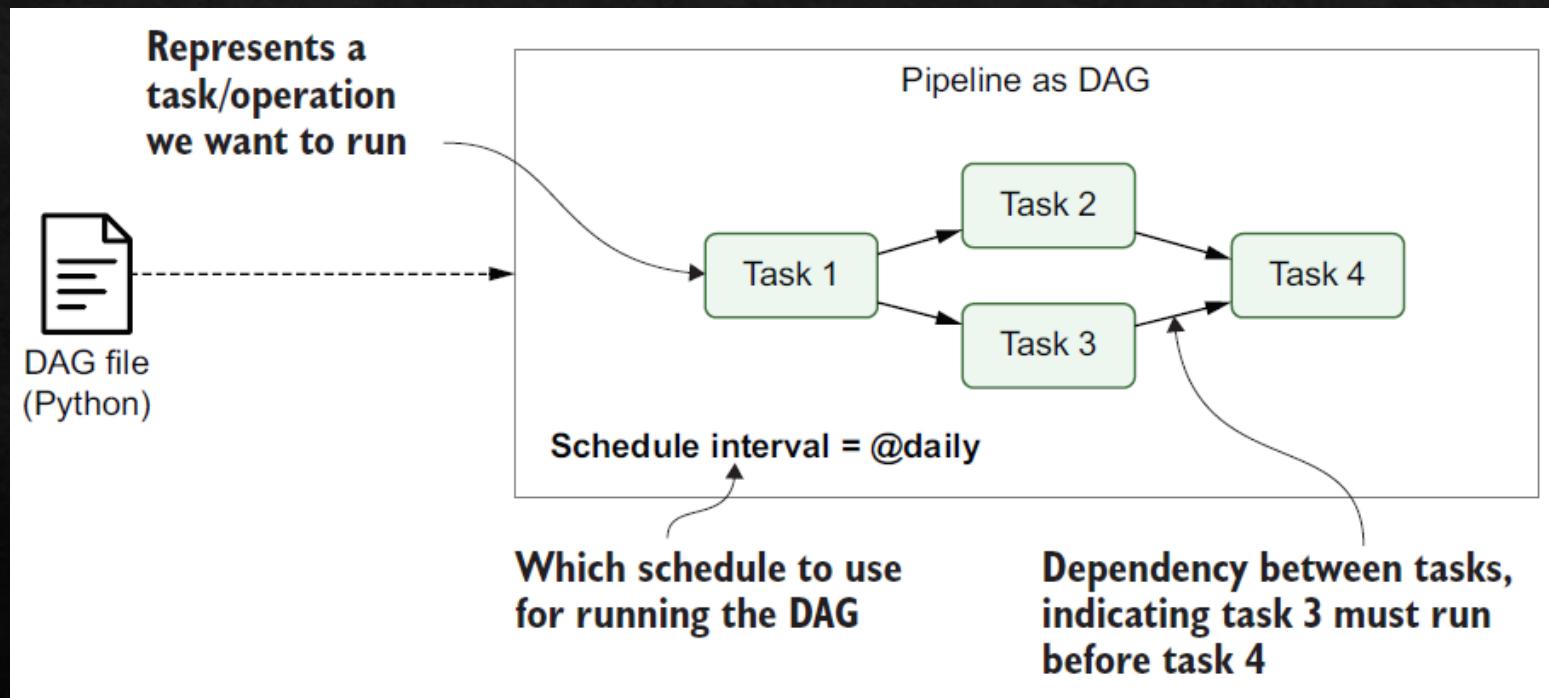
Meet Apache Airflow

- ❖ Introducing Data Pipelines
- ❖ Introducing Apache Airflow



Defining pipelines flexibly in (Python) code

- ❖ In Airflow, you define your DAGs using Python code in DAG files, which are essentially Python scripts that describe the structure of the corresponding DAG.



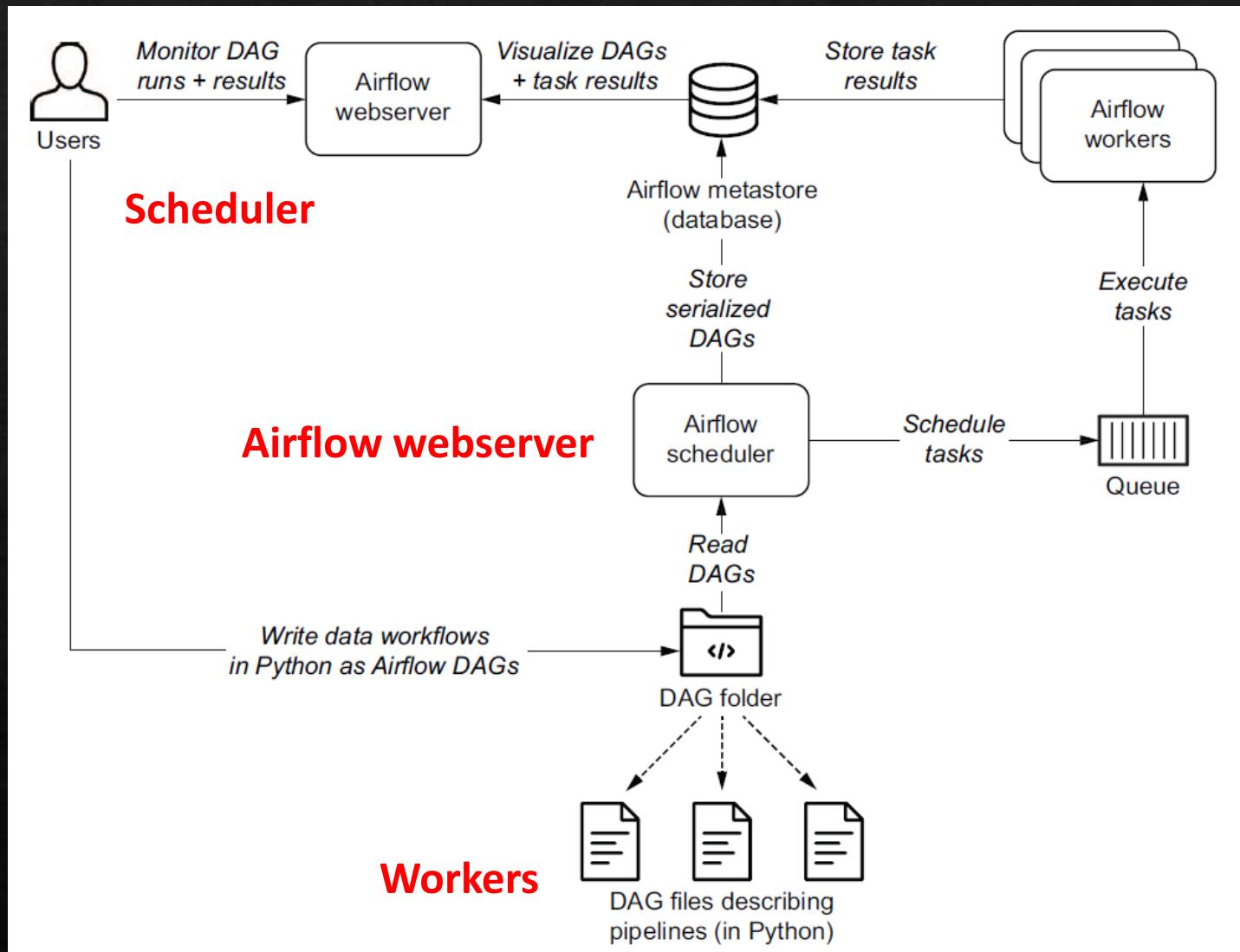
- Airflow pipelines are defined as DAGs using Python code in DAG files.
- Each DAG file typically defines one DAG, which describes the different tasks and their dependencies.
- Besides this, the DAG also defines a schedule interval that determines when the DAG is executed by Airflow.

Scheduling and executing pipelines

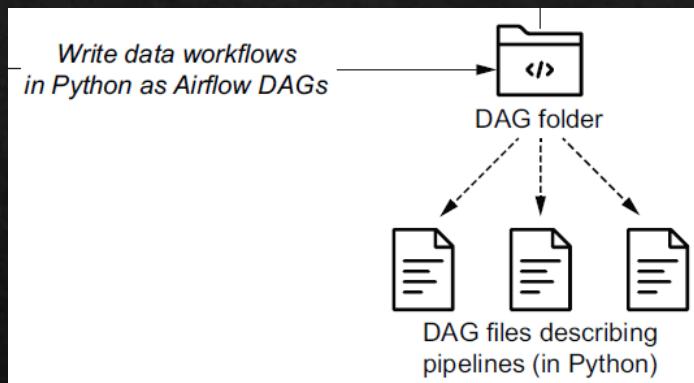
- ❖ The Airflow scheduler
 - ❖ Parses DAGs, checks their schedule interval, and (if the DAGs' schedule has passed) starts scheduling the DAGs' tasks for execution by passing them to the Airflow workers.
- ❖ The Airflow workers
 - ❖ Pick up tasks that are scheduled for execution and execute them. As such, the workers are responsible for actually “doing the work.”
- ❖ The Airflow webserver
 - ❖ Visualizes the DAGs parsed by the scheduler and provides the main interface for users to monitor DAG runs and their results.



Overview of the main components involved in Airflow

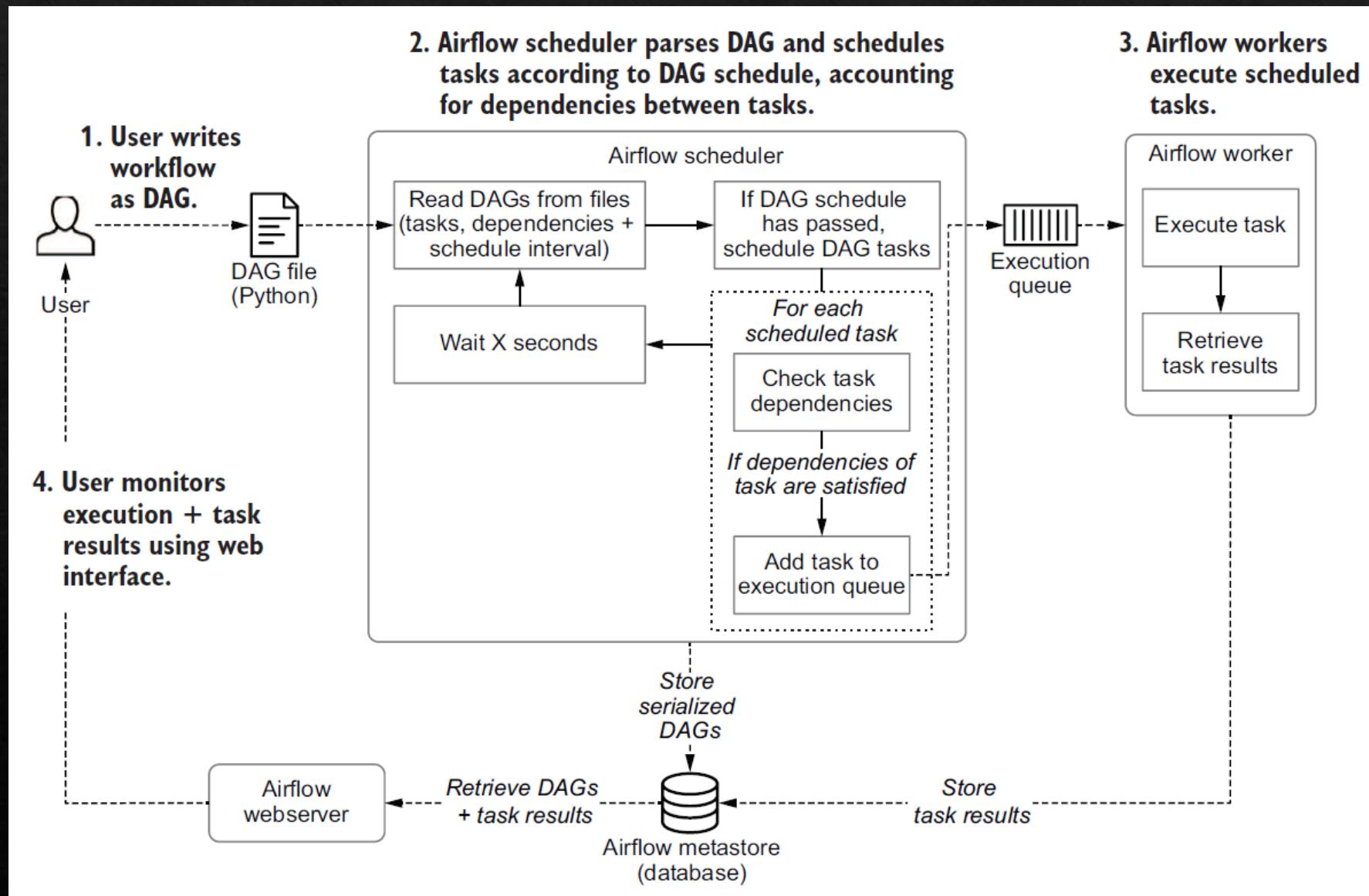


Workers



```
File Edit Selection View Go Run ... ← → airflow dags_base_branch_operator.py dags_python_with_trigger_rule_eg1.py dags_01_umbrella.py  
EXPLORER OPEN EDITORS 5 unsaved AIRFLOW dags _pycache_ dags_01_umbrella.py 1 dags_02_download_rocket_launches.py dags_02_mkdir_download_rocket_launches.py dags_04_listing_04_01.py dags_0301_unscheduled.py dags_base_branch_operator.py dags_bash_operator.py dags_bash_select_fruit.py dags_bash_with_template.py dags_branch_python_operator.py dags_conn_test.py dags_email_operator.py dags_python_email_xcom.py dags_python_import.py dags_python_operator.py dags_python_show_templates.py dags_python_task_decorator.py dags_python_with_branch_decorator.py dags_python_with_op_args.py dags_python_with_op_kwargs.py dags_python_with_template.py  
1 """DAG demonstrating the umbrella use case with dummy operators."""  
2  
3 import airflow.utils.dates  
4 from airflow import DAG  
5 from airflow.operators.dummy import DummyOperator  
6  
7 dag = DAG(  
8     dag_id="01_umbrella",  
9     description="Umbrella example with DummyOperators.",  
10    start_date=airflow.utils.dates.days_ago(5),  
11    schedule_interval="@daily",  
12 )  
13  
14 fetch_weather_forecast = DummyOperator(task_id="fetch_weather_forecast", dag=dag)  
15 fetch_sales_data = DummyOperator(task_id="fetch_sales_data", dag=dag)  
16 clean_forecast_data = DummyOperator(task_id="clean_forecast_data", dag=dag)  
17 clean_sales_data = DummyOperator(task_id="clean_sales_data", dag=dag)  
18 join_datasets = DummyOperator(task_id="join_datasets", dag=dag)  
19 train_ml_model = DummyOperator(task_id="train_ml_model", dag=dag)  
20 deploy_ml_model = DummyOperator(task_id="deploy_ml_model", dag=dag)  
21  
22 # Set dependencies between all tasks  
23 fetch_weather_forecast >> clean_forecast_data  
24 fetch_sales_data >> clean_sales_data  
25 [clean_forecast_data, clean_sales_data] >> join_datasets  
26 join_datasets >> train_ml_model >> deploy_ml_model
```

Schematic overview of the Airflow Process



- Airflow pipelines Schematic overview of the process involved in developing and executing pipelines as DAGs using Airflow

The Extensive web interface

- ❖ Monitoring and handling failures
- ❖ Incremental loading and backfilling



The Extensive web interface

Airflow DAGs Cluster Activity Datasets Security ▾ Browse ▾ Admin ▾ Docs ▾ 22:13 UTC ▾ AA ▾

DAGs

All 74 Active 21 Paused 53 Running 0 Failed 2 Filter DAGs by tag Search DAGs Auto-refresh C

i DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks
01_umbrella	airflow	6	@daily	2023-12-28, 00:00:00	2023-12-29, 00:00:00	7
01_unscheduled	airflow	1	None	2023-12-29, 14:16:23		1
02_mkdir_download_rocket_launches	airflow	6	@daily	2023-12-29, 13:51:07	2023-12-29, 00:00:00	2
03_download_rocket_launches	airflow	7	@daily	2023-12-29, 14:05:23	2023-12-29, 00:00:00	4
04_listing_4_01	airflow	90	@hourly	2023-12-29, 16:00:00	2023-12-29, 17:00:00	1
dags_base_branch_operator	airflow	1	None	2023-12-27, 20:38:46		3
dags_bash_fruit	airflow	1	10 0 * * #1	2023-11-03, 15:10:00	2023-12-01, 15:10:00	1

The Extensive web interface

Airflow DAGs Cluster Activity Datasets Security Browse Admin Docs 22:14 UTC AA

DAG: 01_umbrella Umbrella example with DummyOperators. Schedule: @daily Next Run: 2023-12-29, 00:00:00

Grid Graph Calendar Task Duration Task Tries Landing Times Gantt Details Code Audit Log

2023-12-29 오후 10:14:10 25 All Run Types All Run States Clear Filters Auto-refresh

Press shift + / for Shortcuts deferred failed queued removed restarting running scheduled skipped success up_for_reschedule up_for_retry upstream_failed no_status

Duration Dec 26, 00:00

fetch_weather_forecast
fetch_sales_data
clean_forecast_data
clean_sales_data
join_datasets
train_ml_model
deploy_ml_model

» DAG 01_umbrella

Details Graph Gantt Code

DAG Runs Summary

Total Runs Displayed	6
Total success	6
First Run Start	2023-12-28, 00:41:27 UTC
Last Run Start	2023-12-29, 03:50:43 UTC
Max Run Duration	00:00:07

The Extensive web interface

Airflow DAGs Cluster Activity Datasets Security ▾ Browse ▾ Admin ▾ Docs ▾ 22:15 UTC ▾ AA ▾

DAG: 01_umbrella Umbrella example with DummyOperators.

Schedule: @daily i Next Run: 2023-12-29, 00:00:00

Grid Graph Calendar Task Duration Task Tries Landing Times Gantt Details Code Audit Log

2023-12-29 오후 10:14:10 25 All Run Types All Run States Clear Filters Auto-refresh

Press shift + / for Shortcuts deferred failed queued removed restarting running scheduled skipped success up_for_reschedule up_for_retry upstream_failed no_status

Duration Dec 26, 00:00

» DAG 01_umbrella

Details Graph Gantt Code

Layout: Left -> Right

```
graph LR; subgraph Top [Top]; A[fetch_sales_data<br/>EmptyOperator]; B[clean_sales_data<br/>EmptyOperator]; C[join_datasets<br/>EmptyOperator]; D[train_ml_model<br/>EmptyOperator]; E[deploy_ml_model<br/>EmptyOperator]; end; subgraph Bottom [Bottom]; F[fetch_weather_forecast<br/>EmptyOperator]; G[clean_forecast_data<br/>EmptyOperator]; H[join_datasets<br/>EmptyOperator]; I[train_ml_model<br/>EmptyOperator]; J[deploy_ml_model<br/>EmptyOperator]; end; A --> B; B --> C; C --> D; D --> E; F --> G; G --> H; H --> I; I --> J; C --- H;
```

Tasks listed on the left:

- fetch_weather_forecast
- fetch_sales_data
- clean_forecast_data
- clean_sales_data
- join_datasets
- train_ml_model
- deploy_ml_model

The Extensive web interface

The screenshot shows the Airflow web interface with the following details:

- Header:** Airflow, DAGs, Cluster Activity, Datasets, Security, Browse, Admin, Docs, 22:15 UTC, AA.
- DAG Overview:** DAG 01_umbrella. Duration: Dec 26, 00:00. A Gantt chart on the left shows task durations from 00:00:00 to 00:00:07.
- Task List:** fetch_weather_forecast, fetch_sales_data, clean_forecast_data, clean_sales_data, join_datasets, train_ml_model, deploy_ml_model.
- Code View:** Parsed at: 2023-12-29, 17:42:52 UTC.

```
3 import airflow.utils.dates
4 from airflow import DAG
5 from airflow.operators.dummy import DummyOperator
6
7 dag = DAG(
8     dag_id="01_umbrella",
9     description="Umbrella example with DummyOperators.",
10    start_date=airflow.utils.dates.days_ago(5),
11    schedule_interval="@daily",
12 )
13
14 fetch_weather_forecast = DummyOperator(task_id="fetch_weather_forecast", dag=dag)
15 fetch_sales_data = DummyOperator(task_id="fetch_sales_data", dag=dag)
16 clean_forecast_data = DummyOperator(task_id="clean_forecast_data", dag=dag)
17 clean_sales_data = DummyOperator(task_id="clean_sales_data", dag=dag)
18 join_datasets = DummyOperator(task_id="join_datasets", dag=dag)
19 train_ml_model = DummyOperator(task_id="train_ml_model", dag=dag)
20 deploy_ml_model = DummyOperator(task_id="deploy_ml_model", dag=dag)
21
22 # Set dependencies between all tasks
23 fetch_weather_forecast >> clean_forecast_data
```

The Extensive web interface

The screenshot shows the Airflow web interface with the following details:

- Header:** Airflow, DAGs, Cluster Activity, Datasets, Security, Browse, Admin, Docs, 22:16 UTC, AA.
- Shortcuts:** Press shift + / for Shortcuts.
- Filter Buttons:** deferred, failed, queued, removed, restarting, running, scheduled, skipped, success, up_for_reschedule, up_for_retry, upstream_failed, no_status.
- DAG Run Information:** DAG 01_umbrella / Run 02023-12-29, 00:00:00 UTC / Task fetch_weather_forecast.
- Task Navigation:** Details, Graph, Gantt, Code, Logs (selected), XCom.
- Logs Section:** (by attempts) All Levels, All File Sources, Wrap, Download, See More.

*** Could not read served logs: Request URL is missing an 'http://' or 'https://' protocol.
- Left Sidebar:** A vertical list of tasks: fetch_weather_forecast, fetch_sales_data, clean_forecast_data, clean_sales_data, join_datasets, train_ml_model, deploy_ml_model. The first task is highlighted with a blue background.
- Timeline:** A bar chart showing the duration of the task across multiple attempts. The x-axis is labeled "Dec 26, 00:00" and the y-axis is labeled "Duration".