

Introduction to Airflow

INTRODUCTION TO AIRFLOW IN PYTHON



Mike Metzger
Data Engineer

What is data engineering?

Data engineering is:

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

Data engineering refers to the building of systems to enable the collection and usage of data.

↳ This system should be

- (i) maintainable
- (ii) Reliable
- (iii) Repeatable

What is a workflow?

A *workflow* is:

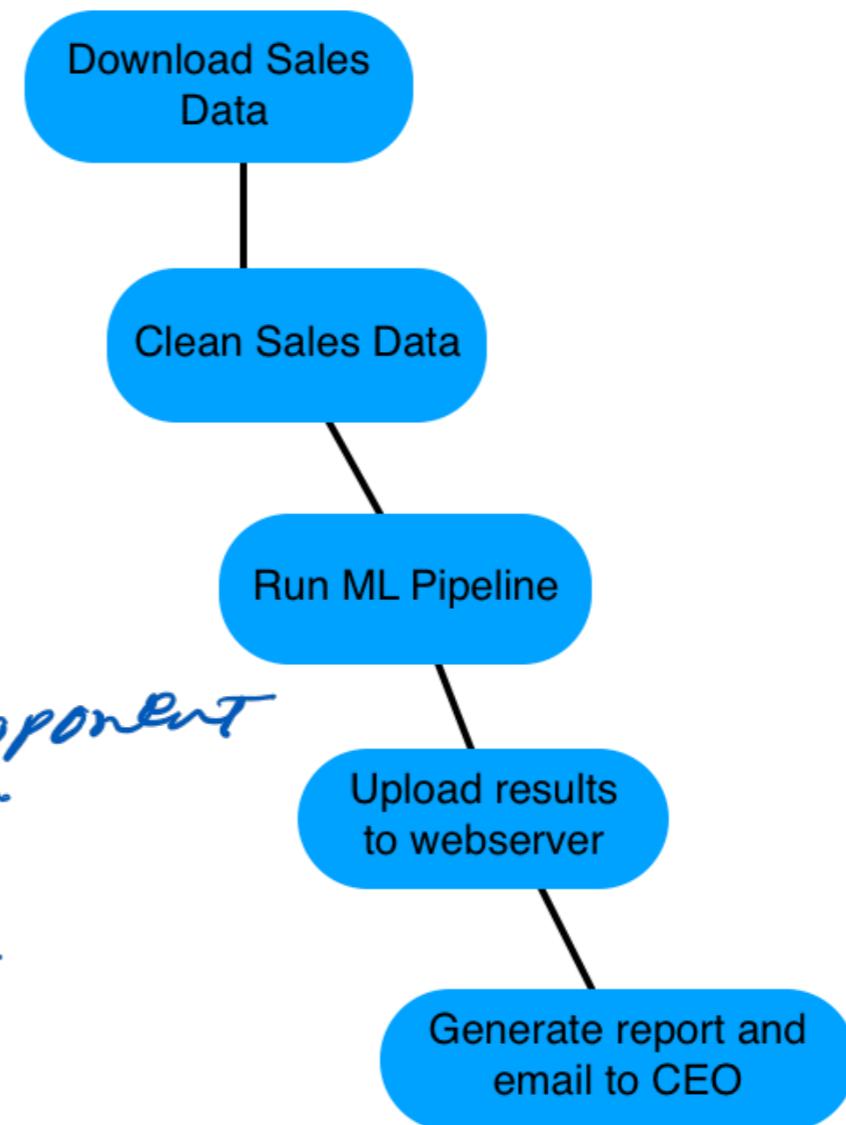
- A set of steps to accomplish a given data engineering task
 - Such as: downloading files, copying data, filtering information, writing to a database, etc

- Of varying levels of complexity

*from 2-3 to 100's component
in one workflow.*

- A term with various meaning depending on

context → *The context for above is data engineering*



What is Airflow? → scalable to infinite tasks.

Airflow is a platform to program workflows, including:

- Creation → author
- Scheduling
- Monitoring



Apache *Boekfiling?*
Airflow

Airflow allows to define almost any workflow in python code, no matter how complex.

Use Case : (i) Business Operation / Data Pipeline

(ii) ETL

(iii) infra management like AWS | spin/manage/tear down .

(iii) Use for MLOps / MLops pipeline .

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



Apache
Airflow

Note: Do not use Airflow for infinitely running event based workflows.
it was built for finite batch workflow.

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

Other workflow tools

Other tools:

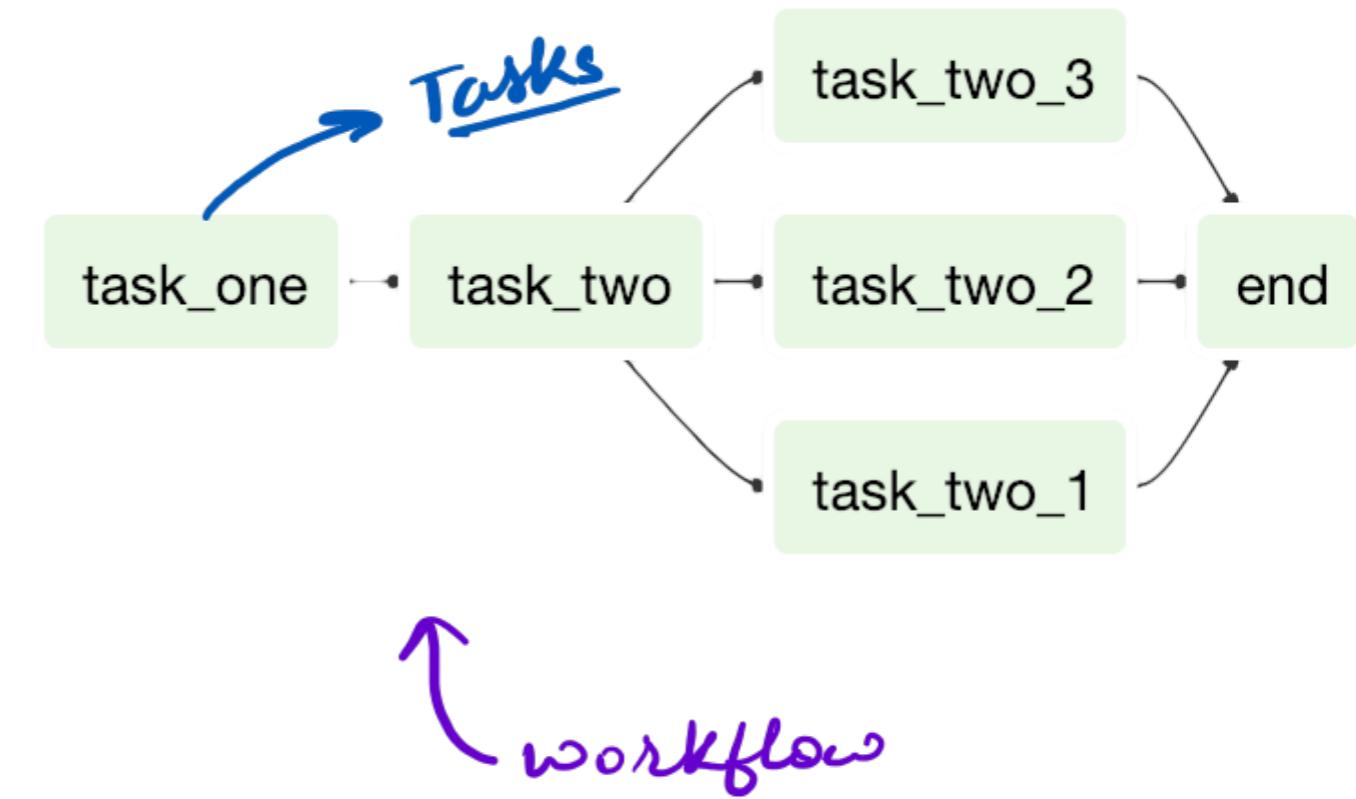
- Luigi → Spotify's
- SSIS → Microsoft
- Bash scripting



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 → set of task → workflow → tasks and dependencies b/w tasks.

DAG code example

Simple DAG definition:

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

In any Python code a dag referred by this

when using Airflow shell command dag will
be referred by this.

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 AIRFLOW IN PYTHON

Airflow DAGs

INTRODUCTION TO AIRFLOW IN PYTHON



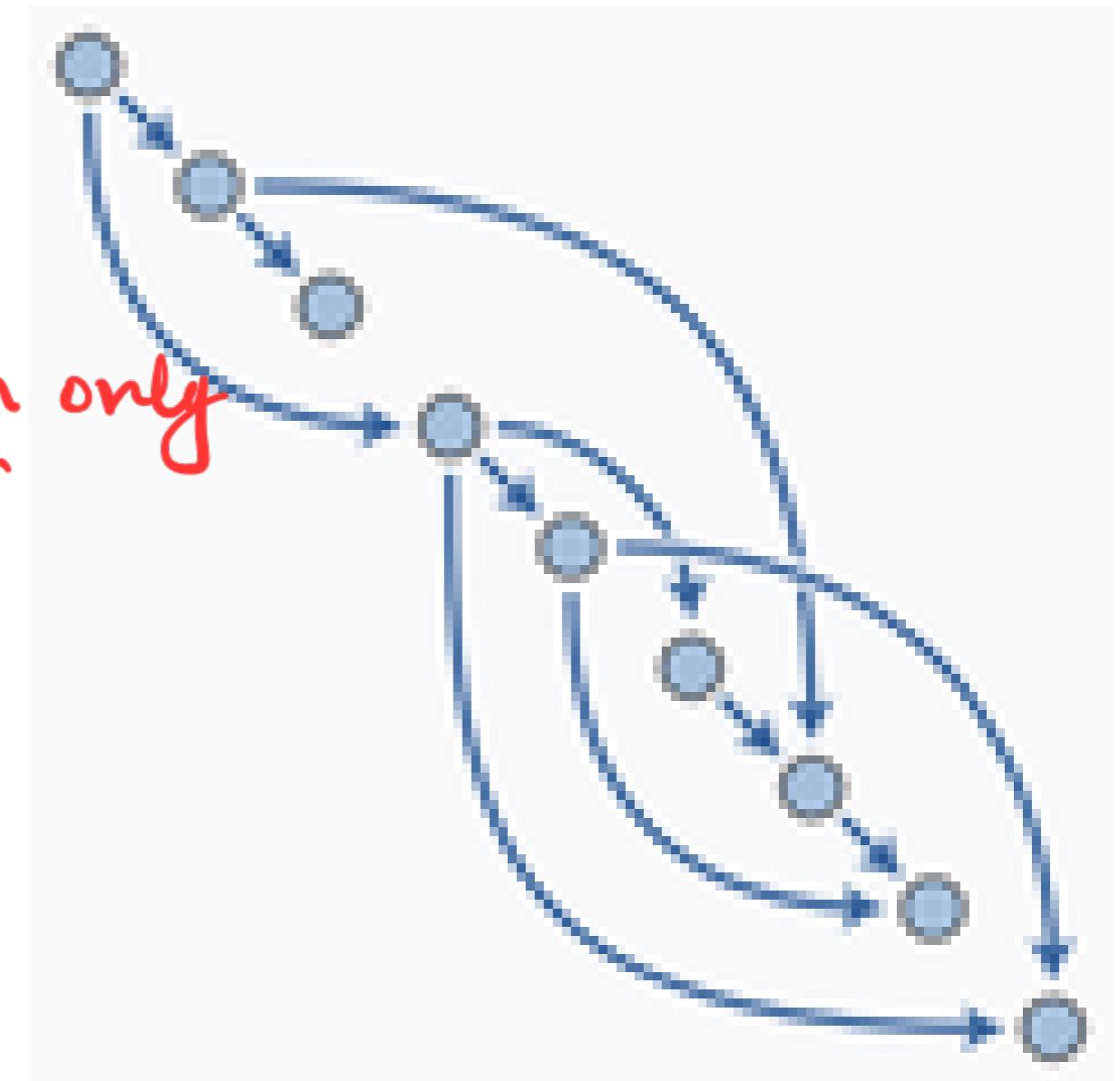
Mike Metzger
Data Engineer

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

provides context how to order the running of components



¹ https://en.m.wikipedia.org/wiki/Directed_acyclic_graph

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:
```

↳ dogobject

↳ alias.

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 )
```

↳ older method before 2.0

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 AIRFLOW IN PYTHON

Airflow webUI is useful in developing and administering workflows on the airflow platform.

→ made up of several primary page groups.

Airflow web interface

INTRODUCTION TO AIRFLOW IN PYTHON



Mike Metzger

Data Engineer

DAGs view



DAGs

All 2	Active 0	Paused 2	Running 0	Failed 0	Filter DAGs by tag	Search DAGs
<input checked="" type="checkbox"/> Auto-refresh						
i DAG ▾	Owner ▾	Runs i	Schedule	Last Run ▾ i	Next Run ▾ i	Recent Tasks i
<input type="checkbox"/> example_dag airflow	airflow	1 day, 0:00:00	1 day, 0:00:00	2024-01-10, 00:00:00 i		
<input type="checkbox"/> update_state airflow	airflow	1 day, 0:00:00	1 day, 0:00:00	2024-01-10, 00:00:00 i		

DAGs view DAGs

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
<input type="checkbox"/>	example_dag	airflow	○ ○ ○ ○	1 day, 0:00:00	2024-01-10, 00:00:00 i	○ ○ ○ ○ ○ ○	
<input type="checkbox"/>	update_state	airflow	○ ○ ○ ○	1 day, 0:00:00	2024-01-10, 00:00:00 i	○ ○ ○ ○ ○ ○	

DAGs view owner

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	Schedule	Last Run	Next Run	Recent Tasks
<input type="checkbox"/> example_dag	airflow	4	1 day, 0:00:00	2024-01-10, 00:00:00		6
<input type="checkbox"/> update_state	airflow	4	1 day, 0:00:00	2024-01-10, 00:00:00		6

DAGs view runs

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	Next Run	Recent Tasks i
<input type="checkbox"/> example_dag	airflow		1 day, 0:00:00	2024-01-10, 00:00:00 i		
<input type="checkbox"/> update_state	airflow		1 day, 0:00:00	2024-01-10, 00:00:00 i		

DAGs view schedule

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	Next Run	Recent Tasks i
<input type="checkbox"/> example_dag	airflow		1 day, 0:00:00	2024-01-10, 00:00:00 i		
<input type="checkbox"/> update_state	airflow		1 day, 0:00:00	2024-01-10, 00:00:00 i		

DAGs view last run

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

DAGs view next run

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
<input type="checkbox"/>	example_dag	airflow	○○○○	1 day, 0:00:00		2024-01-10, 00:00:00 i	○○○○○○
<input type="checkbox"/>	update_state	airflow	○○○○	1 day, 0:00:00		2024-01-10, 00:00:00 i	○○○○○○

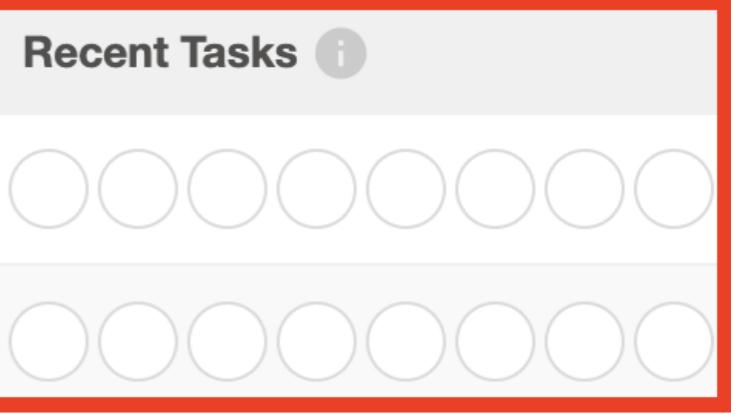
DAGs view recent tasks

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
<input type="checkbox"/>	example_dag	airflow	○○○○	1 day, 0:00:00	2024-01-10, 00:00:00 i		
<input type="checkbox"/>	update_state	airflow	○○○○	1 day, 0:00:00	2024-01-10, 00:00:00 i		

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

i 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	i	○ ○ ○ ○ ○ ○
<input type="checkbox"/> update_state	airflow	○ ○ ○ ○	1 day, 0:00:00	2024-01-10, 00:00:00	i	○ ○ ○ ○ ○ ○

DAG detail view → info about dag itself

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, and Log In. The time is listed as 21:30 UTC. Below the navigation, it says 'DAG: example_dag' with a status of 'Schedule: 1 day, 0:00:00' and 'Next Run: 2023-02-01, 00:00:00'. There are several tabs at the top: Grid (highlighted with a yellow box and a blue arrow), Graph, Calendar, Task Duration, Task Tries, Landing Times, Gantt, Details, Code, and Audit Log. A date range selector shows '01/28/2024, 09:29:58 PM' to '25'. Filter options include 'All Run Types' and 'All Run States' with a 'Clear Filters' button, and an 'Auto-refresh' toggle. Below these are various task status filters like deferred, failed, 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' with one total task, all of which are BashOperator tasks. The DAG details section shows no owners and no tags, with a schedule interval of 'No schedule interval'. Handwritten annotations include:

- A large blue arrow points from the text 'several views of dags are available.' to the 'Grid' tab.
- A blue arrow points from the text 'start/delete refresh.' to the start/stop and refresh icons in the top right.
- A blue arrow points from the text 'only one task' to the 'generate_random_number' task in the DAG summary.

several views of dags are available.

start/delete refresh.

only one task

DAG: example_dag

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

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	

generate_random_number

DAG graph view

The screenshot shows the 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 with the DAG name `example_dag`, its schedule (1 day, 0:00:00), and next run (2023-02-01, 00:00:00). A yellow highlight is on the `Graph` tab in the main navigation bar, which is currently active. Other tabs include Grid, Calendar, Task Duration, Task Tries, Landing Times, and Gantt. Below the tabs are buttons for Details, Code, and Audit Log. The main area features a search bar with date (01/28/2024, 09:30:30 PM), limit (25), run types (All Run Types), run states (All Run States), a Clear Filters button, and an Auto-refresh toggle. A keyboard shortcut keytip for `shift + /` is shown above the filters. A list of run statuses is provided: 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 graph view. The graph shows a single task named `generate_random_number` (BashOperator). A yellow highlight is on this task node. To the right of the graph is 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 are navigation links: Airflow, DAGs, Cluster Activity, Datasets, Security, Browse, Admin, Docs, 22:14 UTC, and Log In. Below the header, it displays the DAG name `example_dag`, its schedule (1 day, 0:00:00), and next run time (2023-02-01, 00:00:00). A yellow box highlights the `<> Code` tab in the navigation bar. The interface includes a toolbar with Grid, Graph, Calendar, Task Duration, Task Tries, Landing Times, Gantt, Details, Audit Log, and a refresh button. Below the toolbar are date/time filters (01/28/2024, 10:14:28 PM, 25 days, All Run Types, All Run States, Clear Filters, Auto-refresh), a status filter (deferred, failed, queued, removed, restarting, running, scheduled, shutdown, skipped, success, up_for_reschedule, up_for_retry, upstream_failed, no_status), and a note about keyboard shortcuts. The main content area shows the DAG code under the `example_dag` heading, with tabs for Details, Graph, Gantt, and Code. The code is as follows:

```
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 blue arrow points from the handwritten note to the word "Toggle Wrap" in the code editor.

it is read
only, any change
will done through
original script only

Audit logs

The screenshot shows the Airflow web interface. At the top, there is a navigation bar with links: Airflow, DAGs, Cluster Activity, Datasets, Security, Browse, Admin, Docs, 04:14 UTC, and Log In. The 'Browse' link is highlighted with a red box. A dropdown menu for 'Browse' is open, showing options: DAG Runs, Jobs, Audit Logs, Task Instances, Task Reschedules, Triggers, SLA Misses, and DAG Dependencies. The 'Audit Logs' option is also highlighted with a red box and has a blue arrow pointing to it from the right. To the right of the dropdown, the text 'Provides troubleshooting' is handwritten in blue. Below the dropdown, a message 'Record Count: 3' is displayed. The main content area is titled 'List Log' and contains a table with three rows of audit log data. The columns are: Id, Dttm, Dag Id, Task Id, Event, Log, and xtra. The first row shows an event for 'cli_dag_reserialize' at 2024-01-29, 04:13:04. The second row shows an event for 'cli_scheduler' at 2024-01-29, 04:12:44. The third row is partially visible.

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 AIRFLOW IN PYTHON